Elementi di analisi per Visione Artificiale

Paolo Medici
Dipartimento di Ingegneria dell'Informazione di Parma

2 ottobre 2025

Questo libro vuole essere una introduzione ragionevolmente sintetica ai fondamenti di geometria, algebra e statistica necessari alla comprensione e all'utilizzo delle tecniche più avanzate di visione artificiale. Come si vedrà, sono stati inseriti diversi elementi non propriamente di elaborazione di immagini ma che risultano utili a chi interessa sviluppare applicazioni complesse basate sull'elaborazione di immagini, coinvolgendo concetti come il *tracking* o la fusione sensoriale di alto livello. Per non appesantire la trattazione ho cercato, ove possibile, di non entrare nelle dimostrazioni dei diversi teoremi ma, con lo scopo di stimolare la curiosità, ha lasciato la loro trattazione al lettore. L'obiettivo originale di questo libro infatti non è mai stato quello di realizzare una trattazione rigorosa ed esaustiva dove spesso ci si perde in calcoli e dimostrazioni con il rischio di stancare il lettore e distogliere l'attenzione verso invece alcuni concetti importanti. Allo stesso modo non mi sono posto come obiettivo quello di voler parlare di ogni qualsiasi argomento relativo all'elaborazione di immagini e alla visione artificiale ma mi sono limitato di fatti a quei soli argomenti inerenti alle sperimentazioni che ho direttamente affrontato nelle mie attività di ricerca, di cui sono più confidente e su cui posso dare un minimo di contributo. La stesura di questo libro è stata infatti fortemente influenzata dalle mie aree di ricerca le quali riguardano principalmente applicazioni della Visione Artificiale alla percezione di robot e allo sviluppo e controllo di veicoli autonomi.

La *Computer Vision* è un campo della scienza estremamente stimolante, anche per i non addetti ai lavori. Il fatto stesso che nella visione artificiale geometria, statistica, ottimizzazione sono argomenti così strettamente correlati ne fa un ambito di studio completo e degno di interesse anche ad esterni alla materia. Questa ampia correlazione tra gli argomenti tuttavia non ha aiutato l'attività di divisione in capitoli di questo libro e di conseguenza i rimandi tra un capitolo e gli altri, come si vedrà, sono ampiamente diffusi.

Le citazioni inserite nel testo sono molto ridotte e faccio riferimento solo a testi che io ritengono fondamentali e, quando possibile, ho citato i primi che hanno proposto l'idea alla base della teoria: la lettura degli articoli citati in bibliografia è caldamente consigliata.

Ho introdotto, quando possibile, il termine inglese corrispondente al termine italiano non per anglofilia ma per suggerire le eventuali parole chiave da cercare su internet in modo da individuare argomenti collegati a quello trattato.

Per l'organizzazione di questo volume ho tratto spunto da diversi libri, libri di cui consiglio la lettura, tra cui *"Multiple View Geometry"* [HZ04] di Hartley e Zisserman, *"Pattern Recognition and Machine Learning"* [Bis06] e *"Emerging Topics In Computer Vision"* [MK04] redatto da Medioni e Kang. Per tematiche più strettamente legate all'elaborazione delle immagini, un ottimo libro, disponibile anche online, può essere *"Computer Vision: Algorithms and Applications"* di Szeliski [Sze10].

Per ultima cosa, sottolineo che questo libro è stato scritto negli ultimi 20 anni e alcune cose potrebbero essere molto outdated (sopratutto a valle della rivoluzione del machine learning avvenuta a metà del decennio scorso) ma mi piace tenerle per motivi storici.

La sintassi matematica che verrà usata è minimalista:

- le matrici verranno indicate con lettere in grassetto maiuscolo $\mathbf{A}$ mentre i vettori in grassetto minuscolo $\mathbf{x}$;
- la trasposta dell'inversa di una matrice $\mathbf{A}$, ovvero $\left(\mathbf{A}^{-1}\right)^{\top} = \left(\mathbf{A}^{\top}\right)^{-1}$, si scriverà $\mathbf{A}^{-\top}$;
- negli ambiti statistici, la sintassi $\hat{x}$ indica il valore stimato della quantità x;
- salvo indicazioni differenti, la sintassi ${}^{b}\mathbf{R}_{a}$ indica una matrice di cambiamento di base che trasforma il sistema di riferimento a nel sistema di riferimento b mentre ${}^{a}\mathbf{v}$ indica un vettore espresso nel sistema di riferimento a.

Rimando infine all'appendice B per avere una breve carrellata sul significato dei simboli usati in questo libro.

È possibile trovare l'ultima versione di questo documento a http://www.ce.unipr.it/medici. Tutto il materiale di "Elementi di analisi per Visione Artificiale" è rilasciato sotto licenza Creative Commons 4.0. Il testo completo della licenza è disponibile, in inglese, alla pagina https://creativecommons.org/licenses/by-nc-sa/4.0/.

This document is a brief introduction to the fundamentals of geometry, algebra and statistics needed to understand and use computer vision techniques. You can find the latest version of this document at http://www.ce.unipr.it/medici. This manual aim to give technical elements about image elaboration and artificial vision. Demonstrations are usually not provided in order to stimulate the reader and left to him. This work may be distributed and/or modified under the conditions of the Creative Commons 4.0. The latest version of the license is in https://creativecommons.org/licenses/by-nc-sa/4.0/.

Capitolo 1

Elementi

Questo primo capitolo espone diversi argomenti di analisi matematica e geometria analitica necessari alla comprensione e utilizzo degli algoritmi di algebra, di statistica e, ovviamente, di visione delle macchine che verranno discussi poi nei capitoli successivi.

1.1 Sistemi lineari sovradimensionati

Analizzando sistemi reali è facile imbattersi nel problema di dover ricavare la "soluzione" di un sistema lineare sovradimensionato.

L'importanza di questo argomento è evidente: quando si eseguono osservazioni su un sistema reale questo risulta naturalmente affetto da rumore, appunto, di osservazione. Questo rumore chiaramente compromette il risultato della singola osservazione ma, fortunatamente, è normalmente possibile acquisire molte più osservazioni che incognite ottenendo così un sistema sovradimensionato. In queste condizioni, per ottenere una soluzione al problema che minimizzi l'errore, è richiesto l'utilizzo di una tecnica di regressione numerica, per esempio, ai minimi quadrati. In questa prima sezione sono presentate tecniche matematiche ampiamente utilizzate in tutto il libro: per ulteriori dettagli riguardo queste tecniche si può fare riferimento al capitolo 3 incentrato totalmente su questo argomento.

Si abbia pertanto un sistema lineare *sovradimensionato* (*overdetermined*)

$$\mathbf{A}\mathbf{x} = \mathbf{y} \tag{1.1}$$

dove $\mathbf{A}$ è una matrice rettangolare $m \times n$ e con $m \geq n$. Tale matrice, essendo rettangolare, non ammette inversa ma è comunque possibile definire per ogni possibile soluzione $\mathbf{x} \in \mathbb{R}^n$ un valore dell'errore, detto anche residuo, che questa eventuale soluzione comporterebbe. Non esiste una soluzione generale per un sistema sovradimensionato ma solo soluzioni che minimizzano il residuo sotto una particolare metrica.

Definiamo[1] come metrica dell'errore il modulo del residuo

$$\epsilon(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2 \tag{1.2}$$

La soluzione cosiddetta "ai minimi quadrati" di un sistema lineare (1.1) è rappresenta dal vettore $\mathbf{x}$ che minimizza la distanza euclidea del residuo (1.2) ovvero cercare la soluzione ottima del sistema, nei sensi di una regressione ai minimi quadrati, equivale a trovare il minimo di tale funzione errore al variare di $\mathbf{x}$.

Se si moltiplica l'equazione (1.1) per $\mathbf{A}^\top$ si ottiene un sistema lineare "tradizionale" che ammette come soluzione

$$\mathbf{A}^\top\mathbf{A}\mathbf{x} = \mathbf{A}^\top\mathbf{y} \tag{1.3}$$

Questo è un primo metodo risolutivo per sistemi sovradimensionati e viene indicato in letteratura come tecnica delle equazioni *perpendicolari* (*normal equations*): il problema originale viene ricondotto a un sistema lineare classico dove la matrice dei coefficienti è quadrata e pertanto invertibile con tecniche classiche. Tuttavia la soluzione proposta in equazione (1.3) è numericamente instabile in quanto $\text{cond} \approx \mathbf{A}^2$. Dettagli ulteriori sul condizionamento delle matrici e sulla propagazione dei disturbi nella soluzione dei sistemi lineari ben dimensionati o sovradimensionati saranno presentati in sezione 2.7.

Se il sistema è ben condizionato, la tecnica più stabile per risolvere un problema alle *normal equations* è la fattorizzazione di Cholesky.

Si può dimostrare che una soluzione $\mathbf{x}$, meglio condizionata e che minimizza la funzione (1.2), esiste e vale:

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{y} \\ \mathbf{A}^\top\mathbf{A}\mathbf{x} &= \mathbf{A}^\top\mathbf{y} \\ \mathbf{x} &= \left(\mathbf{A}^\top\mathbf{A}\right)^{-1}\mathbf{A}^\top\mathbf{y} \end{aligned} \tag{1.4}$$

[1] la motivazione di questa scelta verrà discussa nei dettagli nel capitolo sullo studio dei modelli.

Per costruzione $\mathbf{x}$ è una soluzione del sistema (1.1) ed è anche il vettore che minimizza la funzione (1.2). Viene indicata con $\mathbf{A}^+$ la matrice pseudoinversa (*pseudoinverse matrix*) di $\mathbf{A}$ e vale

$$\mathbf{A}^+ = \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top \tag{1.5}$$

Questa soluzione del sistema è detta pseudoinversa di Moore-Penrose.
La pseudoinversa ha le seguenti proprietà

- La pseudoinversa di una matrice esiste se esiste l'inversa di $\mathbf{A}^\top \mathbf{A}$;
- La pseudoinversa di una matrice quadrata coincide con la sua inversa;
- La pseudoinversa di una matrice, se esiste, è unica.

È necessario precisare fin da subito che nel minimizzare la quantità (1.2) non si è fatta nessuna ipotesi sulla distribuzione del rumore all'interno delle varie componenti di cui la matrice è composta: senza tale informazione non c'è garanzia che la soluzione sarà ottima dal punto di vista statistico. Senza ipotesi sulla distribuzione del rumore, la soluzione ottenuta con questa minimizzazione è infatti una soluzione puramente algebrica che minimizza appunto un errore algebrico (*algebraic error*).

È possibile ottenere una soluzione leggermente migliore dal punto di vista statistico quando il rumore è gaussiano bianco a media nulla e si conosce il valore della varianza del rumore su ogni osservazione. In questo caso è possibile assegnare ad ogni equazione del sistema pesi differenti, moltiplicando ogni riga del sistema per un opportuno peso in modo da pesare in maniera differente ogni dato acquisito. Discussione più approfondita su questo argomento si trova in sezione 3.2 e in generale nel capitolo 2 si affronterà il caso generale dove si conosce il modo con cui l'errore sui dati osservati incide sulla stima dei parametri.

Esistono invece delle tecniche stabili basate su fattorizzazioni che permettono di ricavare la soluzione partendo direttamente dalla matrice $\mathbf{A}$.

Usando per esempio la fattorizzazione QR, algoritmo notoriamente stabile dal punto di vista numerico, della matrice $\mathbf{A}$ il problema originale (1.1) si trasforma nel problema $\mathbf{QRx} = \mathbf{y}$ e la soluzione si può ricavare da $\mathbf{Rx} = \mathbf{Q}^\top \mathbf{y}$, sfruttando l'ortogonalità della matrice $\mathbf{Q}$. Nella fattorizzazione QR vige la relazione $\mathbf{R}^\top \mathbf{R} = \mathbf{A}^\top \mathbf{A}$ ovvero $\mathbf{R}$ è fattorizzazione di Cholesky di $\mathbf{A}^\top \mathbf{A}$: attraverso questa relazione si può ricavare infine la pseudoinversa in maniera esplicita.

Attraverso invece la Decomposizione ai Valori Singolari *Singular Value Decomposition* (SVD), la matrice sovradimensionata $\mathbf{A}$ viene scomposta in 3 matrici dalle proprietà interessanti. Sia $\mathbf{A} = \mathbf{USV}^*$ la decomposizione ai valori singolari (SVD) di $\mathbf{A}$. $\mathbf{U}$ è una matrice unitaria di dimensioni $m \times n$ (a seconda del formalismo usato, *complete SVD* o *economic SVD*, le dimensioni delle matrici possono cambiare, e $\mathbf{U}$ diventare $m \times m$), $\mathbf{S}$ è una matrice diagonale che contiene i valori singolari (gli autovalori della matrice $\mathbf{AA}^\top$, di dimensioni, a seconda del formalismo, $n \times n$ o $m \times n$) e $\mathbf{V}^*$ è una matrice ortonormale, trasposta coniugata, di dimensioni $n \times n$.

Attraverso un procedimento puramente matematico si ottiene che la pseudoinversa di $\mathbf{A}$ equivale a

$$\mathbf{A}^+ = \mathbf{VS}^+\mathbf{U}^* \tag{1.6}$$

dove la pseudoinversa di una matrice diagonale $\mathbf{S}^+$ equivale alla sua inversa ovvero una matrice diagonale costituita dai reciproci dei rispettivi valori.

Riassumendo, i modi per risolvere un sistema lineare sovradimensionato sono

- Usando le normal-equations $(A^\top A)x = A^\top b$ ovvero la pseudo-inversa di Moore-Penrose $x = (A^\top A)^{-1} A^\top b$ (siccome $(A^\top A)$ è semidefinita positiva si può usare Cholesky come risolutore);
- La decomposizione QR $A = QR \rightarrow x = R^{-1} Q^\top b$;
- la decomposizione SVD $A = U\Sigma V^\top \rightarrow x = A^+ b = V\Sigma^+ U^\top b$;
- La decomposizione di Cholesky $A^\top Ax = U^\top Ux = A^\top b \rightarrow x = U^{-1}\left(U^{-\top} A^\top b\right)$ (essendo U la matrice triangolare superiore U^{-1} e $U^{-\top}$ sono facili da calcolare);

Dal punto di vista del calcolo numerico si può ridurre il numero di condizione della matrice da invertire scalando le colonne di A.

Dettagli ulteriori sulla pseudoinversa di Moore-Penrose possono essere trovati in molti libri, per esempio in [CM09] o nel testo fondamentale di calcolo numerico [GVL96].

Esaminiamo ora il caso in cui il sistema lineare da risolvere sia invece omogeneo.

Un sistema lineare omogeneo ha la forma

$$\mathbf{Ax} = 0 \tag{1.7}$$

e normalmente la soluzione ovvia $\mathbf{x} = 0$, che è si ottiene anche attraverso l'equazione (1.4), non risulta utile ai fini del problema. In questo caso è necessario trovare, sempre ai sensi di una regressione ai minimi quadrati, un $\mathbf{x} \in \mathbb{R}^n$, non nullo,

rappresentate un sottospazio vettoriale ovvero il *kernel* di $\mathbf{A}$. Il vettore generatore del sottospazio è conosciuto a meno di uno o più fattori moltiplicativi (a seconda della dimensione del sottospazio nullo). Per ottenere una soluzione unica è necessario imporre un vincolo aggiuntivo, per esempio $|\mathbf{x}| = 1$, tale da poter così formalizzare

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x}\|^2 \tag{1.8}$$

con il vincolo $|\mathbf{x}| = 1$ ovvero realizzando una minimizzazione vincolata.

Anche in questo caso la SVD si dimostra una tecnica estremamente efficace e computazionalmente stabile: le basi del *kernel* di $\mathbf{A}$ infatti sono esattamente le colonne di $\mathbf{V}$ associate ai valori singolari (autovalori) nulli della matrice diagonale $\mathbf{S}$. In genere, a causa della presenza di rumore, non esisterà un valore singolare esattamente nullo ma deve essere scelta la colonna associata al minimo valore singolare.

Gli autovettori associati a valori singolari nulli della matrice $\mathbf{S}$ rappresentano pertanto il *kernel* della matrice stessa e il numero di autovalori nulli rappresenta la dimensione del *kernel* stesso. Va notato come nell'equazione (1.6) la presenza di zeri nella matrice diagonale $\mathbf{S}$ fosse problematica: ora si capisce che tale presenza è sintomo del fatto che una delle componenti del problema è totalmente incorrelata con la soluzione e, in quanto tale, potrebbe essere trascurata: tale risultato infatti sarà utilizzato nella sezione 2.10.1 nella trattazione dell'agoritmo PCA.

La soluzione del sottospazio di $\mathbf{A}$ è pertanto

$$\mathbf{x} = \sum_{i=1}^{N} \beta_i \mathbf{v}_i \tag{1.9}$$

dove $\mathbf{v}_i$ sono le colonne della matrice $\mathbf{V}$, vettori singolari "destri", di $\mathbf{A}$ corrispondenti ad N valori singolari, autovalori di 0 della matrice $\mathbf{A}^\top\mathbf{A}$.

La decomposizione SVD risulta una delle tecniche più stabili e versatili sviluppata negli ultimi anni per la risoluzione di sistemi lineari e, in tutto questo libro, si farà larghissimo uso di tale tecnologia.

1.2 Autovalori e Autovettori

Gli autovalori e autovettori sono stati già anticipati nella sezione precedente. In questa verrà fatta una trattazione minimale per poterli usare in maniera proficua.

Definizione 1 *Data una matrice quadrata* $\mathbf{A}$ *di ordine* n*, un numero (reale o complesso)* λ *e un vettore non nullo* $\mathbf{x}$ *sono detti rispettivamente* autovalore *e* autovettore *di* $\mathbf{A}$ *se vale la relazione*

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \tag{1.10}$$

$\mathbf{x}$ *è anche detto* autovettore *associato all'*autovalore λ.

Un autovettore è un vettore $\mathbf{x} \neq 0$ che non cambia direzione a seguito della trasformazione (applicazione) lineare geometrica $\mathbf{A}$ ma cambia solo il modulo di un fattore λ detto autovalore.

Riscrivendo il sistema (1.10) usando la matrice identità $\mathbf{I}$, segue che autovalore e autovettore associato si ottengono come soluzione del sistema omogeneo:

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0 \tag{1.11}$$

Se $\mathbf{x}$ è un autovettore di $\mathbf{A}$ associato all'autovalore λ e $t \neq 0$ un numero (reale o complesso), allora anche $t\mathbf{x}$ è un autovettore di λ.

In generale l'insieme dei vettori $\mathbf{x}$ associati a un autovalore λ di $\mathbf{A}$ forma un sottospazio di $\mathbb{R}^n$ chiamato *autospazio*. La dimensione di questo sottospazio è detta molteplicità geometrica dell'autovalore.

Definizione 2 *Il* polinomio caratteristico *di* $\mathbf{A}$ *nella variabile* x *è il polinomio definito nel modo seguente:*

$$p(x) = \det(\mathbf{A} - x\mathbf{I}) \tag{1.12}$$

Dalla definizione (1.11) si evince che λ è un autovalore se e solo se $p(\lambda) = 0$. Le radici del polinomio caratteristico sono gli autovalori di $\mathbf{A}$ e di conseguenza il polinomio caratteristico ha grado pari alla dimensione della matrice. Le matrici 2×2 e 3×3 hanno polinomi caratteristici notevoli.

Proprietà degli Autovalori e Autovettori

- $\mathbf{A}$ e $\mathbf{A}^\top$ hanno gli stessi autovalori;
- Se $\mathbf{A}$ è non singolare, e λ è un suo autovalore, allora λ^{-1} è autovalore di $\mathbf{A}^{-1}$;
- Se $\mathbf{A}$ è ortogonale, allora $|\lambda| = 1$;

- $\lambda = 0$ è autovalore di $\mathbf{A}$ se e solo se $\det(\mathbf{A}) = 0$;
- Gli autovalori di matrici diagonali e triangolari (superiori e inferiori) sono gli elementi della diagonale principale;
- La somma degli elementi diagonali è uguale alla somma degli autovalori ovvero $\operatorname{trace} \mathbf{A} = \sum \lambda_i$;
- Il determinante di una matrice è uguale alla produttoria dei propri autovalori ovvero $\det \mathbf{A} = \prod \lambda_i$;
- Le matrici simmetriche hanno autovalori reali e autovettori ortogonali pertanto l'insieme corrispondente di autovettori unitari $\{\mathbf{v}_i\}$ è un insieme di vettori ortonormali;
- Una matrice quadrata $\mathbf{A}$ può essere espressa in termine della matrice degli autovalori $\mathbf{D} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ e autovalori $\mathbf{V}$ nella forma
$$\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1} \tag{1.13}$$
- Una matrice quadrata simmetrica $\mathbf{A}$ può essere espressa in termine dei suoi autovalori λ_i reali e autovalori $\{\mathbf{v}_i\}$ nella forma
$$\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^{\top} = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^{\top} \tag{1.14}$$
detta decomposizione spettrale di $\mathbf{A}$.

1.3 Parametrizzazioni alternative nello spazio e nelle varietà

I punti nei vari spazi $\mathbb{R}^n$ possono essere descritti con coordinate diverse da quelle cartesiane. In questa sezione vengono presentate alcune parametrizzazioni utili che verranno usate nel resto del libro.

Introduciamo la seguente definizione:

Definizione 3 *S^n è la sfera unitaria in $\mathbb{R}^n$ tale che:*

$$S^n := \left\{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|^2 = 1\right\} \tag{1.15}$$

Siccome una generica parametrizzazione su S^n avrà $n+1$ componenti e 1 solo vincolo questa deve possedere per definizione n gradi di libertà (DOF).

Nel caso $n = 0$ la "sfera" unitaria $S^0 = \{-1, +1\}$ è formata da solo 2 punti e perciò non è una varietà connessa.

Con $n = 1$ la varietà è esattamente la stessa di $SO(2)$ ovvero con la parametrizzazione $S^1 = \left\{ \begin{bmatrix} \cos\alpha \\ \sin\alpha \end{bmatrix} ; \alpha \in \mathbb{R} \right\}$.

La sfera S^2 (la superficie della sfera o una direzione in $\mathbb{R}^3$) invece è una 2-varietà che non ha struttura di gruppo. Può essere parametrizzata da due parametri (ad esempio le coordinate polari come vedremo tra poco) ma presenterà sempre delle singolarità.

1.3.1 Coordinate Polari

Dando per assodate le coordinate cartesiane, in questa sezione vengono introdotte le coordinate polari e in particolare verranno mostrate le relazioni che legano le coordinate cartesiane a quelle polari.

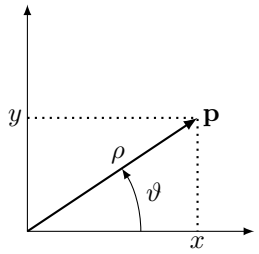


Figura 1.1: Corrispondenza tra coordinate polari e cartesiane.

Per un punto nello spazio bidimensionale la relazione che lega questi due sistemi di coordinate si scrive come:

$$\begin{aligned} x &= \rho\cos\vartheta \\ y &= \rho\sin\vartheta \end{aligned} \tag{1.16}$$

La trasformazione inversa, da cartesiane a polari, è

$$\begin{aligned} \rho &= \sqrt{x^2 + y^2} \\ \vartheta &= \operatorname{atan2}(y, x) \end{aligned} \tag{1.17}$$

Un punto su una sfera non ha invece una rappresentazione univoca: per lo stesso motivo, come verrà sottolineato più volte in appendice, esistono infinite rappresentazioni di una rotazione nello spazio tridimensionale.

Una scelta molto diffusa sono le coordinate polari sferiche (*sperical coordinate system*).

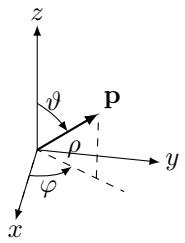


Figura 1.2: Coordinate polari in 3 dimensioni: coordinate sferiche.

Con questa convenzione la relazione tra le coordinate cartesiane e quelle polari si scrive

$$\begin{aligned} x &= \rho \sin\vartheta \cos\varphi \\ y &= \rho \sin\vartheta \sin\varphi \\ z &= \rho \cos\vartheta \end{aligned} \tag{1.18}$$

dove ϑ è definito come *zenit* mentre φ è chiamato *azimuth*.

La trasformazione inversa, da coordinate cartesiane a polari, si ottiene come

$$\begin{aligned} \rho &= \sqrt{x^2+y^2+z^2} \\ \varphi &= \operatorname{atan2}(y,x) \\ \vartheta &= \operatorname{atan2}(\sqrt{x^2+y^2}, z) = \arccos(z/\rho); \end{aligned} \tag{1.19}$$

1.3.2 Proiezione Stereografica

Una alternativa abbastanza comune per parametrizzare la sfera S^n è usare la proiezione stereografica per trasformare coordinate dallo spazio della varietà $\mathbb{R}^n$ (spazio parametri) a $\mathbb{R}^{n+1}$ (spazio cartesiano) e viceversa.

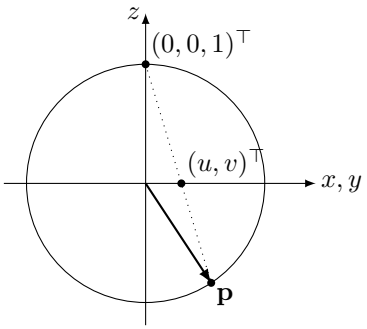


Figura 1.3: Proiezione stereografica.

Nella sfera tridimensionale S^2, è possibile definire una funzione φ_{+3} come proiezione stereografica dallo spazio $U_{+3} := S^2 / \left[0,0,1\right]^\top$ a $\mathbb{R}^2$:

$$\begin{aligned} \varphi_{+3} &: U_{+3} \mapsto \mathbb{R}^2 \\ \varphi_{+3}\left(\left[x,y,z\right]^\top\right) &= \frac{1}{1-z}\begin{bmatrix} x \\ y \end{bmatrix} \end{aligned} \tag{1.20}$$

insieme alla sua inversa

$$\varphi_{+3}^{-1}\left(\left[u,v\right]^\top\right) = \frac{1}{1+u^2+v^2}\begin{bmatrix} 2u \\ 2v \\ -1+u^2+v^2 \end{bmatrix} \tag{1.21}$$

avendo indicato con $\left[0,0,1\right]^\top$ il "polo nord" della sfera. φ_{+3} e φ_{+3}^{-1} sono continue in U_{+3} e dunque la proiezione stereografica è un omeomorfismo. In questa proiezione si crea una relazione tra il punto $(u,v,0)$ sul piano $z=0$ e il punto sulla sfera (x,y,z) intersezione tra il raggio proiettivo che unisce l'origine $(0,0,1)^\top$ (unico punto di singolarità) con il punto del piano e la sfera di raggio unitario, come mostrato in figura 1.3.

Allo stesso modo possono essere definiti degli spazi $U_{\pm i} := S^2 / \pm e_i$ dove gli e_i sono i versori unitari dentro i quali definire 6 parametrizzazioni simili (ognuna con la sua diversa singolarità) e questo permette di scegliere la parametrizzazione più opportuna in modo da operare nel punto più distante dalla singolarità di quella specifica formula.

1.4 Coordinate Omogenee

In questa sezione vengono introdotte le coordinate omogenee, un artificio matematico che risulta molto utile per la discussione del problema della geometria proiettiva ma anche di diversi formalismi affrontati in diversi capitoli di questo libro.

Chiameremo coordinate omogenee (*homogeneous coordinates*) di un punto del piano $\mathbf{p} = (x, y) \in \mathbb{R}^2$ una qualsiasi terna ordinata $\tilde{\mathbf{p}} = (x', y', w') \in \mathbb{R}^3$ di numeri reali tali che $w' \neq 0$, $\frac{x'}{w'} = x$ e $\frac{y'}{w'} = y$. Allo stesso modo coordinate omogenee di un punto $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ saranno una quadrupla di numeri $\tilde{\mathbf{p}} = (x', y', z', w') \in \mathbb{R}^4$ tali che $w' \neq 0$ e $\frac{x'}{w'} = x$, $\frac{y'}{w'} = y$ e $\frac{z'}{w'} = z$.

Il punto $\tilde{\mathbf{p}}$ espresso in coordinate omogenee equivale al punto *reale* $\mathbf{p}$ (*inhomogeneous*):

$$\tilde{\mathbf{p}} = (x', y', w') = w'(\frac{x'}{w'}, \frac{y'}{w'}, 1) = w'(x, y, 1) = w'\mathbf{p}$$

Il vettore $(x, y, 1)$ è chiamato *augmented vector*.

Le coordinate omogenee hanno le seguenti proprietà:

- Le coordinate omogenee sono definite a meno di un coefficiente di proporzionalità. Ad esempio, la terna $(x, y, 1)$ ed ogni suo multiplo $\lambda \neq 0$, ovvero $(x, y, 1) \cong (\lambda x, \lambda y, \lambda)$, sono coordinate omogenee dello stesso punto dello spazio (x, y);
- I punti in coordinate omogenee con coordinata $w = 0$ sono detti **impropri**, *points at infinity* o *ideal points*, e non hanno nessun significato geometrico nello spazio cartesiano, ma possono rappresentare un punto all'infinito, nella direzione del vettore (x, y).

In coordinate omogenee c'è pertanto distinzione tra vettore ($w = 0$) e punto ($w \neq 0$), cosa che non accade con le coordinate euclidee. L'insieme costituito da tutte le terne/quaterne non nulle forma uno spazio proiettivo bidimensionale/tridimensionale.

Le coordinate omogenee permettono di rappresentare punti all'infinito e consentono di esprimere tutte le trasformazioni di coordinate geometriche usate in visione artificiale in forma matriciale. L'uso di coordinate omogenee è usato in *computer graphics* per la proprietà di rappresentare, esattamente come nel caso cartesiano, le trasformazioni affini attraverso l'uso di matrici ed in più permettono di rappresentare con lo stesso formalismo anche le proiezioni prospettiche.

1.5 Linee, Piani e Iperpiani

In questa sezione viene fatto un breve riassunto delle equazioni delle rette e, per estensione, degli iperpiani. Una retta è un insieme di punti che separa il piano cartesiano in due parti, il piano è l'insieme di punti che separa lo spazio tridimensionale in due parti e, generalizzando, l'iperpiano è quell'insieme di punti che separa lo spazio $\mathbb{R}^n$ in due parti. Questa definizione tornerà utile quando si parlerà in seguito di classificazione.

1.5.1 Retta

Esistono diverse formulazioni per esprimere il concetto di retta.

Nel caso più generale, quello multidimensionale, una retta, luogo dei punti $\mathbf{x} \in \mathbb{R}^n$ di dimensione 1, assume la forma

$$\mathbf{x} = \mathbf{p} + t\mathbf{v} \tag{1.22}$$

dove $\mathbf{p} \in \mathbb{R}^n$ è un generico punto di origine, $\mathbf{v} \in \mathbb{R}^n$ è il vettore direzione e $t \in \mathbb{R}$ è uno scalare. In questo caso si parla di raggio parametrico (*parametric ray*).

In buona parte delle applicazioni la retta è un concetto tipico dello spazio bidimensionale. In questo spazio, trascurando l'equazione della retta scritta in forma esplicita $y = mx + q$ in quanto presenta singolarità, dedichiamo l'attenzione alla retta scritta in forma implicita. L'equazione della retta scritta in forma implicita è:

$$ax + by + c = 0 \tag{1.23}$$

Tale rappresentazione è molto utile perché permette di considerare sia rette orizzontali che verticali senza singolarità alcuna. Il parametro c vale zero quando la retta passa per l'origine e, ovviamente, la retta passa per un punto (x', y') quando $c = -ax' - by'$.

Nel caso bidimensionale l'equazione del raggio parametrico (1.22) si riduce all'equazione della retta implicita di parametri

$$(a, b) \cdot \mathbf{v} = 0 \quad c = -(a, b) \cdot \mathbf{p} \tag{1.24}$$

Dalla prime delle equazioni (1.24) si vede come il vettore formato dai parametri (a, b) e il vettore direttrice siano ortogonali tra loro. Il vettore generatore dalla retta è infatti proporzionale per esempio a $\mathbf{v} \propto (-b, a)$ o $\mathbf{v} \propto (\frac{1}{a}, -\frac{1}{b})$. Il vettore $\mathbf{v}'$ ortogonale alla retta data è semplicemente $\mathbf{v}' \propto (a, b)$ e la retta ortogonale a quella data ha una equazione implicita scritta nella forma

$$bx - ay + c' = 0 \tag{1.25}$$

dove c' si ottiene selezionando il punto della retta originale in cui deve passare la perpendicolare.

I parametri della retta scritta in forma implicita sono omogenei (l'equazione (1.23) viene infatti chiamata equazione omogenea della retta) ovvero rappresentano un sottospazio vettoriale di $\mathbb{R}^3$: qualunque multiplo di tali parametri rappresenta la medesima retta. Tali parametri sono pertanto definiti a meno di un fattore moltiplicativo. Questa considerazione suggerisce un ulteriore modo per rappresentare una retta e un generico iperpiano.

Le rette, scritte in forma omogenea implicita, devono soddisfare l'equazione (prodotto scalare):

$$\mathbf{l}^\top \mathbf{x} = 0 \tag{1.26}$$

con $\mathbf{x} \in \mathbb{R}^3$ punto in coordinate omogenee e $\mathbf{l} = (a, b, c)^\top$ i parametri della retta. Per le coordinate omogenee si veda la precedente sezione 1.4 mentre si veda per le implicazioni di questa scrittura, sul dualismo punto-retta, il paragrafo 1.5.7.

Siccome la retta implicita è conosciuta a meno di un fattore moltiplicativo, esistono infiniti modi di esprimere la medesima retta. Una possibile normalizzazione della retta si ottiene dividendo i parametri per la lunghezza $\sqrt{a^2 + b^2}$. In tal caso si ottiene una soluzione particolare della retta in quanto i parametri sono quelli di una retta scritta in coordinate polari nella stessa forma di equazione (1.46) e conseguentemente con questa normalizzazione il parametro c rappresenta la minima distanza tra la retta e l'origine degli assi.

Infine, essendo la retta un iperpiano in 2 dimensioni, la sua equazione può essere scritta come in equazione (1.49).

1.5.2 Retta passante per due punti

Per due punti (x_0, y_0) e (x_1, y_1) dello spazio cartesiano $\mathbb{R}^2$ passa una retta implicita di equazione

$$(y_1 - y_0)x - (x_1 - x_0)y - y_1 x_0 + x_1 y_0 = 0 \tag{1.27}$$

dove è ben visibile il fatto che non esistano singolarità e tutti i valori sono ammissibili. Indicando con (d_x, d_y) la differenza tra i due punti, la retta passante per un punto (x_0, y_0) e diretta lungo il vettore (d_x, d_y) ha equazione

$$d_y x - d_x y + y_0 d_x - x_0 d_y = 0 \tag{1.28}$$

Generalizzando al caso n-dimensionale, l'equazione della retta in $\mathbb{R}^n$, passante per due punti $\mathbf{p}$ e $\mathbf{q}$ scritti in forma omogenea, è il luogo dei punti $\mathbf{x} \in \mathbb{R}^n$ tale che

$$\mathbf{x} = (1 - t)\mathbf{p} + t\mathbf{q} = \mathbf{p} + t(\mathbf{q} - \mathbf{p}) \tag{1.29}$$

equazione del raggio parametrico con $t \in \mathbb{R}$ valore scalare. I valori di $\mathbf{x}$ associati a valori $t \in [0, 1]$ sono punti interni al segmento $(\mathbf{p}, \mathbf{q})$.

Usando invece le coordinate omogenee, limitatamente al caso cartesiano bidimensionale, si ottiene il seguente risultato notevole: la retta di parametri $\mathbf{l} = (a, b, c)^\top$, passante per i punti $\mathbf{x}_1$ e $\mathbf{x}_2$, si ottiene come

$$\mathbf{l} = \mathbf{x}_1 \times \mathbf{x}_2 \tag{1.30}$$

in quanto, un qualsiasi punto $\mathbf{x}$, per appartenere alla retta deve soddisfare l'equazione (1.26).

1.5.3 Distanza punto-retta

La distanza di un punto (x', y') da una retta retta (*line-point distance*), intesa come distanza ortogonale, ovvero distanza tra il punto dato e il punto più vicino sulla retta, vale:

$$d = \frac{|ax' + by' + c|}{\sqrt{a^2 + b^2}} \tag{1.31}$$

Nel caso n-dimensionale il punto $\mathbf{x}$ appartenente alla retta di equazione (1.22) più vicino a un punto $\mathbf{m}$ è quel punto per il quale lo scalare t assume il valore

$$t = (\mathbf{m} - \mathbf{p}) \cdot \mathbf{v} \tag{1.32}$$

proiezione scalare sulla direttrice $\mathbf{v}$ del segmento $\mathbf{m} - \mathbf{p}$.

Questa versione risulta molto interessante nel caso si voglia misurare la distanza tra un punto $\mathbf{m}$ e un segmento $(\mathbf{p}, \mathbf{q})$ sfruttando la retta generata come in equazione (1.29). In questo caso un valore di t compreso tra $[0, 1]$ sta ad indicare che il punto più vicino a $\mathbf{m}$ cade all'interno del segmento, in quanto proiezione scalare del segmento $(\mathbf{p}, \mathbf{m})$ sul segmento $(\mathbf{p}, \mathbf{q})$.

Infine nella sezione 1.5.10, in equazione (1.54), verrà mostrato come trovare il punto su un iperpiano più vicino a un generico punto. Tale formulazione si può applicare anche alle rette scritte in forma di iperpiano e di conseguenza il punto (x, y) appartenente alla retta (a, b, c) più vicino al punto (x', y') è

$$(x, y) = \left(x' - a\frac{ax' + by' + c}{a^2 + b^2}, y' - b\frac{ax' + by' + c}{a^2 + b^2} \right) \tag{1.33}$$

1.5.4 Vicinanza punto-segmento

In diversi problemi è necessario conoscere la distanza tra un punto $\mathbf{p}$ e una polilinea formata da molteplici segmenti. Il peso computazionale di questo problema cresce linearmente con il numero di punti con cui è formata la retta: per poter eseguire queste analisi è necessario che il confronto con il singolo punto sia pertanto molto veloce.

In questa sezione verrà definito come segmento come quella parte di retta limitata tra i punti $\mathbf{a}$ e $\mathbf{b}$. Il punto $\mathbf{p}$ e il segmento possono relazionarsi in 3 modi: il punto più vicino è $\mathbf{a}$, il punto più vicino è $\mathbf{b}$ o il punto più vicino è un punto compreso tra i due estremi. Da un punto di vista prettamente computazionale calcolare queste 3 distanze richiederebbe 9 moltiplicazioni, 6 somme e una divisione, oltre ai necessari 3 confronti. Questa sezione mostra come si può migliorare computazionalmente il confronto facendo uso del prodotto scalare.

Senza perdita di generalità si può supporre che $\mathbf{a} = (0, 0)^\top$. Dalla definizione di prodotto scalare

$$\mathbf{p} \cdot \mathbf{b} = \cos\alpha \|\mathbf{p}\| \|\mathbf{b}\| \tag{1.34}$$

e della lunghezza della proiezione ortogonale di $\mathbf{p}$ su $\mathbf{b}$

$$\cos\alpha \|\mathbf{p}\| = \frac{\mathbf{p} \cdot \mathbf{b}}{\|\mathbf{b}\|} \tag{1.35}$$

è possibile calcolare la distanza punto-segmento in maniera più efficiente. Il punto più vicino a $\mathbf{p}$ è $\mathbf{a}$ se e solo se $\alpha > \pi/2$ ovvero $\mathbf{p} \cdot \mathbf{b} < 0$, mentre il punto più vicino è $\mathbf{b}$ se e solo se la proiezione di $\mathbf{p}$ su $\mathbf{b}$ è maggiore di $\|\mathbf{b}\|$ ovvero $\mathbf{p} \cdot \mathbf{b} > \|\mathbf{b}\|^2$. In questo caso per ottenere la sola informazione della vicinanza bastano 4 moltiplicazioni e 2 somme. Se e solo se il punto vicino risulta interno si potrà procedere con la tradizionale distanza punto-retta.

1.5.5 Rette in $\mathbb{R}^3$

Si può vedere una generica retta in uno spazio $\mathbb{R}^n$ come interpolazione di due punti dello stesso spazio:

$$\mathbf{x} = \lambda\mathbf{p} + (1 - \lambda)\mathbf{q} \tag{1.36}$$

Nel caso specifico di $\mathbb{R}^3$ queste equazioni richiedono 6 parametri da stimare (una "bounded 3D line" ha in effetti 6 gradi di libertà).

Una retta nello spazio $\mathbb{R}^n$ può essere vista come un punto più un versore:

$$\mathbf{x} = \mathbf{x}_0 + t\hat{\mathbf{v}} \tag{1.37}$$

Nel caso specifico di $\mathbb{R}^3$ queste equazione richiedono 5 parametri (in quanto un versore può essere descritto da solo 2 variabili). In questo caso il luogo dei punti si può ricavare moltiplicando per $\times\hat{\mathbf{v}}$:

$$\mathbf{x} \times \hat{\mathbf{v}} = \mathbf{x}_0 \times \hat{\mathbf{v}} = \mathbf{n} \tag{1.38}$$

Il vettore $\mathbf{x}_0 \times \hat{\mathbf{v}}$ descrive ovviamente un vettore ortogonale agli altri due ma la cui lunguezza è importante. Questa rappresentazione è identica a quella che si ottiene usando il sistema di coordinate Pluckeriane.

Nello spazio $\mathbb{R}^3$ la retta è il luogo di punti dell'intersezione di 2 piani (di cui uno potenzialmente passante per l'origine). Anche in questo caso parliamo di almeno 5 parametri da stimare.

Tuttavia in $\mathbb{R}^3$ le rette hanno solo 4 gradi di libertà: possiamo infatti vedere che ogni linea è tangente a una sfera di raggio r, intersecante nel punto $m = (r, \theta, \phi)$ in coordinate sferiche. L'ultimo parametro è un angolo di rotazione γ intorno al vettore m per indicare la direzione della linea (questa parte richiede un paio di condizioni ulteriori per evitare singolarità).

1.5.6 Incrocio di due rette

Siano ora ℓ_1 e ℓ_2 due rette di parametri $\mathbf{l}_1$ e $\mathbf{l}_2$ intersecanti nel punto $\mathbf{x}$ espresso in coordinate omogenee. Per ottenere il punto di incontro è necessario risolvere un sistema, omogeneo, nella forma

$$\begin{cases} \mathbf{l}_1^\top \mathbf{x} &= 0 \\ \mathbf{l}_2^\top \mathbf{x} &= 0 \end{cases} \tag{1.39}$$

Il sistema, del tipo $\mathbf{A}\mathbf{x} = 0$, può anche essere esteso al caso di n rette intersecanti, con $n > 2$, ottenendo un sistema sovradimensionato risolvibile con la tecnica della decomposizione SVD o QR. La soluzione del problema sovradimensionato, affetto da rumore, rappresenta il punto che minimizza il residuo algebrico di equazione (1.39).

Nel caso di due sole rette, il sistema (1.39) fornisce direttamente la soluzione. L'intersezione tra due rette $\mathbf{l}_1$ e $\mathbf{l}_2$, scritte in forma implicita (1.23), è il punto $\mathbf{x} = \mathbf{l}_1 \times \mathbf{l}_2$ espresso in coordinate omogenee, dove $\times$ è il prodotto vettoriale.

È da notare che, siccome le coordinate omogenee possono rappresentare punti all'infinito, questo particolare formalismo ammette anche il caso in cui le due rette siano parallele.

1.5.7 Principio di Dualità

Un concetto che tornerà utile di seguito è il principio di dualità punto-retta. Tale principio si basa sulla proprietà commutativa del prodotto scalare applicata all'equazione della retta scritta in forma implicita dove i luoghi dei punti della retta sono espressi sotto forma di coordinate omogenee:

$$\mathbf{l}^\top \mathbf{x} = \mathbf{x}^\top \mathbf{l} = 0 \tag{1.40}$$

È pertanto possibile ottenere formazioni duali quando ai parametri di una retta $\mathbf{l}$ si sostituisce quelli di un suo punto $\mathbf{x}$.

Da questa considerazione nasce il principio di dualità (*Duality Principle*) che garantisce che la soluzione del problema duale, dove il significato di retta e punto vengono scambiati, è anche soluzione del problema originale.

Per esempio, come si è visto nelle sezioni precedenti, dati due punti $\mathbf{p}$ e $\mathbf{q}$ è possibile definire una linea $\mathbf{l} = \mathbf{p} \times \mathbf{q}$ passante per essi mentre date due linee $\mathbf{l}$ e $\mathbf{m}$ è possibile definire un punto $\mathbf{x} = \mathbf{l} \times \mathbf{m}$ come loro intersezione.

1.5.8 Distanza tra rette in $\mathbb{R}^3$

Nello spazio $\mathbb{R}^3$, e in generale in tutti gli spazi di dimensione superiore, due rette $\mathbf{l}_1$ e $\mathbf{l}_2$ possono non incrociarsi in nessun punto anche se non sono parallele. Tali rette si definiscono sghembe (*skew lines*). Per queste particolari rette un parametro di interesse è la loro distanza minima e, conseguentemente, i punti sulle due rette che rappresentano tale minimo.

Siano due rette formate da punti $\mathbf{x}_1$ e $\mathbf{x}_2$ di equazione

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{p}_1 + t_1 \mathbf{v}_1 \\ \mathbf{x}_2 &= \mathbf{p}_2 + t_2 \mathbf{v}_2 \end{aligned} \tag{1.41}$$

dove $\mathbf{p}_1$ e $\mathbf{p}_2$ sono due generici punti appartenenti alle rette, $\mathbf{v}_1$ e $\mathbf{v}_2$ sono i vettori direzione, e $t_1, t_2 \in \mathbb{R}$ sono valori scalari, incognite del problema.

La "distanza" tra due generici punti sulle due rette è

$$\mathbf{d} = \mathbf{x}_2 - \mathbf{x}_1 = (\mathbf{p}_2 + t_2 \mathbf{v}_2) - (\mathbf{p}_1 + t_1 \mathbf{v}_1) = \mathbf{r} + t_2 \mathbf{v}_2 - t_1 \mathbf{v}_1 \tag{1.42}$$

avendo definito $\mathbf{r} = \mathbf{p}_2 - \mathbf{p}_1$. La quantità da minimizzare è $\|\mathbf{d}\|^2$, funzione di t_1 e t_2, il cui gradiente si annulla in

$$\begin{aligned} \mathbf{v}_1 \cdot \mathbf{v}_1 t_1 - \mathbf{v}_1 \cdot \mathbf{v}_2 t_2 &= \mathbf{v}_1 \cdot \mathbf{r} \\ \mathbf{v}_2 \cdot \mathbf{v}_1 t_1 - \mathbf{v}_2 \cdot \mathbf{v}_2 t_2 &= \mathbf{v}_2 \cdot \mathbf{r} \end{aligned} \tag{1.43}$$

Questo è un sistema lineare in t_1 e t_2 facilmente risolvibile e, con tale soluzione, si possono ricavare i due punti di minimo $\mathbf{p}_1$ e $\mathbf{p}_2$.

Esiste anche una formulazione alternativa al risolvere il sistema lineare giungendo allo stesso risultato attraverso considerazioni puramente geometriche. Si può dimostrare che la distanza tra le due rette in $\mathbb{R}^3$ vale

$$d = \frac{|\mathbf{r} \cdot \mathbf{n}|}{\|\mathbf{n}\|} \tag{1.44}$$

avendo definito $\mathbf{n} = \mathbf{v}_1 \times \mathbf{v}_2$. Il vettore $\mathbf{n}$, prodotto vettoriale, è per definizione ortogonale ad entrambe le rette e la distanza è la proiezione del segmento $\mathbf{r}$ lungo tale vettore. Si vede bene che quando le linee sono parallele ($\mathbf{n} = \mathbf{0}$) non è possibile stabilire un valore ragionevole per la triangolazione.

Il piano formato dalla traslazione della seconda retta lungo $\mathbf{n}$ interseca la prima retta nel punto di minima distanza

$$\begin{aligned} t_1 &= \frac{\mathbf{r} \cdot \mathbf{v}_2 \times \mathbf{n}}{\mathbf{v}_1 \cdot \mathbf{v}_2 \times \mathbf{n}} \\ t_2 &= \frac{\mathbf{r} \cdot \mathbf{v}_1 \times \mathbf{n}}{\mathbf{v}_2 \cdot \mathbf{v}_1 \times \mathbf{n}} \end{aligned} \tag{1.45}$$

Indipendentemente dal formalismo scelto, mettendo questi valori dentro le equazioni 1.41 si ricavano le coordinate tridimensionali dei punti tra loro più vicini delle rette.

1.5.9 Retta in coordinante polari

Le rette viste finora tendono ad avere una rappresentazione sovradimensionata rispetto ai gradi di libertà. La retta sul piano $\mathbb{R}^2$ infatti ha solo 2 gradi di libertà mentre la retta scritta in forma implicita dipende da ben 3 parametri, conosciuti a meno di un fattore moltiplicativo e senza un significato geometrico ben visibile. Dall'altra parte, l'equazione esplicita della retta a due parametri $y = mx + q$ presenta la singolarità delle rette verticali.

Una soluzione al problema è cambiare parametrizzazione e sfruttare le coordinate polari. Usando le coordinate polari risulta possibile esprimere una retta in uno spazio bidimensionale senza singolarità e usando solo 2 parametri:

$$x \cos\theta + y \sin\theta = \rho \tag{1.46}$$

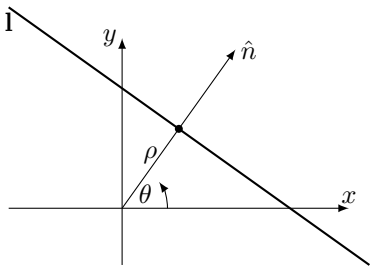


Figura 1.4: Retta espressa in coordinate polari.

dove ρ è la distanza tra la retta e il punto $(0,0)$ e θ è l'angolo che forma tale segmento distanza (ortogonale alla retta) e l'asse delle ascisse (figura 1.4). Si confronti tale rappresentazione con quella espressa in equazione (1.49). Sotto questa formulazione il legame tra questi due parametri e l'equazione della retta diventa non lineare.

Tale equazione è comunemente usata nella trasformata di Hough per le rette (sezione 3.11) per poter sfruttare uno spazio dei parametri bidimensionale e limitato.

Con questa particolare forma, la distanza tra un punto dello spazio (x_i, y_i) e la retta si scrive in maniera molto compatta come

$$d = |x_i \cos\theta + y_i \sin\theta - \rho| \tag{1.47}$$

1.5.10 Piani

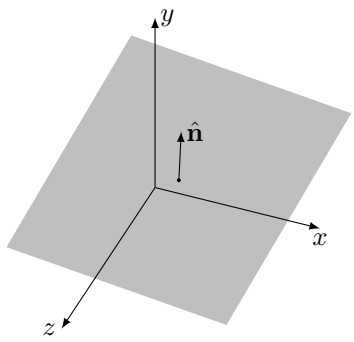


Figura 1.5: Esempio di piano in $\mathbb{R}^3$.

È possibile generalizzare il discorso delle rette a piani ed iperpiani nello spazio $\mathbb{R}^n$. Come per le rette infatti esiste una forma implicita e omogenea dell'equazione di un piano intesa come luogo dei punti espressi dalla coordinata $\tilde{\mathbf{x}} \in \mathbb{R}^{n+1}$ omogenea a $\mathbf{x} \in \mathbb{R}^n$:

$$\mathbf{m}^\top \tilde{\mathbf{x}} = 0 \tag{1.48}$$

Il prodotto scalare tra coordinate omogenee codifica sempre degli iperpiani.

Le coordinate omogenee sono conosciute a meno di un fattore moltiplicativo e pertanto si può forzare un vincolo opzionale: come per le rette si può pensare che i primi n parametri della coordinata omogenea formino un vettore di lunghezza unitaria.

Un generico piano, o iperpiano, è dunque il luogo dei punti $\mathbf{x} \in \mathbb{R}^n$ che soddisfano la condizione

$$\mathbf{x} \cdot \mathbf{n} - \rho = 0 \tag{1.49}$$

dove $\mathbf{n} \in \mathbb{R}^n$ è la normale al piano e $\rho = 0$ se e solo se il piano passa per l'origine. Nel caso di $\rho \neq 0$ una scrittura alternativa del piano è

$$\frac{1}{\rho}\mathbf{p} \cdot \mathbf{n} = 1 \tag{1.50}$$

e un'altra scrittura dell'equazione (1.49) che si può trovare in letteratura è

$$(\mathbf{x} - \mathbf{x}_0) \cdot \mathbf{n} = 0 \tag{1.51}$$

con $\mathbf{x}_0 \in \mathbb{R}^n$ un generico punto del piano da cui si può ricavare la corrispondenza $\rho = \mathbf{x}_0 \cdot \mathbf{n}$.

Bisogna ricordare che i gradi di libertà sono comunque sempre e solo n.

Quando introdotto, il vincolo di normalizzazione $|\hat{\mathbf{n}}| = 1$ rappresenta un caso particolare: sotto questa condizione, come nel caso delle rette, ρ assume il significato di minima distanza euclidea tra il piano e l'origine.

Se il piano (o l'iperpiano) è normalizzato, la distanza tra un generico punto $\mathbf{p}$ e il piano si misura come

$$d = |\mathbf{p} \cdot \hat{\mathbf{n}} - \rho| \tag{1.52}$$

altrimenti, come nel caso delle rette, è necessario dividere la distanza per $\|\mathbf{n}\|$.

Il punto $\mathbf{x}$ più vicino a un generico punto $\mathbf{p}$ appartenente all'iperpiano si trova nell'intersezione tra la retta di direzione $\mathbf{n}$ passante per $\mathbf{p}$ e il piano stesso:

$$\begin{cases} \mathbf{p} + t\mathbf{n} = \mathbf{x} \\ \mathbf{x} \cdot \mathbf{n} = \rho \end{cases} \tag{1.53}$$

ovvero in

$$\mathbf{x} = \mathbf{p} - \frac{\mathbf{p} \cdot \mathbf{n} - \rho}{|\mathbf{n}|^2}\mathbf{n} \tag{1.54}$$

Tale formulazione è applicabile anche alle rette come si è già visto.

Per quanto riguarda i vari metodi per la generazione, nella sezione 3.6.3 verrà mostrato come ottenere la regressione ai minimi quadrati di un insieme di punti all'equazione di piano.

Come nel caso della retta, anche i parametri del piano in $\mathbb{R}^3$ possono essere espressi attraverso l'uso di 3 coordinate polari (azimuth, zenith e ρ):

$$x \sin\vartheta \cos\varphi + y \sin\vartheta \sin\varphi + z \cos\vartheta = \rho \tag{1.55}$$

equazione del piano espressa in coordinate polari sferiche (1.18).

1.5.11 La divisione del piano

La retta (iperpiano) separa il piano (lo spazio) in due parti e all'interno di ognuna di queste parti la funzione $\mathbf{m}^\top\mathbf{x}$ assume il medesimo segno. Attraverso questa considerazione è possibile facilmente scoprire se un insieme di punti si trovano tutti dallo stesso lato rispetto a una retta/iperpiano o meno.

Per esempio, nel caso della retta, a seconda di come è orientato il vettore generatore, è possibile capire in quale dei due semipiani (sinistro, destro) cade un generico punto, attraverso lo studio di $s = ax_i + by_i + c$: quando $s < 0$ il punto si trova a sinistra della retta, $s > 0$ il punto si trova alla destra e infine quando $s = 0$ il punto è sulla retta.

Questa considerazione, ovvero che l'equazione di un piano permette in maniera molto efficiente di individuare in quale semipiano cade un generico punto, verrà utilizzata nel capitolo sui classificatori: la semplice equazione di una retta o di un piano può essere usata come classificatore se lo spazio delle categorie, generato da opportune n caratteristiche misurabili dell'immagine, è separabile da una superficie lineare.

1.6 Coniche

La conica è una curva algebrica luogo dei punti ottenibili come intersezione tra un cono a base circolare e un piano. L'equazione di una conica scritta in forma implicita è

$$ax^2 + bxy + cy^2 + dx + ey + f = 0 \tag{1.56}$$

È da notare che i parametri della conica sono conosciuti a meno di un fattore moltiplicativo.

L'equazione (1.56) mostra l'equazione della conica scritta in coordinate cartesiane tradizionali, inomogenee. L'uso di coordinate omogenee permette la scrittura di equazioni quadratiche in forma matriciale.

Se al posto delle coordinate cartesiane vengono usate le coordinate omogenee, applicando la sostituzione $x = x_1/x_3$ e $y = x_2/x_3$, si può ottenere l'equazione della conica espressa in forma omogenea:

$$ax_1^2 + bx_1x_2 + cx_2^2 + dx_1x_3 + ex_2x_3 + fx_3^2 = 0 \tag{1.57}$$

In questo modo è possibile rappresentare l'equazione (1.56) in forma matriciale

$$\mathbf{x}^\top\mathbf{C}\mathbf{x} = 0 \tag{1.58}$$

dove $\mathbf{C}$ è la matrice simmetrica 3×3 dei parametri e $\mathbf{x}$ è il luogo dei punti (espresso in coordinate omogenee) della conica. Essendo espressa da rapporti omogenei questa matrice è definita a meno di un fattore moltiplicativo. La conica è definita da 5 gradi di libertà ovvero dai 6 elementi della matrice simmetrica meno il fattore di scala.

Per il dualismo punto-retta, la linea $\mathbf{l}$ tangente a una conica $\mathbf{C}$ nel punto $\mathbf{x}$ è semplicemente $\mathbf{l} = \mathbf{C}\mathbf{x}$.

La scrittura della conica in equazione (1.58) ha la forma di una curva definita da un luogo di punti e perciò è anche chiamata *point conic* perché definisce l'equazione della conica usando punti dello spazio. Usando il teorema di dualità è anche possibile esprimere una conica $\mathbf{C}^* \propto \mathbf{C}^{-1}$, duale della $\mathbf{C}$, in funzione, questa volta, di rette: una linea tangente $\mathbf{l}$ alla conica $\mathbf{C}$ soddisfa $\mathbf{l}^\top\mathbf{C}^*\mathbf{l} = 0$.

Nella sezione 3.6.7 verranno presentate tecniche atte a stimare i parametri che codificano una conica dati i punti.

1.7 Prodotto Vettoriale

Nello spazio $\mathbb{R}^3$ è possibile trasformare l'operatore prodotto vettoriale in una applicazione lineare, ovvero dare una rappresentazione matriciale al prodotto vettoriale, tale che $[\mathbf{x}]_\times \mathbf{y} = \mathbf{x} \times \mathbf{y}$.

Nel testo verrà indicata con $[\mathbf{x}]_\times$ la matrice 3×3 associata al prodotto vettoriale. La forma di tale matrice, antisimmetrica, è

$$[\mathbf{x}]_\times = \begin{bmatrix} 0 & -x_2 & x_1 \\ x_2 & 0 & -x_0 \\ -x_1 & x_0 & 0 \end{bmatrix} \tag{1.59}$$

dove $\mathbf{x} = (x_0, x_1, x_2)^\top$. Questa matrice ha il determinante nullo e rango massimo 2.

1.8 Trasformazioni geometriche

Le trasformazioni geometriche dei punti del piano sono trasformazioni biunivoche che ad ogni punto del piano associano uno ed un solo punto del piano stesso.

Le trasformazioni geometriche si possono classificare in

Affinità Nel piano cartesiano la trasformazione affine è una applicazione biettiva che associa il punto $\mathbf{p}$ al punto $\mathbf{p}'$ attraverso una funzione del tipo

$$\mathbf{p}' = \mathbf{A}\mathbf{p} + \mathbf{t} \tag{1.60}$$

Una affinità gode delle seguenti proprietà:

- trasforma rette in rette;
- conserva la colinearità tra i punti;
- conserva il parallelismo e incidenza tra rette;
- in generale non conserva la forma ne gli angoli.

Essendo biettiva la trasformazione affine è invertibile, e l'inversa è anche essa una trasformazione affine di parametri

$$\mathbf{p} = \mathbf{A}^{-1}\mathbf{p}' - \mathbf{A}^{-1}\mathbf{t} = \mathbf{A}'\mathbf{p}' + \mathbf{t}' \tag{1.61}$$

Similitudine Una similitudine è una trasformazione affine che preserva il rapporto tra le dimensioni e gli angoli.

La forma dell'equazione è uguale a quella trasformazione affine (1.60) ma può rappresentare solo cambiamenti di scala, riflessioni, rotazioni e traslazioni. A seconda del segno del determinante di $\mathbf{A}$ le similitudini si dividono in *dirette* (determinante positivo) che preservano l'orientazione o *inverse* (determinante negativo) dove l'orientazione risulta ribaltata.

Isometria Le isometrie sono trasformazioni simili che conservano le distanze:

$$\|f(\mathbf{x}) - f(\mathbf{y})\| = \|x - y\| \tag{1.62}$$

per ogni $x, y \in \mathbb{R}^n$.

Le isometrie tra spazi euclidei si scrivono come in equazione (1.60) dove però $\mathbf{A}$, condizione necessaria e sufficiente perché sia una isometria, deve essere una matrice *ortogonale*.

Essendo ortogonale la matrice $\mathbf{A}$ deve avere determinante ±1. Come per le similitudini, se $\det \mathbf{A} = 1$ si dice che l'isometria è *diretta*, mentre se $\det \mathbf{A} = -1$ l'isometria è *inversa*.

Sono per esempio isometrie

- traslazioni;
- rotazioni;
- simmetrie centrali ed assiali.

1.9 Posa relativa tra sensori

Introduciamo per nomenclatura, le relazioni che intercorrono tra vari sistemi di riferimento e che verrà usata all'interno di questo libro. Ulteriore nomenclatura sui sistemi di coordinate si troverà poi nella futura sezione 8.2 a cui bisogna in parte far riferimento per alcuni termini.

Sia ${}^{w}\mathbf{x} \in \mathbb{R}^3$ un punto espresso in coordinate "globali", "mondo" (*world coordinates*) e sia ${}^{s}\mathbf{x}$ lo stesso punto espresso però in coordinate "locali", "sensore" (o *body* nel caso generico di un sistema in movimento rispetto a un altro). La coordinata ${}^{s}\mathbf{x}$ rappresenta una informazione spaziale rilevata dal mero sensore: tale coordinata pertanto non possiede la conoscenza di come il sensore sia posizionato e orientato nello spazio mondo. Anche se rappresentanti il medesimo punto fisico, le coordinate espresse dai due punti sono infatti differenti in quanto sono rappresentate in due sistemi di riferimento differenti: uno rappresenta una posizione intesa come assoluta (il mondo), mentre il secondo rappresenta il punto visto dal sensore, dove il sensore è al centro del sistema di riferimento allineato rispetto agli assi.

Definizione 4 *La relazione che lega le coordinate mondo a quelle sensore è*

$$ {}^{w}\mathbf{x} = {}^{w}\mathbf{R}_s \, {}^{s}\mathbf{x} + {}^{w}\mathbf{t} \tag{1.63} $$

con ${}^{w}\mathbf{R}_s$ *matrice di rotazione che permette di trasformare un punto da coordinate sensore a coordinate mondo e* $\mathbf{t}$ *posizione del sensore rispetto all'origine del sistema di riferimento.*

Siano ora, indicati con i numeri 1 e 2, due generici sensori legati al comune sistema di riferimento mondo w attraverso i parametri $({}^{w}\mathbf{R}_1, {}^{w}\mathbf{t}_1)$ e $({}^{w}\mathbf{R}_2, {}^{w}\mathbf{t}_2)$ rispettivamente, espressi come in definizione 4.

Sia $({}^{1}\mathbf{R}_2, {}^{1}\mathbf{t}_{2,1})$ la posa "relativa" del sensore 2 rispetto al sensore 1, posa che permette di convertire un punto dal sistema di riferimento sensore 2 al sistema di riferimento sensore 1:

$$ {}^{1}\mathbf{x} = {}^{1}\mathbf{R}_2 \, {}^{2}\mathbf{x} + {}^{1}\mathbf{t}_{2,1} \tag{1.64} $$

La matrice ${}^{1}\mathbf{R}_2$, che rappresenta l'orientazione del sensore 2 rispetto al sensore 1, trasforma le coordinate sensore mentre ${}^{1}\mathbf{t}_{2,1}$ è la posa del sensore 2 rispetto al sensore 1 espressa nel sistema di riferimento 1.

I parametri della posa relativa si ricavano dalle pose dei singoli sensori, pose espresse rispetto ad un terzo sistema di riferimento (il sistema mondo), attraverso le relazioni:

$$ \begin{aligned} {}^{1}\mathbf{R}_2 &= \mathbf{R}_1^{-1} \mathbf{R}_2 \\ {}^{1}\mathbf{t}_{2,1} &= \mathbf{R}_1^{-1} (\mathbf{t}_2 - \mathbf{t}_1) \end{aligned} \tag{1.65} $$

D'ora in poi, per alleggerire la notazione, sottintendiamo il sistema di riferimento mondo w e pertanto, quando non indicato, le coordinate sono riferite a questo sistema e il cambiamento di base porta anche esso verso quest'ultimo.

La posa relativa opposta $({}^{2}\mathbf{R}_1, {}^{2}\mathbf{t}_{1,2})$, che trasforma dal sistema 2 al sistema 1, si può ottenere da $({}^{1}\mathbf{R}_2, {}^{1}\mathbf{t}_{2,1})$ come

$$ \begin{aligned} {}^{2}\mathbf{R}_1 &= \mathbf{R}_2^{-1} \mathbf{R}_1 = {}^{1}\mathbf{R}_2^{-1} = {}^{1}\mathbf{R}_2^{\top} \\ {}^{2}\mathbf{t}_{1,2} &= -\mathbf{R}_2^{-1} (\mathbf{t}_2 - \mathbf{t}_1) = - {}^{1}\mathbf{R}_2^{\top} \, {}^{1}\mathbf{t}_{2,1} \end{aligned} \tag{1.66} $$

Data la conoscenza della posa relativa tra i sensori e della posa assoluta di uno dei due (in questo caso per semplicità il sensore 1) è possibile ricavare la posa assoluta del secondo sensore attraverso la trasformazione

$$ \begin{aligned} \mathbf{R}_2 &= \mathbf{R}_1 \, {}^{1}\mathbf{R}_2 \\ \mathbf{t}_2 &= \mathbf{R}_1 \, {}^{1}\mathbf{t}_{2,1} + \mathbf{t}_1 \end{aligned} \tag{1.67} $$

1.9.1 Stima delle rotazioni e traslazioni

Ci sono diverse tecniche per ottenere una stima ottima della trasformazione di rotazione e traslazione rigida tra punti appartenenti al medesimo spazio. Per stima ottima si intende la stima alla massima verosimiglianza, dove il rumore di osservazione ξ_i è totalmente additivo nello spazio $\mathbb{R}^n$ in cui vivono i punti.

Siano pertanto due insiemi di punti $\left\{{}^{1}\mathbf{x}_i\right\}$ e $\left\{{}^{2}\mathbf{x}_i\right\}$ tali che siano messi in relazione da

$$ {}^{2}\mathbf{x}_i = {}^{2}\mathbf{R}_1 \, {}^{1}\mathbf{x}_i + {}^{2}\mathbf{t} + \xi_i \tag{1.68} $$

come in equazione (1.64) ma con il vettore rumore ξ_i.

Per risolvere il problema ottimo $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$ che trasforma tutti i punti da $\mathbf{x}^1$ a $\mathbf{x}^2$ è richiesto un criterio ai minimi quadrati che ottimizzi una funzione costo del tipo

$$ S = \sum_i w_i \| \left({}^{2}\mathbf{R}_1 \, {}^{1}\mathbf{x}_i + {}^{2}\mathbf{t} \right) - {}^{2}\mathbf{x}_i \|^2 \tag{1.69} $$

dove w_i sono eventuali prior associati ad ogni campione. La soluzione ai minimi quadrati in forma non lineare chiaramente ha problemi di minimi locali.

Un risultato notevole che riguarda il vettore traslazione si ricava applicando le classiche derivate $0 = \partial S/\partial \mathbf{t}$ all'equazione (1.69) che si annulla in

$$\hat{\mathbf{t}} = {}^2\bar{\mathbf{x}} - \hat{\mathbf{R}}\,{}^1\bar{\mathbf{x}} \tag{1.70}$$

e pertanto data $\mathbf{R}$ è immediato trovare la traslazione in forma ottima o viceversa è sufficiente ricavare la sola rotazione $\hat{\mathbf{R}}$ per risolvere il problema delle pose. A questo risultato si arriva anche da un punto di vista intuitivo: il centroide delle due serie di punti a seguito di una trasformazione rigida deve seguire la relazione di equazione (1.68).

La rotazione $\mathbf{R}$ ottima deve minimizzare pertanto

$$S = \sum_i w_i \| {}^2\mathbf{R}_1 \, {}^1\mathbf{p}_i - {}^2\mathbf{p}_i \|^2 \tag{1.71}$$

avendo definito $\mathbf{p}_i = \mathbf{x}_i - \bar{\mathbf{x}}$. Pertanto qualunque problema di registrazione di pose in n dimensioni si riduce sempre ad un problema a n DOF.

Una prima tecnica per calcolare la rotazione fa uso delle componenti principali (si veda sezione 2.10.1). Le componenti principali estratte da ogni insieme di punti *separatamente* sono una base dello spazio. È possibile determinare una rotazione che le fa combaciare tali basi in quanto ricavate le matrici degli autovettori colonna $\mathbf{R}_1$ e $\mathbf{R}_2$ si ottiene direttamente $\mathbf{R} = \mathbf{R}_2 \left(\mathbf{R}_1\right)^\top$. Possono esistere tuttavia più soluzioni (e ognuna andrebbe verificata) e a causa del rumore ricavare gli assi attraverso PCA può diventare estremamente inaffidabile (ad esempio se la distribuzione risultasse circolare risulterebbe impossibile qualsiasi stima).

La cosa migliore per minimizzare (1.71) consiste nel minimizzare o meglio massimizzare:

$$\begin{aligned}
&\arg\min_{\mathbf{R}} \textstyle\sum_i w_i \| {}^2\mathbf{R}_1 \, {}^1\mathbf{p}_i - {}^2\mathbf{p}_i \|^2 \\
&= \arg\min_{\mathbf{R}} \left(-2 \textstyle\sum_i w_i \mathbf{p}_{i,2}^\top \mathbf{R} \mathbf{p}_{i,1}\right) \\
&= \arg\max_{\mathbf{R}} \left(\textstyle\sum_i w_i \mathbf{p}_{i,2}^\top \mathbf{R} \mathbf{p}_{i,1}\right) \\
&= \arg\max_{\mathbf{R}} \operatorname{trace}\left(\mathbf{W}\mathbf{P}_2^\top \mathbf{R} \mathbf{P}_1\right) \\
&= \arg\max_{\mathbf{R}} \operatorname{trace}\left(\mathbf{R}\mathbf{P}_1 \mathbf{W}\mathbf{P}_2^\top\right)
\end{aligned} \tag{1.72}$$

ovvero minimizzare il sistema di equazioni (1.69) è equivalente a massimizzare la traccia della matrice $\hat{\mathbf{R}}\mathbf{H}$ dove $\mathbf{H}$ è la matrice di correlazione tra le due nuvole di punti definita come

$$\mathbf{H} = \operatorname{cov}({}^1\mathbf{x}, {}^2\mathbf{x}) = \sum_i \left({}^1\mathbf{x}_i - {}^1\bar{\mathbf{x}}\right)\left({}^2\mathbf{x}_i - {}^2\bar{\mathbf{x}}\right)^\top \tag{1.73}$$

Si dimostra che la matrice $\hat{\mathbf{R}}$ che massimizza la traccia di $\hat{\mathbf{R}}\mathbf{H}$ è

$$\hat{\mathbf{R}} = \mathbf{V}\mathbf{U}^\top \tag{1.74}$$

avendo decomposto a valori singolari $\mathbf{H} = \mathbf{U}\Sigma\mathbf{V}^\top$. Questo algoritmo per la prima volta viene descritto in [Kab76], riscoperto in [AHB87], ma sicuramente migliorato in [Ume91] (in particolare il caso $\det \hat{\mathbf{R}} = -1$) e normalmente viene chiamato algoritmo di Kabsch-Umeyama.

Questa soluzione, molto più stabile di quella basata su PCA e sempre valida per $n > 3$, richiede una attenzione particolare nei soli casi bidimensionali e tridimensionali per gestire eventuali riflessioni (in tal caso infatti il determinante della matrice risultante potrebbe risultare negativo).

Lo "svantaggio" della tecnica basata su SVD rispetto a quella basata su PCA è la richiesta che le associazioni tra i punti delle due distribuzioni siano corrette.

L'unione delle due tecniche insieme ad un approccio iterativo prende il nome di *Iterative Closest Point* (ICP).

1.10 Trasformazioni omografiche

Le coordinate omogenee (sezione 1.4) permettono di rappresentare uno spettro molto ampio di trasformazioni unificando sotto lo stesso formalismo sia trasformazioni lineari (affini, rotazioni, traslazioni) che trasformazioni prospettiche.

Dati due piani distinti Π_i e Π_j si dice che essi sono riferiti a una trasformazione omografica (*homographic transformation*) quando esiste una corrispondenza biunivoca tale che:

- ad ogni punto o a ogni retta di Π_i corrisponde un solo punto e una sola retta di Π_i
- ad ogni fascio di rette di Π_j corrisponde un fascio proiettivo su Π_j

Sia il piano Π osservato da due viste differenti. Nello spazio $\mathbb{R}^2$ l'omografia (la trasformazione omografica) è rappresentata da equazioni del tipo:

$$\begin{aligned} u_j &= \frac{h_0 u_i + h_1 v_i + h_2}{h_6 u_i + h_7 v_i + h_8} \\ v_j &= \frac{h_3 u_i + h_4 v_i + h_5}{h_6 u_i + h_7 v_i + h_8} \end{aligned} \tag{1.75}$$

dove (u_i, v_i) sono coordinate dei punti appartenenti al piano Π_i, mentre (u_j, v_j) sono punti del piano Π_j.

Per la sua particolare forma tale trasformazione è descrivibile attraverso una trasformazione lineare usando le coordinate omogenee (sezione 1.4):

$$\begin{bmatrix} u_j \\ v_j \\ 1 \end{bmatrix} = \mathbf{H}_{ij}^{\Pi} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \tag{1.76}$$

avendo definito

$$\mathbf{H}_{ij}^{\Pi} = \begin{bmatrix} h_0 & h_1 & h_2 \\ h_3 & h_4 & h_5 \\ h_6 & h_7 & h_8 \end{bmatrix} \tag{1.77}$$

Nello spazio $\mathbb{R}^2$ le omografie sono codificate da matrici 3×3 (omografie 2D): allo stesso modo è possibile definire trasformazioni omografiche per spazio di dimensione maggiore. Per compattezza e per mantenere il riferimento a una rappresentazione in memoria *row-major*, come in `C`, la matrice $\mathbf{H}_{ij}^{\Pi}$ è stata espressa usando i coefficienti $h_0 \dots h_8$ piuttosto che la classica sintassi per indicare gli elementi della matrice.

Viene definita matrice omografica $\mathbf{H}_{ij}^{\Pi}$ la matrice che converte punti omogenei $\mathbf{x}_i$ appartenenti al piano Π_i della immagine i in punti $\mathbf{x}_j$ omogenei dell'immagine j con la relazione

$$\mathbf{x}_j = \mathbf{H}_{ij}^{\Pi} \mathbf{x}_i \tag{1.78}$$

Essendo una relazione tra grandezze omogenee il sistema è definito a meno di un fattore moltiplicativo: qualunque multiplo dei parametri della matrice omografica definisce la medesima trasformazione perché qualunque multiplo dei vettori di ingresso o uscita soddisfa ugualmente la relazione (1.75). Come conseguenza di ciò i gradi di libertà del problema non sono 9, come in una generica trasformazione affine in $\mathbb{R}^3$, ma 8 in quanto è sempre possibile fissare un vincolo aggiuntivo sugli elementi della matrice. Esempi di vincoli usati spesso sono $h_8 = 1$ o $\|\mathbf{H}\|_F = 1$. È da notare che $h_8 = 1$ non è generalmente un vincolo ottimo dal punto di vista computazionale in quanto l'ordine di grandezza che assume h_8 può essere molto diverso da quello degli altri elementi della matrice stessa e potrebbe generare singolarità, oltre al caso limite in cui h_8 potrebbe essere zero. Il vincolo alternativo $\|\mathbf{H}\|_F = 1$, soddisfatto gratuitamente dell'uso di risolutori basati su fattorizzazioni SVD o QR, è invece computazionalmente ottimo.

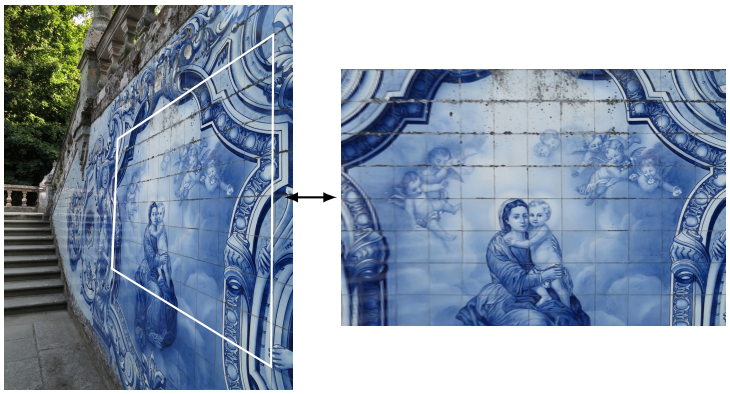


Figura 1.6: Esempio di trasformazione Omografica: l'omografia mette in relazione piani in prospettiva con piani non in prospettiva.

Le applicazioni che coinvolgono trasformazioni omografiche sono molteplici. Verranno in dettaglio affrontante nel capitolo 8 della camera pin-hole, ma in sintesi tali trasformazioni permettono la rimozione della prospettiva da piani in immagine, la proiezione di piani in prospettiva e associare i punti di piani osservati da punti di vista differenti. Un modo per ottenere delle trasformazioni prospettiche è mettere in relazione punti tra i piani che si vogliono trasformare e determinare in questo modo i parametri della matrice omografica (1.75) anche in maniera sovradimensionata, per esempio attraverso il metodo dei minimi quadrati. Un modo per ricavare i coefficienti sarà mostrato nell'equazione (8.52). Va ricordato che tale trasformazione, che lega punti di piani tra due viste prospettiche, vale solo e solamente per i punti dei piani considerati: l'omografia mette in relazione punti di piani tra loro, ma solo quelli. Qualsiasi punto non appartenente al piano verrà riproiettato in una posizione errata.

Figura 1.7: Esempio di trasformazione Omografica: l'omografia mette in relazione piani "virtuali" tra loro.

È facile vedere che ogni omografia è sempre invertibile e l'inversa della trasformazione è una trasformazione omografica anche essa:

$$\left(\mathbf{H}_{ij}^{\Pi}\right)^{-1} = \mathbf{H}_{ji}^{\Pi} \tag{1.79}$$

Una possibile forma per l'inversa dell'omografia (1.75) è

$$\begin{aligned} u_i &= \frac{(h_5h_7 - h_4h_8)u_j + (h_1h_8 - h_2h_7)v_j + h_4h_2 - h_1h_5}{(h_4h_6 - h_3h_7)u_j + (h_0h_7 - h_1h_6)v_j + h_1h_3 - h_4h_0} \\ v_i &= \frac{(h_3h_8 - h_5h_6)u_j + (h_2h_6 - h_0h_8)v_j + h_0h_5 - h_2h_3}{(h_4h_6 - h_3h_7)u_j + (h_0h_7 - h_1h_6)v_j + h_1h_3 - h_4h_0} \end{aligned} \tag{1.80}$$

e, essendo conosciuta a meno di un fattore moltiplicativo, non è stato fatto uso di divisioni nel ricavare i parametri della trasformazione inversa (*unnormalized inverse homographic matrix*).

È da segnalare che quando i due piani messi in relazione sono paralleli, allora $h_6 = 0 \wedge h_7 = 0$, la trasformazione omografica si riduce ad una trasformazione affine (*affine transformation*) rappresentata dalle classiche equazioni

$$\begin{aligned} u_j &= h_0u_i + h_1v_i + h_2 \\ v_j &= h_3u_i + h_4v_i + h_5 \end{aligned} \tag{1.81}$$

già incontrate in precedenza.

1.10.1 Omografia e Rette

Esistono applicazioni interessanti dell'omografia in diversi ambiti.

Una trasformazione omografica trasforma generalmente rette in rette. In casi particolari però può trasformare rette in punti, come ad esempio nella proiezione prospettica di elementi all'orizzonte: le coordinate omogenee infatti rappresentano diversamente punti e vettori, e quando una retta si riduce a un punto, la sua coordinata omogenea diventa 0.

La trasformazione omografica applicata a una retta (effetto del dualismo punto-retta) è esattamente la trasformazione inversa di quella che trasforma i punti corrispondenti tra gli spazi: la trasformazione $\mathbf{H}_{ij}$ che trasforma punti $\mathbf{x}_i$ dall'immagine i a punti $\mathbf{x}_j$ dell'immagine j trasforma equazioni delle rette dall'immagine j all'immagine i:

$$\begin{aligned} \mathbf{x}_j &= \mathbf{H}_{ij}\mathbf{x}_i \\ \mathbf{l}_i &= \mathbf{H}_{ij}^{\top}\mathbf{l}_j \end{aligned} \tag{1.82}$$

Esaminando punti e rette all'infinito (esempio all'orizzonte) si vede come un punto all'infinito abbia coordinate $(x, y, 0)^{\top}$. Esiste pertanto una linea speciale $\mathbf{l}_{\infty} = (0, 0, 1)^{\top}$ che congiunge tutti questi punti.

Il principio di dualità permette di spiegare come, data una trasformazione $\mathbf{M}$ (proiettiva o omografica), la trasformazione che trasforma un punto $\mathbf{x}$ in $\mathbf{x}'$ si scriva

$$\mathbf{x}' = \mathbf{M}\mathbf{x} \tag{1.83}$$

mentre la trasformazione che trasforma una retta $\mathbf{l}$ diventi invece

$$\mathbf{l}' = \mathbf{M}^{-\top}\mathbf{l} \tag{1.84}$$

1.10.2 Omografia e Coniche

Una conica si trasforma attraverso una trasformazione omografica $\mathbf{x}' = \mathbf{H}\mathbf{x}$ in una conica. Infatti consegue che

$$\mathbf{x}^\top\mathbf{C}\mathbf{x} = \mathbf{x}'^\top\mathbf{H}^{-\top}\mathbf{C}\mathbf{H}^{-1}\mathbf{x}' \tag{1.85}$$

che è ancora una forma quadratica $\mathbf{C}' \equiv \mathbf{H}^{-\top}\mathbf{C}\mathbf{H}^{-1}$. L'uso del simbolo di equivalenza $\equiv$ è necessario in quanto la conica è conosciuta a meno di un fattore moltiplicativo.

Questo risultato notevole permette di dimostrare che una conica vista in prospettiva è ancora una conica.

1.11 Punti dentro triangoli e quadrilateri

Si consideri il problema di verificare se un punto è all'interno di poligoni relativamente semplici come i triangoli o i quadrilateri. La trattazione del caso di poligono generico è più complessa e viene lasciata al lettore (e opzionalmente riconducibile al caso triangolo/quadrilatero).

Nel caso di triangoli e quadrilateri gli approcci che saranno mostrati risultano molto efficienti nel caso in cui sia necessario eseguire confronti tra un singolo poligono verso un numero elevato di punti in quanto tali tecniche permettono l'utilizzo di precalcoli: l'idea base infatti è quella di sfruttare una trasformazione dello spazio delle coordinate del poligono in uno spazio dove risulti più facile eseguire il confronto.

Per esempio, un parallelogramma formato da due vettori generatori può sempre essere trasformato in un quadrato unitario $(0,0)-(1,1)$ attraverso una trasformazione affine $\mathbf{p} \mapsto (X, Y)$. Un punto $\mathbf{p}$ cade all'interno del parallelogramma se $0 < X(\mathbf{p}) < 1$ e $0 < Y(\mathbf{p}) < 1$. La stessa trasformazione vale anche per i triangoli formati dagli stessi vettori generatori, con però il confronto $0 < X(\mathbf{p}) < 1$ e $0 < Y(\mathbf{p}) < 1 - X(\mathbf{p})$. Per valutare se un punto cade all'interno del parallelogramma sono necessarie appena 4 moltiplicazioni e 6 somme. Il costo di creazione della trasformazione affine è più elevato ma se il numero di confronti è alto tale peso, costante, risulta trascurabile.

Per trasformare quadrilateri generici in un quadrato unitario $(0,0)-(1,1)$ si deve usare invece la trasformazione omografica (si veda la sezione 1.10). A scapito di un costo iniziale elevato di creazione della trasformazione, il controllo se un punto appartiene o meno alla figura geometrica è relativamente semplice

$$\begin{aligned} 0 &< h_0 p_x + h_1 p_y + h_2 < h_6 p_x + h_7 p_y + h_8 \\ 0 &< h_3 p_x + h_4 p_y + h_5 < h_6 p_x + h_7 p_y + h_8 \end{aligned} \tag{1.86}$$

limitato pertanto a 6 moltiplicazioni, 6 somme e 4 confronti.

1.12 Trasformazioni tra immagini e Look Up Table

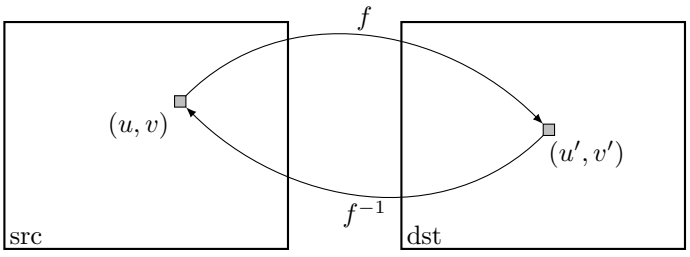


Figura 1.8: Trasformazione diretta e inversa tra immagini.

Essendo un argomento abbastanza delicato che potrebbe portare alcune ambiguità conviene dedicare una sezione a come in pratica vengono applicate le trasformazioni tra immagini.

Sia f una generica trasformazione biettiva

$$f : \mathbb{R}^2 \to \mathbb{R}^2 \tag{1.87}$$

tale che trasformi il punto $(u, v)^\top$ appartenente all'immagine sorgente nel punto $(u', v')^\top$ dell'immagine destinazione, ovvero

$$(u', v')^\top = f(u, v) \tag{1.88}$$

Questa trasformazione verrà chiamata *Forward Warping*.

Essendo le immagini non continue ma quantizzate in pixel, la trasformazione f non è usabile direttamente nelle applicazioni reali perché potrebbe sia lasciare dei buchi nella seconda immagine sia proiettare più volte lo stesso punto della prima. Per queste ragioni quando un immagine viene processata, si lavora sempre con la trasformazione inversa f^{-1} che, per ogni punto dell'immagine destinazione $(u', v')^\top$, restituisce il punto dell'immagine sorgente (u, v) da cui estrarre il colore, ovvero:

$$(u, v)^\top = f^{-1}(u', v') \tag{1.89}$$

Questa trasformazione verrà indicata con *Inverse Warping*.

È chiaro che anche l'immagine sorgente è composta da pixel, ma la conoscenza del punto $(u, v)^\top$ permette in maniera molto semplice l'utilizzo di tecniche, quali l'interpolazione lineare, per ricavare il valore del pixel.

Se la funzione f^{-1} è molto complicata e si vuole applicare la medesima trasformazione a più immagini, per risparmiare tempo computazionale si può creare una *Look Up Table* (LUT) di elementi $(u, v)^\top$ grande come l'immagine destinazione dove memorizzare per ogni elemento il risultato della trasformazione (1.89).

1.13 Accuratezza Sub-Pixel

Ogni quantità, informazione, caratteristica, che può essere estratta da un immagine è limitata dalla quantizzazione in pixel dell'immagine stessa. Tuttavia, esaminando un intorno del punto da estrarre, è possibile fornire una stima, approssimata, della posizione della caratteristica con precisione inferiore al pixel. Questi approcci sono tutti basati sul tentativo di modellare localmente la funzione immagine, quantizzata, cercando di ricostruire l'informazione distrutta dalla quantizzazione spaziale.

Ogni problema ha una sua tecnica specifica per estrarre tale informazione. In questa sezione ne verranno esaminate alcune.

1.13.1 Minimi e Massimi in 1D

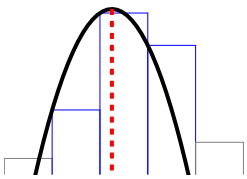


Figura 1.9: Costruzione del modello a parabola e individuazione del massimo con precisione sub-pixel.

Se il punto da esaminare è il massimo o il minimo di una sequenza monodimensionale, si può approssimare il primo vicinato del punto con una quadrica di equazione $ax^2 + bx + c = y$. La quadrica è il grado di funzione minimo che permetta l'individuazione di minimi o massimi locali.

Siano pertanto y_{-1}, y_0 e y_{+1} i valori della funzione con scostamento di -1, 0 e $+1$ rispetto al minimo/massimo individuato con precisione del pixel. L'equazione della quadrica passante per questi 3 punti assume la forma notevole

$$a = \frac{y_{+1} - 2y_0 + y_{-1}}{2} \quad b = \frac{y_{+1} - y_{-1}}{2} \quad c = y_0 \tag{1.90}$$

Tale curva ha il punto di massimo/minimo, notevole, in

$$\hat{\delta}_x = -\frac{b}{2a} = -\frac{y_{+1} - y_{-1}}{2(y_{+1} - 2y_0 + y_{-1})} \tag{1.91}$$

$\hat{\delta}_x$ è da intendersi come scostamento rispetto al punto di massimo/minimo precedentemente individuato ovvero rappresenta solamente la sua parte *sub-pixel*.

Questa equazione fornisce anche un ulteriore risultato notevole: se y_0 è un punto di massimo/minimo locale significa che tale valore sarà, per definizione, minore/maggiore sempre sia di y_{+1} che di y_{-1}. Grazie a questa considerazione, si dimostra facilmente che $\hat{\delta}_x$ è sempre compreso tra $-1/2$ e $1/2$.

Esiste una formulazione alternativa: chiamando $\delta_+ = y_{+1} - y_0$ e $\delta_- = y_{-1} - y_0$ l'equazione della parabola diventa

$$a = \frac{\delta_+ + \delta_-}{2} \quad b = \frac{\delta_+ - \delta_-}{2} \tag{1.92}$$

e il minimo si trova in

$$\hat{\delta}_x = -\frac{\delta_+ - \delta_-}{2(\delta_+ + \delta_-)} \tag{1.93}$$

dove si vede bene che la posizione del minimo è ovviamente indipendente da y_0 ma solamente funzione dei delta.

1.13.2 Minimi e Massimi in più dimensioni

In due dimensioni, ma lo stesso discorso vale per qualunque dimensione, bisogna estendere il problema di ricerca del massimo a funzioni via via sempre più complesse.

La soluzione più immediata è analizzare il punto lungo ogni direzione spaziale in maniera indipendente: in questo modo il problema si riconduce totalmente al caso monodimensionale.

Se si vuole sfruttare invece un intorno più ampio, il successivo modello più semplice da utilizzare è il paraboloide, quadrica scritta nella forma

$$m_0x^2 + m_1x + m_2y^2 + m_3y + m_4 = z \tag{1.94}$$

dove i punti (x, y) sono sempre da intendersi come scostamenti rispetto al punto da modellare e z è il valore che assume la funzione in quel determinato punto. Rispetto alla soluzione con gli assi totalmente separati, con questa equazione anche i punti non sugli assi contribuiscono attivamente alla soluzione del problema. Chiaramente se nel sistema si inseriscono solamente i 5 punti lungo gli assi, la soluzione sarà esattamente la stessa del caso visto nella sezione precedente.

Ogni elemento del problema fornisce pertanto un vincolo nella forma

$$\mathbf{m} \cdot \mathbf{x}_i = z_i \tag{1.95}$$

e tutti i vincoli insieme generano un sistema lineare potenzialmente sovradimensionato. In questo caso non esistono risultati notevoli con cui ottenere in forma chiusa la soluzione ma la cosa più semplice da fare è precalcolare una fattorizzazione del sistema formato dagli elementi $\mathbf{x}_i$, rappresentante un particolare intorno di $(0, 0)$, in modo da velocizzare la successiva risoluzione nel momento in cui i valori z_i saranno conosciuti.

L'equazione (1.94) assume gradiente nullo nel punto

$$\left(-\frac{m_1}{2m_0}, -\frac{m_3}{2m_2}\right) \tag{1.96}$$

esattamente come per il caso monodimensionale, in quanto le due componenti, quella lungo la x e quella lungo la y, rimangono comunque separate in fase di valutazione. Tale risultato è estendibile a casi n-dimensionali.

1.13.3 Minimi, Massimi e Punti di Sella

Per come è scritto, il modello presentato in precedenza vale sia per punti di minimo/massimo ma anche punti di sella. Tale modello non tiene tuttavia conto di eventuali rotazioni che localmente può la funzione. Se per punti di minimo e massimo tale rotazione è comunque in prima approssimazione ininfluente, nel caso dei punti di sella questa può assumere una certa rilevanza.

La versione dell'equazione (1.94) che tiene conto di eventuali rotazioni degli assi è

$$m_0x^2 + m_1x + m_2y^2 + m_3y + m_4xy + m_5 = z \tag{1.97}$$

Il sistema è totalmente compatibile con quello mostrato nella sezione precedente con l'unica differenza che ora le incognite sono 6 e perciò è necessario processare almeno 6 punti nell'intorno del minimo/massimo/punto di sella. Anche in questo caso non esistono soluzioni notevoli, ma conviene fattorizzare la matrice dei termini noti.

Il gradiente della funzione (1.97) si annulla nel punto corrispondente alla soluzione del sistema lineare

$$\begin{cases} 2m_0x + m_4y = -m_1 \\ m_4x + 2m_2y = -m_3 \end{cases} \tag{1.98}$$

risolvibile facilmente con la regola di Cramer.

I punti di sella possono essere utili per esempio per trovare con precisione *subpixel* marcatori a forma di scacchiera.

1.14 L'immagine Integrale

Sia I una generica immagine a toni di grigio. Il valore del pixel (x, y) dell'immagine integrale $\mathcal{I}$ rappresenta la somma dei valori di ogni pixel dell'immagine sorgente contenuti all'interno del rettangolo $(0, 0) - (x, y)$:

$$\mathcal{I}(x, y) = \sum_{v=0}^{y} \sum_{u=0}^{x} I(u, v) \tag{1.99}$$

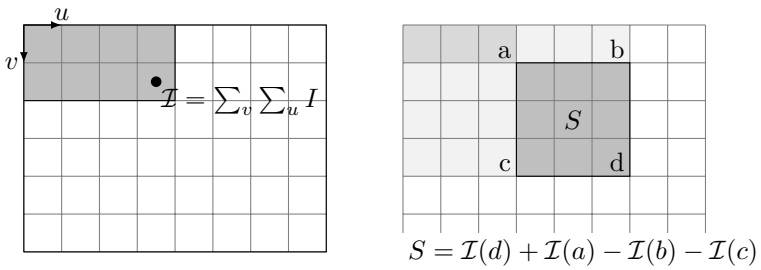


Figura 1.10: Costruzione dell'immagine integrale e utilizzo per calcolare aree.

Con questa definizione è da notare che gli estremi del rettangolo sono compresi nella sommatoria (figura 1.10).

L'artificio computazione di usare l'immagine integrale permette di ottimizzare diversi algoritmi mostrati in questo libro in particolare SURF (sezione 5.4) e l'estrazione delle *feature* di Haar (sezione 6.1).

Grazie all'immagine integrale è possibile, ad un costo computazionale costante di 4 somme, ottenere la sommatoria di una qualunque sottoparte rettangolare dell'immagine I:

$$\begin{aligned} &\textstyle\sum_{y=y_0}^{y_1} \sum_{x=x_0}^{x_1} I(x,y) = \\ &\mathcal{I}(x_1,y_1) + \mathcal{I}(x_0-1,y_0-1) - \mathcal{I}(x_1,y_0-1) - \mathcal{I}(x_0-1,y_1) \end{aligned} \tag{1.100}$$

Il valore così ottenuto rappresenta la somma degli elementi dell'immagine originale all'interno del rettangolo (estremi inclusi).

Oltre a poter calcolare velocemente la sommatoria di una qualsiasi sottoparte dell'immagine, è possibile ottenere facilmente convoluzioni con kernel di forma particolare in maniera molto agevole e sempre con prestazioni invarianti rispetto alla dimensione del filtro. Esempi di maschere di convoluzione si possono vedere in sezione 6.1.

Capitolo 2

Elementi di Statistica

La visione artificiale si pone come obiettivo quello di permettere la percezione del mondo attraverso gli *occhi* di un calcolatore. Tuttavia, ogni qualvolta si voglia esaminare una quantità reale osservabile e ricondurla ad un modello matematico, bisogna confrontarsi con la statistica. Per questa ragione, in questo secondo capitolo verranno mostrate alcune tecniche di statistica, fondamentali per chi sviluppa algoritmi di visione delle macchine.

2.1 Media e Varianza

È facile supporre che la nozione della media tra numeri sia un concetto conosciuto a tutti, almeno da un punto di vista puramente intuitivo. In questa sezione ne viene comunque fatto un breve riassunto, ne vengono date le definizioni e verranno sottolineati alcuni aspetti interessanti.

Per n campioni di una quantità osservata x la media campionaria *sample mean* si indica $\bar{x}$ e vale

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.1}$$

La media campionaria, per definizione, è una quantità *empirica*.

Se si potessero campionare infiniti valori di x, $\bar{x}$ convergerebbe al valore teorico, atteso (*expected value*). Questa è la legge dei grandi numeri (*Law of Large Numbers*).

Il valor medio atteso (*expectation*, *mean*) di una variabile casuale X si indica con $\mathrm{E}[X]$ o μ e si può calcolare da variabili aleatorie discrete attraverso la formula

$$\mathrm{E}[X] = \mu_x = \sum_{-\infty}^{+\infty} x_i p_X(x_i) \tag{2.2}$$

e per le variabili continue attraverso

$$\mathrm{E}[X] = \mu_x = \int_{-\infty}^{+\infty} x p_X(x) dx \tag{2.3}$$

data la conoscenza della distribuzione di probabilità $p_X(x)$.

Introduciamo ora il concetto di media di una funzione di variabile aleatoria.

Definizione 5 *Sia X una variabile aleatoria con funzione di probabilità $p_X(x)$ e $g(x)$ una generica funzione misurabile in x. Se assolutamente convergente l'integrale*

$$E[g(X)] = \sum_{-\infty}^{+\infty} g(x_i) p_i \qquad E[g(X)] = \int_{-\infty}^{+\infty} g(x) p_X(x) dx \tag{2.4}$$

prende il nome di "valor medio della variabile aleatoria $Y = g(X)$".

Esistono alcune funzioni la cui media assume un significato notevole. Quando $g(x) = x$ si parla di statistiche di primo ordine (*first statistical moment*), e in generale quando $g(x) = x^k$ si parla di statistiche di k-ordine. Il valor medio è pertanto la statistica di primo ordine e un'altra statistica di particolare interesse è il momento di secondo ordine:

$$\mathrm{E}[X^2] = \int_{-\infty}^{+\infty} x^2 p_X(x) dx \tag{2.5}$$

Tale statistica è importante perché permette di stimare la varianza di X.

La varianza è definita come il valore atteso del quadrato della variabile aleatoria X a cui viene tolto il suo valor medio, ovvero momento di secondo ordine della funzione $g(X) = X - \mathrm{E}[X]$:

$$\mathrm{var}(X) = \sigma_X^2 = \mathrm{E}[(X - \mathrm{E}[X])^2] \tag{2.6}$$

e, supponendo X e $\mathrm{E}[X]$ processi indipendenti, si ottiene la forma più semplice e ampiamente usata della varianza

$$\mathrm{var}(X) = \sigma_X^2 = \mathrm{E}[X^2] - \mathrm{E}[X]^2 \tag{2.7}$$

La radice quadrata della varianza è conosciuta come deviazione standard (*standard deviation*) e ha il vantaggio di avere la stessa unità di misura della grandezza osservata:

$$\sigma_X = \sqrt{\mathrm{var}(X)} \tag{2.8}$$

Estendiamo i concetti visti finora al caso multivariabile. Il caso multivariabile può essere visto come estensione a più dimensioni dove ad ogni dimensione è associata una diversa variabile.

La matrice delle covarianze Σ è l'estensione a più dimensioni (o a più variabili) del concetto di varianza. È costruita come

$$\Sigma_{ij} = \mathrm{cov}(X_i, X_j) \tag{2.9}$$

dove ogni elemento della matrice contiene la covarianza tra le varie componenti del vettore aleatorio X. La covarianza indica come le differenti variabili aleatorie che compongono il vettore X sono tra loro legate.

I possibili modi di indicare la matrice di covarianza sono

$$\Sigma = \mathrm{E}\left[(X - \mathrm{E}[X])(X - \mathrm{E}[X])^\top\right] = \mathrm{var}(X) = \mathrm{cov}(X) = \mathrm{cov}(X, X) \tag{2.10}$$

La notazione della cross-covarianza è invece univoca

$$\mathrm{cov}(X, Y) = \mathrm{E}\left[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])^\top\right] \tag{2.11}$$

generalizzazione del concetto di matrice delle covarianze. La matrice di cross-covarianza $\mathbf{\Sigma}$ ha come elementi nella posizione (i, j) la covarianza tra la variabile aleatoria X_i e la variabile Y_j:

$$\mathbf{\Sigma} = \begin{bmatrix} \mathrm{cov}(X_1, Y_1) & \cdots & \mathrm{cov}(X_n, Y_1) \\ \vdots & & \vdots \\ \mathrm{cov}(X_1, Y_m) & \cdots & \mathrm{cov}(X_n, Y_m) \end{bmatrix} \tag{2.12}$$

La matrice di covarianza $\mathrm{cov}(X, X)$ è conseguentemente simmetrica.

La matrice di covarianza, descrivendo come le variabili sono tra di loro in relazione e di conseguenza quanto sono tra loro slegate, è anche chiamata matrice di dispersione (*scatter plot matrix*). L'inversa della matrice di covarianza si chiama matrice di concentrazione o matrice di precisione.

La matrice di correlazione $\mathbf{r}(X, Y)$ è la matrice di cross-covarianza normalizzata rispetto alle matrici di covarianza:

$$\mathbf{r}(X, Y) = \frac{\mathrm{cov}(X, Y)}{\sqrt{\mathrm{var}(X)\mathrm{var}(Y)}} \tag{2.13}$$

Questa matrice ha valori sempre nell'intervallo $[-1, 1]$ o $[-100\%, 100\%]$.

2.2 La distribuzione Gaussiana

La distribuzione Gaussiana è una delle distribuzioni di probabilità più diffuse nei problemi pratici in quanto modella buona parte della distribuzione di probabilità in eventi reali. In questo documento in particolare è usata nei filtri (sezione 2.12) e nei classificatori Bayesiani (sezione 4.2), in LDA (sezione 4.3).

Definizione 6 *La distribuzione gaussiana standard che si indica con il simbolo $\mathcal{N}(0; 1)$, è quella di densità*

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{\left(-\frac{1}{2}x^2\right)} \tag{2.14}$$

Definizione 7 *La distribuzione gaussiana generale $\mathcal{N}(\mu; \sigma^2)$, con $\mu, \sigma \in \mathbb{R}, \sigma^2 \geqslant 0$, è quella che si ottiene dalla distribuzione standard con la trasformazione $x \mapsto \sigma x + \mu$.*

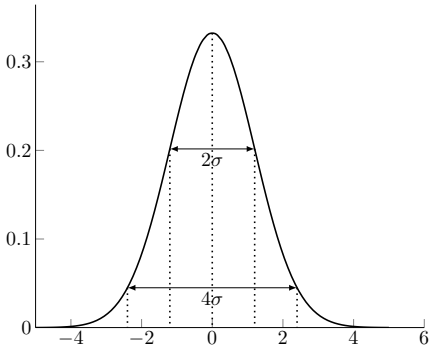


Figura 2.1: Distribuzione gaussiana

Nel caso univariabile (gaussiana univariata) la gaussiana ha la seguente funzione di distribuzione:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{2.15}$$

dove μ è il valor medio e σ^2 è la varianza. All'interno di $\pm\sigma$ da μ si concentra il 68% della probabilità, in $\pm 2\sigma$ il 95% e in $\pm 3\sigma$ il 99.7%.

La distribuzione gaussiana multivariabile (gaussiana multidimensionale) è data da un vettore $\boldsymbol{\mu}$ di dimensione n, rappresentante il valor medio delle varie componenti, e da una matrice di covarianza $\boldsymbol{\Sigma}$ di dimensioni $n \times n$:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{|\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} \tag{2.16}$$

distribuzione normale di valor medio $\boldsymbol{\mu} = [\mu_1, \mu_2, \dots \mu_n]^T$ e covarianza $\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11} & \cdots & \sigma_{1n} \\ \vdots & \ddots & \vdots \\ \sigma_{n1} & \cdots & \sigma_{nn} \end{bmatrix}$.

Si può anticipare che la quantità a esponente dell'equazione (2.16) è la distanza di Mahalanobis (sezione 2.4) tra $\mathbf{x}$ e $\boldsymbol{\mu}$.

Quando le variabili aleatorie sono tra loro indipendenti e di varianza uguale, la matrice $\boldsymbol{\Sigma}$ è una matrice diagonale con valori tutti uguali a σ^2 e la distribuzione di probabilità normale multivariata si riduce a

$$p(\mathbf{x}) = \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{|\mathbf{x}-\boldsymbol{\mu}|^2}{2\sigma^2}} \tag{2.17}$$

2.2.1 Gaussiana campionata

In applicazioni pratiche di elaborazione di segnali discreti, dove la gaussiana viene usata come filtro convolutivo, anche essa deve essere rappresentata a passi discreti g_k. La gaussiana viene normalmente campionata a passo uniforme ma, siccome ha supporto infinito, vengono presi tanti campioni per solo 3 o 4 volte la deviazione standard della gaussiana:

$$g_k = \begin{cases} ce^{-\frac{k^2}{2\sigma^2}} & |k| < 3\sigma \\ 0 & \text{otherwise} \end{cases} \tag{2.18}$$

con c fattore di normalizzazione scelto in modo tale che $\sum_k g_k = 1$.

È possibile estendere la gaussiana al caso multidimensionale in modo molto semplice come:

$$g_{k_1,k_2,\dots,k_n} = g_{k_1} \cdot g_{k_2} \dots g_{k_n} \tag{2.19}$$

2.3 Modelli a Miscela

I *modelli a miscela* sono un tipo di modello di densità costituito da certo numero di funzioni di densità, solitamente gaussiane (*Gaussian Mixture Models*) e queste funzioni sono unite per fornire una densità multimodale. I modelli a miscela permettono di rappresentare distribuzioni di probabilità in presenza di sottopopolazioni. Possono, per esempio, essere impiegate per modellare i colori di un oggetto e sfruttare tale informazione per eseguire il *tracking* o la segmentazione basata sul colore.

Il *mixture model* è un formalismo matematico sufficiente per modellare una distribuzione di probabilità come somma di distribuzioni parametriche. In termini matematici

$$p_X(x) = \sum_{k=1}^{n} a_k h(x|\lambda_k) \tag{2.20}$$

dove $p_X(x)$ è la funzione distribuzione modellata, n è il numero di componenti nel modello, e a_k è il fattore di proporzione del componente k. Per definizione $0 < a_k < 1 \ \forall k = 1, \ldots, n$ e $a_1 + \cdots + a_n = 1$. $h(x|\lambda_k)$ è una distribuzione di probabilità parametrizzata da un vettore (in generale) λ_k. Nel caso di modelli a miscela di gaussiane, il vettore dei parametri è formato da media e varianza delle singole componenti.

I *mixture models* sono spesso utilizzanti quando si conosce $h(x)$, si può campionare $p_X(x)$ e si vuole solo determinare i parametri a_k e λ_k. Un esempio di situazione pratica dove tale formalismo è impiegato, è quando si vuole analizzare una popolazione formata da distinte sottopopolazioni.

2.4 La distanza di Mahalanobis

Un problema molto diffuso è quello di capire quanto un elemento $\mathbf{x}$ possa appartenere o meno a una distribuzione di probabilità, permettendo di fornire una stima approssimativa se tale elemento sia un *inlier*, ovvero appartenga alla distribuzione, o un *outlier*, ovvero esserne esterno.

La distanza di Mahalanobis [Mah36] permette di ottenere la misura di una osservazione normalizzata rispetto alla varianza della stessa e per questo motivo viene anche indicata come "distanza generalizzata".

Definizione 8 *La distanza di Mahalanobis di un vettore* $\mathbf{x}$ *rispetto a una distribuzione di valor medio* μ *e matrice di covarianza* $\mathbf{\Sigma}$ *è definita come*

$$d(\mathbf{x}) = \sqrt{(\mathbf{x} - \mu)^\top \mathbf{\Sigma}^{-1} (\mathbf{x} - \mu)} \tag{2.21}$$

distanza generalizzata del punto rispetto alla media.

Tale distanza può venir estesa (*generalized squared interpoint distance*) al caso di due vettori $\mathbf{x}$ e $\mathbf{y}$ realizzazioni della medesima variabile aleatoria con distribuzione di covarianza $\mathbf{\Sigma}$:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{y})} \tag{2.22}$$

Nel caso particolare di matrice covarianza diagonale, si riottiene la distanza euclidea normalizzata, mentre quando la matrice di covarianza è esattamente la matrice identità (ovvero le componenti della distribuzione fossero di fatto incorrelate tra loro) la formulazione sopra si ricondurrebbe alla classica distanza euclidea.

La distanza di Mahalanobis permette di misurare distanze su campioni di cui non si conoscono le unità di misura, assegnando di fatto un fattore di scala automatico ai dati.

2.4.1 Standard Score

Una formulazione alternativa alla distanza di Mahalanobis è la *Standard Score*. Una variabile casuale X viene standardizzata, usando le sue statistiche empiriche, applicando la trasformazione

$$Z = \frac{X - \mu}{\sigma} \tag{2.23}$$

con μ media e σ deviazione standard di X. La nuova variabile casuale Z ha, per definizione, media nulla e varianza unitaria.

È possibile usare questa *Z-score* per scremare potenziali *outlier* della distribuzione.

2.5 Trasformazioni di Variabili Aleatorie

Uno dei problemi fondamentali in statistica è capire come una variabile aleatoria si propaghi all'interno di un sistema complesso e in che misura renda aleatoria l'uscita di tale sistema.

Sia $f(\cdot)$ una funzione che trasforma la variabile aleatoria X nella variabile aleatoria Y, ovvero $y = f(x)$, con x realizzazioni della variabile aleatoria X, e supponiamo che f sia invertibile, ovvero che esiste una funzione $x = g(y)$ tale che $g(f(x)) = x$.

Sia $\mathcal{I}_x$ un generico intervallo del dominio di esistenza dei valori x e $\mathcal{I}_y = \{y : y = f(x), x \in \mathcal{I}_x\}$ la sua corrispondente immagine. È ovvio che le probabilità degli eventi di x in $\mathcal{I}_x$ e y in $\mathcal{I}_y$ devono essere uguali ovvero

$$\int_{\mathcal{I}_y} p_Y(y) dy = \int_{\mathcal{I}_x} p_X(x) dx \tag{2.24}$$

Senza perdita di generalità è possibile porre a infinitesimo l'intervallo $\mathcal{I}_x$. Sotto questa condizione la relazione (2.24) si riduce a

$$p_Y(y)|dy| = p_X(x)|dx| = p_X(g(y))|dx| \tag{2.25}$$

da cui

$$p_Y(y) = p_X(g(y))\frac{|dx|}{|dy|} = p_X(g(y))|g'(y)| = \left.\frac{p_X(x)}{|f'(x)|}\right|_{x=g(y)} \tag{2.26}$$

Questa relazione si può facilmente estendere al caso di funzione non iniettiva, sommando i diversi intervalli, e al caso multidimensionale, usando lo Jacobiano al posto della derivata.

2.6 Propagazione dell'incertezza

Per capire come si propaga esattamente l'incertezza in un sistema è pertanto necessario un processo, più o meno complesso, sia di inversione che di derivazione del sistema stesso.

In molte applicazioni risulta pertanto difficoltoso, se non impossibile, ottenere in forma analitica la distribuzione di probabilità all'uscita di una trasformazione di una generica distribuzione in ingresso. Per fortuna, in applicazioni pratiche, spesso è richiesta una precisione inferiore nell'affrontare un problema di propagazione dell'incertezza, limitandosi normalmente alle sole statistiche di primo e secondo ordine e limitandosi ai casi di distribuzione di probabilità di tipo gaussiano.

Somma o differenza di grandezze La variabile aleatoria $Z = X \pm Y$, somma/differenza di variabili aleatorie indipendenti, ha varianza (covarianza) pari a

$$\mathrm{var}(Z) = \mathrm{var}(X) + \mathrm{var}(Y) \tag{2.27}$$

La varianza della variabile risultante pertanto è la somma delle singole varianze.

Trasformazioni Lineari Sia $\mathbf{y} = \mathbf{A}\mathbf{x}$ un sistema lineare dove al vettore aleatorio $\mathbf{x}$ è associata la matrice di covarianza $\mathrm{var}(X)$. La matrice di covarianza della variabile aleatoria $\mathbf{y}$ risultante, uscita del sistema, è

$$\mathrm{var}(Y) = \mathrm{var}(\mathbf{A}X) = \mathbf{A}\mathrm{var}(X)\mathbf{A}^\top \tag{2.28}$$

Tale relazione vale anche nel caso di proiezioni $y = \mathbf{b} \cdot \mathbf{x}$ e, in modo simile al sistema lineare, la varianza della variabile Y diventa

$$\mathrm{var}(Y) = \mathrm{var}(\mathbf{b}^\top X) = \mathbf{b}^\top \mathrm{var}(X)\mathbf{b} \tag{2.29}$$

Generalizzando i casi precedenti, la cross-covarianza tra $\mathbf{A}\mathbf{x}$ e $\mathbf{B}\mathbf{y}$ si può scrivere come:

$$\mathrm{cov}(\mathbf{A}X, \mathbf{B}Y) = \mathbf{A}\mathrm{cov}(X, Y)\mathbf{B}^\top \tag{2.30}$$

e, come caso particolare, la cross-covarianza tra $\mathbf{x}$ e $\mathbf{A}\mathbf{x}$

$$\mathrm{cov}(X, \mathbf{A}X) = \mathrm{var}(X)\mathbf{A}^\top \tag{2.31}$$

È da notare che $\mathrm{cov}(Y, X) = \mathrm{cov}(X, Y)^\top = \mathbf{A}\mathrm{var}(X)$.

Gli esempi di propagazione dell'incertezza visti finora si possono ulteriormente generalizzare, anticipando risultati importanti per il caso non-lineare, presentato la trasformazione affine $f(\mathbf{x})$ definita come

$$f(\mathbf{x}) = f_{\bar{\mathbf{x}}} + \mathbf{A}(\mathbf{x} - \bar{\mathbf{x}}) \tag{2.32}$$

ovvero una trasformazione di variabili casuali $Y = f(X)$ che di fatto di valor medio $\bar{y} = f_{\bar{\mathbf{x}}}$ e matrice di covarianza $\mathbf{\Sigma}_Y = \mathbf{A}\mathbf{\Sigma}_X\mathbf{A}^\top$.

Trasformazioni non lineari La propagazione della covarianza nel caso non-lineare non è infatti facilmente ottenibile in forma chiusa e in genere si ottiene solo in forma approssimata. Tecniche come la simulazione Monte Carlo possono essere usate per simulare in maniera molto accurata a diversi ordini di precisione la distribuzione di probabilità a seguito una generica trasformazione. L'approssimazione lineare è comunque ampiamente usata nei problemi pratici ma, come si vedrà nella sezione successiva, tecniche moderne permettono la stima della covarianza a ordini di precisione elevati in maniera abbastanza semplice.

Normalmente, per statistiche di primo ordine (*first-order error propagation*), la trasformazione f non lineare viene approssimata, attraverso l'espansione in serie, da una trasformazione affine

$$f(\mathbf{x}) \approx f(\bar{\mathbf{x}}) + \mathbf{J}_f(\mathbf{x} - \bar{\mathbf{x}}) \tag{2.33}$$

con $\mathbf{J}_f$ matrice delle derivate parziali (Jacobiano) della funzione f. Con questa approssimazione, il risultato del caso lineare affine mostrato in precedenza in equazione (2.32) può essere usato per determinare la matrice di covarianza della variabile $f(\mathbf{x})$, sostituendo alla matrice $\mathbf{A}$ lo Jacobiano, ottenendo la covarianza

$$\mathbf{\Sigma}_Y = \mathbf{J}_f\mathbf{\Sigma}_X\mathbf{J}_f^\top \tag{2.34}$$

e usando come valor medio atteso $\bar{y} = f(\bar{\mathbf{x}})$.

2.6.1 Esempi di propagazione degli errori

Risulta importante nel campo della visione artificiale la teoria della propagazione degli errori, in quanto sono comuni operazioni base di estrazione di caratteristiche affette da rumore, come per esempio la misura dell'intensità di colore o la misura della posizione di una particolare *feature* sull'immagine, ed è importante capire quanto questo rumore influisce nei calcoli successivi.

L'errore di misura dovuto a rumore additivo si formalizza come $x = \hat{x} + \varepsilon$, dove x è il valore osservato, $\hat{x}$ il valore reale e ε è il rumore additivo, per esempio gaussiano bianco di varianza σ_x^2.

Nel caso della visione potrebbe essere interessante stimare come si propaga nel sistema l'errore generato dall'osservazione imprecisa di un punto sull'immagine. In questo caso le variabili osservate saranno (x, y) coordinate immagine affette entrambe da errore di localizzazione di varianza σ_x^2 e σ_y^2 rispettivamente, variabili normalmente (almeno in prima approssimazione) non correlate tra di loro.

Usando il risultato di equazione (2.33), la generica funzione $z(x, y)$, funzione in due variabili aleatorie, si può approssimare al primo ordine attraverso l'espansione in serie di Taylor come

$$z(x', y') \approx z(x, y) + \left.\frac{\partial z}{\partial x}\right|_{x,y} (x' - x) + \left.\frac{\partial z}{\partial y}\right|_{x,y} (y' - y) \tag{2.35}$$

da cui si può stimare l'incertezza sul valore di z, applicando la propagazione della varianza vista nella sezione precedente, ottenendo

$$\sigma_z^2 = \left(\frac{\partial z}{\partial x}\right)^2 \sigma_x^2 + \left(\frac{\partial z}{\partial y}\right)^2 \sigma_y^2 \tag{2.36}$$

avendo scelto di calcolare la derivata in (x, y).

Con questa formulazione, si possono presentare alcuni esempi:

Esempio 1 La propagazione dell'errore di $z = x + y$ è

$$\sigma_z^2 = \sigma_x^2 + \sigma_y^2 \tag{2.37}$$

risultato già visto in precedenza.

Esempio 2 La propagazione dell'errore di $z = xy$ è

$$\sigma_z^2 = y^2\sigma_x^2 + x^2\sigma_y^2 \tag{2.38}$$

Esempio 3 La propagazione dell'errore di $z = \frac{1}{x \pm y}$ è

$$\sigma_z^2 = \frac{\sigma_x^2 + \sigma_y^2}{(x \pm y)^4} \tag{2.39}$$

Esempio 4 La propagazione dell'errore di $z = \frac{x}{y}$ è

$$\sigma_z^2 = \frac{1}{y^2}\sigma_x^2 + \frac{x^2}{y^4}\sigma_y^2 \tag{2.40}$$

Esempio 5 La propagazione dell'errore di $z = \sqrt{x^2 + y^2}$ è

$$\sigma_z^2 = \frac{x^2\sigma_x^2 + y^2\sigma_y^2}{x^2 + y^2} \tag{2.41}$$

È interessante notare da queste equazioni come il valore assoluto che assumono le variabili (x e y negli esempi) influisca direttamente sulla stima dell'errore sulla variabile finale z: alcune variabili producono risultati a varianza inferiore man mano che aumentano di intensità, mentre altre possono avere un comportamento contrario. Per questi motivi, a seconda della trasformazione e pertanto della stima del modello che si vuole ottenere, alcuni punti dell'immagine possono essere più importanti da osservare rispetto ad altri.

2.6.2 Propagazione dell'errore attraverso statistiche linearizzate

L'approccio a Punti Sigma (*Sigma-Point Approach* o *SPA*) permette di stimare il valor medio e la varianza di una variabile casuale all'uscita di un sistema modellato da una funzione $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ non lineare.

Per stimare valor medio e varianza, la variabile casuale in ingresso $\mathbf{x} \in \mathbb{R}^n$ viene approssimata da $2n + 1$ punti $\boldsymbol{\mathcal{X}}_i$, chiamati *sigma points*, ognuno pesato con un peso w_i, in modo da ottenere una distribuzione con media e varianza $\bar{\mathbf{x}}$ e $\boldsymbol{\Sigma}_\mathbf{x}$ rispettivamente, ovvero parametri esattamente uguali a quelli di $\mathbf{x}$.

Un modo per ottenere un insieme di punti, la cui distribuzione ha media e varianza uguali a quelli della distribuzione originale, consiste nel prendere $2n+1$ *sigma-points* e i rispettivi pesi nel modo seguente:

$$\begin{aligned} \boldsymbol{\mathcal{X}}_0 &= \bar{\mathbf{x}} \\ \boldsymbol{\mathcal{X}}_i &= \bar{\mathbf{x}} + \zeta \left(\sqrt{\boldsymbol{\Sigma}_{\mathbf{x}}}\right)_i \\ \boldsymbol{\mathcal{X}}_{i+n} &= \bar{\mathbf{x}} - \zeta \left(\sqrt{\boldsymbol{\Sigma}_{\mathbf{x}}}\right)_i \end{aligned} \tag{2.42}$$

dove ζ è un fattore scalare che tiene conto di quanto i punti sigma siano diffusi rispetto al valor medio $\bar{\mathbf{x}}$. Associato a ogni punto sigma è presente una coppia di pesi w_i^m e w_i^c usati nel calcolo, rispettivamente, della media e della covarianza.

A differenza dei metodi *montecarlo*, i *sigma-points* sono scelti in maniera deterministica in modo da rappresentare al meglio le statistiche della variabile.

Ottenuti i *sigma-points*, questi vengono trasformati (*unscented transformation*) attraverso la funzione f in punti sigma trasformati

$$\boldsymbol{\mathcal{Y}}_i = f(\boldsymbol{\mathcal{X}}_i) \quad {\scriptstyle i=0,\ldots,2n} \tag{2.43}$$

Da questi punti è possibile calcolare media e varianza della variabile di uscita attraverso

$$\begin{aligned} \bar{\mathbf{y}} &\approx \textstyle\sum_{i=0}^{2n} w_i^m \boldsymbol{\mathcal{Y}}_i \\ \boldsymbol{\Sigma}_{\mathbf{y}} &\approx \textstyle\sum_{i=0}^{2n} w_i^c (\boldsymbol{\mathcal{Y}}_i - \bar{\mathbf{y}})(\boldsymbol{\mathcal{Y}}_i - \bar{\mathbf{y}})^\top \end{aligned} \tag{2.44}$$

per ogni punto $i = 0, \ldots, 2n$. Media e varianza così ottenuti sono una buona approssimazione della media e varianza della distribuzione in ingresso trasformata attraverso la funzione f.

Il problema affrontato dall'approccio a punti Sigma è comunque un problema mal definito perché esistono infinite distribuzioni di probabilità possedenti la stessa la media e la covarianza. La *Unscented transform* (UT) [JU97], una dei possibili *Sigma-Point Approach*, fissa come valori $\zeta = \sqrt{n+\lambda}$, dove n è la dimensione dello spazio e λ è un numero definito come $\lambda = \alpha^2(n+\kappa) - n$ con $\alpha \in]0.001, 1]$ un numero piccolo positivo e κ solitamente posto a 0 o $3-n$. In alcuni articoli viene posto $\alpha = 1$ e $\kappa = 3 - n$ per le distribuzioni gaussiane.

Anche nella trasformazione *unscented* i punti sigma sono punti pesati e i pesi sono differenti nel calcolo del valor medio e della matrice di covarianza. La trasformazione *unscented* fissa pertanto questi pesi a

$$\begin{aligned} w_0^m &= \tfrac{\lambda}{n+\lambda} \\ w_0^c &= \tfrac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \\ w_i &= w_{i+n} = \tfrac{1}{2(n+\lambda)} \end{aligned} \tag{2.45}$$

La differenza tra i pesi w_i^m e w_i^c è solo nel termine centrale. Viene fissato $\beta = 2$ per le distribuzioni gaussiane.

È da sottolineare che le varianti degli approcci *sigma-point* hanno tali pesi calcolati in maniera differente.

2.7 Condizionamento nei sistemi lineari sovradimensionati

Nella sezione 2.6 e seguenti si è discusso di come il rumore si propaghi attraverso trasformazioni lineare e non lineari.

In questa sezione invece si studia il caso complementare dove è conosciuta la stima del rumore sulla variabile in uscita dal sistema lineare mentre si vuole sapere la stima del rumore sulle variabili in ingresso ovvero la bontà con cui si è ottenuta la soluzione di un sistema lineare. Per buona parte di questa sezione si fa riferimento alla teoria discussa in sezione 1.1 e ne è di fatto la continuazione, per integrarla poi, con nella sezione 3.5 al discorso più generale di regressione a modelli non lineari.

Sia pertanto

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.46}$$

un sistema lineare, ideale ovvero non affetto da rumore, con $\mathbf{x}$ la soluzione esatta del problema.

Una perturbazione $\delta\mathbf{b}$ sulla colonna dei termini noti (osservazioni,uscite), in

$$\mathbf{A}\mathbf{x} = \tilde{\mathbf{b}} \tag{2.47}$$

con $\tilde{\mathbf{b}} = \mathbf{b} + \delta\mathbf{b}$, provoca una perturbazione $\tilde{\mathbf{x}} = \mathbf{x} + \delta\mathbf{x}$ sulla soluzione di entità pari a

$$\delta\mathbf{x} = \mathbf{A}^{-1}\delta\mathbf{b} \tag{2.48}$$

In questo modo si ricade nel caso visto in precedenza di propagazione di rumore in un sistema lineare.

Un indice interessante consiste nel calcolare la norma dell'errore in relazione al valore atteso. Tale relazione vale

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}\|\|\mathbf{A}^{-1}\|\frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} = \kappa(\mathbf{A})\frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} \tag{2.49}$$

avendo definito $\kappa(\mathbf{A})$ numero di condizionamento (*condition number*) della matrice dei coefficienti (*sensitivity matrix*) $\mathbf{A}$. Nel caso particolare in cui $\mathbf{A}$ sia singolare, il condizionamento della matrice si pone pari a $\kappa(\mathbf{A}) = \infty$.

Estendiamo ora l'analisi al caso in cui il sistema sia un sistema lineare sovradimensionato. A questo scopo è possibile ricavare il condizionamento di una matrice usando un'ulteriore proprietà della decomposizione SVD. Sia

$$\mathbf{x} = \mathbf{V}\mathbf{S}^{-1}\mathbf{U}^{*}\mathbf{b} \tag{2.50}$$

la soluzione di un problema lineare sovradimensionato attraverso il metodo della decomposizione SVD. Se si esplicita l'equazione (2.50), si può mostrare come la soluzione di un sistema lineare, soluzione ottenuta attraverso la decomposizione SVD, ha come forma

$$\mathbf{x} = \sum \frac{\mathbf{u}_i^{\top}\mathbf{b}}{\sigma_i}\mathbf{v}_i \tag{2.51}$$

Da questa formulazione si vede che, quando i valori singolari σ_i sono bassi, ogni piccola variazione al numeratore viene amplificata: sotto la norma euclidea il numero di condizionamento di una matrice è esattamente il rapporto tra il più grande valore singolare rispetto al più piccolo. Il condizionamento è sempre positivo e un condizionamento prossimo all'unità indica una matrice ben condizionata.

Riassumendo il condizionamento ha le seguenti importanti proprietà:

- $\kappa(\mathbf{A}) = \kappa(\mathbf{A}^{-1})$
- $\kappa(c\mathbf{A}) = \kappa(\mathbf{A})$ per ogni $c \neq 0$
- $\kappa(\mathbf{A}) \geq 1$
- $\kappa(\mathbf{A}) = \frac{\sigma_1}{\sigma_n}$ se la norma è euclidea
- $\kappa(\mathbf{A}) = 1$ se $\mathbf{A}$ è ortogonale

Come è stato fatto notare nella sezione 1.1, la soluzione alle equazioni perpendicolari tende invece ad amplificare gli errori rispetto a soluzioni alternative. È facile dimostrare infatti che in questo caso

$$\kappa\left(\mathbf{A}^{\top}\mathbf{A}\right) = \left(\frac{\sigma_1}{\sigma^n}\right)^2 \tag{2.52}$$

2.8 Il Maximum Likelihood estimator

Da un punto di vista statistico il vettore dei dati $\mathbf{x} = \{x_1 \dots x_n\}$ sono realizzazioni di una variabile aleatoria di una popolazione sconosciuta. Il compito dell'analisi dei dati è quella di individuare la popolazione che più probabilmente ha generato quei campioni. In statistica, ogni popolazione è identificata da una corrispondente distribuzione di probabilità e associata a ogni distribuzione di probabilità c'è una parametrizzazione unica $\boldsymbol{\vartheta}$: variando questi parametri deve essere generata una differente distribuzione di probabilità.

Sia $f(\mathbf{x}|\boldsymbol{\vartheta})$ la funzione di densità di probabilità (PDF) che indica la probabilità di osservare $\mathbf{x}$ data una parametrizzazione $\boldsymbol{\vartheta}$. Se le osservazioni singole x_i sono statisticamente indipendenti una dall'altra la PDF di $\mathbf{x}$ può essere espressa come prodotto delle singole PDF:

$$f(\mathbf{x} = \{x_1 \dots x_n\}\,|\boldsymbol{\vartheta}) = f_1(x_1|\boldsymbol{\vartheta})f_2(x_2|\boldsymbol{\vartheta}) \dots f_n(x_n|\boldsymbol{\vartheta}) \tag{2.53}$$

Data una parametrizzazione $\boldsymbol{\vartheta}$ è possibile definire una specifica PDF che mostra la probabilità di comparire di alcuni dati rispetto ad altri. Nel caso reale abbiamo esattamente il problema reciproco: i dati sono stati osservati e c'è da individuare quale $\boldsymbol{\vartheta}$ ha generato quella specifica PDF.

Definizione 9 *Per risolvere il problema inverso, definiamo la funzione* $\mathcal{L} : \boldsymbol{\vartheta} \mapsto [0, \infty)$*, funzione di verosimiglianza (*likelihood*), definita come*

$$\mathcal{L}(\boldsymbol{\vartheta}|\mathbf{x}) = f(\mathbf{x}|\boldsymbol{\vartheta}) = \prod_{i=1}^{n} f_i(x_i|\boldsymbol{\vartheta}) \tag{2.54}$$

nel caso di osservazioni statisticamente indipendenti.

$\mathcal{L}(\boldsymbol{\vartheta}|\mathbf{x})$ indica la verosimiglianza del parametro $\boldsymbol{\vartheta}$ a seguito della osservazione degli eventi $\mathbf{x}$.

Il principio dello stimatore a massima verosimiglianza (*MLE*) $\hat{\boldsymbol{\vartheta}}_{MLE}$, sviluppato originariamente da R.A. Fisher negli anni '20 del novecento, sceglie come migliore parametrizzazione quella che fa adattare meglio la distribuzione di probabilità generata con i dati osservati.

Nel caso di distribuzione di probabilità gaussiana è utile una ulteriore definizione.

Definizione 10 *Sia* ℓ *la funzione di verosimiglianza logaritmica (*log likelihood*) definita come*

$$\ell = \log \mathcal{L}(\boldsymbol{\vartheta}|x_1 \dots x_n) = \sum_{i=1}^{n} \log f_i(x_i|\boldsymbol{\vartheta}) \tag{2.55}$$

avendo sfruttato le proprietà del logaritmo.

La miglior stima dei parametri del modello è quella che massimizza la verosimiglianza, ovvero la verosimiglianza logaritmica

$$\hat{\boldsymbol{\vartheta}}_{ML} = \arg\max_{\boldsymbol{\vartheta}} \mathcal{L}(\boldsymbol{\vartheta}|x_1 \dots x_n) = \arg\max_{\boldsymbol{\vartheta}} \sum_{i=1}^{n} \log f_i(x_i|\boldsymbol{\vartheta}) \tag{2.56}$$

siccome il logaritmo è una funzione monotona crescente.

È possibile trovare in letteratura, come stimatore ottimo, invece del massimo della funzione di verosimiglianza, il minimo dell'opposta

$$\hat{\boldsymbol{\vartheta}}_{ML} = \arg\min_{\boldsymbol{\vartheta}} \left(-\sum_{i=1}^{n} \log f_i(x_i|\boldsymbol{\vartheta}) \right) \tag{2.57}$$

ovvero il minimo del *negative log likelihood*.

Questa formulazione risulta molto utile quando la distribuzione del rumore è gaussiana. Siano (x_i, y_i) le realizzazioni della variabile aleatoria. Nel caso infatti di una generica funzione $y_i = g(x_i; \boldsymbol{\vartheta}) + \epsilon$ con rumore a distribuzione normale, tempo costante e media nulla, la Likelihood è

$$\mathcal{L}(\boldsymbol{\vartheta}|\mathbf{x}) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y_i - g(x_i; \boldsymbol{\vartheta}))^2}{2\sigma^2} \right) \tag{2.58}$$

e pertanto la MLE stimata attraverso il minimo della *negative log likelihood* si scrive come

$$\hat{\boldsymbol{\vartheta}}_{ML} = \arg\min_{\boldsymbol{\vartheta}} \left(-\sum_{i=1}^{n} \log N(x_i, y_i|\boldsymbol{\vartheta}) \right) = \arg\min_{\boldsymbol{\vartheta}} \sum_{i=1}^{n} (y_i - g(x_i; \boldsymbol{\vartheta}))^2 \tag{2.59}$$

ovvero la tradizionale soluzione ai minimi quadrati è lo stimatore alla massima verosimiglianza in caso di rumore additivo gaussiano a media nulla.

Ora, le m derivate parziali della log-verosimiglianza formano un vettore $m \times 1$

$$\mathbf{u}(\boldsymbol{\beta}) = \frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \begin{bmatrix} \dfrac{\partial \ell}{\partial \beta_1} \\ \vdots \\ \dfrac{\partial \ell}{\partial \beta_m} \end{bmatrix} \tag{2.60}$$

Il vettore $\mathbf{u}(\boldsymbol{\beta})$ è chiamato *score vector* (o *Fisher's score function*) della log-verosimiglianza. Se la log-verosimiglianza è concava, lo stimatore alla massima verosimiglianza individua pertanto il punto per il quale

$$\mathbf{u}(\hat{\boldsymbol{\beta}}) = \mathbf{0} \tag{2.61}$$

I momenti di $\mathbf{u}(\boldsymbol{\beta})$ soddisfano pertanto importanti proprietà: come abbiamo visto poco sopra la media di $\mathbf{u}(\boldsymbol{\beta})$ calcolata nel punto di massima verosimiglianza è uguale a zero e la matrice di varianza-covarianza è

$$\operatorname{var}(\mathbf{u}(\boldsymbol{\beta})) = \mathrm{E}\left[\mathbf{u}(\boldsymbol{\beta})\mathbf{u}(\boldsymbol{\beta})^\top\right] = -\mathrm{E}\left[\frac{\partial^2 \ell(\boldsymbol{\beta})}{\partial \beta_j \partial \beta_k}\right] = \mathcal{I}(\boldsymbol{\beta}) \tag{2.62}$$

La matrice $\mathcal{I}$, definita come il negativo dell'Hessiana, è chiamata *expected Fisher information matrix* e la sua inversa *observed information matrix*.

2.8.1 Stima del Massimo a Posteriori

Il *Maximum a Posteriori estimator*, o *maximum a posteriori probability* (MAP), fornisce come stima (una delle) moda della distribuzione a posteriori. A differenza della stima alla massima verosimiglianza, la MAP ottiene una densità a posteriori facendo uso della teoria bayesiana, unendo la conoscenza a priori $f(\boldsymbol{\vartheta})$ con la densità condizionale $\mathcal{L}(\boldsymbol{\vartheta}|\mathbf{x}) = f(\mathbf{x}|\boldsymbol{\vartheta})$ di verosimiglianza, ottenendo la nuova stima

$$\hat{\boldsymbol{\vartheta}}_{MAP} = \arg\max_{\boldsymbol{\vartheta}} f(\boldsymbol{\vartheta}|\mathbf{x}) = \arg\max_{\boldsymbol{\vartheta}} \frac{f(\mathbf{x}|\boldsymbol{\vartheta}) f(\boldsymbol{\vartheta})}{f(\mathbf{x})} = \arg\max_{\boldsymbol{\vartheta}} f(\mathbf{x}|\boldsymbol{\vartheta}) f(\boldsymbol{\vartheta}) \tag{2.63}$$

e nel caso di eventi non correlati la formula si trasforma in

$$\hat{\boldsymbol{\vartheta}}_{MAP} = \arg\max_{\boldsymbol{\vartheta}} \prod_{i=1}^{n} f(x_i|\boldsymbol{\vartheta}) f(\boldsymbol{\vartheta}) = \arg\max_{\boldsymbol{\vartheta}} \left\{ \sum_{i=1}^{n} \log f(x_i|\boldsymbol{\vartheta}) \right\} + \log f(\boldsymbol{\vartheta}) \tag{2.64}$$

dove, sempre per semplificare i conti, si sono sfruttate le proprietà del logaritmo.

Chiaramente se la probabilità a priori $f(\boldsymbol{\vartheta})$ è uniforme, MAP e MLE sono coincidenti.

2.9 Media ponderata con la varianza

Avendo molteplici osservazioni con varianza differente σ_i^2, si vogliono fondere le varie osservazioni. Questo è il caso per esempio di più osservazioni dello stesso osservabile eseguite da diversi sensori nello stesso istante o con lo stesso sensore di una quantità supposta costante ma con rumore di osservazione variabile nel tempo. L'obiettivo è ottenere una media pesata di ogni singola osservazione del tipo

$$\bar{x} = \sum_i w_i x_i \tag{2.65}$$

La varianza della variabile $\bar{x}$ sarà per definizione.

$$\sigma_{\bar{x}}^2 = \sum_i w_i^2 \sigma_i^2 \tag{2.66}$$

La soluzione ottima (stimatore di massima verosimiglianza) si ottiene minimizzando questa quantitá sotto il vincolo aggiuntivo $\sum_i w_i = 1$.

Il peso che minimizza questa quantità è

$$w_i = \frac{\frac{1}{\sigma_i^2}}{\sum_j \frac{1}{\sigma_j^2}} \tag{2.67}$$

In questo modo la varianza della media è inferiore alla varianza dei singoli strumenti di misura ed equivale a

$$\sigma_{\bar{x}}^2 = \frac{1}{\sum 1/\sigma_i^2} \tag{2.68}$$

Conseguenza diretta è il poter unire n letture dello stesso sensore e dello stesso osservabile (supponendo il rumore di osservazione a varianza costante) ma in istanti di tempo differenti. La varianza finale si riduce a

$$\sigma_{\bar{x}}^2 = \frac{\sigma_0^2}{n} \tag{2.69}$$

È possibile costruire in modo iterativo questo risultato attraverso la successione:

$$\bar{x}_{i+1} = (1-k)\bar{x}_i + k x_{i+1} \quad k = \frac{\sigma_{\bar{x}}^2}{\sigma_{\bar{x}}^2 + \sigma_{i+1}^2} \tag{2.70}$$

con k fattore di *blending*. Scritta in questo modo, la stima dell'osservabile è nella stessa forma del filtro di Kalman monodimensionale (si confronti questo risultato con quello di sezione 2.12.2): senza rumore di processo, il guadagno k è tendente a zero.

2.10 Analisi ad autovalori

Questa sezione è a cavallo tra analisi, statistica e classificazione e tratta quelle tematiche riguardanti l'analisi dei dati sfruttando le informazioni fornite dagli autovalori e autovettori.

2.10.1 PCA

La *Principal Component Analysis*, o trasformazione discreta di Karhunen-Loeve *KLT*, è una tecnica che ha due importanti applicazioni nell'analisi dei dati:

- permette di "ordinare" in una distribuzione vettoriale dei dati in modo da massimizzarne la varianza e, attraverso questa informazione, ridurre le dimensioni del problema: si tratta pertanto di una tecnica di compressione dei dati a perdita, o altrimenti una tecnica per rappresentare con meno dati la medesima quantità di informazione;
- trasforma i dati in ingresso in modo che la matrice di covarianza dei dati in uscita sia diagonale e pertanto le componenti dei dati siano tra loro scorrelate.

Allo stesso modo esistono due formulazioni della definizione di PCA:

- proietta i dati su uno spazio a dimensione inferiore tale che la varianza dei dati proiettati sia massima;
- proietta i dati su uno spazio a dimensione inferiore tale che la distanza tra il punto e la sua proiezione sia minima.

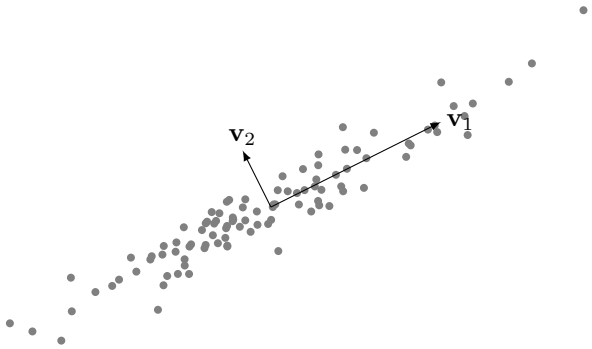


Figura 2.2: Componenti Principali.

Un esempio pratico di riduzione delle dimensioni di un problema è l'equazione di un iperpiano in d dimensioni: esiste una base dello spazio che trasforma l'equazione del piano riducendola a $d-1$ dimensioni senza perdere informazione, facendo risparmiare così una dimensione al problema.

Siano pertanto $\mathbf{x}_i \in \mathbb{R}^d$ vettori aleatori rappresentanti i risultati di un qualche esperimento, realizzazioni di una variabile aleatoria a media nulla, che possono essere memorizzati nelle *righe*[1] della matrice $\mathbf{X}$ di dimensioni $d \times n$, matrice pertanto che memorizza n vettori aleatori di dimensionalità d e con $n > d$. Ogni riga corrisponde a un diverso risultato $\mathbf{x}$ e la distribuzione di questi esperimenti deve avere media, quantomeno quella empirica, nulla.

Assumendo che i punti abbiano media zero (cosa che si può sempre ottenere con la semplice sottrazione del centroide), la loro covarianza delle occorrenze di $\mathbf{x}$ è data da

$$\mathbf{\Sigma} = E(\mathbf{x}\mathbf{x}^\top) \approx \frac{1}{n}\mathbf{X}^\top\mathbf{X} \tag{2.71}$$

Se i dati in ingresso $\mathbf{x}$ sono correlati, la matrice di covarianza $\mathbf{\Sigma}$ non è una matrice diagonale.

L'obiettivo di PCA è trovare una trasformazione $\mathbf{V}$ ottima che trasformi i dati da correlati a decorrelati

$$\mathbf{y} = \mathbf{V}^\top\mathbf{x} \tag{2.72}$$

ed ordinati in base al loro contenuto informativo in maniera tale che, se preso un sottoinsieme delle basi, possa tale approccio ridurre la dimensione del problema.

Se esiste una base ortonormale $\mathbf{V}$, tale che la matrice di covarianza di $\mathbf{\Sigma}_X$ espressa con questa base sia diagonale, allora gli assi di questa nuova base si chiamano componenti principali di $\mathbf{\Sigma}$ (o della distribuzione di X). Quando si ottiene una matrice di covarianza dove tutti gli elementi sono 0 tranne che sulla diagonale, significa che sotto questa nuova base dello spazio gli eventi sono tra loro scorrelati.

Questa trasformazione può essere trovata risolvendo un problema agli autovalori: si può infatti dimostrare che gli elementi della matrice di correlazione diagonale devono essere gli autovalori di Σ_X e per questa ragione le varianze della proiezione del vettore $\mathbf{x}$ sulle componenti principali sono gli autovalori stessi:

$$\mathbf{\Sigma V} = \mathbf{V\Delta} \tag{2.73}$$

dove $\mathbf{V}$ è la matrice degli autovettori (matrice ortogonale $\mathbf{V}\mathbf{V}^\top = \mathbf{I}$) e $\mathbf{\Delta}$ è la matrice diagonale degli autovalori $\lambda_1 \geq \ldots \geq \lambda_d$.

Per ottenere questo risultato esistono due approcci. Siccome $\mathbf{\Sigma}$ è una matrice simmetrica, reale, definita positiva, può essere scomposta in

$$\mathbf{\Sigma} = \mathbf{V\Delta V}^\top \tag{2.74}$$

chiamata decomposizione spettrale, con $\mathbf{V}$ matrice ortonormale, autovalori destri di $\mathbf{\Sigma}$, e $\mathbf{\Delta}$ è la matrice diagonale che contiene gli autovalori. Siccome la matrice $\mathbf{\Sigma}$ è definita positiva, tutti gli autovalori saranno positivi o nulli. Moltiplicando a destra l'equazione (2.74) per $\mathbf{V}$ si mostra che è esattamente la soluzione del problema (2.73).

Tale tecnica tuttavia richiede il calcolo esplicito di $\mathbf{\Sigma}$. Data una matrice rettangolare $\mathbf{X}$, la tecnica SVD permette esattamente di trovare gli autovalori e gli autovettori della matrice $\mathbf{X}^\top\mathbf{X}$ ovvero di $\mathbf{\Sigma}$ e pertanto è la tecnica più efficiente e numericamente stabile per ottenere questo risultato. Attraverso la SVD è possibile decomporre la matrice degli eventi $\mathbf{X}$ in modo che

$$\mathbf{X} = \mathbf{USV}^\top$$

usando come rappresentazione la Economy/Compact SVD dove $\mathbf{U}$ sono gli autovettori sinistri (*left singular vectors*), $\mathbf{S}$ gli autovalori di $\mathbf{\Sigma}$ e $\mathbf{V}$ gli autovettori destri. È da notare che usando la SVD non è necessario calcolare esplicitamente la matrice di covarianza $\mathbf{\Sigma}$. Tale matrice può essere tuttavia ricavata in un secondo momento attraverso l'equazione

$$\mathbf{\Sigma} = \mathbf{X}^\top\mathbf{X} = \mathbf{VS}^2\mathbf{V}^\top \tag{2.75}$$

[1] In questo documento si è scelta la convezione per righe: in letteratura si trova in ugual maniera la rappresentazione per riga o per colonna dei dati e di conseguenza la nomenclatura potrebbe essere differente e far riferimento a $\mathbf{U}$ invece che a $\mathbf{V}$ e viceversa.

Confrontando questa relazione con quella di equazione (2.74) si ottiene anche che $\mathbf{\Delta} = \mathbf{S}^2$.

Vanno ricordate le proprietà degli autovalori:

- Gli autovalori di $\mathbf{X}\mathbf{X}^\top$ e di $\mathbf{X}^\top\mathbf{X}$ sono i medesimi.
- I valori singolari sono gli autovalori della matrice $\mathbf{X}^\top\mathbf{X}$, ovvero la matrice di covarianza;
- Gli autovalori maggiori sono associati ai vettori direzione di massima varianza;

e anche una importante proprietà della SVD

$$\mathbf{x}^{(l)} = \sum_{i=1}^{l} \mathbf{u}_i \sigma_i \mathbf{v}_i^\top \tag{2.76}$$

che è l'approssimazione di rango l più vicina a $\mathbf{X}$. Questo fatto, unito alla caratteristica propria di SVD di ritornare i valori singolari di $\mathbf{X}$ ordinati dal maggiore al minore, permette l'approssimazione di una matrice a una di rango inferiore.

Selezionando il numero di autovettori con autovalori abbastanza grandi è possibile creare una base ortonormale $m \times n$ dello spazio $\tilde{\mathbf{V}}$ tale che $\mathbf{y} \in \mathbb{R}^m$ ottenuto come proiezione

$$\mathbf{y} = \tilde{\mathbf{V}}^\top \mathbf{x}$$

rappresenti uno spazio di dimensioni ridotte ma che comunque contenga la maggior parte dell'informazione del sistema.

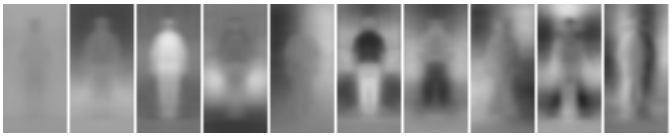


Figura 2.3: Esempio dei primi 10 autovettori 24×48 estratti dal dataset di pedoni Daimler-DB

2.10.2 ZCA

PCA è una tecnica che permette di decorrelare le componenti ma questo non impedisce agli autovalori di essere essere differenti. Se si forzano tutti gli autovalori ad essere uguali (si veda anche 2.4.1), e di fatto viene cambiata l'unita di misura, in modo tale che tutte le componenti principali siano uguali (le varianze siano uguali) la distribuzione viene detta sferizzata e il procedimento viene indicato come sbiancamento (*whitening*) dei dati.

$\mathbf{W}$ è chiamata matrice di sbiancamento (*whitening matrix*) ed è indicata come la soluzione *Zero Components Analysis* (*ZCA*) dell'equazione

$$\mathbf{Y}^\top\mathbf{Y} = \mathbf{I} \tag{2.77}$$

Dopo la trasformazione di sbiancamento, i dati, oltre ad avere media zero e decorrelati, avranno covarianza identità.

La matrice sbiancata dalla PCA è ottenuta come

$$\mathbf{X}_{PCA} = \mathbf{V}^\top\mathbf{X}^\top = \mathbf{S}\mathbf{U}^\top \tag{2.78}$$

ovvero $\mathbf{W}_{PCA} = \mathbf{V}^\top$ mentre la matrice sbiancante dalla ZCA si può ottenere da

$$\mathbf{X}_{ZCA} = \mathbf{\Delta}^{-1}\mathbf{X}_{PCA} = \mathbf{S}^{-1}\mathbf{X}_{PCA} = \mathbf{S}^{-1}\mathbf{V}^\top\mathbf{X}^\top = \mathbf{U}^\top \tag{2.79}$$

ovvero $\mathbf{W}_{ZCA} = \mathbf{S}^{-1}\mathbf{V}^\top$ ma soprattutto il risultato notevole $\mathbf{X}_{ZCA} = \mathbf{U}^\top$.

È da notare che la matrice dopo la trasformazione PCA potrebbe avere un numero di componenti inferiore ai dati di ingresso, mentre ZCA ha sempre lo stesso numero di componenti.

2.11 Elementi di probabilità

In questa sezione sono riportati alcune relazioni di probabilità utili poi nella sezione successiva.

Definiamo la funzione di densità di probabilità (probability density function, PDF) come

$$p_X(x) = P(X = x) \tag{2.80}$$

per poter fare i passaggi dal caso discreto al caso continuo.

Il teorema di Bayes (o formula di Bayes) è una relazione che si ottiene unendo il teorema della probabilità composta con il teorema della probabilità assoluta.

Partendo dalla definizione di probabilità condizionata $P(A,B) = P(A|B)P(B)$ (*multiplication rule*) si ottiene:

$$P(A|B) = \frac{P(A,B)}{P(B)} \tag{2.81}$$

e il viceversa

$$P(B|A) = \frac{P(B,A)}{P(A)} \tag{2.82}$$

con la considerazione che $P(A,B) = P(B,A)$ si ottiene

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.83}$$

Si può applicare lo stesso ragionamento nel caso di tre variabili:

$$P(A,B,C) = P(A|B,C)P(B,C) = P(B|A,C)P(A,C) = P(B,A,C) \tag{2.84}$$

si ottiene la formula di Bayes

$$P(A|B,C) = \frac{P(B|A,C)P(A|C)}{P(B|C)} \tag{2.85}$$

dove si vede come si propaga la dipendenza da una terza variabile C.

Un'altra formula importante che verrà usata nella prossima sezione è il teorema della probabilità assoluta (*law of total probability*):

$$P(B) = \sum P(A_i, B) = \sum P(A_i)P(B|A_i) \tag{2.86}$$

o nel caso continuo

$$p_X(x) = \int p_{X,Y}(X = x, Y = y)dy = \int p(x|Y = y)p(y)dy \tag{2.87}$$

densità marginale di $\mathbf{X}$.

2.12 Filtri Bayesiani

In questa sezione viene discusso il problema dei filtraggi statistici ovvero quella classe di problemi dove sono a disposizione dati proveniente da uno o più sensori affetti da rumore, dati che rappresentano l'osservazione dello stato dinamico di un sistema, non direttamente osservabile ma di cui è richiesta una stima. Il procedimento attraverso il quale si cerca di trovare la miglior stima dello stato interno di un sistema viene chiamato "filtraggio" in quanto è un metodo per filtrare via le diverse componenti di rumore. L'evoluzione di un sistema (l'evoluzione del suo stato interno) deve seguire leggi fisiche conosciute su cui va ad agire una componente di rumore (rumore di processo). È proprio attraverso la conoscenza delle equazioni che regolano l'evoluzione dello stato che è possibile fornire una stima migliore dello stato interno.

Un processo fisico può essere visto, nella sua rappresentazione di spazio di stato (*State Space Model*), attraverso una funzione che descrive come lo stato $\mathbf{x}_t$ si evolve nel tempo:

$$\dot{\mathbf{x}}_t = f(t, \mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t) \tag{2.88}$$

con $\mathbf{u}_t$ eventuali ingressi al sistema, conosciuti, e $\mathbf{w}_t$ parametro rappresentante il rumore *di processo*, ovvero l'aleatorietà che ne regola l'evoluzione. Allo stesso modo anche l'osservazione dello stato è un processo su cui agisce un rumore, in questo caso definito *di osservazione*. Anche in questo caso è possibile definire una funzione che modella l'osservazione $\mathbf{z}_t$ come

$$\mathbf{z}_t = h(t, \mathbf{x}_t, \mathbf{v}_t) \tag{2.89}$$

con $\mathbf{v}_t$ rumore di osservazione e funzione solo dello stato attuale.

Questo formalismo è descritto nel dominio continuo del tempo. Nelle applicazioni pratiche i segnali vengono campionati a tempo discreto k e pertanto viene normalmente usata una versione a tempo discreto nella forma

$$\begin{aligned} x_{k+1} &= f_k(x_k, u_k, w_k) \\ z_{k+1} &= h_k(x_k, v_k) \end{aligned} \tag{2.90}$$

dove w_k e v_k possono essere visti come sequenze di rumore bianco di statistiche note.

Nei sistemi che soddisfano le equazioni (2.90), l'evoluzione dello stato è solo funzione dello stato precedente, mentre l'osservazione è solo funzione dello stato attuale (figura 2.4). Se un sistema soddisfa tali ipotesi si dice che il processo è markoviano: l'evoluzione del sistema e l'osservazione devono essere solo funzione dello stato corrente e non degli stati passati. L'accesso all'informazione sullo stato avviene sempre per via indiretta attraverso l'osservazione (*Hidden Markov Model*).

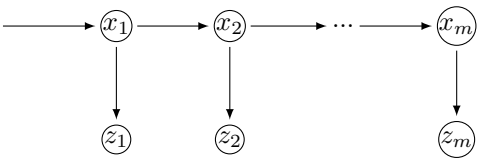


Figura 2.4: Esempio di evoluzione e osservazione di un sistema markoviano.

Molti approcci per stimare da un insieme di misure lo stato sconosciuto di un sistema non tengono conto della natura rumorosa di tali osservazioni. È possibile infatti costruire un algoritmo che esegua una regressione non lineare sulle osservazioni per ottenere la stima di tutti gli stati del problema, risolvendo un problema di ottimizzazione con un elevato numero di incognite.

I filtri, a differenza delle regressioni, si pongono come obiettivo quello di fornire la miglior stima di delle variabili (stato) man mano che i dati delle osservazioni arrivano. Dal punto di vista teorico le regressioni sono il caso ottimo, mentre i filtraggi convergono al risultato corretto solo dopo un numero di campioni sufficientemente elevato.

I filtri bayesiani si pongono come obiettivo quello di stimare all'istante di tempo k, discreto, lo stato della variabile aleatoria $\mathbf{x}_k \in \mathbb{R}^n$ data un'osservazione del sistema, indiretta, $\mathbf{z}_k \in \mathbb{R}^m$.

Le tecniche di filtraggio permettono sia di ottenere la stima migliore dello stato sconosciuto $\mathbf{x}_k$ ma anche la distribuzione di probabilità multivariata $p(\mathbf{x}_k)$ rappresentante la conoscenza che si ha dello stato stesso.

Data l'osservazione del sistema è possibile definire una densità di probabilità di $\mathbf{x}_k$ *a posteriori* dell'osservazione dell'evento $\mathbf{z}_k$ dovuta proprio alla conoscenza in più che si ottiene da tale osservazione:

$$p^+(\mathbf{x}_k) = p(\mathbf{x}_k|\mathbf{z}_k) \tag{2.91}$$

dove, probabilità condizionata, $p(\mathbf{x}_k|\mathbf{z}_k)$ indica la probabilità che lo stato nascosto sia $\mathbf{x}_k$ data l'osservazione $\mathbf{z}_k$. La "funzione" $p(\mathbf{x}_k|\mathbf{z}_k)$ rappresenta il modello della misurazione dello stato (*measurement model*). In letteratura la distribuzione a posteriori $p^+(\mathbf{x}_k)$ viene anche indicata come *belief*.

Applicando il teorema di Bayes all'equazione (2.91) si ottiene

$$p(\mathbf{x}_k|\mathbf{z}_k) = c_k p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k) \tag{2.92}$$

con c_k fattore di normalizzazione tale che $\int p(\mathbf{x}_k|\mathbf{z}_k) = 1$. La conoscenza di $p(\mathbf{z}_k|\mathbf{x}_k)$ risulta indispensabile, conoscenza che rappresenta la probabilità che l'osservazione sia proprio la quantità $\mathbf{z}_k$ osservata dato il possibile stato $\mathbf{x}_k$. L'utilizzo del teorema di Bayes per stimare lo stato data l'osservazione è il motivo per il quale questa classe di filtraggi è detta *bayesiana*.

Oltre alla conoscenza *a posteriori* della distribuzione di probabilità, è possibile sfruttare un'ulteriore informazione per migliorare la stima: la conoscenza *a priori* rispetto all'osservazione, ottenuta dal vincolo secondo il quale lo stato non si evolve in maniera totalmente imprevedibile ma viceversa può solo evolversi in determinati modi con determinate probabilità. Tali modi in cui il sistema si può evolvere sono funzione solamente dello stato corrente. L'ipotesi di processo Markoviano implica infatti che l'unico stato passato che influisca sull'evoluzione del sistema sia quello di tempo $k-1$, ovvero $p(x_k|x_{1:k-1}) = p(x_k|x_{k-1})$.

È pertanto possibile eseguire la predizione *a priori*, grazie all'equazione di Chapman-Kolmogorov:

$$p^-(\mathbf{x}_k) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)p(\mathbf{x}_{k-1})d\mathbf{x}_{k-1} \tag{2.93}$$

dove $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$ rappresenta la dinamica del sistema (*dynamic model*) e $\mathbf{u}_k$ sono gli eventuali ingressi, che influenzano l'evoluzione del sistema, di cui però la conoscenza è totale.

Dalla conoscenza dello stato *a priori* e dall'osservazione z_k è possibile riscrivere l'equazione (2.91) nell'equazione di aggiornamento dello stato

$$p^+(\mathbf{x}_k) = c_k p(\mathbf{z}_k|\mathbf{x}_k)p^-(\mathbf{x}_k) \tag{2.94}$$

Lo stato viene stimato alternando una fase di predizione (stima *a priori*) a una fase di osservazione (stima *a posteriori*). Questo processo, iterativo, prende il nome di stima bayesiana ricorsiva (*Recursive Bayesian Estimation*).

Le tecniche descritte in questa sezione faranno riferimento solo all'ultima osservazione disponibile per stimare lo stato. Dal punto di vista formale è possibile estendere la discussione al caso in cui vengano sfruttate tutte le osservazioni per ottenere una stima più accurata dello stato. In questo caso le equazioni di filtraggio e predizione diventano

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{z}_{1:k}) &= \int p(\mathbf{x}_{1:k}|\mathbf{z}_{1:k})d\mathbf{x}_{1:k-1} \\ p(\mathbf{x}_{k+1}|\mathbf{z}_{1:k}) &= \int p(\mathbf{x}_{k+1}|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k})d\mathbf{x}_k \end{aligned} \tag{2.95}$$

Per motivi di semplicità e per il ridotto peso computazionale normalmente viene valutata solo l'ultima osservazione, ma in determinati casi (per esempio nei filtri particellari) è possibile introdurre la conoscenza di tutta la storia passata nelle equazioni in maniera abbastanza agevole.

In quanto stima di variabili continue, non risulta possibile sfruttare la teoria bayesiana "direttamente" ma sono state proposte in letteratura diversi approcci per permettere la stima in maniera efficiente sia dal punto di vista computazionale che di utilizzo della memoria.

A seconda che il problema sia lineare o non-lineare, che la distribuzione di probabilità del rumore sia gaussiana o meno, ognuno di questi filtri si comporta in maniera più o meno ottima.

Il Filtro di Kalman (sezione 2.12.2) è il filtro ottimo nel caso in cui il problema sia lineare e la distribuzione del rumore gaussiana. I filtri di Kalman Estesi e a Punti Sigma, sezioni 2.12.4 e 2.12.5 rispettivamente, sono filtri sub-ottimi per problemi non-lineari e distribuzione del rumore gaussiana (o poco discostanti da essa). Infine i filtri particellari sono una soluzione sub-ottima per i problemi non lineari con distribuzione del rumore non gaussiana.

I filtri *grid-based* (sezione 2.12.1) e i filtri particellari (sezione 2.12.8) lavorano su una rappresentazione discreta dello stato, mentre i filtri Kalman, Extendend e Sigma-Point lavorano su una rappresentazione continua dello stato.

Kalman, Kalman Esteso e Kalman a Punti Sigma stimano la distribuzione dell'incertezza (dello stato, del processo, dell'osservazione) come una singola gaussiana. Esistono estensioni multimodali come *Multi-hypothesis tracking* (*MHT*) che permettono di applicare i filtri di Kalman a distribuzioni come miscela di gaussiane, mentre i filtri particellari e *grid-based* sono per loro natura multimodali.

Un ottimo *survey* sui filtraggi bayesiani è [Che03].

2.12.1 Metodi Grid-based

Gli approcci *grid-based* si adattano perfettamente a quei problemi dove lo stato assume solo un numero limitato di valori discreti (vengono infatti detti Filtri Discreti) mentre permettono di fornire una stima approssimata nel caso di stato continuo (*histogram filters*) trasformato in discreto attraverso una quantizzazione spaziale. Ad ogni elemento della griglia (o dell'istogramma) è associata la probabilità che lo stato sia effettivamente in quella particolare cella. La teoria dei filtri bayesiani (perciò distribuzioni multimodali e sistemi fortemente non lineari) è sfruttata direttamente, limitata però ai soli punti discreti in cui lo stato può vivere.

Si supponga che vengano usati m punti per rappresentare lo stato $\mathbf{x} \in \mathbb{R}^n$. Se lo stato originale è continuo questa è chiaramente una approssimazione ed è preferibile che $m \gg n$. Ad ogni iterazione k, esistono pertanto $\mathbf{x}_{i,k} \in \mathbb{R}^n$ con $i = 1, \ldots, m$ stati possibili a cui è associata una distribuzione di probabilità $p_{i,k}$ che si evolve nel tempo in base alla dinamica del problema.

Valgono le equazioni viste in precedenza, ovvero la stima *a priori*:

$$p^-_{i,k} = \sum_{j=1}^{m} p(x_{i,k}|x_{j,k-1}) p^+_{j,k-1} = \sum_{j=1}^{m} f_{i,j} p^+_{j,k-1} \quad \forall i \tag{2.96}$$

e l'equazione di aggiornamento dello stato *a posteriori* dell'osservazione z_k:

$$p^+_{i,k} = c_k p(z_k|x_{i,k}) p^-_{i,k} \quad \forall i \tag{2.97}$$

con c_k sempre fattore di normalizzazione tale che $\sum p^+_i = 1$.

I metodi *grid-based* permettono di applicare pertanto la teoria ricorsiva bayesiana direttamente.

2.12.2 Filtro di Kalman

Il filtro di Kalman [WB95] cerca di stimare in presenza di disturbi lo stato interno $\mathbf{x} \in \mathbb{R}^n$, non accessibile, di un sistema tempo discreto, la cui conoscenza del modello è completa. Di fatto il filtro di Kalman è lo stimatore ricorsivo ottimo: se il rumore del problema è gaussiano, il filtro di Kalman fornisce la stima ai minimi quadrati dello stato interno del sistema.

Per ragioni storiche il filtro di Kalman si riferisce propriamente al solo filtraggio di un sistema dove la transizione di stato e l'osservazione sono funzioni lineari dello stato corrente.

Seguendo la teoria dei sistemi lineari, la dinamica di un sistema "lineare" tempo continuo è rappresentata da una equazione differenziale del tipo

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{w}(t) \tag{2.98}$$

equazione di aggiornamento dello stato, a cui è associata un'osservazione indiretta di questo stato attraverso un sistema lineare:

$$\mathbf{z}(t) = \mathbf{H}(t)\mathbf{x}(t) + \mathbf{v}(t) \tag{2.99}$$

con $\mathbf{z} \in \mathbf{R}^m$ l'osservabile.

Il filtro di Kalman a tempo discreto viene in aiuto dei sistemi reali dove il mondo viene campionato a intervalli discreti, trasformando il sistema lineare tempo continuo in un sistema lineare del tipo

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k \\ \mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \end{cases} \tag{2.100}$$

Se il sistema si evolve seguendo questo modello è chiamato *Linear-Gaussian State Space Model* o *Linear Dynamic System*. Se i valori delle matrici sono indipendenti dal tempo, il modello è chiamato *stazionario*.

Le variabili w_k e v_k rappresentano rispettivamente il rumore di processo e di osservazione, valor medio nullo $\bar{w_k} = \bar{v_k} = 0$ e varianza rispettiva $\mathbf{Q}$ e $\mathbf{R}$ conosciute (si suppone rumore gaussiano bianco). $\mathbf{A}$ è una matrice $n \times n$ di transizione dello stato, $\mathbf{B}$ è una matrice $n \times l$ che collega l'ingresso di controllo opzionale $\mathbf{u} \in \mathbb{R}^l$ con lo stato $\mathbf{x}$ e infine $\mathbf{H}$ è una matrice $m \times n$ che collega lo stato con la misura $\mathbf{z}_k$. Tutte queste matrici, rappresentanti il modello del sistema, devono essere conosciute con assoluta precisione, pena l'introduzione di errori sistematici.

Il filtro di Kalman è un filtro di stima ricorsivo e richiede ad ogni iterazione la conoscenza dello stato stimato dal passo precedente $\hat{\mathbf{x}}_{k-1}$ e l'osservazione corrente $\mathbf{z}_k$, osservazione indiretta dello stato del sistema.

Sia $\hat{\mathbf{x}}_k^-$ la stima *a priori* dello stato del sistema, basata sulla stima ottenuta al tempo $k-1$ e dalla dinamica del problema, e $\hat{\mathbf{x}}_k$ la stima dello stato del problema *a posteriori* dell'osservazione $\mathbf{z}_k$ e basata su di essa. Da queste definizioni è possibile definire l'errore della stima *a priori* e a *a posteriori* come

$$\begin{aligned} \mathbf{e}_k^- &= \mathbf{x}_k - \hat{\mathbf{x}}_k^- \\ \mathbf{e}_k &= \mathbf{x}_k - \hat{\mathbf{x}}_k \end{aligned} \tag{2.101}$$

A questi errori è possibile associare

$$\begin{aligned} \mathbf{P}_k^- &= \mathrm{E}[\mathbf{e}_k^- \mathbf{e}_k^{-\top}] \\ \mathbf{P}_k &= \mathrm{E}[\mathbf{e}_k \mathbf{e}_k^\top] \end{aligned} \tag{2.102}$$

le matrici di covarianza *a priori* e *a posteriori* rispettivamente.

L'obiettivo del filtro di Kalman è minimizzare la covarianza dell'errore *a posteriori* $\mathbf{P}_k$ e fornire un metodo per ottenere la stima di $\hat{\mathbf{x}}_k$ data la stima a priori $\hat{\mathbf{x}}_k^-$ e l'osservazione $\mathbf{z}_k$.

Il filtro di Kalman fornisce una stima dello stato *a posteriori* attraverso una combinazione lineare della stima dello stato precedente e dell'errore di osservazione:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-) \tag{2.103}$$

spostando il problema della stima a quello di ricavare il fattore di guadagno $\mathbf{K}_k$ (*blending factor*). La differenza $\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-$ è chiamata *residuo*, o *innovation*, e rappresenta la discrepanza tra l'osservazione predetta e quella realmente avvenuta. È da notare che la metrica usata per calcolare il residuo può dipendere dalle peculiarità del problema.

Il filtro di Kalman viene normalmente presentato in due fasi: aggiornamento del tempo (fase di predizione) e aggiornamento della misura (fase di osservazione).

Nella prima fase si ottiene la stima *a priori* sia di $\hat{\mathbf{x}}_k$ che della covarianza $\mathbf{P}_k$. La stima *a priori* $\hat{\mathbf{x}}_k^-$ viene dalla buona conoscenza della dinamica del sistema (2.100):

$$\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k \tag{2.104}$$

e allo stesso modo viene aggiornata la stima a priori della covarianza dell'errore:

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top + \mathbf{Q}_k \tag{2.105}$$

Queste sono le miglior stime dello stato e della covarianza dell'istante k ottenibili *a priori* dell'osservazione del sistema.

Nella seconda fase viene calcolato il guadagno

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^\top \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\top + \mathbf{R}_k\right)^{-1} \tag{2.106}$$

che minimizza la covarianza *a posteriori* e, con questo fattore, viene aggiornato lo stato a posteriori attraverso l'equazione (2.103).

Usando questo valore per il guadagno $\mathbf{K}$, la stima *a posteriori* della matrice di covarianza diventa

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^- \tag{2.107}$$

Per poter unificare le diverse varianti dei filtri di Kalman si possono tradurre queste equazioni usando le matrici di varianza-covarianza

$$\begin{aligned} \mathrm{cov}(x_k, z_k) &= \mathbf{P}_k^- \mathbf{H}_k^\top \\ \mathrm{cov}(z_k) &= \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\top \end{aligned} \tag{2.108}$$

in modo da poter scrivere l'equazione (2.106) come

$$\mathbf{K}_k = \mathrm{cov}(x_k, z_k)\left(\mathrm{cov}(z_k) + \mathbf{R}_k\right)^{-1} \tag{2.109}$$

e, sostituendo le covarianze (2.108) in (2.107) si ottiene

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k \,\mathrm{cov}(x_k, z_k)^\top \tag{2.110}$$

Come si può facilmente notare la matrice di covarianza e il guadagno di Kalman non dipendono minimamente né dallo stato, ne dalle osservazioni, ne tanto meno dal residuo, e hanno una storia indipendente.

Kalman richiede tuttavia un valore iniziale della variabile di stato e della matrice di covarianza: il valore iniziale dello stato deve essere il più simile possibile al valore vero e la somiglianza a questo valore va inserita nella matrice di covarianza iniziale.

Filtro di Kalman monodimensionale

È interessante mostrare, come esempio, il caso semplificato di filtro di Kalman di stato monodimensionale e coincidente con l'osservabile. Le equazioni di transizione e di osservazioni sono

$$\begin{aligned} x_i &= x_{i-1} + u_i + w_i \\ z_i &= x_i + v_i \end{aligned} \tag{2.111}$$

dove w_i è il rumore di processo la cui varianza q_i rappresenta la stima della probabilità di variazione del segnale stesso (bassa se il segnale varia poco nel tempo, alta se il segnale varia molto) mentre v_i è il rumore di osservazione di varianza r_i, rumore associato all'osservazione dello stato.

Il ciclo di predizione è molto semplice e diventa:

$$\begin{aligned} x_i^- &= x_{i-1} + u_i \\ p_i^- &= p_{i-1} + q_i \end{aligned} \tag{2.112}$$

Il guadagno di Kalman k diventa

$$k_i = \frac{p_i^-}{p_i^- + r_i} \tag{2.113}$$

e infine la fase di osservazione diventa

$$\begin{aligned} x_i &= x_i^- + k_i(z_i - x_i^-) = k_i z_i + (1 - k_i)x_i^- \\ p_i &= (1 - k_i)p_i^- \end{aligned} \tag{2.114}$$

È solitamente possibile stimare a priori il valore di r, mentre quello di q va impostato attraverso esperimenti.

Come si vede nella prima delle equazioni (2.114), il fattore k è di fatto un *blending factor* tra l'osservazione dello stato e lo stato stimato precedente.

Nel caso monodimensionale è facile vedere come il guadagno k e la varianza p sono processi indipendenti dallo stato e dalle osservazioni, tanto meno dall'errore. Se r e q non variano nel tempo, k e p sono sequenze numeriche che convergono a un numero costante determinato solamente dalla caratterizzazione del rumore, indipendentemente dai valori assunti all'inizio. Si confronti questo risultato con quello che si ottiene dall'equazione (2.70).

2.12.3 Rumore correlato

Nel caso in cui il rumore non sia semplicemente additivo, ma si propaghi nel sistema attraverso una trasformazione comunque lineare, il sistema di Kalman si generalizza in

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \mathbf{W}_k\mathbf{w}_k \\ \mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{V}_k\mathbf{v}_k \end{cases} \tag{2.115}$$

Il rumore di processo è correlato attraverso una matrice $\mathbf{W}_k$ alla sorgente, e il rumore di osservazione attraverso una matrice $\mathbf{V}_k$.

È possibile in questo caso applicare le stesse equazioni del sistema di Kalman introducendo le sostituzioni

$$\begin{aligned} \mathbf{Q}'_k &= \mathbf{W}_k\mathbf{Q}_k\mathbf{W}_k^\top \\ \mathbf{R}'_k &= \mathbf{V}_k\mathbf{R}_k\mathbf{V}_k^\top \end{aligned} \tag{2.116}$$

Tale risultato tornerà utile nella sezione seguente sul filtro di Kalman esteso.

Chiaramente se le matrici $\mathbf{W}_k$ e $\mathbf{V}_k$ sono delle identità, ovvero il rumore è semplicemente additivo, la forma si semplifica e ridiventa quella vista in precedenza.

2.12.4 Filtro di Kalman Esteso

Il filtro di Kalman esteso *Extended Kalman Filter* (EKF) è una versione non-lineare del filtro di Kalman usata quando l'evoluzione o l'osservazione dello stato del sistema sono non-lineari.

Un sistema non lineare a tempo discreto, formato dall'evoluzione dello stato e dalla sua osservazione, può essere scritto in forma generalizzata come

$$\begin{cases} \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \\ \mathbf{z}_k &= h(\mathbf{x}_k, \mathbf{v}_k) \end{cases} \tag{2.117}$$

dove, oltre allo stato $\mathbf{x}_k$ e agli ingressi $\mathbf{u}_k$, anche gli errori di processo $\mathbf{w}_k$ e di osservazione $\mathbf{v}_k$ possono influire in maniera non lineare nell'evoluzione dello stato f e nell'osservazione h, generalizzando anche il concetto di rumore additivo usato in precedenza.

Per poter essere applicato, EKF richiede il calcolo degli Jacobiani sia di f che di h. Applicando la teoria mostrata nella sezione 2.6 sulla propagazione dell'incertezza in funzioni non lineari, attraverso le matrici delle derivate è possibile sfruttare le stesse formulazioni matematiche fatte per il caso di Kalman lineare su funzioni non-lineari usando come matrici

$$\begin{aligned} \mathbf{A}_k &= \left.\frac{\partial f(\mathbf{x},\mathbf{u}_k,\bar{\mathbf{w}})}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^-} & \mathbf{W}_k &= \left.\frac{\partial f(\hat{\mathbf{x}}_k^-,\mathbf{u}_k,\mathbf{w})}{\partial \mathbf{w}}\right|_{\bar{\mathbf{w}}} \\ \mathbf{H}_k &= \left.\frac{\partial h(\mathbf{x},\bar{\mathbf{v}})}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^-} & \mathbf{V}_k &= \left.\frac{\partial h(\mathbf{x},\mathbf{v})}{\partial \mathbf{v}}\right|_{\bar{\mathbf{x}}} \end{aligned} \tag{2.118}$$

e usando come equazione di aggiornamento

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-)) \tag{2.119}$$

È comunque da notare che anche il calcolo del residuo $\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-)$ può essere una funzione non lineare (per esempio quando si esegue un confronto tra angoli ed esiste una periodicità dell'errore).

Rispetto a Kalman lineare, la versione EKF risulta una scelta sub-ottima come stimatore ma comunque ampiamente accettata e usata in applicazioni pratiche. Il filtro di Kalman esteso, per sua costruzione, raggiunge solo una precisione di primo ordine ma permette comunque risultati vicini all'ottimo nel caso di funzionamento del filtro in punti in cui le derivate seconde sono nulle.

2.12.5 Filtro di Kalman Sigma-Point

Un'alternativa nel caso non-lineare al filtro di Kalman Esteso è il filtro di Kalman a Punti Sigma. In risultati riportati da diversi esperimenti, nel caso di funzioni f e h non lineari, il *Sigma Point Kalman Filter* (SPKF) tende a fornire prestazioni migliori rispetto a EKF: la propagazione dell'errore linearizzata dal punto di vista statistico (SPKF) è generalmente migliore della propagazione dell'espansione in serie di Taylor (EKF).

Non solo lo stato, ma i diversi punti intorno la media (i *sigma points*) vengono propagati attraverso le funzioni che compongono l'aggiornamento e l'osservazione dello stato di Kalman. Il vantaggio di SPKF è quello di non richiedere il calcolo degli Jacobiani e normalmente permette una stima migliore di media e varianza del processo.

Il filtro di Kalman Unscented (*Unscented Kalman filter*) è una delle varie versioni del filtro di Kalman a Punti Sigma. In questo caso si fa uso della teoria per la propagazione dell'incertezza discussa in sezione 2.6.2 per stimare valor medio e covarianza dello stato a priori e dell'errore di osservazione.

Anche con il filtro Unscented è possibile gestire il caso in cui il rumore si inserisce nel sistema in maniera non additiva. Per generalizzare il caso di rumore non additivo definiamo, allo scopo di mantenere una sintassi uguale a quella discussa in sezione 2.6.2, una variabile chiamata *stato aumentato* $\mathbf{x}^a \in \mathbb{R}^{n^a}$ con $n^a = n + q$ formata dallo stato $\mathbf{x} \in \mathbb{R}^n$ e dal rumore di processo w, a media nulla, in modo da usare la funzione

$$\boldsymbol{\mathcal{X}}^- = f(\mathbf{x}_{k-1}^a, \mathbf{u}_k) \tag{2.120}$$

di aggiornamento dello stato che permetta di tener conto in maniera non lineare e non additiva anche del contributo del rumore di processo. Allo stesso modo definiamo la matrice di covarianza aumentata come:

$$\mathbf{P}_{\mathbf{x}}^a = \begin{bmatrix} \mathbf{P}_{\mathbf{x}} & 0 \\ 0 & \mathbf{Q} \end{bmatrix} \tag{2.121}$$

Nel caso in cui il rumore di processo sia additivo il sistema ridiventa simile a quello di Kalman lineare nella forma

$$\mathbf{P}_k^- = \sum_{i=0}^{2n} w_i^c (\boldsymbol{\mathcal{X}}_i^- - \bar{\boldsymbol{\mathcal{X}}}_i^-)(\boldsymbol{\mathcal{X}}_i^- - \bar{\boldsymbol{\mathcal{X}}}_i^-)^\top + \mathbf{Q}_k \tag{2.122}$$

Dai *sigma points* $\boldsymbol{\mathcal{X}}_i^-$, proiettati attraverso f e rappresentanti la distribuzione dello stato *a priori*, è possibile generare altri punti sigma in modo da ottenere la stima dell'osservazione *a priori*:

$$\boldsymbol{\mathcal{Z}}_i = h(\boldsymbol{\mathcal{X}}_i^-) \tag{2.123}$$

con cui calcolare il valore più probabile dell'osservazione $\hat{\mathbf{z}}$ pesando i risultati $\boldsymbol{\mathcal{Z}}_i$ con i pesi dei *sigma point* associati come nell'equazione (2.44). Anche in questo caso il rumore di osservazione può essere inserito come stato aumentato o, se supposto additivo e indipendente, può venire sommato alla matrice di covarianza.

Attraverso la conoscenza dei punti sigma $\boldsymbol{\mathcal{X}}_i^-$ e $\boldsymbol{\mathcal{Z}}_i$ è possibile ottenere facilmente la covarianza $\mathrm{cov}(\boldsymbol{\mathcal{Z}})$ e anche la cross-covarianza $\mathrm{cov}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}})$ generalizzando l'equazione (2.44):

$$\mathrm{cov}\left(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}}\right) \approx \sum_{i=0}^{2n} w_i^c (\boldsymbol{\mathcal{X}}_i - \bar{\mathbf{x}})(\boldsymbol{\mathcal{Z}}_i - \bar{\mathbf{z}})^\top \tag{2.124}$$

Data la conoscenza della covarianza $\mathrm{cov}(\boldsymbol{\mathcal{Z}})$ e la cross-covarianza $\mathrm{cov}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}})$ il guadagno di Kalman *sigma-point* diventa esattamente come quello espresso dall'equazione (2.109) e l'aggiornamento della covarianza $\mathbf{P}_k$ segue l'equazione (2.110).

2.12.6 IEKF e ISPKF

Il filtro di Kalman esteso fa uso dello Jacobiano della funzione di osservazione h centrato in $\hat{\mathbf{x}}^-$, stato *a priori* e, grazie alla conoscenza dell'osservazione, permette di ottenere la stima dello stato *a posteriori*.

Di fatto questo procedimento è esattamente una singola iterazione del metodo di Gauss-Newton.

È possibile aumentare le iterazioni in modo da ottenere la classe dei filtri di Kalman iterativi, i quali normalmente mostrano prestazioni nettamente migliori della loro controparte non iterativa.

L'unica differenza rispetto ai rispettivi filtri non iterativi è nella parte di osservazione (cfr. equazione (2.119)), sostituita da iterazioni nella forma:

$$\mathbf{x}_{i+1} = \hat{\mathbf{x}} + \mathbf{K}(\mathbf{z} - h(\mathbf{x}_i) - \mathbf{H}_i(\hat{\mathbf{x}} - \mathbf{x}_i)) \tag{2.125}$$

con il guadagno $\mathbf{K}$ calcolato in maniera iterativa come

$$\mathbf{K} = \mathbf{P}\mathbf{H}_i^\top (\mathbf{H}_i \mathbf{P} \mathbf{H}_i^\top + \mathbf{R})^{-1} \tag{2.126}$$

e usando come valore iniziale per la minimizzazione, il valore $\mathbf{x}_0 = \hat{\mathbf{x}}^-$.

Il valore di $\mathbf{K}$, associato all'ultima iterazione, viene infine usato per aggiornare la matrice di covarianza di processo.

Lo stesso procedimento si può applicare al filtro SPKF per ottenere l'*Iterated Sigma Point Kalman Filter* [SSM06], dove l'iterazione per calcolare lo stato è nella forma

$$\mathbf{x}_{i+1} = \hat{\mathbf{x}} + \mathbf{K}\left(\mathbf{z} - h(\mathbf{x}_i) - \operatorname{cov}(\mathcal{X}, \mathcal{Z})^\top \mathbf{P}^{-1}(\hat{\mathbf{x}} - \mathbf{x}_i)\right) \tag{2.127}$$

2.12.7 Filtro di Kalman a miscela di Gaussiane

Il *Gaussian sum Kalman filters* (GS-KF) descrive un filtro bayesiano dove la distribuzione multimodale viene approssimata a una miscela di Gaussiane. Viene trattato allo stesso modo del filtro di Kalman lineare nel caso di trasformazioni lineari trattando ogni stato separatamente.

2.12.8 Particle Filter

Gli approcci lineari e quasi lineari proposti da Kalman possono essere usati in quei problemi dove lo stato è gaussiano o quasi gaussiano con distribuzione unimodale: la stima dello stato all'istante di tempo k è funzione diretta dell'unica stima dello stato all'istante di tempo $k-1$ e della covarianza di tale stima.

Quando è richiesto di ricavare la distribuzione di probabilità non gaussiana dello stato del sistema $p(x_k; u_{k-1}; z_k)$ all'istante di tempo k, funzione degli ingressi e delle osservazioni, gli approcci di tipo Kalman non sono più soddisfacenti.

Gli approcci *grid based* sono adatti a quei problemi, di fatto poco comuni, dove lo stato è discretizzabile e finito. Gli approcci *histogram based/occupacy grid* si adattano a una classe di problemi maggiore che però, a causa del campionamento uniforme dello stato, scalano molto male con l'aumentare delle dimensioni.

Si consideri nuovamente il risultato espresso dall'equazione (2.4): per estrarre una generica statistica $h(\cdot)$ (per esempio media, o varianza) da una distribuzione di probabilità $p(x)$, si fa uso dell'espressione

$$\bar{h} \stackrel{def}{=} \int_X h(x)p(x)dx \tag{2.128}$$

Nel caso in cui tale stima non si possa ottenere per via analitica, è comunque possibile ricavarla per via indiretta, attraverso l'analisi di x_i campioni indipendenti, con $1 \leq i \leq N$, estratti casualmente con distribuzione esattamente p.

Dati i campioni x_i generati in questo modo, la stima Monte Carlo di $h(\cdot)$ è data da

$$\bar{h} \approx \frac{1}{N} \sum_{i=1}^{N} h(x_i) \tag{2.129}$$

Monte Carlo non risolve tutti i problemi né suggerisce come ottenere i campioni casuali in maniera efficiente. Il problema diventa sensibile nei casi multidimensionali dove le aree in cui la probabilità assume valori significativi sono estremamente esigue. L'obiettivo che si pone infatti l'*Important Sampling* (IS) è campionare la distribuzione $p(x)$ in regioni "importanti" in modo da massimizzare l'efficienza computazionale.

L'idea dell'*Important Sampling* è quella di prendere una più semplice distribuzione $q(x)$ (*Importance density*), al posto della vera $p(x)$ normalmente difficile da campionare (o da riprodurre), effettuando la sostituzione

$$\int_X h(x)p(x)dx = \int_X h(x)\frac{p(x)}{q(x)}q(x)dx = \int_X h(x)w(x)q(x)dx$$

avendo introdotto il sistema di pesi $w(x)$. Attraverso l'uso di adeguati pesi pertanto è possibile modificare l'equazione (2.129) in

$$\bar{h} \approx \frac{1}{N} \sum_{i=1}^{N} w_i h(x_i) \tag{2.130}$$

dove $w_i \propto W_i = p(x_i)/q(x_i)$ rappresenta un peso correttivo, fattore di importanza (*important weights*), per convertire la distribuzione di supporto q a quella reale p. I pesi W_i devono essere normalizzati

$$w_i = \frac{W_i}{\sum W_i} \tag{2.131}$$

per poter essere utilizzati.

Più la distribuzione $q(x)$ è simile alla $p(x)$, più la stima risulterà corretta. D'altra parte la distribuzione $q(x)$ deve essere molto semplice da campionare, scegliendo per esempio la distribuzione uniforme o gaussiana.

Data la conoscenza dei filtri bayesiani e con le tecniche Monte Carlo è possibile affrontare la teoria dei filtri particellari. Lo stato all'istante k è rappresentato da un insieme di campioni (*particles*) e ogni campione è un ipotesi dello stato da vagliare. Si può parlare di una serie di particelle ottenute *a priori* dell'osservazione, applicando l'equazione (2.130) alla funzione di evoluzione dello stato.

Se si applica direttamente la teoria bayesiana ai campioni della distribuzione stimata è possibile modificare i pesi w_i associati ai campioni usando contemporaneamente il modello del sistema e della percezione (*Sequential Important Sampling*):

$$w_{k,i} \propto w_{k-1,i} \frac{p(z_k|x_{k,i})p(x_{k,i}|x_{k-1,i})}{q(x_{k,i}|x_{k-1,i}, z_k)} \tag{2.132}$$

In questo modo i campioni iniziali sono sempre gli stessi, ma cambiano solo i pesi w_i associati.

Quando possibile è conveniente usare come *Important density* la distribuzione *a priori*

$$q(x_{k,i}|x_{k-1,i}, z_k) = p(x_{k,i}|x_{k-1,i}) \tag{2.133}$$

in modo che, introdotta in (2.132), si ottenga

$$w_{k,i} \propto w_{k-1,i} p(z_k|x_{k,i}) \tag{2.134}$$

Il problema dell'approccio SIS è che dopo poche iterazioni solo alcune particelle avranno il fattore peso non trascurabile (*weight degeneracy*).

BootStrap/Sequential Importance Resampling

Una soluzione più semplice è la *Sequential Important Resampling* dove i pesi non dipendono dalle iterazioni precedenti ma sono invece i campioni a cambiare, in seguito a una fase di *resampling*.

La fase di ricampionamento consiste nel generare un nuovo insieme di particelle x' ricampionando N_s volte una versione discreta approssimata di $p(\mathbf{x}_k|\mathbf{z}_k)$ data da

$$p(\mathbf{x}_k|\mathbf{z}_k) \approx \sum_{i=1}^{N_s} w_{k,i} \delta(\mathbf{x}_k - \mathbf{x}_{k,i}) \tag{2.135}$$

avendo definito

$$w_{k,i} \propto p(\mathbf{z}_k|\mathbf{x}_k) \tag{2.136}$$

I filtri SIR non evitano il caso degenere (di fatto anzi eliminano definitivamente le particelle poco probabili), tuttavia portano a un notevole risparmio computazionale e concentrano la ricerca della soluzione intorno agli stati più probabili.

Esistono svariati algoritmi per eseguire il ricampionamento. Un'elenco, non sicuramente esaustivo, di tali algoritmi è: *Simple Random Resampling*, *Roulette Wheel / Fitness proportionate selection*, *Stochastic universal sampling*, *Multinomial Resamping*, *Residual Resampling*, *Stratified Resampling*, *Systematic Resampling*.

2.12.9 Stima dei Parametri

Kalman, in tutte le sue varianti, è classicamente visto come filtro o stimatore di uno stato. Tuttavia è largamente diffuso, principalmente in *machine learning*, l'utilizzo di queste tecniche per stimare i parametri di un modello (il meta-modello):

$$\mathbf{y}_k = f(\mathbf{x_k}, \boldsymbol{\beta}) \tag{2.137}$$

dove $\mathbf{y}_k$ sono le uscite del sistema, $\mathbf{x_k}$ gli ingressi e f una funzione basata sui parametri $\boldsymbol{\beta}$ da stimare. Il concetto di addestramento, o *fitting*, del modello consiste nel determinare i parametri $\boldsymbol{\beta}$.

Kalman permette di determinare i parametri, eventualmente variabili, del modello usando come stato da determinare proprio $\boldsymbol{\beta}$ in modo da ottenere un sistema iterativo del tipo

$$\begin{cases} \boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \mathbf{w}_k \\ \mathbf{y}_k = f(\mathbf{x}_k, \boldsymbol{\beta}_k) \end{cases} \tag{2.138}$$

dove il rumore opzionale $\mathbf{w}_k$ viene usato per modellare eventuali variazioni del modello nel tempo: la scelta della varianza di $\mathbf{w}$ determina la reattività alle variazioni dei parametri del modello.

2.12.10 Filtro alfa beta

L'*alpha-beta filter* si può vedere come una versione semplificata del filtro di Kalman dove lo stato è rappresentato da sole due variabili di cui una è l'integrale dell'altra. Da una semplice similitudine con sistemi fisici possiamo chiamare queste variabili posizione $\mathbf{x}$ e velocità $\mathbf{v}$. Se si suppone che la velocità rimanga costante nell'intervallo di tempo piccolo ΔT si ha la stima *a priori* (predizione) della posizione all'istante k come

$$\hat{\mathbf{x}}_k^- = \hat{\mathbf{x}}_{k-1} + \Delta T \mathbf{v}_{k-1} \tag{2.139}$$

mentre la velocità viene sempre ritenuta costante:

$$\hat{\mathbf{v}}_k^- = \hat{\mathbf{v}_{k-1}} \tag{2.140}$$

L'uscita tuttavia è affetta da rumore e il valore osservato $\mathbf{x}_k$ è differente dal valore predetto $\hat{\mathbf{x}}_k^-$. Questo errore di predizione $\mathbf{r}$ è chiamato residuo (stima dell'errore a posteriori):

$$\mathbf{r}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k^- \tag{2.141}$$

Definiamo due parametri α e β in modo da ottenere la stima a posteriori come

$$\begin{cases} \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \alpha \mathbf{r}_k \\ \hat{\mathbf{v}}_k &= \hat{\mathbf{v}}_k^- + \beta \frac{\mathbf{r}_k}{\Delta T} \end{cases} \tag{2.142}$$

In questo modo si ottiene un osservatore asintotico delle variabili posizione e velocità. A differenza del filtro di Kalman, il filtro alfa-beta è un filtro subottimo dove i parametri α e β sono tarati per via sperimentale senza nessun riscontro statistico. Questo approccio è solitamente avvallato dal fatto che anche nel filtro di Kalman a volte è necessario imporre le matrici del rumore per via totalmente empirica.

Capitolo 3

Metodi di Regressione e Ottimizzazione per l'Analisi di Modelli

Uno dei problemi più diffusi all'interno della visione artificiale (e in generale all'interno della teoria dell'informazione) è quello di far adattare un insieme di misure affette da rumore (per esempio i *pixel* di un'immagine) a un modello predefinito.

Oltre alla presenza di rumore, che potrebbe essere sia gaussiano bianco ma potenzialmente di qualunque distribuzione statistica, c'è da considerare il problema dell'eventuale presenza di *outlier*, termine utilizzato in statistica per indicare dati troppo distanti dal modello per farne effettivamente parte.

In questo capitolo vengono presentate sia diverse tecniche regressive volte a ricavare i parametri $\boldsymbol{\beta}$ di un modello stazionario dato un insieme di dati affetti da rumore sia tecniche per individuare e rimuovere gli *outlier* dai dati in ingresso.

Nel capitolo successivo verranno presentate invece tecniche di "regressione" più legate al tema della classificazione.

Per stimare i parametri di un modello alcune tecniche presenti in letteratura sono le seguenti:

Least Squares Fitting Se i dati sono tutti *inliers*, non ci sono *outliers* e l'unico disturbo è rumore additivo gaussiano bianco, la regressione ai minimi quadrati è la tecnica ottima (sezione 3.2);

M-Estimator La presenza anche di pochi *outlier* sposta di molto il modello in quanto gli errori vengono pesati al quadrato [Hub96]: pesare in maniera non quadratica i punti lontani del modello stimato produce miglioramenti nella stima stessa (sezione 3.8);

IRLS *iteratively reweighted least squares* viene usata quando gli *outliers* sono molto distanti dal modello e in bassa quantità: in questa condizione si può eseguire una regressione iterativa (sezione 3.9), dove a ogni ciclo i punti con errore troppo elevato vengono rimossi (*ILS*) o pesati in maniera differente (*IRLS*);

Hough Se i dati in ingresso sono sia affetti da errore che da molti *outliers* e potenzialmente c'è presenza di una distribuzione multimodale, ma con il modello formato da pochi parametri, la trasformata di Hough [Hou59] permette di ottenere il modello più diffuso dal punto di vista statistico (sezione 3.11);

RANSAC Se gli *outliers* sono comparabili in numero con gli *inliers* e il rumore è molto basso (rispetto alla posizione degli *outliers*), il *RANdom SAmpling and Consensus* [FB87] permette di ottenere il miglior modello presente sulla scena (sezione 3.12);

LMedS Il *Least Median of Squares* è un algoritmo, simile a RANSAC, che ordina i punti in base alla distanza del modello generato casualmente e sceglie fra tutti il modello con mediana dell'errore minore [Rou84] (sezione 3.12.2);

Kalman È possibile infine usare un filtro di Kalman per ricavare i parametri di un modello (vedi 2.12.9) quando tale informazione è richiesta a tempo di esecuzione.

Solamente RANSAC e la Trasformata di Hough permettono di gestire ottimamente il caso in cui nella misura siano presenti due o più distribuzioni che contemporaneamente si avvicinano al modello.

Nulla infine impedisce di usare tecniche miste, per esempio un Hough abbastanza grossolano (pertanto veloce e con basso impatto in termini di memoria) per rimuovere gli *outliers* e successivamente una regressione ai minimi quadrati per ottenere i parametri del modello in maniera più precisa.

3.1 Il limite di Cramer-Rao

Il limite di Cramer-Rao (*Cramer-Rao Lower Bound CRLB*) stabilisce un limite inferiore per la varianza di ogni stimatore corretto del parametro θ (per mantenere una simbologia comune con la letteratura, *beta* nel nostro caso).

Sia X una variabile aleatoria multidimensionale e θ un parametro deterministico sconosciuto. Sia $f_x^{\vartheta}(X)$ la densità di probabilità di X dato θ. Assumiamo che tale densità di probabilità esista e sia due volte differenziabile rispetto a θ.

Teorema 1 (Disuguaglianza di Cramer-Rao) *Sia $T(\cdot)$ uno stimatore corretto del parametro scalare ϑ, e si supponga che lo spazio delle osservazioni X sia indipendente da θ. Allora (sotto alcune ipotesi di regolarità...)*

$$E^{\theta}\left[(T(X)-\theta)^2\right] \geq [I_n(\theta)]^{-1} \tag{3.1}$$

dove $I_n(\theta) = E^{\theta}\left[\left(\frac{\partial \ln f_x^{\theta}(X)}{\partial \theta}\right)^2\right]$ *(quantità di Informazione di Fisher).*

Siccome il parametro θ non è conosciuto il teorema di Cramer-Rao permette solo di capire se lo stimatore è ottimo o meno.

3.2 Regressione ai minimi quadrati

Esaminiamo per primo il caso più diffuso in applicazioni reali quando il rumore sulle osservazioni è di tipo additivo gaussiano bianco.

Sia pertanto

$$y = f(\mathbf{x}, \boldsymbol{\beta}) + \varepsilon \tag{3.2}$$

funzione, in generale non lineare, di alcuni parametri $\boldsymbol{\beta}$ e di alcuni ingressi $\mathbf{x}$ a cui viene sommato del rumore additivo, gaussiano, a media nulla e varianza σ. Per poter stimare in maniera robusta i parametri, il numero di campioni in ingresso $\mathbf{x} = \{\mathbf{x}_1 \dots \mathbf{x}_n\}$ deve essere elevato, molto di più dei parametri.

Si può pensare che la funzione dei parametri non sia la stessa per tutti i campioni ma potrebbero essercene di differenti, osservando di fatto quantità differenti, funzione sempre dei medesimi parametri $\boldsymbol{\beta}$. In tal caso l'equazione (3.2) può venire generalizzata come

$$y_i = f_i(\boldsymbol{\beta}) + \varepsilon_i \tag{3.3}$$

avendo sottinteso con il pedice i sia il tipo di funzione sia l'i-esimo campione in ingresso (di fatto un parametro costante della funzione).

Si introduce il vettore $\mathbf{r}$ definito come

$$r_i = y_i - f_i(\boldsymbol{\beta}) \tag{3.4}$$

contenete il residuo associato all'osservazione i-esima (o alla funzione i-esima). r_i è funzione di $\boldsymbol{\beta}$ tanto come f_i e ne condivide le derivate (a meno di un segno con questo formalismo).

Per ottenere uno stimatore a massima verosimiglianza, la quantità da minimizzare è la *negative log likelihood* (sezione 2.8) della funzione (3.2). Nel caso di rumore gaussiano la funzione di verosimiglianza si scrive infatti come

$$\mathcal{L}(r_i|\boldsymbol{\beta}, \sigma) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{r_i^2}{2\sigma_i^2}} \tag{3.5}$$

nel caso di osservazioni indipendenti. Applicando alla funzione di verosimiglianza la definizione di *negative log likelihood* si ottiene che nel caso di rumore gaussiano lo stimatore alla massima verosimiglianza è il metodo dei minimi quadrati.

La regressione ai minimi quadrati è una tecnica di ottimizzazione standard per sistemi sovradimensionati che individua i parametri $\boldsymbol{\beta} = (\beta_1, \dots, \beta_m)$ di una funzione $f(\mathbf{x}, \boldsymbol{\beta}) : \mathbb{R}^m \mapsto \mathbb{R}^n$ che minimizzano un errore S calcolato come somma del quadrato (*Sum Of Squared Error*) dei residui r_i su un insieme di n osservazioni $y_1 \dots y_n$:

$$S(\boldsymbol{\beta}) = SSE(\boldsymbol{\beta}) = \mathbf{r} \cdot \mathbf{r} = \sum_{i=1}^{n} \|r_i\|^2 = \sum_{i=1}^{n} \|y_i - f_i(\boldsymbol{\beta})\|^2 \tag{3.6}$$

$S(\boldsymbol{\beta})$ è definito come *residual sum of squares* o alternativamente come *expected squared error*.

$S : \mathbb{R}^m \mapsto \mathbb{R}$ è una funzione che viene analizzata, al variare dei parametri $\boldsymbol{\beta} \in \mathbb{R}^m$, per cercare il suo valor minimo

$$\beta^+ = \arg\min_{\beta} S(\beta) \tag{3.7}$$

Per questa ragione viene chiamata *funzione obiettivo* o *funzione costo*. Un minimo ottenuto attraverso un procedimento come quello descritto dall'equazione (3.7) viene definito *minimo globale*.

Un minimo globale è difficile, dal punto di vista prettamente computazionale, da individuare e normalmente si possono sfruttare tecniche per individuare solamente i minimi locali.

Sia pertanto $S(\boldsymbol{\beta})$[1] differenziabile, ovvero f differenziabile. La condizione necessaria che $\boldsymbol{\beta}$ sia un minimo è che, in quel punto dello spazio dei parametri, il gradiente di $S(\boldsymbol{\beta})$ si annulli, ovvero

$$\frac{\partial S(\boldsymbol{\beta})}{\partial \beta_j} = 2\mathbf{J}^{\top}\mathbf{r} = -2\sum_{i=1}^{n} r_i \frac{\partial f_i(\boldsymbol{\beta})}{\partial \beta_j} = 0 \qquad j = 1, \dots, m \tag{3.8}$$

[1] In letteratura la funzione S viene spesso codificata con un fattore $1/2$ di scala per rendere il gradiente di S non viziato dal fattore 2 ed il segno concorde con f per semplificare la notazione.

Condizione sufficiente che un punto stazionario ($S'(\boldsymbol{\beta}) = 0$) sia un minimo è che $S''(\boldsymbol{\beta})$ (l'Hessiana) sia definita positiva. Chiaramente l'esistenza del minimo locale garantisce solo che esiste un intorno δ di $\boldsymbol{\beta}$ tale che la funzione $S(\boldsymbol{\beta} + \delta) \geq S(\boldsymbol{\beta})$.

Tutta la discussione affrontata fin ora ha come ipotesi che il rumore sia additivo ε con varianza costante tra tutti i campioni (*homoscedasticity*). Nel caso in cui il rumore di misura sia comunque gaussiano additivo a media nulla ma con varianza non costante, ogni singola osservazione y_i è una variabile aleatoria indipendente a cui è associata la varianza σ_i^2. Intuitivamente si capisce che la regressione ottima in questo caso dovrà pesare di più i campioni con varianza bassa mentre dovranno essere pesati meno i campioni con varianza elevata. Per ottenere questo risultato si fa uso di una normalizzazione, simile a quella mostrata in sezione 2.4.1 e diretta conseguenza della *likelihood* di equazione (3.5), e pertanto non si deve più minimizzare la semplice somma dei residui al quadrato, ma piuttosto la somma *pesata* dei residui:

$$\chi^2 = \sum_{i=1}^{n} \frac{\|r_i\|^2}{\sigma_i} \tag{3.9}$$

La funzione costo, ora somma di una variabile aleatoria di varianza unitaria al quadrato, diventa una distribuzione chi-quadro e per questo motivo viene indicata come χ^2. Il minimo di questa funzione costo coincide con quello ottenuto in precedenza dai minimi quadrati quando la varianza è invece costante. La condizione (3.8) per ottenere il minimo si modifica anch'essa di conseguenza:

$$\sum_{i=1}^{n} \frac{r_i}{\sigma_i} \frac{\partial f_i(\boldsymbol{\beta})}{\partial \beta_j} = 0 \qquad j = 1, \ldots, m \tag{3.10}$$

Generalizzando ulteriormente questo concetto, quando sull'osservazione è presente del rumore gaussiano con matrice di covarianza nota $\boldsymbol{\Sigma}$, la *Weighted Sum of Squared Error* (*WSSE*) si può scrivere infine come

$$\chi^2 = \mathbf{r}^\top \boldsymbol{\Sigma}^{-1} \mathbf{r} \tag{3.11}$$

È da notare che questa formulazione della funzione di costo equivale a quella di equazione (3.6) dove però, invece della distanza euclidea, viene usata la distanza di Mahalanobis (sezione 2.4).

Qualunque *Weighted Least Squares* può essere ricondotto a un problema non pesato $\boldsymbol{\Sigma} = I$ premoltiplicando i residui $\mathbf{r}$ (e di conseguenza le derivate) per una matrice $\mathbf{L}^\top$ tale che $\boldsymbol{\Sigma}^{-1} = \mathbf{L}\mathbf{L}^\top$, usando per esempio una decomposizione di Cholesky nel caso in cui tale matrice non sia diagonale.

Tutti questi stimatori, che tengono conto della varianza dell'osservazione, coincidono con il *negative log likelihood* per la variabile $\mathbf{y}$ perturbata da rumore gaussiano di media zero e covarianza $\boldsymbol{\Sigma}$.

3.2.1 Regressione lineare ai minimi quadrati

Quando f è una funzione lineare rispetto ai parametri $\boldsymbol{\beta}$ si parla di regressione lineare ai minimi quadrati (*Linear Least Squares* o *Ordinary Least Squares OLS*). Tale funzione può essere rappresentata nella forma di sistema lineare

$$y_i = \mathbf{x}_i \boldsymbol{\beta} + \varepsilon_i \tag{3.12}$$

dove $\boldsymbol{\beta}$ sono i parametri sconosciuti da ricavare e ε_i è rumore additivo gaussiano bianco a media nulla. I parametri $\boldsymbol{\beta}$ sono i coefficienti della regressione: permettono di misurare l'associazione tra la variabile $\mathbf{x}$ e la variabile y.

Ogni osservazione è un vincolo e tutti i singoli vincoli possono essere raccolti in forma matriciale

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{3.13}$$

$\mathbf{y} \in \mathbb{R}^n$ è il vettore delle *risposte* (variabili dipendenti), la matrice $\mathbf{X} \in \mathbb{R}^{n \times m}$ che raccoglie le variabili indipendenti (*explanatory variables*) viene chiamata *design matrix*, e infine $\boldsymbol{\varepsilon}$ è il vettore del rumore additivo a media nulla $\mathrm{E}[\boldsymbol{\varepsilon}] = 0$ e varianza $\boldsymbol{\Sigma}$. Il vettore dei parametri $\boldsymbol{\beta}$ è chiamato *Linear Projection Coefficient* o *Linear Predictor*. La variabile casuale $\mathbf{y}$ è pertanto formata da una parte *deterministica* e da una parte *stocastica*.

L'obiettivo è quello di trovare l'iperpiano $\boldsymbol{\beta}$ in m dimensioni che meglio si adatta ai dati $(\mathbf{y}, \mathbf{X})$.

Il valore $\boldsymbol{\beta}$ che minimizza la funzione costo definita in equazione (3.6), limitatamente al caso di rumore sull'osservazione a valor medio nullo e varianza costante fra tutti i campioni, di fatto è il miglior stimatore lineare che minimizza la varianza (*Best Linear Unbiased Estimator BLUE*).

Definizione 11 *Il Best Linear Unbiased Estimate (BLUE) di un parametro $\boldsymbol{\beta}$ basato su un set di dati Y è*

1. *una funzione lineare di Y, in modo che lo stimatore possa essere scritto come $\hat{\boldsymbol{\beta}} = \mathbf{A}Y$;*
2. *deve essere* unbiased *($E[\mathbf{A}Y] = 0$),*
3. *fra tutti gli stimatori lineari possibili è quello che produce la varianza minore.*

Il teorema di Gauss-Markov dimostra che uno stimatore ai minimi quadrati è la miglior scelta tra tutti gli stimatori a minima varianza *BLUE* quando la varianza sull'osservazione è costante (*homoscedastic*).

La miglior stima ai minimi quadrati $\hat{\boldsymbol{\beta}}$ che minimizza la somma dei residui è la soluzione del problema lineare

$$\hat{\boldsymbol{\beta}} = \arg\min_{\mathbf{b}} \|\boldsymbol{\varepsilon}\|^2 = \arg\min_{\mathbf{b}} \sum \|\mathbf{y}_i - \mathbf{x}_i\mathbf{b}\|^2 = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} \tag{3.14}$$

Lo stesso risultato era già pervenuto nella sezione 1.1 riguardante la pseudoinversa di una matrice: una decomposizione SVD della matrice $\mathbf{X}$ ritorna anche la soluzione migliore dal punto di vista della propagazione degli errori di calcolo.

La matrice $\mathbf{P}$, definita come

$$\mathbf{P} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top \tag{3.15}$$

è una matrice di proiezione (*projection matrix*) che trasforma le uscite (*response vector*) $\mathbf{y}$ nella loro stima $\hat{\mathbf{y}}$ (stima dell'osservazione senza rumore):

$$\mathbf{P}\mathbf{y}_i = \mathbf{x}_i\hat{\boldsymbol{\beta}} = \hat{\mathbf{y}}_i \tag{3.16}$$

Grazie a questa proprietà, $\mathbf{P}$ è chiamata *hat matrix*.

Nel caso di rumore a varianza non costante tra i campioni osservati (*heteroscedastic*) la regressione a minimi quadrati pesata è la scelta *BLUE*

$$w_i = \frac{1}{\sigma_i} \tag{3.17}$$

con $w_i > 0$ che tengono conto le varie incertezze legate ad ogni osservazione y_i così che $1/w_i$ sia la deviazione standard della misura i-esima. Inseriti i pesi w_i in una matrice diagonale $\mathbf{W}$ si ottiene un nuovo sistema lineare dove ogni riga di fatto ha la medesima varianza di osservazione. La soluzione che minimizza $\boldsymbol{\varepsilon}$, può sempre essere espressa come

$$\hat{\boldsymbol{\beta}} = (\mathbf{W}\mathbf{X})^+\mathbf{W}\mathbf{y} \tag{3.18}$$

con $\mathbf{W} = \boldsymbol{\Sigma}^{-1}$.

Generalizzando ulteriormente, nel caso di rumore con varianza non costante tra i campioni osservati e tra loro correlato, la miglior stima *BLUE* nel caso lineare deve tenere conto della covarianza del rumore $\boldsymbol{\Sigma}$:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\boldsymbol{\Sigma}^{-1}\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{\Sigma}^{-1}\mathbf{y} \tag{3.19}$$

Tale estimatore è chiamato *Generalized Least Squares* (*GLS*).

Tale sistema minimizza la varianza

$$Var[\hat{\boldsymbol{\beta}}_{GLS}] = (\mathbf{X}^\top\boldsymbol{\Sigma}^{-1}\mathbf{X})^{-1} \tag{3.20}$$

3.2.2 Total Least Squares

Estendiamo ora il problema lineare $\mathbf{A}\mathbf{x} = \mathbf{b} + \delta$ al caso più generale dove anche la matrice dei coefficienti $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{E}$ è perturbata (*Errors-In-Variables model EIV* [VHV91]). Questo tipo di problema di regressione ai minimi quadrati è chiamato *Total Least squares* (TLS).

La soluzione del sistema perturbato

$$(\mathbf{A} + \mathbf{E})\mathbf{x} = \mathbf{b} + \boldsymbol{\delta} \tag{3.21}$$

corrisponde a trovare la soluzione $\mathbf{x}$ che minimizzi la norma di Frobenius $\|(\mathbf{E}\ \boldsymbol{\delta})\|_F$, soggetta al vincolo (3.21). Con il TLS classico tutte le colonne della matrice dei dati contengono rumore. Se alcune colonne sono senza errori, allora la soluzione è chiamata *mixed TLS-LS*.

Il sistema (3.21) può essere riscritto come

$$([\mathbf{A}|\mathbf{b}] + [\mathbf{E}|\boldsymbol{\delta}]) \begin{bmatrix} \mathbf{x} \\ -1 \end{bmatrix} = \mathbf{0} \tag{3.22}$$

Sfruttando la decomposizione SVD e il teorema di Eckart-Young-Mirsky (la matrice formata dai primi n termini della decomposizione SVD è la matrice che meglio approssima la matrice $\mathbf{Z}$ sotto la norma di Frobenius) è possibile trovare la soluzione del problema (3.21). Sia pertanto

$$\mathbf{C} := [\mathbf{A}|\mathbf{b}] = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top \tag{3.23}$$

la Decomposizione a Valori Singolari della matrice $\mathbf{C}$, dove $\boldsymbol{\Sigma} = \text{diag}\,(\sigma_1 \dots \sigma_{n+d})$. La soluzione *Total Least squares*, se esiste, si scrive come

$$\hat{\mathbf{X}}_{tls} = -\mathbf{V}_{12}\mathbf{V}_{22}^{-1} \tag{3.24}$$

'avendo partizionato

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}_{11} & \mathbf{V}_{12} \\ \mathbf{V}_{21} & \mathbf{V}_{22} \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_2 \end{bmatrix} \tag{3.25}$$

'ed è possibile ottenere la miglior stima di $\hat{\mathbf{C}}$ come

$$\hat{\mathbf{C}}_{tls} = \mathbf{C} + \Delta\mathbf{C}_{tls} = \mathbf{U}\,\text{diag}\,(\boldsymbol{\Sigma}_1, 0)\,\mathbf{V}^\top \tag{3.26}$$

3.3 Metodi di ottimizzazione

Consideriamo ora un generico problema di modellizzazione (ottimizzazione) di funzione non vincolata, applicabile per esempio a problemi di classificazione nell'ambito della visione artificiale. Le considerazioni espresse in questa sezione si applicano al caso dei minimi quadrati ma possono essere estese a una generica *loss function*.

Sia $\mathbf{z}$ l'insieme dei dati coinvolti nell'operazione di modellizzazione formati da una coppia $(\mathbf{x}_i, y_i)$ composizione da un ingresso arbitrario $\mathbf{x}_i$ e dall'uscita y_i. Sia $\ell(\hat{y}, y)$ la funzione costo (*loss function*) che ritorna la bontà della stima su y. L'obiettivo è trovare i pesi $\boldsymbol{\beta}$ che parametrizzano la funzione $f(\mathbf{x}; \boldsymbol{\beta})$ che minimizzano una funzione costo $S(\boldsymbol{\beta})$

$$S(\boldsymbol{\beta}) = \int \ell(\mathbf{z}; \boldsymbol{\beta}) dP(\mathbf{z}) \qquad S(\boldsymbol{\beta}) = \sum_{i=1}^{n} \ell_i(\boldsymbol{\beta}) \tag{3.27}$$

sia nel caso continuo che nel caso discreto, avendo definito $\ell_i(\boldsymbol{\beta}) = \ell(f_i(\mathbf{x}_i; \boldsymbol{\beta}), y_i)$. Per semplicità si farà sempre riferimento al secondo caso, quello discreto, per descrivere la funzione costo.

Nel caso di errore additivo gaussiano normale, lo stimatore a massima verosimiglianza è la *loss function* quadratica di equazione (3.6):

$$\ell_i(\boldsymbol{\beta}) = r_i^2(\boldsymbol{\beta}) = (y_i - f_i(\mathbf{x}_i; \boldsymbol{\beta}))^2 \tag{3.28}$$

In applicazioni pratiche non è quasi mai possibile ottenere il minimo della funzione in forma chiusa e pertanto bisogna fare ricorso ad opportuni metodi iterativi, i quali, partendo da uno stato iniziale e muovendosi lungo opportune direzioni $\boldsymbol{\delta}$ si avvicinano man mano al minimo della funzione obiettivo.

3.3.1 Metodo di Newton-Raphson

Il problema di trovare i minimi di una funzione può essere ricondotto al problema di trovare gli zeri di una funzione, nel caso specifico la derivata prima della funzione costo S.

Sia pertanto $\mathbf{g} : \mathbb{R}^m \mapsto \mathbb{R}^n$ una funzione multivariata derivabile di cui sia richiesto di trovare

$$\mathbf{g}(\mathbf{x}) = \mathbf{0} \tag{3.29}$$

Espandendo in serie di Taylor la funzione $\mathbf{g}$, localmente intorno a un punto $\mathbf{x}$ opportuno, si ottiene

$$\mathbf{g}(\mathbf{x} + \boldsymbol{\delta}) = \mathbf{g}(\mathbf{x}) + \mathbf{J}_g \boldsymbol{\delta} + O(\boldsymbol{\delta}^2) \tag{3.30}$$

dove $\mathbf{J}_g$ è la matrice $n \times m$ Jacobiano della funzione $\mathbf{g}$ calcolato in $\mathbf{x}$.

L'obiettivo è modificare il valore di $\mathbf{x}$ di un valore $\boldsymbol{\delta}$ in maniera tale che la funzione costo calcolata in $\mathbf{x}_t + \boldsymbol{\delta}$ sia esattamente zero. Ignorando i contributi di ordine di ordine superiore a $\boldsymbol{\delta}^2$, la stima del $\boldsymbol{\delta}$ che in prima approssimazione fa avvicinare a zero la funzione $\mathbf{g}$ è la soluzione del sistema lineare (3.30) con la condizione (3.29), ovvero

$$\mathbf{J}_g \boldsymbol{\delta} = -\mathbf{g}(\mathbf{x}) \tag{3.31}$$

sistema che, se $\mathbf{J}_g$ non ha deficienze di rango, è un semplice sistema lineare volendo anche sovradimensionato, che si può risolvere con una delle tecniche mostrate in sezione 1.1. L'idea dei metodi iterativi è quello di modificare il punto $\mathbf{x}_t$ della quantità $\boldsymbol{\delta}_t$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \boldsymbol{\delta}_t \tag{3.32}$$

per le iterazioni $t = 1, 2, \ldots$, così calcolata in modo da avvicinarsi progressivamente allo zero della funzione.

Nel caso di singola variabile $n = m = 1$ il metodo di Newton si riduce a

$$x_{t+1} = x_t - \frac{g(x)}{g'(x)} \tag{3.33}$$

In calcolo numerico questo è il cosiddetto metodo di Newton (o di Newton-Raphson) per trovare gli zeri di una funzione.

I punti di massimo e minimo di una funzione sono i punti per i quali si riesce ad annullare il gradiente. Si può pertanto applicare questa tecnica per trovare i massimi e minimi di una funzione $f(\mathbf{x}) : \mathbb{R}^m \mapsto \mathbb{R}$ definendo

$$\begin{aligned} \mathbf{g}(\mathbf{x}) &= \nabla f(\mathbf{x}) \\ \mathbf{J}_g(\mathbf{x}) &= \mathbf{H}_f(\mathbf{x}) \end{aligned} \tag{3.34}$$

dove $\nabla f(\mathbf{x})$ è la funzione gradiente $\mathbb{R}^m \mapsto \mathbb{R}^m$ mentre $\mathbf{H}_f(\mathbf{x})$ la matrice Hessiana $m \times m$, funzioni gradiente ed Hessiana di f calcolate in $\mathbf{x}$. La modifica del punto $\mathbf{x}$ di Newton diventa pertanto

$$\mathbf{H}_f(\mathbf{x}) \boldsymbol{\delta}_t = -\nabla f(\mathbf{x}) \tag{3.35}$$

Quando viene utilizzato per ottimizzazione il metodo di Newton approssima di fatto la funzione $f(\mathbf{x})$ nell'intorno di $\mathbf{x}$ con una quadrica. Se $f(\mathbf{x})$ è una funzione quadratica, la convergenza è garantita in un sola iterazione.

Ora, nel caso specifico dei metodi di ottimizzazione, la funzione $f(\mathbf{x})$ è la funzione costo $S(\boldsymbol{\beta})$. Pertanto, quando la matrice Hessiana di $S(\boldsymbol{\beta})$ è non singolare, si ottiene l'equazione di variazione dei parametri

$$\boldsymbol{\delta}_t = -\mathbf{H}_S^{-1}(\boldsymbol{\beta}_t)\nabla S(\boldsymbol{\beta}_t) \tag{3.36}$$

attraverso il metodo di ottimizzazione di Newton.

3.3.2 Discesa del Gradiente

L'algoritmo di discesa del gradiente (*gradient descent*, GD, o *steepest descent*) aggiorna i pesi $\boldsymbol{\beta}$ a ogni iterazione usando il gradiente (più precisamente, l'antigradiente) della funzione obiettivo $S(\boldsymbol{\beta})$:

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \gamma\nabla S(\boldsymbol{\beta}_t) = \boldsymbol{\beta}_t - \gamma\sum_{i=1}^{n}\nabla\ell_i(\boldsymbol{\beta}_t) \tag{3.37}$$

ovvero, definendo il passo di aggiornamento:

$$\boldsymbol{\delta}_t = -\gamma\sum_{i=1}^{n}\nabla\ell_i(\boldsymbol{\beta}_t) \tag{3.38}$$

dove γ è un fattore di ottimizzazione opportunamente scelto (in *machine learning* è chiamato *learning rate*). Sotto opportune assunzioni, se il punto di partenza è sufficientemente vicino alla soluzione e il valore di γ è abbastanza piccolo, il ritmo di convergenza ottenibile è praticamente lineare.

Poiché il parametro γ è scelto manualmente, l'approccio risulta empirico e dipendente dal problema, se non dall'esperienza dell'utilizzatore. Confrontando l'equazione (3.37) con l'equazione (3.36), si osserva che il metodo di Newton è di fatto un caso particolare di discesa del gradiente, in cui il parametro scalare γ viene sostituito da una matrice definita positiva $\boldsymbol{\Gamma}_t$, ottenuta come inversa dell'Hessiana nel punto corrente:

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \boldsymbol{\Gamma}_t\nabla S(\boldsymbol{\beta}_t) \tag{3.39}$$

La discesa del gradiente del secondo ordine corrisponde quindi all'algoritmo di Newton, che - sotto opportune ipotesi - garantisce una convergenza quadratica, in contrasto con la convergenza lineare della discesa del gradiente classica.

3.3.3 Discesa Stocastica del Gradiente

L'algoritmo di discesa stocastica del gradiente (*Stochastic Gradient Descent*, SGD) è una semplificazione della discesa del gradiente classica. Invece di calcolare il gradiente esatto della funzione obiettivo $S(\boldsymbol{\beta})$, ad ogni iterazione si utilizza il gradiente di un singolo campione ℓ_i scelto casualmente:

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \gamma_t\nabla\ell_i(\boldsymbol{\beta}_t) \tag{3.40}$$

SGD è garantito convergere su funzioni convexe definite su domini convessi, ma viene comunemente utilizzato anche in contesti non convessi, come l'addestramento di reti neurali.

Una variante pratica consiste nell'utilizzare un piccolo gruppo di campioni (*mini-batch*) per ogni aggiornamento, riducendo il rumore rispetto al singolo campione ma mantenendo una buona efficienza computazionale.

Per simulare un'inerzia nel cambiamento dei parametri, si introduce un termine α detto *momentum*:

$$\boldsymbol{\delta}_t = -\gamma_t\nabla S(\boldsymbol{\beta}_t) + \alpha\boldsymbol{\delta}_{t-1} \tag{3.41}$$

dove α è solitamente un valore piccolo (es. 0.05). Il *momentum* è una delle modifiche più semplici ed efficaci a SGD, utile per superare oscillazioni e rallentamenti in direzioni poco informative.

Oltre a SGD con momentum, esistono numerose varianti progettate per accelerare la convergenza e migliorare la stabilità dell'ottimizzazione. Una lista non esaustiva include:

- Nesterov Accelerated Gradient (NAG)
- AdaGrad (Adaptive Gradient)
- RMSProp (Root Mean Square Propagation)
- AdaDelta
- Adam (Adaptive Moment Estimation) 3.3.4
- AdaMax
- Momentum
- Resilient Propagation (RProp)

Una panoramica comparativa di questi algoritmi è disponibile in [Rud16].

3.3.4 Adam

Adam [KB14] (*Adaptive Moment Estimation*) è uno degli algoritmi di ottimizzazione più diffusi nel deep learning, grazie alla sua capacità di combinare i vantaggi di AdaGrad e RMSProp.

AdaGrad assegna un learning rate specifico a ciascun parametro, risultando efficace in presenza di gradienti sparsi. RMSProp, invece, adatta il learning rate in base alla magnitudo del gradiente, rendendolo adatto a scenari online e non stazionari.

Adam estende questi approcci introducendo una stima adattiva sia del momento di primo ordine (la media dei gradienti) che del momento di secondo ordine (la varianza). In particolare, per ogni modello $\boldsymbol{\beta}$, Adam mantiene due variabili:

- m_t: stima del momento di primo ordine (media dei gradienti);
- v_t: stima del momento di secondo ordine (varianza dei gradienti).

Queste stime vengono aggiornate iterativamente secondo:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla S(\boldsymbol{\beta}_t) \tag{3.42}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\left(\nabla S(\boldsymbol{\beta}_t)\right)^2 \tag{3.43}$$

dove β_1 e β_2 sono iperparametri (da non confondere con $\boldsymbol{\beta}$ che sono i parametri del modello) che controllano il decadimento esponenziale (tipicamente $\beta_1 = 0.9$, $\beta_2 = 0.999$).

Poiché m_t e v_t sono inizializzati a zero, le prime iterazioni risultano sottostimate. Per correggere questo bias, si calcolano:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{3.44}$$

L'aggiornamento dei parametri avviene quindi secondo:

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \gamma \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{3.45}$$

dove γ è il learning rate e ε è un termine di stabilizzazione numerica (tipicamente $\varepsilon = 10^{-8}$).

Adam è particolarmente efficace in scenari con dati rumorosi, gradienti sparsi o funzioni obiettivo non stazionarie. Grazie alla sua robustezza e semplicità d'uso, è spesso la scelta predefinita per l'ottimizzazione di reti neurali profonde.

3.3.5 Gauss-Newton

I metodi visti finora lasciano molta libertà nella scelta della *loss function*. Nei casi pratici in cui la funzione costo ℓ è quadratica, si possono introdurre ottimizzazioni ulteriori rispetto al metodo di Newton, evitando il gravoso calcolo dell'Hessiana.

In questo caso, la loss function assume la forma già vista in precedenza:

$$S(\boldsymbol{\beta}) = \frac{1}{2}\mathbf{r}^\top \mathbf{r} = \frac{1}{2}\sum_{i=1}^{n} r_i^2(\boldsymbol{\beta}) \tag{3.46}$$

Il termine $1/2$ serve per ottenere un'espressione del gradiente più compatta.

Con questa funzione costo, gradiente e Hessiana si scrivono come:

$$\begin{aligned} \nabla S(\boldsymbol{\beta}) &= \textstyle\sum_{i=1}^{n} r_i(\boldsymbol{\beta})\nabla r_i(\boldsymbol{\beta}) = \mathbf{J}_r^\top \mathbf{r} \\ \mathbf{H}_S(\boldsymbol{\beta}) &= \textstyle\sum_{i=1}^{n} \nabla r_i \nabla r_i^\top + \sum_{i=1}^{n} r_i \mathbf{H}_{r_i} = \mathbf{J}_r^\top \mathbf{J}_r + \sum_{i=1}^{n} r_i \mathbf{H}_{r_i} \end{aligned} \tag{3.47}$$

Quando i parametri sono vicini alla soluzione, il residuo è piccolo e l'Hessiana può essere approssimata dal primo termine:

$$\mathbf{H}_S(\boldsymbol{\beta}) \approx \mathbf{J}_r^\top \mathbf{J}_r \tag{3.48}$$

In queste condizioni, gradiente e Hessiana della funzione costo S dipendono solo dal Jacobiano delle funzioni $r_i(\boldsymbol{\beta})$. L'Hessiana così approssimata può essere inserita nell'equazione (3.35):

$$-\mathbf{J}_r^\top \mathbf{r} = \mathbf{H}_S \boldsymbol{\delta}_\beta \approx \mathbf{J}_r^\top \mathbf{J}_r \boldsymbol{\delta}_\beta \tag{3.49}$$

Come nel caso del metodo di Newton, si ottiene un problema di minimo lineare risolvibile tramite le *normal equations*:

$$\boldsymbol{\delta}_\beta = -\left(\mathbf{J}_r^\top \mathbf{J}_r\right)^{-1} \mathbf{J}_r^\top \mathbf{r} \tag{3.50}$$

Il significato delle *normal equations* è geometrico: il minimo si ottiene quando $\mathbf{J}\boldsymbol{\delta}_\beta - \mathbf{r}$ è ortogonale allo spazio colonne di $\mathbf{J}$.

Nel caso particolare in cui il residuo sia scritto come:

$$r_i = y_i - f_i(\mathbf{x}_i; \boldsymbol{\beta}) \tag{3.51}$$

ovvero come in equazione (3.6), è possibile usare $\mathbf{J}_f$, Jacobiano di f, invece di $\mathbf{J}_r$:

$$\boldsymbol{\delta}_\beta = \left(\mathbf{J}_f^\top \mathbf{J}_f\right)^{-1} \mathbf{J}_f^\top \mathbf{r} \tag{3.52}$$

avendo osservato che le derivate di r_i e $f_i(\mathbf{x}_i)$ coincidono a meno del segno.[2]

3.3.6 Levenberg-Marquardt

Nelle sezioni precedenti, gli algoritmi di risoluzione di sistemi non lineari sono stati distinti tra metodi di discesa del gradiente e metodi di Gauss-Newton. Per una trattazione più approfondita si rimanda a [MBT04].

Nei metodi Gauss-Newton, quando $\mathbf{J}^\top\mathbf{J}$ è definita positiva, il metodo fornisce sempre una direzione di discesa del costo. Tuttavia, quando $\mathbf{J}^\top\mathbf{J}$ diventa singolare, il metodo può diventare numericamente instabile.

La tecnica proposta da Levenberg-Marquardt cerca di combinare i punti di forza di Gauss-Newton e della discesa del gradiente, traendone vantaggio da entrambi.

L'algoritmo di Levenberg-Marquardt (LM) è una tecnica iterativa ormai considerata standard per la risoluzione di problemi non lineari multivariabili. Una descrizione dettagliata dell'algoritmo è disponibile in [Lou05].

LM può essere visto come composto da una fase iniziale di discesa del gradiente, più lenta ma stabile, seguita da un risolutore di tipo Gauss-Newton, più veloce ma meno robusto.

L'algoritmo risolve una versione modificata dell'equazione (3.49), caso particolare dell'equazione (3.35), nota come *augmented normal equations*:

$$\mathbf{N}\boldsymbol{\delta}_\beta = -\mathbf{J}_r^\top \mathbf{r} \tag{3.53}$$

dove $\mathbf{N} = \mathbf{H}_S + \mu\mathbf{I}$ e $\mu > 0$ è un fattore di attenuazione (*damping factor*). Quando μ è elevato, $\mathbf{N}$ è quasi diagonale e l'algoritmo si comporta come una discesa del gradiente. Quando μ è vicino a zero, LM approssima il metodo di Newton.

A differenza della ricerca lungo una linea (*line search*), LM implementa il concetto di *trust region*, adattando dinamicamente la regione entro cui si assume valida la linearizzazione del modello.

Come nel metodo di Gauss-Newton, LM sfrutta l'approssimazione dell'Hessiana:

$$\mathbf{H}_S(\boldsymbol{\beta}) \approx \mathbf{J}_r^\top \mathbf{J}_r \tag{3.54}$$

valida quando la *loss function* è quadratica.

La scelta iniziale e l'aggiornamento del parametro μ tra le iterazioni è lasciata al risolutore, e diverse strategie sono proposte in letteratura.

Una delle implementazioni più diffuse [Nie99] propone di inizializzare μ come:

$$\mu_0 = \tau \max \operatorname{trace} \mathbf{H} \tag{3.55}$$

dove τ è scelto liberamente dall'utente in base alla fiducia nella stima iniziale di $\boldsymbol{\beta}$.

L'aggiornamento di μ è guidato dal *gain ratio* ρ:

$$\rho = \frac{S(\boldsymbol{\beta}) - S(\boldsymbol{\beta} + \boldsymbol{\delta}_\beta)}{\frac{1}{2}\boldsymbol{\delta}_\beta^\top(\mu\boldsymbol{\delta}_\beta + \mathbf{J}^\top\mathbf{r})} \tag{3.56}$$

Un valore elevato di ρ indica che la linearizzazione del modello è efficace, e si può ridurre μ. Viceversa, se ρ è basso o negativo, μ va aumentato per avvicinarsi a un comportamento da discesa del gradiente. Quando $\rho \approx 1$, si ha una buona corrispondenza tra modello e dati.

L'aggiornamento di μ può essere gestito secondo la seguente regola:

if $\rho > 0$ **then**
  $\mu \leftarrow \mu \cdot \max\left(\frac{1}{3}, 1 - (2\rho - 1)^3\right)$
  $\nu \leftarrow 2$
else
  $\mu \leftarrow \mu \cdot \nu$
  $\nu \leftarrow 2 \cdot \nu$
end if

[2] Le derivate coincidono quando si sceglie un residuo del tipo $r_i = \hat{y}_i - y_i$.

3.3.7 DogLeg

Come in Levenberg-Marquardt, l'algoritmo Dog-Leg prova a combinare il metodo Gauss-Newton con il metodo a discesa del gradiente. Rispetto a Levenberg-Marquardt dove questo comportamento è gestito dalla Dumped Hessian, nel caso di Dog Leg, la scelta è resta esplicita dall'analisi di una *Trust Region*.

3.3.8 Sampson Error

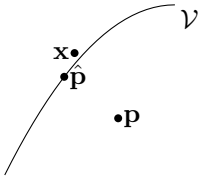


Figura 3.1: Tra una varietà $\mathcal{V}$ e un punto $\mathbf{p}$ si possono individuare il punto a distanza geometrica minima $\hat{\mathbf{p}}$ e il punto determinato dalla distanza di Sampson $\mathbf{x}$.

In molti problemi di regressione di dati è necessario essere in possesso di una qualche metrica per capire quanto un $\mathbf{p}$ è distante dal modello vero e proprio e per fare questo sarebbe utile avere una stima $\hat{\mathbf{p}}$ dell'osservazione senza la componente del rumore, ovvero un dato che appartiene al modello esattamente. Entrambe queste quantità normalmente non sono direttamente ottenibili se non introducendo variabili sussidiarie incognite. È tuttavia possibile ottenere una stima di questi valori, linearizzando la funzione del modello nell'intorno dell'osservazione.

Sia $\mathbf{p}$ una osservazione affetta da rumore e sia $f(\mathbf{x}) = \mathbf{0}$ una varietà multidimensionale *manifold* rappresentante un particolare modello a cui l'osservazione deve appartenere, ovvero $\mathbf{p} = \hat{\mathbf{p}} + \epsilon$.

Il residuo $f(\mathbf{p})$ è una misura *algebrica* della vicinanza tra il punto e il modello e non fornisce nessuna informazione utile in termini assoluti: se la funzione viene sostituita da un suo multiplo (diverso da zero) rappresenterà ovviamente lo stesso luogo dei punti ma il risultato della funzione cambierà di conseguenza. La metrica corretta sotto il punto di vista dello stimatore alla massima verosimiglianza in caso di rumore additivo gaussiano bianco sulle osservazioni è la distanza geometrica tra il punto $\mathbf{p}$ e il punto $\hat{\mathbf{p}}$ appartenente al modello ovvero stimare ϵ.

Esaminiamo pertanto il problema di calcolare una distanza approssimata tra il punto $\mathbf{p} \in \mathbb{R}^m$ rispetto a una varietà geometrica $f(\mathbf{x}) = \mathbf{0}$ dove $f : \mathbb{R}^m \mapsto \mathbb{R}^n$ funzione derivabile in un intorno di $\mathbf{p}$.

Il punto $\hat{\mathbf{p}}$ che giace sulla varietà più vicino al punto $\mathbf{p}$ è per definizione quel punto che minimizza l'errore geometrico

$$\hat{\mathbf{p}} = \arg\min_{\mathbf{x}} \|\mathbf{p} - \mathbf{x}\| \tag{3.57}$$

sotto il vincolo $f(\mathbf{x}) = 0$ (o $\min ||\epsilon||^2$ sotto il vincolo $f(\mathbf{p} + \epsilon) = \mathbf{0}$).

La differenza tra minimizzare una quantità algebrica in maniera lineare e una quantità geometrica in maniera non-lineare ha spinto la ricerca di un eventuale compromesso. Il metodo di Sampson, sviluppato inizialmente per varietà come le coniche, richiede un ipotesi che può essere applicata invece a diversi problemi: le derivate della funzione costo nell'intorno del minimo $\hat{\mathbf{p}}$ devono essere pressoché lineari e dunque approssimabili attraverso l'espansione in serie. La varietà $f(\mathbf{p}) = 0$ può essere approssimata con Taylor in modo tale che

$$\tilde{f}(\mathbf{x}) \approx f(\mathbf{p}) + \mathbf{J}_f \boldsymbol{\delta}_{\mathbf{x}} = \mathbf{0} \tag{3.58}$$

con $\mathbf{J}_f$ matrice $n \times m$ dello Jacobiano della funzione f calcolato in $\mathbf{p}$ e $\boldsymbol{\delta}_{\mathbf{x}} = \mathbf{x} - \mathbf{p}$.

Questa è l'equazione di un iperpiano in $\mathbf{x}$ e la distanza tra il punto $\mathbf{p}$ con il piano $\tilde{f}(\mathbf{x}) = 0$ è la distanza di Sampson o l'*approximate maximum likelihood* (AML). L'errore di Sampson rappresenta la distanza geometrica tra il punto e la versione approssimata della funzione (*geometric distance to first order approximation function*).

Il problema a questo punto diventa quello di trovare il punto $\mathbf{x}$ più vicino a $\mathbf{p}$, ovvero minimizzare $\|\boldsymbol{\delta}_{\mathbf{x}}\|$, soddisfacendo il vincolo lineare

$$\mathbf{J}_f \boldsymbol{\delta}_{\mathbf{x}} = -f(\mathbf{p}) \tag{3.59}$$

Essendo un caso di minimizzazione con vincoli si risolve attraverso l'uso dei moltiplicatori di Lagrange, da cui si ottiene il risultato notevole

$$\boldsymbol{\delta}_{\mathbf{x}} = -\mathbf{J}^\top \left(\mathbf{J}\mathbf{J}^\top\right)^{-1} f(\mathbf{p}) \tag{3.60}$$

risultato interessante se confrontato con il metodo di Gauss-Newton per esempio, equazione (3.50).

Il valore $\boldsymbol{\delta}_{\mathbf{x}}$ rappresenta una stima della distanza del punto $\mathbf{p}$ rispetto alla varietà e può essere usato sia per capire se il punto appartiene o meno alla varietà (per esempio all'interno di algoritmi come RANSAC per discernere gli *outlier*) che potenzialmente come funzione costo alternativa alla norma euclidea. $\boldsymbol{\delta}_{\mathbf{x}}$ è l'errore di Sampson e la sua norma, data da

$$\|\boldsymbol{\delta}_x\|^2 = \boldsymbol{\delta}_x^\top \boldsymbol{\delta}_x = f(\mathbf{p})^\top \left(\mathbf{J}\mathbf{J}^\top\right)^{-1} f(\mathbf{p}) \tag{3.61}$$

indica la distanza (al quadrato) tra il punto e (l'approssimazione al primo grado di) un punto della varietà.

Nel caso notevole $n = 1$, la distanza di Sampson si riduce a

$$\|\boldsymbol{\delta}_x\|^2 = \frac{(f(\mathbf{p}))^2}{\|\nabla f(\mathbf{p})\|^2} \tag{3.62}$$

Applicazioni pratiche dell'uso dell'errore di Sampson sono per esempio la distanza tra un punto e la conica (vedi sezione 3.6.7), distanza di una coppia di punti da una omografia o la distanza di una coppia di punti omologhi rispetto alla matrice Fondamentale (sezione 9.4.2).

La distanza di Sampson può venire generizzata nel caso di molteplici vincoli usando la distanza di Mahalanobis, ovvero minimizzando

$$\min_{\epsilon} \sum ||\epsilon||^2_{\Sigma} = \min_{\epsilon} \sum \epsilon^{\top} \boldsymbol{\Sigma}^{-1} \epsilon \tag{3.63}$$

sotto al vincolo $f(\mathbf{p} + \epsilon) = 0$. L'equazione sopra si generalizza pertanto in

$$\|\boldsymbol{\delta}_x\|^2 = \boldsymbol{\delta}_x^{\top} \boldsymbol{\delta}_x = f(\mathbf{p})^{\top} \left(\mathbf{J}\boldsymbol{\Sigma}\mathbf{J}^{\top}\right)^{-1} f(\mathbf{p}) \tag{3.64}$$

3.3.9 Regressione con rumore Anisotropo

Nel caso in cui il rumore di osservazione non sia isotropico non è più possibile usare la distanza euclidea per misurare l'errore ma è necessario passare alla distanza di Mahalanobis. Sotto questa differente metrica la funzione costo (3.46) si scrive

$$S(\boldsymbol{\beta}) = (\mathbf{r}(\boldsymbol{\beta}))^{\top} \boldsymbol{\Omega} (\mathbf{r}(\boldsymbol{\beta})) \tag{3.65}$$

dove $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$ è la matrice dell'informazione (*information matrix*) detta anche matrice di concentrazione o matrice di precisione. La distanza di Mahalanobis è lo stimatore ottimo nel senso di *Maximum Likelihood* quando il rumore è Gaussiano anisotropo a media nulla. Nel caso particolare in cui la matrice di covarianza sia diagonale questo approccio può ricondursi totalmente all'approccio ai minimi quadrati pesato.

L'espansione in serie di Taylor dell'equazione (3.65) si scrive

$$\begin{aligned} S(\boldsymbol{\beta} + \boldsymbol{\delta}) &= (\mathbf{r}(\boldsymbol{\beta} + \boldsymbol{\delta}))^{\top} \boldsymbol{\Omega} (\mathbf{r}(\boldsymbol{\beta} + \boldsymbol{\delta})) \\ &\approx (\mathbf{r} + \mathbf{J}\boldsymbol{\delta})^{\top} \boldsymbol{\Omega} (\mathbf{r} + \mathbf{J}\boldsymbol{\delta}) \\ &= \mathbf{r}\boldsymbol{\Omega}\mathbf{r}^{\top} + 2\mathbf{r}^{\top}\boldsymbol{\Omega}\mathbf{J}\boldsymbol{\delta} + \boldsymbol{\delta}^{\top}\mathbf{J}^{\top}\boldsymbol{\Omega}\mathbf{J}\boldsymbol{\delta} \\ &= \mathbf{r}\boldsymbol{\Omega}\mathbf{r}^{\top} + 2\mathbf{b}\boldsymbol{\delta} + \boldsymbol{\delta}^{\top}\mathbf{H}\boldsymbol{\delta} \end{aligned} \tag{3.66}$$

con $\mathbf{r}$ e $\mathbf{J}$ calcolate in $\boldsymbol{\beta}$. La matrice $\mathbf{H} = \mathbf{J}^{\top}\boldsymbol{\Omega}\mathbf{J}$ è la matrice dell'informazione dell'intero sistema in quanto ottenuta dalla proiezione dell'errore di misura nello spazio dei parametri attraverso lo Jacobiano $\mathbf{J}$ mentre $\mathbf{b} = \mathbf{r}^{\top}\boldsymbol{\Omega}\mathbf{J}$ è stato introdotto per compattezza.

Le derivate della funzione S di conseguenza diventano

$$\frac{\partial S(\boldsymbol{\beta} + \boldsymbol{\delta})}{\partial \boldsymbol{\delta}} \approx 2\mathbf{b} + 2\mathbf{H}\boldsymbol{\delta} \tag{3.67}$$

Da questo risultato, se si vuole trovare il minimo della funzione costo S usando Gauss-Newton si ottiene un risultato simile a quello visto in precedenza

$$\mathbf{H}\boldsymbol{\delta} = -\mathbf{b} \tag{3.68}$$

risultato molto simile a quella di equazione (3.49) ottenuta da Gauss-Newton con rumore isotropo.

3.3.10 Ottimizzazione su una Varietà

Tutti i metodi di ottimizzazione visti finora sono stati progettati per lavorare su uno spazio Euclideo "piatto". Quando si vuole ottimizzare un vettore di stato che contiene una o più variabili dove lo spazio euclideo perde significato (esempio rotazioni o matrici) ogni parametrizzazione risulta in soluzioni sub-ottime ed affette da singolarità. Negli ultimi anni hanno preso molto piede tecniche che usano la versione overparametrizzata per il vettore di stato [Her08] per poi ottimizzare il problema direttamente sulla varietà (*manifold*), varietà che localmente è 'omeomorfica rispetto a uno spazio lineare.

L'idea è trasformare la classica minimizzazione di $S \in \mathcal{M}$, con $\mathcal{M}$ una varietà n-dimensionale,

$$\boldsymbol{\delta} \Leftarrow \left.\frac{\partial S(\mathbf{x} + \boldsymbol{\delta})}{\partial \boldsymbol{\delta}}\right|_{\boldsymbol{\delta}=0} = 0 \qquad \mathbf{x} \Leftarrow \mathbf{x} + \boldsymbol{\delta} \tag{3.69}$$

'in

$$: \boldsymbol{\epsilon} \Leftarrow \left.\frac{\partial S(\mathbf{x} \boxplus \boldsymbol{\epsilon})}{\partial \boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=0} = 0 \qquad \mathbf{x} \Leftarrow \mathbf{x} \boxplus \boldsymbol{\epsilon} \tag{3.70}$$

con $\boldsymbol{\epsilon} \in \mathbb{R}^n$, supponendo che nell'intorno $\boldsymbol{\epsilon} = 0$ la funzione lavori in uno spazio euclideo. L'operatore $\boxplus$ permette l'addizione tra elementi dello spazio della varietà con elementi dello spazio euclideo $\mathbb{R}^n$.

Un esempio molto classico è considerare l'ottimizzazione di una orientazione espressa in 3 dimensioni attraverso l'uso di un quaternione in 4 dimensioni.

3.4 Funzioni Convesse e Problemi di Ottimizzazione Convessa e non

Definizione 12 *Una funzione $f : \mathbb{R}^n \to \mathbb{R}$ si dice **convessa** se, per ogni coppia di punti $\boldsymbol{x}, \boldsymbol{y}$ nel dominio di f e per ogni $\lambda \in [0, 1]$, vale la disuguaglianza:*

$$f\left(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}\right) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}) \tag{3.71}$$

Una funzione è detta **concava** se la sua opposta, $-f$, è convessa. Le funzioni convesse presentano proprietà strutturali che le rendono particolarmente adatte alla formulazione e risoluzione di problemi di ottimizzazione:

- L'intersezione di insiemi convessi è ancora un insieme convesso.
- L'epigrafo della funzione, definito come $(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R}, |, f(\boldsymbol{x}) \leq t$, è un insieme convesso.
- Se f è due volte differenziabile ($f \in C^2$), la convessità è equivalente alla semidefinità positiva della matrice hessiana $\boldsymbol{\nabla}^2 f(\boldsymbol{x})$ in ogni punto del dominio.
- La disuguaglianza di Jensen fornisce una caratterizzazione alternativa della convessità:

$$f\left(\mathbf{E}[\boldsymbol{X}]\right) \leq \mathbf{E}[f(\boldsymbol{X})] \tag{3.72}$$

Esempi tipici di funzioni convesse includono: funzioni quadratiche con matrice $\boldsymbol{Q} \succeq 0$, norme ℓ_p come $|\boldsymbol{x}|_1$, $|\boldsymbol{x}|_2$, $|\boldsymbol{x}|_\infty$, e la funzione log-sum-exp $f(\boldsymbol{x}) = \log\left(\sum_i e^{x_i}\right)$, frequentemente impiegata in machine learning.

In visione artificiale, formulazioni convesse compaiono in numerosi contesti: stima di parametri tramite minimi quadrati, matching tra feature come problema lineare, segmentazione e clustering basati su rilassamenti convessi. I problemi convessi risultano particolarmente vantaggiosi perché ogni minimo locale coincide con il minimo globale, esistono algoritmi numerici efficienti per la loro risoluzione (metodi del gradiente, simplesso, metodi a punto interno), e molti problemi pratici possono essere riformulati in modo convesso.

Un problema di **programmazione lineare** (LP) assume la forma:

$$\min_{\boldsymbol{x}} \boldsymbol{c}^\top \boldsymbol{x} \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \tag{3.73}$$

dove $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ e $\boldsymbol{b} \in \mathbb{R}^m$. La regione ammissibile è un poliedro convesso, e la soluzione ottima, se esiste, si trova in corrispondenza di uno dei suoi vertici. Esempi di programmazione lineare in visione artificiale sono i problemi di assegnamento e matching, flussi su grafo per segmentazione o stereo.

Un problema di **programmazione quadratica** (QP) si scrive come:

$$\min_{\boldsymbol{x}} \tfrac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \tag{3.74}$$

con $\boldsymbol{Q} \succeq 0$. Si tratta di un'estensione della LP che include un termine quadratico convesso. Esempi di programmazione quadratica in visione artificiale sono Support Vector Machines (SVM) e fitting di modelli geometrici con vincoli.

Oltre a LP e QP, esistono classi più generali di problemi convessi:

- **Second-Order Cone Programming (SOCP)**: impiegato in contesti di ottimizzazione robusta.
- **Semidefinite Programming (SDP)**: utilizzato nei rilassamenti convessi di problemi combinatori.

Tuttavia, molti problemi di visione artificiale non sono convessi. In tali casi la funzione obiettivo può presentare più minimi locali e non esistono garanzie di convergenza verso la soluzione globale. Esempi tipici sono la stima della posa (PnP), il bundle adjustment o la ricostruzione tridimensionale. Per affrontare tali difficoltà si ricorre a:

- inizializzazioni robuste, per ridurre il rischio di convergere verso minimi locali indesiderati;
- rilassamenti convessi o approssimazioni, che consentono di trattare un problema non convesso come se fosse convesso;
- strategie iterative (RANSAC, multi-start), che esplorano più soluzioni candidate;
- metodi che sfruttano la struttura del problema, come l'ottimizzazione su gruppi di Lie.

Nella pratica, gran parte dell'ottimizzazione in visione artificiale avviene su problemi non convessi, e la qualità della soluzione dipende fortemente dalla bontà dell'inizializzazione e dalla modellazione del problema.

3.5 Valutazione dei parametri del modello

Trascurando la presenza di *outlier* nei dati in ingresso su cui eseguire la regressione, rimangono come importanti questioni aperte sia quella di dare un giudizio su quanto è buono il modello ottenuto e allo stesso tempo fornire un indice su quanto tale stima sia distante dal modello vero, a causa degli errori sui dati in ingresso.

In questa sezione viene trattato ampiamente il caso non-lineare: il caso lineare è equivalente usando al posto dello Jacobiano $\mathbf{J}$ la matrice dei parametri $\mathbf{X}$ in parte già affrontato in sezione 2.7.

Sia $\mathbf{y} = (y_1, \dots, y_n)^\top$ un vettore di realizzazioni di variabili aleatorie statisticamente indipendenti $y \in \mathbb{R}$ e $\boldsymbol{\beta} \in \mathbb{R}^m$ parametri del modello. Uno stimatore intuitivo della bontà del modello è il *root-mean-squared residual error* (*RMSE*), chiamato anche *standard error of the regression*:

$$s = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}} \tag{3.75}$$

con $\hat{y}_i = f(\mathbf{x}_i, \hat{\boldsymbol{\beta}})$ valore stimato grazie al modello f da cui sono stati ricavati i parametri $\hat{\boldsymbol{\beta}}$. Normalmente si è già vista questa funzione espressa sotto forma del residuo $r_i = \mathbf{y}_i - \hat{\mathbf{y}}_i$. Se lo stimatore non è effetto da *bias* (come accade per esempio nella regressione ai minimi quadrati) $\mathrm{E}[r_i] = 0$. Pertanto nel caso in cui il rumore sulle osservazioni sia gaussiano a media nulla il valore di $s \geq \sigma$ e i due valori sono uguali quando il modello è ottimo.

Questo tuttavia non è un indice diretto della bontà della soluzione individuata ma solo quanto il modello trovato combacia con i dati in ingresso: si pensi ad esempio al caso limite dei sistemi non sovradimensionati dove il residuo sarà sempre zero, indipendentemente dalla quantità di rumore che agisce sulle singole osservazioni.

L'indice più adatto a stimare il modello è la matrice di varianza-covarianza dei parametri (*Parameter Variances and Covariances matrix*).

La propagazione in avanti della covarianza (*covariance forward propagation*) è stata già mostrata nella sezione 2.6 e, facendo un veloce rimando, esistono 3 metodi per eseguire tale operazione: 'il primo è basato sulla approssimazione lineare del modello e coinvolge l'uso dello Jacobiano, il secondo è basato sulla più generica tecnica della simulazione Monte Carlo, e infine una via moderna alternativa, media tra le prime due, è la *Unscent Transformation* (sezione 2.12.5) che permette, empiricamente, stime fino al terzo ordine in caso di rumore gaussiano.

Il voler valutare la bontà dei parametri individuati $\hat{\boldsymbol{\beta}}$ data la covarianza del rumore stimata (*Covariance Matrix Estimation*) è esattamente il caso opposto perché richiede di calcolare la propagazione all'indietro della varianza (*backward propagation*). Infatti, ottenuta tale matrice di covarianza, è possibile definire un intervallo di confidenza nell'intorno di $\hat{\boldsymbol{\beta}}$.

Tale bontà della stima dei parametri $\hat{\boldsymbol{\beta}}$, nel caso non-lineare, può essere valutata in prima approssimazione attraverso l'inversione della versione linearizzata del modello (ma anche in questo caso tecniche come la Montecarlo o la UT possono essere utilizzate per stime più rigorose).

È possibile individuare la matrice di covarianza associata alla soluzione proposta $\hat{\boldsymbol{\beta}}$ nel caso in cui la funzione f sia biunivoca e derivabile nell'intorno di tale soluzione. Sia pertanto $f : \mathbb{R}^m \to \mathbb{R}^n$ funzione multivariata multidimensionale, è possibile stimare il valor medio $\bar{\mathbf{r}} = \mathrm{E}\left[\mathbf{y} - f(\hat{\boldsymbol{\beta}})\right] \approx \mathbf{0}$ e la matrice di cross-covarianza $\boldsymbol{\Sigma}_r$ dei residui allora la trasformazione inversa f^{-1} avrà valor medio $\hat{\boldsymbol{\beta}}$ e matrice di covarianza

$$\Sigma_{\boldsymbol{\beta}} = (\mathbf{J}^\top \Sigma_r^{-1} \mathbf{J})^{-1} \tag{3.76}$$

con $\mathbf{J}$ Jacobiano del modello f calcolato nel punto $\hat{\boldsymbol{\beta}}$:

$$J_{i,j} = \frac{\partial r_i}{\partial \beta_j}(\hat{\boldsymbol{\beta}}) = -\frac{\partial f_i}{\partial \beta_j}(\hat{\boldsymbol{\beta}}) \tag{3.77}$$

L'equazione (3.76) si ricava manipolando l'equazione (2.34), equazione che calcola la propagazione in avanti dell'incertezza.

Si noti che questo (l'inverso della matrice dell'informazione) è il limite inferiore di Cramer-Rao sulla covarianza che può avere uno stimatore corretto del parametro $\boldsymbol{\beta}$.

Nei casi in cui la trasformazione f sia sottodeterminata, il rango dello Jacobiano d, con $d < m$, è chiamato numero dei parametri essenziali (*essential parameters*). In caso di trasformazione f sottodeterminata la formula (3.76) non è invertibile ma è possibile dimostrare che la migliore approssimazione della matrice di covarianza può essere ottenuta attraverso l'uso della pseudo-inversa:

$$\Sigma_{\boldsymbol{\beta}} = (\mathbf{J}^\top \Sigma_r^{-1} \mathbf{J})^{+}$$

Alternativamente è possibile eseguire una decomposizione QR con Pivot dello Jacobiano, individuare le colonne linearmente dipendenti (attraverso l'analisi della diagonale della matrice R) e rimuoverle durante l'inversione stessa della matrice.

Nel caso invece molto comune in cui f sia una funzione scalare e il rumore di osservazione sia indipendente di varianza costante, la matrice di covarianza stimata asintoticamente (*Asymptotic Covariance Matrix*) si può scrivere in maniera più semplice come

$$\Sigma_{\boldsymbol{\beta}} = (\mathbf{J}^\top \mathbf{J})^{-1} \sigma^2 \tag{3.78}$$

con σ^2 varianza del rumore di osservazione avendo applicato l'ipotesi $\mathbf{\Sigma}_r = \sigma^2\mathbf{I}$ valida in caso di realizzazioni indipendenti. Siccome $\mathbf{J}$ è funzione solo della geometria del problema, la matrice $(\mathbf{J}^\top\mathbf{J})^{-1}$ è anche essa sola funzione del problema e non delle osservazioni. Asintoticamente la stima tende a $\boldsymbol{\beta} = \mathcal{N}\left(\hat{\boldsymbol{\beta}}, \Sigma_{\boldsymbol{\beta}}\right)$. La matrice dello Jacobiano, in quanto indica quanto le uscite sono suscettibili dai parametri, è anche chiamata sensitivity matrix.

La stima del rumore di osservazione può essere empirica, ipotizzando per la legge dei grandi numeri $\sigma = s$, calcolata attraverso

$$\sigma^2 \approx \frac{\sum_{i=1}^{n} r_i^2}{n-m} \tag{3.79}$$

'usando le statistiche a posteriori dell'errore sui dati r_i. Il denominatore $n-m$ rappresenta i gradi di libertà statistici del problema: in questo modo la varianza stimata è infinita quando il numero di incognite del modello equivale al numero di dati raccolti.

Lo stimatore di covarianza di Eicker-White è leggermente differente e viene lasciato al lettore il suo studio.

La matrice di varianza-covarianza dei parametri rappresenta l'elissoide dell'errore.

Una metrica utile per dare un voto al problema è la configurazione D-ottima (D-optimal design):

$$\det\left(\mathbf{J}^\top\mathbf{J}\right)^{-1} \tag{3.80}$$

che minimizza il determinante della matrice di varianza-covarianza, o nel caso opposto, che massimizza la matrice dell'informazione di Fisher:

$$\det\mathbf{F}\left(\boldsymbol{\beta}\right) \tag{3.81}$$

Geometricamente questo approccio minimizza il volume dell'elissoide dell'errore.

Altre metriche sono per esempio la configurazione E-ottima (E-optimal design) che consiste nel massimizzare il minimo autovalore della matrice di Fisher ovvero minimizzare il più grande autovalore della matrice di varianza-covarianza. Geometricamente questo minimizza il massimo diametro dell'elissoide.

3.6 Regressioni notevoli

In questa sezione verranno esaminate alcune regressioni notevoli a modelli molto semplici, come rette piani e circonferenze.

3.6.1 Regressione a una retta

Sia

$$y = mx + q + \varepsilon \tag{3.82}$$

l'equazione della retta scritta in forma esplicita con l'errore di misura totalmente inserito lungo l'asse delle y. Con l'errore lungo l'asse y la funzione costo da minimizzare è

$$S = \frac{1}{2n}\sum_{i=1}^{n}\left(mx_i + q - y_i\right)^2 \tag{3.83}$$

La soluzione del problema è il punto in cui il gradiente di S in m e q si annulla

$$\begin{aligned} \tfrac{\partial S}{\partial m} &= \tfrac{1}{n}\left(m\textstyle\sum x_i^2 + q\sum x_i - \sum y_i x_i\right) = m\bar{x^2} + q\bar{x} - (\bar{xy}) = 0 \\ \tfrac{\partial S}{\partial q} &= \tfrac{1}{n}\left(m\textstyle\sum x_i + qn - \sum y_i\right) = m\bar{x} + q - \bar{y} = 0 \end{aligned} \tag{3.84}$$

'ovvero:

$$\begin{aligned} m &= \frac{(\bar{xy}) - \bar{x}\bar{y}}{\bar{x^2} - \bar{x}^2} = \frac{\operatorname{cov}(x,y)}{\operatorname{var}(x)} \\ q &= -m\bar{x} + \bar{y} \end{aligned} \tag{3.85}$$

con $\bar{x}$ il valor medio dei campioni x_i (con lo stesso formalismo sono indicate anche le altre quantità). La retta passa per il punto $(\bar{x}, \bar{y})$ centroide della distribuzione.

È facile modificare tale risultato nel caso in cui si voglia minimizzare lo scarto lungo le x invece che lungo le y, o rappresentare l'equazione della retta in forma implicita.

3.6.2 Orthogonal Distance Fit

Nel caso in cui l'errore sia presente su entrambi gli assi (rumore funzione della distanza), la scrittura della funzione costo S che massimizza la verosimiglianza è quella che viene chiamata *Orthogonal least-squares line fit*. L'errore può essere espresso infatti usando la distanza tra il punto e la retta, secondo equazione (1.31). La regressione che usa questa metrica, pertanto detta *Perpendicular Regression* o *Total least squares* (si veda sezione 3.2.2), ha senso quando entrambe le coordinate sono affette da errore ovvero sono entrambe variabili aleatorie. L'ammontare del rumore sulle due componenti è supposto uguale (per il caso più generale si veda la discussione in sezione 2.4). La funzione errore S da minimizzare è la distanza tra il punto e la retta:

$$S = \frac{1}{2n} \sum_{i=1}^{n} \frac{(ax_i + by_i + c)^2}{a^2 + b^2} \tag{3.86}$$

'e il minimo si trova in $\nabla S = 0$. È da notare che nel caso di distanza perpendicolare esiste come soluzione sia un minimo che un massimo e pertanto esisteranno due valori di rette (ortogonali tra loro) entrambe soluzioni del sistema.

Dalla derivata parziale $\frac{\partial S}{\partial c} = 0$ si ricava che la retta di regressione passa per il centroide $(\bar{x}, \bar{y})$ della distribuzione, ovvero

$$c = -a\bar{x} - b\bar{y} \tag{3.87}$$

con $\bar{x}$ e $\bar{y}$ medie dei campioni x_i e y_i rispettivamente.

La funzione errore (3.86), usando la relazione (3.87), si può scrivere come:

$$S = \frac{a^2 \left(\bar{x^2} - \bar{x}^2\right) + 2ab \left(\overline{xy} - \bar{x}\bar{y}\right) + b^2 \left(\bar{y^2} - \bar{y}^2\right)}{a^2 + b^2} \tag{3.88}$$

'ovvero, facendo sostituzioni adeguate $S_{xx} = \mathrm{var}(x)$, $S_{yy} = \mathrm{var}(y)$ e $S_{xy} = \mathrm{cov}(x, y)$:

$$S = \frac{a^2 S_{xx} + 2ab S_{xy} + b^2 S_{yy}}{a^2 + b^2} \tag{3.89}$$

più facilmente derivabile. L'espressione (3.89) dell'errore non è di carattere generale, ma vale solamente per tutte le rette che passano per il centroide della distribuzione. Essendo una forma omogenea è conosciuta a meno di un fattore moltiplicativo: non esiste pertanto una sola soluzione ma una relazione che lega i parametri. Escludendo i casi $a = 0$, $b = 0$ (da trattare a parte) il vincolo per ricavare il minimo/massimo ha la forma del tipo

$$(a^2 - b^2)S_{xy} + ab(S_{yy} - S_{xx}) = 0 \tag{3.90}$$

soluzione del problema.

È da notare infine che il medesimo risultato si ottiene in maniera molto più semplice applicando la decomposizione SVD sull'equazione delle rette. Nel caso di regressione lineare la decomposizione SVD minimizza sia l'errore algebrico che geometrico (l'errore algebrico e geometrico coincidono quando tutti i termini affetti da rumore rimangono limitati al termine noto).

3.6.3 Regressione ortogonale a un piano

Si possono estendere le considerazioni fatte sulla retta anche per il piano. Va sottolineato che le regressione ortogonali di una retta, di un piano, o di un iperpiano, sono da considerarsi come un problema di autovalori e risolvibile attraverso la decomposizione SVD (è esattamente la principale applicazione della PCA).

Sia $\mathbf{p_0} = \mathrm{E}[\mathbf{p}]$ il centroide dei punti coinvolti nella regressione. Data l'equazione del piano (1.49) e come funzione errore la sommatoria delle distanze (1.52) si ottiene immediatamente il vincolo:

$$k = -\mathbf{p_0} \cdot \hat{n} \tag{3.91}$$

'ovvero, come già rilevato nel caso lineare, il centroide della distribuzione appartiene al piano. Partendo da questo primo vincolo, è possibile descrivere il piano come

$$(\mathbf{p} - \mathbf{p_0}) \cdot \hat{n} = 0 \tag{3.92}$$

sistema omogeneo sovradimensionato, la cui soluzione si può ottenere con la pseudoinversa (ad esempio con la fattorizzazione QR o SVD). Il valore di $\hat{n}$ così ricavato sarà conosciuto a meno di un fattore moltiplicativo e per questo motivo si può sempre normalizzare, forzandolo alla lunghezza unitaria (le soluzioni ottenute attraverso fattorizzazioni sono solitamente già normalizzate).

3.6.4 Regressione lineare a funzione polinomiale

Il metodo applicato per ottenere la regressione lineare a una retta espressa in forma esplicita si può generalizzare a una qualunque funzione polinomiale del tipo:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_m x^m + \varepsilon \tag{3.93}$$

dove $\beta_0 \ldots \beta_m$ sono i parametri della curva da ricavare, parametri che si ottengono cercando il minimo della funzione errore descritta in (3.6). Le derivate di una funzione polinomiale sono notevoli:

$$\begin{array}{ll} \frac{\partial S}{\partial \beta_j} & = \sum_{i=0}^{n} (\beta_0 + \ldots + \beta_m x_i^m - y_i) x_i^j \\ & = \beta_0 \sum x_i^j + \ldots + \beta_m \sum x_i^{j+m} - \sum y_i x_i^j \end{array} \tag{3.94}$$

Il porre il gradiente nullo significa risolvere pertanto il sistema associato:

$$\begin{bmatrix} \sum 1 & \ldots & \sum x_i^m \\ \sum x_i & \ldots & \sum x_i^{m+1} \\ \vdots & \ddots & \vdots \\ \sum x_i^m & \ldots & \sum x_i^{2m} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_m \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum y_i x_i \\ \vdots \\ \sum y_i x_i^m \end{bmatrix} \tag{3.95}$$

che è una matrice simmetrica.

Alternativamente è possibile sfruttare la teoria della pseudoinversa (sezione 1.1) e usare direttamente l'equazione (3.93) per costruire un sistema lineare sovradimensionato:

$$\begin{bmatrix} 1 & x_1 & \ldots & x_1^m \\ 1 & x_2 & \ldots & x_2^m \\ \vdots & & & \vdots \\ 1 & x_n & \ldots & x_n^m \end{bmatrix} \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{3.96}$$

matrice di Vandermonde. La soluzione di questo sistema permette di ottenere i coefficienti del polinomio che minimizza il quadrato dei residui. Se si pensa alla pseudoinversa risolta con il metodo delle *normal equations* si vede come il sistema risultante è esattamente lo stesso di equazione (3.95).

Come si vedrà in altre parti di questo libro, matrici come quella di Vandermonde, dove le diverse colonne hanno ordini di grandezza differenti, sono mal condizionate e richiedono una normalizzazione per migliorarne la stabilità numerica.

3.6.5 Regressione a una circonferenza

La regressione di una serie di punti all'equazione di una circonferenza (*circular regression*) si può ottenere minimizzando sia una distanza algebrica che geometrica.

Se si vuole calcolare la regressione lineare di una serie di dati verso l'equazione della circonferenza di centro in (x_0, y_0) e raggio r la funzione da minimizzare è

$$S = \sum \left((x_i - x_0)^2 + (y_i - y_0)^2 - r^2 \right)^2 \tag{3.97}$$

dove si minimizza la distanza ortogonale tra i punti e il modello. Per risolvere il problema conviene eseguire un cambio di variabile e minimizzare la forma algebrica:

$$S = \sum \left(z_i + B x_i + C y_i + D \right)^2 \tag{3.98}$$

dove è stato introdotto $z_i = x_i^2 + y_i^2$ per semplicità. Il problema si riduce alla soluzione di un sistema lineare 3×3 di equazione

$$\begin{array}{llll} \sum z_i x_i & + B \sum x_i^2 & + C \sum y_i x_i & + D \sum x_i = 0 \\ \sum z_i y_i & + B \sum x_i y_i & + C \sum y_i^2 & + D \sum y_i = 0 \\ \sum z_i & + B \sum x_i & + C \sum y_i & + D \sum 1 = 0 \end{array} \tag{3.99}$$

simmetrico, facilmente risolvibile. Ricavati i parametri B, C e D è possibile ottenere i parametri originali del cerchio:

$$x_0 = -\frac{B}{2} \quad y_0 = -\frac{C}{2} \quad r^2 = x_0^2 + y_0^2 - D \tag{3.100}$$

Lo stesso risultato si può ottenere usando i risolutori lineari visti in precedenza. Si consideri per esempio una rappresentazione algebrica di un cerchio

$$f(\mathbf{x}) = a \mathbf{x}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c = 0 \tag{3.101}$$

dove $\mathbf{x}$ 'e il luogo dei punti della circonferenza.

Dato un elenco di punti che appartengono alla circonferenza affetti da rumore, i parametri (a, b_x, b_y, c) che descrivono la circonferenza si ottengono dalla soluzione del sistema omogeneo di vincoli (3.101). Come si vedrà in dettaglio in successivi problemi, per motivi puramente computazionali, risulta conveniente normalizzare i dati in ingresso, in quanto le diverse incognite sono associate a dati di magnitudine molto differenti.

La soluzione algebrica è spesso usata come soluzione iniziale per tecniche iterative che minimizzano una metrica differente. Per eseguire una regressione geometrica è necessario minimizzare le distanze $d_i^2 = \left(\|\mathbf{x}_i - (x_0, y_0)^\top\| - r\right)^2$. Per minimizzare questa quantità è richiesto un risolutore non lineare ai minimi quadrati, ad esempio Levenberg-Marquardt, e il calcolo delle derivate della funzione costo.

Una alternativa è infine parametrizzare il problema in un altro spazio diverso da quello cartesiano. Usando infatti la forma parametrica dell'equazione del cerchio

$$\begin{aligned} x &= x_0 + r\cos\varphi \\ y &= y_0 + r\sin\varphi \end{aligned} \tag{3.102}$$

le quantità da minimizzare diventano

$$\begin{aligned} x_i - x_0 + r\cos\varphi_i &\approx 0 \\ y_i - y_0 + r\sin\varphi_i &\approx 0 \end{aligned} \tag{3.103}$$

facilmente derivabili. Ad ogni dato in ingresso (x_i, y_i) viene associata una incognita aggiuntiva φ_i, variabile sussidiaria. In questo modo si crea un sistema non lineare in $3 + n$ incognite con $2n$ equazioni.

3.6.6 Regressione ad un ellisse

Come per il cerchio è possibile eseguire sia una minimizzazione algebrica, che geometrica.

L'equazione quadratica di un ellisse è

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c = 0 \tag{3.104}$$

dove $\mathbf{A}$ è una matrice simmetrica, definita positiva. Anche in questo caso la soluzione del problema omogeneo (3.104) permette di ricavare le 6 incognite (conosciute a meno di un fattore moltiplicativo) del sistema.

La soluzione non lineare che minimizza la quantità geometrica si può ottenere usando la rappresentazione parametrica dell'ellisse

$$\mathbf{x} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} a\cos\varphi \\ b\sin\varphi \end{bmatrix} \tag{3.105}$$

dove (x_0, y_0) rappresenta il centro dell'ellissi, (a, b) la lunghezza dei due semiassi e α la rotazione dell'ellissi rispetto al centro. Come per il cerchio, le φ_i saranno variabili sussidiarie e il problema non lineare diventa di $5 + n$ incognite con $2n$ equazioni.

3.6.7 Regressione ad un conica

È chiaramente possibile generalizzare la regressione della parabola, della circonferenza e dell'ellissi a una qualsiasi conica (sezione 1.6) arbitrariamente orientata.

Siano $(x_i, y_i)^\top$, con $i = 1, \ldots, n$, punti affetti da rumore appartenenti al luogo dei punti da stimare.

L'equazione (1.56) può essere riscritta nella forma

$$\mathbf{a}_i^\top \boldsymbol{\beta} = 0 \tag{3.106}$$

dove $\mathbf{a}_i = \left\{x_i^2, x_i y_i, y_i^2, x_i, y_i, 1\right\}$ e $\boldsymbol{\beta} = \{a, b, c, d, e, f\}$ da cui risulta evidente che per ottenere i parametri $\boldsymbol{\beta}$ di una qualsiasi conica si può procedere con la soluzione di un problema omogeneo di tipo $\mathbf{A}\boldsymbol{\beta} = 0$ in 6 incognite, minimizzando una quantità del tipo

$$S = \sum_{i=1}^{n} \mathbf{a}_i^\top \boldsymbol{\beta} \tag{3.107}$$

Tale soluzione chiaramente minimizza un errore algebrico e non geometrico, pertanto questo non è lo stimatore ottimo.

Una formulazione alternativa per ricavare i parametri delle coniche si può trovare in [FPF99].

Infine, per capire se un punto è vicino all'equazione di una conica ovvero per ottenere una approssimazione geometrica della distanza punto-conica, si può calcolare l'errore di Sampson (sezione 3.3.8) sfruttando il fatto che, per una conica di equazione (1.56), il gradiente della varietà assume una forma molto semplice da calcolare:

$$\nabla f(x, y) = (2ax + by + d, bx + 2cy + e) \tag{3.108}$$

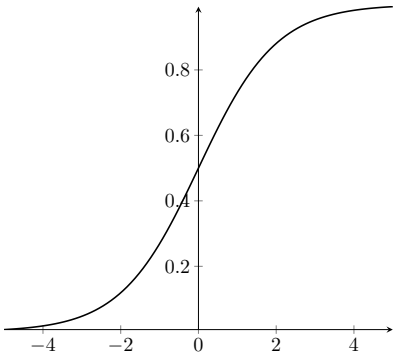


Figura 3.2: Funzione Logistica

3.7 Regressione Logistica

Esiste una famiglia di modelli lineari, che mettono in relazione la variabile dipendente con le variabili esplicative attraverso una funzione non lineare, chiamati modelli lineari generalizzati (*generalized linear model*). La regressione logistica si situa in questa classe di modelli, nel caso particolare in cui la variabile y sia dicotomica, ovvero possa assumere solo valori 0 o 1. Per sua natura, questo genere di problemi, assume una notevole importanza nei problemi di classificazione.

Nel caso di problemi binari è possibile definire la probabilità di successo e insuccesso

$$\begin{aligned} P[Y=1|\mathbf{x}] &= p(\mathbf{x}) \\ P[Y=0|\mathbf{x}] &= 1-p(\mathbf{x}) \end{aligned} \tag{3.109}$$

La risposta di un predittore lineare del tipo

$$y' = \boldsymbol{\beta}\cdot\mathbf{x}+\varepsilon \tag{3.110}$$

non è limitata tra 0 e 1 perciò non è adatta a questo scopo. Risulta necessario associare la risposta del predittore lineare con la risposta di una certa funzione g, funzione della probabilità $p(\mathbf{x})$

$$g(p(\mathbf{x})) = \boldsymbol{\beta}\cdot\mathbf{x}+b \tag{3.111}$$

dove $g(p)$, *mean function*, è una funzione non lineare definita tra $[0,1]$. $g(p)$ deve essere invertibile e l'inversa $g^{-1}(y')$ è la *link function*.

Un modello ampiamente usato per la funzione $g(p)$ è la funzione *logit* definita come:

$$logit(p) = \log\frac{p}{1-p} = \boldsymbol{\beta}\cdot\mathbf{x} \tag{3.112}$$

La funzione $\frac{p}{1-p}$, siccome rappresenta quante volte il successo è maggiore dell'insuccesso, è detta *odds-ratio* e di conseguenza la funzione (3.112) rappresenta il logaritmo della probabilità che accada un evento rispetto alla probabilità che il medesimo evento non accada (*log-odds*).

La sua funzione inversa esiste e vale

$$\mathrm{E}[Y|\mathbf{x}] = p(\mathbf{x}) = \frac{e^{\boldsymbol{\beta}\cdot\mathbf{x}}}{1+e^{\boldsymbol{\beta}\cdot\mathbf{x}}} \tag{3.113}$$

'ed è la funzione logistica.

Il metodo della massima verosimiglianza in questo caso non coincide con il metodo dei minimi quadrati ma con

$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{i=1}^{n} f(y_i|\mathbf{x}_i) = \prod_{i=1}^{n} p^{y_i}(\mathbf{x}_i)\left(1-p^{y_i}(\mathbf{x}_i)\right) \tag{3.114}$$

da cui la funzione di verosimiglianza logaritmica

$$\log\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{n} y_i(\boldsymbol{\beta}\cdot\mathbf{x}_i) - \log\left(1+e^{\boldsymbol{\beta}\cdot\mathbf{x}_i}\right) \tag{3.115}$$

la cui massimizzazione, attraverso tecniche iterative, permette la stima dei parametri $\boldsymbol{\beta}$.

3.8 M-Estimator

L'utilizzo della regressione ai minimi quadrati (*Least squares*) dell'errore rispetto ad altre funzioni di peso è sia scelta per via della funzione di massima verosimiglianza ma soprattutto per via della semplicità delle derivate che si ottengono nello Jacobiano.

Nei problemi visti finora si è quasi sempre supposta la varianza costante e gli errori di osservazioni distribuiti secondo una distribuzione normale. Se il rumore fosse solamente gaussiano questo approccio è teoricamente corretto, ma applicazioni reali presentano distribuzioni solitamente formate da rumore gaussiano appartenente al modello e rumore associato a elementi che non appartengono al modello stesso (*outlier*). In questa condizione la regressione ai minimi quadrati ha come conseguenza quella di trattare tutti i punti come se l'errore fosse gaussiano, ovvero pesando poco i punti vicini al modello e pesando invece molto i punti lontani dal modello i quali, per un puro discorso di probabilità, sono solitamente *outlier*.

Il modo di trattare in maniera univoca questi problemi è stato indicato da John Nelder che ha battezzato tali tecniche con il nome di modelli lineari generalizzati (*GLM*, *General Linear Models*).

Per risolvere questo problema è necessario cambiare la metrica attraverso la quale vengono valutati gli errori: 'un primo esempio di metrica differente che potrebbe risolvere il problema è la regressione al valore assoluto. Il calcolo tuttavia del minimo della funzione errore espresso come distanza in valore assoluto (*Least absolute deviations regression*) non è facile, in quanto la derivata non è continua e richiede l'utilizzo di tecniche iterative di ottimizzazione: metriche derivabile sono preferibili in questo caso.

Peter Huber ha proposto nel 1964 una generalizzazione del concetto di minimizzazione alla massima verosimiglianza introducendo gli *M-estimator*.

Alcuni esempi di funzioni di regressione sono mostrate in figura 3.3.

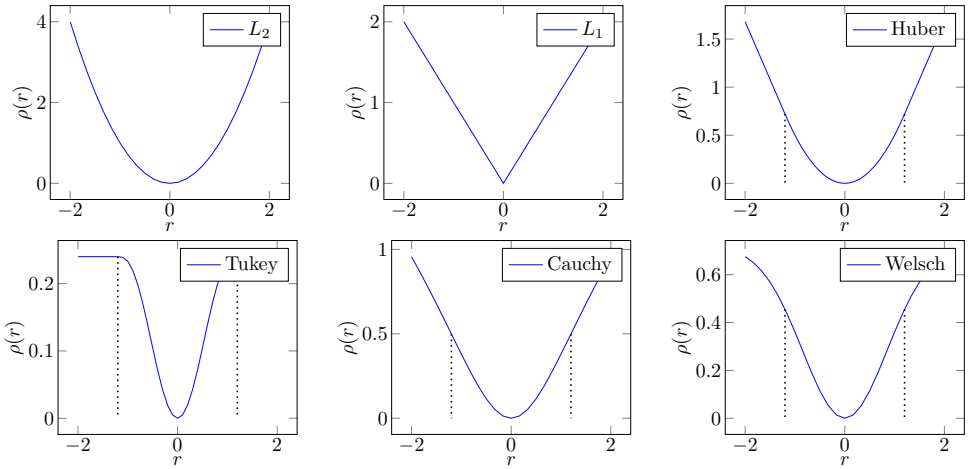


Figura 3.3: Alcuni esempi di funzioni peso per regressioni: la regressione ai minimi quadrati (metrica L2), la regressione lineare (L1), gli stimatori di Huber, la biquadratica di Tukey, la funzione Lorentziana (Cauchy) e la funzione di Welsch (Leclerc).

Un *M-Estimator* sostituisce la metrica basata sulla somma dei quadrati a una metrica basata su una funzione ρ (*loss function*) generica con un unico minimo in zero e con crescita sub-quadratica. Gli M-Estimator generalizzano la regressione ai minimi quadrati: ponendo $\rho(\mathbf{r}) = \|\mathbf{r}\|^2$ si ottiene la forma classica della regressione.

Infine, se la funzione perdita è monotona crescente si parla di *M-estimator* mentre se la funzione perdita è crescente vicino a zero ma decrescente lontano da 0 si parla di *Redescending M-estimator*.

La stima dei parametri si ottiene attraverso la minimizzazione di una sommatoria di quantità pesate generiche:

$$\min_{\beta} \sum \rho \left(\frac{\mathbf{r}_i}{\sigma_i} \right) \tag{3.116}$$

la cui soluzione, in forma chiusa o iterativa, rispetto ai minimi quadrati è diversa a causa della differente derivata della funzione ρ:

$$\sum_{i=1}^{n} \frac{1}{\sigma_i} \rho' \left(\frac{\mathbf{r}_i}{\sigma_i} \right) \frac{\partial \mathbf{r}_i}{\partial \beta_j} = 0 \qquad j = 1, \ldots, m \tag{3.117}$$

3.9 Minimi quadrati riponderati iterativamente

Una tecnica ortogonale agli M-Estimator è la tecnica dei minimi quadrati riponderati iterativamente (*IRLS*, *Iteratively Reweighted Least Squares*) [gre84]. È una tecnica dove ad ogni iterazione vengono stimati nuovi pesi con i quali ricavare una nuova soluzione. Tale tecnica si può applicare sia a problemi lineari che non-lineari.

Nel caso lineare l'obiettivo è quello di minimizzare una funzione costo del tipo

$$\|\mathbf{Wr}\|^2 = \sum_i w_i^2 r_i^2 = \mathbf{r}^\top \mathbf{W}^\top \mathbf{W}\mathbf{r} \tag{3.118}$$

dove la matrice $\mathbf{W}$ è una matrice diagonale con i pesi w_i posti lungo la diagonale.

Nel caso di problema sovradimensionato questo ha soluzione

$$\mathbf{x} = \left[\mathbf{A}^\top \mathbf{W}^\top \mathbf{W}\mathbf{A}\right]^{-1} \mathbf{A}^\top \mathbf{W}^\top \mathbf{W}\mathbf{b} \tag{3.119}$$

Si applica lo stesso approccio a sistemi non-lineari.

3.10 Black-Rangarajan duality

La tecnica dei minimi quadrati iterativi diventa una tecnica ottima se per i pesi vengono selezionati opportunamente, cercando di unire l'aspetto di stima robusta con l'aspetto di rigetto degli Outlier.

La dualità descritta da Black-Rangarajan [BR96] collega questi due aspetti mostrando che la stima robusta può essere vista come un processo di rimozione degli Outlier. Questo dualismo permette l'uso della Non-Convessità Graduata (*Graduated Non-Convexity*, GNC), una tecnica che trasforma gradualmente un problema non convesso in uno convesso, rendendo più facile trovare una soluzione globale senza bisogno di un'ipotesi iniziale.

In pratica, questo dualismo unito alla GNC sono utilizzati per sviluppare algoritmi che sono robusti a una percentuale elevata di outlier, superando i metodi tradizionali come RANSAC in termini di accuratezza e velocità.

La dualità di Black-Rangarajan fornisce una teoria per costruire la relazione tra M-estimatori e processi lineari. Le tipiche funzioni di Loss robuste includono Welsch (Leclerc), Cauchy (Lorentziana), Charbonnier (pseudo-Huber, ℓ_1-ℓ_2), Huber, Geman-McClure, quadratica troncata smooth, quadratica troncata, funzioni biweight di Tukey, ecc. La dualità di Black-Rangarajan di queste funzioni può essere trovata in [ZB17] di cui riporto un estratto qua in tabella:

Nome	$\rho(x)$	$\omega(x)$
Quadratica	$\frac{x^2}{2}$	1
Cauchy	$\frac{\tau^2}{2}\log\left(1+x^2/\tau^2\right)$	$\frac{\tau^2}{\tau^2+x^2}$
Huber	$\begin{cases} x^2/2 & \lvert x\rvert \le \tau \\ \tau\lvert x\rvert - \tau^2/2 & \lvert x\rvert \ge \tau \end{cases}$	$\begin{cases} 1 & \lvert x\rvert \le \tau \\ \tau/\lvert x\rvert & \lvert x\rvert \ge \tau \end{cases}$
Welsch	$\frac{\tau^2}{2}\left(1-e^{-x^2/\tau^2}\right)$	e^{-x^2/τ^2}
Quadratica troncata	$\min\{\tau, x\}^2/2$	$\begin{cases} 1 & \lvert x\rvert \le \tau \\ 0 & \lvert x\rvert > \tau \end{cases}$

dove c'è la *loss function* $\rho(x)$ e la sua corrispondente funzione di aggiornamento dei pesi $\omega(x)$ e sia $\tau \stackrel{\text{def}}{=\!=} \max\{x : \omega(x) = 1\}$ il raggio degli *inlier* incondizionati.

3.11 Trasformata di Hough

Sia $g(\mathbf{x}, \boldsymbol{\beta}) = 0$ una varietà continua in $\mathbf{x}$ di cui è richiesto stimare i parametri $\boldsymbol{\beta} \in \mathbb{R}^m$. Per ricavare tali parametri e poter definire completamente la funzione, sono disponibili un insieme di coordinate $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ che appartengono al luogo dei punti della funzione, potenzialmente affetti da rumore ma sopratutto potenzialmente *outlier*.

La trasformata di Hough (*Hough Transform*) è una tecnica che permette di raggruppare un insieme "molto probabile" di punti che soddisfano alcuni vincoli parametrici [PIK92].

Per ogni possibile punto $\boldsymbol{\beta}^*$ nello spazio dei parametri è possibile associare un voto $H(\boldsymbol{\beta})$ del tipo

$$H(\boldsymbol{\beta}^*) = \{\mathbf{x} : g(\mathbf{x}, \boldsymbol{\beta}^*) = 0, \mathbf{x} \in S\} \tag{3.120}$$

'ovvero il numero degli elementi di S che soddisfano il vincolo espresso da g. Il parametro $\boldsymbol{\beta}^*$ che massimizza tale voto è la soluzione statisticamente più probabile al problema.

Sia ora la funzione $p(\mathbf{x}, \boldsymbol{\beta})$ un indice di verosimiglianza tra la coppia $(\mathbf{x}, \boldsymbol{\beta})$ e il vincolo espresso da $g(\mathbf{x}, \boldsymbol{\beta}) = 0$. La funzione p normalmente è una funzione binaria, ma generalizzando può rappresentare tranquillamente una probabilità. Attraverso la funzione p è possibile costruire la trasformata di Hough $H(\boldsymbol{\beta})$ in maniera incrementale attraverso

$$H(\boldsymbol{\beta}) = \sum_{i=1}^{n} p(\mathbf{x}_i, \boldsymbol{\beta}) \tag{3.121}$$

La trasformata di Hough è la somma di tutte queste funzioni.

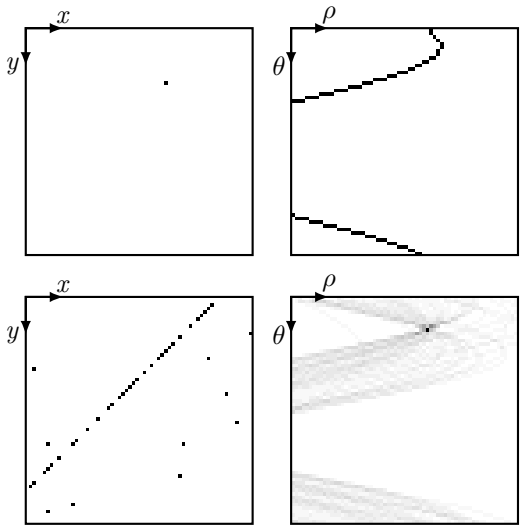


Figura 3.4: Esempio di Trasformata di Hough per individuare rette in coordinate polari: mappa accumulatore (in alto a destra) di un singolo punto (in alto a sinistra), e mappa accumulatore (in basso a destra) di una serie di punti colineari insieme ad *outlier* (in basso a sinistra).

Per particolari vincoli è possibile semplificare ulteriormente quest'approccio, in modo da ridurre il peso computazionale e l'utilizzo della memoria.

Siano pertanto $\beta_1 \dots \beta_m$ parametri da stimare, quantizzabili e limitati, e siano f e β_1 una funzione e un parametro tali che si possa scrivere la funzione $g(\mathbf{x}, \boldsymbol{\beta}) = 0$ come

$$\beta_1 = f(\mathbf{x}, \beta_2 \dots \beta_m) \tag{3.122}$$

Se la funzione g è esprimibile come in equazione (3.122), è possibile attraverso il metodo della trasformata di Hough discreta stimare i parametri $\boldsymbol{\beta}$ che rappresentano il modello più "probabile" fra tutti i punti $\mathbf{x}$ forniti. Per ogni elemento $\mathbf{x}$ è possibile far variare i parametri $\beta_2 \dots \beta_m$ nel loro intervallo e inserire nell'immagine accumulatore $H(\boldsymbol{\beta})$ i valori di β_1 restituiti dalla funzione (3.122).

In questo modo è possibile generare una mappa n-dimensionale di probabilità usando osservazioni $\mathbf{x}$ affette da rumore e potenzialmente *outliers*. Allo stesso modo il metodo di Hough permette di stimare un modello in presenza di una mistura di modelli con parametri differenti.

Il metodo di Hough permette prestazioni via via migliori man mano che il numero di vincoli aumenta, limitando dinamicamente per esempio il campo dei parametri associati al campione $\mathbf{x}$. L'algoritmo di Hough può essere visto come una forma degenere di *template matching*.

Normalmente risulta interessante l'uso di Hough dove il modello ha solo 2 parametri in quanto facilmente graficabile su una mappa bidimensionale.

Un esempio molto comune della trasformata di Hough è quello dove g (il modello) è una retta, espressa in forma polare come in equazione (1.46), dove i parametri da ricavare sono θ e ρ: risulta evidente che per ogni coppia di punti (x, y) e per tutti i possibili angoli di θ quantizzati e limitati (in quanto angolo è un parametro limitato) esiste uno e un solo ρ che soddisfa l'equazione (1.46).

È pertanto possibile creare mappa $H(\theta, \rho)$ dove per ogni punto $(x, y) \in S$ e per ogni $\theta \in [\theta_{min}, \theta_{max}]$ viene incrementata sulla mappa accumulatore l'elemento associato a $(\theta, \cos\theta x + \sin\theta y)$, relazione che soddisfa l'equazione (1.46) della retta scritta sotto forma di coordinate polari.

3.12 RANSAC

L'algoritmo di *RANdom Sample And Consesus* è un algoritmo iterativo per la stima dei parametri di un modello dove l'insieme dei dati è fortemente condizionato dalla presenza di *outlier*, algoritmo [FB81] non deterministico basato sulla selezione casuale degli elementi generatori del modello.

RANSAC, e tutte le sue varianti, possono essere viste come algoritmi che iterativamente alternano tra due fasi: la fase di generazione delle ipotesi (*hypothesis generation*) e la fase di valutazione delle ipotesi (*hypothesis evaluation*).

L'algoritmo, in breve, consiste nel selezionare casualmente s campioni fra tutti gli n campioni in ingresso $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, con s grande abbastanza per ricavare un modello (l'ipotesi). Ottenuta un'ipotesi, vengono contati quanti degli n elementi di X sono abbastanza vicini ad essa per appartenergli. Un campione $\mathbf{x} \in S$ appartiene o meno al modello ipotetico (ovvero è un *inlier* o un *outlier* ipotetico) se la sua distanza rispetto al modello $d_{\boldsymbol{\beta}}(\mathbf{x})$ è inferiore o superiore a una soglia data τ, soglia normalmente dipendente dal problema. La soglia τ si scontra con quei problemi pratici dove l'errore additivo è

di tipo gaussiano, ovvero dove il supporto è infinito. In questo caso è comunque necessario definire una probabilità p di individuazione degli *inlier* per definire una soglia τ.

Tutti i camptioni che soddisfano l'ipotesi si chiamano consensi (*consensus*).

L'insieme dei consensi S associati all'ipotesi $\boldsymbol{\beta}$ è il *consensus set* di $\boldsymbol{\beta}$:

$$S(\boldsymbol{\beta}) = \{\mathbf{x} \in X : d_{\boldsymbol{\beta}}(\mathbf{x}) < \tau\} \tag{3.123}$$

Tra tutti i modelli generati casualmente viene scelto il modello che soddisfa una determinata metrica, per esempio, per RANSAC originale, quella che ha il consenso di cardinalità massima.

Uno dei problemi è scegliere quante ipotesi generare per avere una buona probabilità di ottenere il modello corretto.

Esiste una relazione statistica tra il numero di iterazioni N e la probabilità p di individuare una soluzione di soli *inlier*. Il numero di tentativi N deve soddisfare $(1-P)^N \leq 1-p$ ovvero

$$N \geq \frac{\log(1-p)}{\log(1-P)} \tag{3.124}$$

dove P è la probabilità di avere scelto una soluzione fatta solamente di *inlier*.

Normalmente come approssimazione di P si può usare $P = (1-\epsilon)^s$[3] e perciò

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)} \tag{3.125}$$

con ϵ la probabilità a priori di selezionare un *outlier* e s il numero di elementi necessari a definire il modello. La dimensione di un *consensus set* minimo può essere dedotta in via statistica semplicemente come $T = (1-\epsilon)n$.

Normalmente s viene scelto uguale al numero di elementi necessari per creare il modello ma, se superiore a questo numero, il modello che viene generato deve essere un modello costruito attraverso una regressione numerica rispetto ai vincoli forniti, condizione necessaria quando la varianza del rumore è elevata, incrementando tuttavia il rischio di inglobare nei vincoli anche *outliers*.

3.12.1 M-SAC

La politica di RANSAC è quella di restituire, fra tutte le ipotesi generate, quella che possiede il minor numero di elementi esterni a una soglia fissata. Questa politica può essere vista come un *M-estimator* che minimizza una *loss function* del tipo

$$\rho = \begin{cases} 0 & |e| < \tau \\ 1 & |e| > \tau \end{cases} \tag{3.126}$$

'ovvero che assegna come voto 1 a tutti gli elementi più distanti della soglia dal modello valutato e a 0 gli elementi all'interno della soglia τ.

Il concetto si può pertanto generalizzare, nelle tecniche M-SAC (*M-Estimator Sample and Consensus*), dove la *loss function* di RANSAC viene modificata.

Come segnalato nella sezione precedente il rumore sui dati può essere visto in parte come rumore gaussiano sugli *inliers* associato a una distribuzione uniforme di *outliers*. La *negative Maximum Likelihood* è di fatto la *loss function* teoricamente corretta, base dei metodi MLESAC, ma abbastanza onerosa dal punto di vista computazionale.

Una buona approssimazione, propria delle tecniche M-SAC, è usare come *loss function*

$$\rho = \begin{cases} e^2 & |e| < \tau \\ \tau^2 & |e| > \tau \end{cases} \tag{3.127}$$

Questa *loss function* modella abbastanza bene il caso di *inlier* affetti da errore gaussiano a media nulla, e *outlier* distribuiti uniformemente.

3.12.2 LMedS

L'algoritmo di rigetto degli *outlier* chiamato *Least Median of Squares* (*LMedS*) è molto simile concettualmente a RANSAC: come per RANSAC viene generato un modello partendo da campionamenti casuali dai dati in ingresso ma, invece che scegliere il modello che raccoglie il maggior numero di consensi (o che minimizza una *loss function*), LMdeS seleziona fra tutti il modello che ha il valore mediano degli errori minore. Tutti i dati in ingresso pertanto vengono confrontati con il modello, ordinati per errore, ed esaminato il valore mediano.

La relazione tra probabilità di individuare *inlier* e numero di iterazioni è lo stesso di RANSAC. RANSAC tuttavia richiede due parametri (il numero di iterazioni e la soglia per discriminare se un elemento appartiene o meno al data-set), mentre LMedS richiede solo il primo. Per costruzione, LMedS tuttavia tollera al massimo la presenza del 50% di *outlier*.

Una buona panoramica delle tecniche RANSAC, M-SAC e LMedS si può trovare in [CKY09].

[3] La stima corretta è $P = \frac{\binom{pn}{s}}{\binom{n}{s}}$ con pn numero totale di *inlier*, n numero totale di elementi e s il numero di elementi necessari.

Capitolo 4

Classificazione

Un ruolo predominante nella Visione Artificiale rivestono le tecniche di Classificazione e di *machine learning*. La grande quantità di informazione che si può estrarre da un sensore video supera di gran lunga in quantità quella che si può ottenere da altri sensori ma richiedono tecniche complesse che permettano di sfruttare questa ricchezza di informazione.

Come già detto in precedenza, statistica, classificazione e *fitting* di modelli si possono vedere di fatto come diverse facce di un unico argomento. La statistica ricerca il modo più corretto dal punto di vista bayesiano per estrarre i parametri (dello stato o modello) nascosti di un sistema, affetto eventualmente da rumore, cercando, dati gli ingressi, di restituire l'uscita più probabile mentre la classificazione propone tecniche e modi su come modellare il sistema in maniera efficiente. Infine se si conoscesse il modello esatto che sottostà a un sistema fisico, qualunque problema di classificazione si ricondurrebbe a un problema di ottimizzazione. Per queste ragioni non è pertanto facile ne netto capire dove finisca un argomento e inizi l'altro.

Il problema della classificazione si riconduce a quello di ricavare i parametri di un modello generico che permetta di generalizzare il problema avendo a disposizione un numero limitato di esempi.

Un classificatore può essere visto in due modi, a seconda di che tipo di informazione tale sistema voglia fornire:

1. come funzione di "verosomiglianza" verso un modello, come in equazione (4.1);
2. come partizionamento dello spazio degli ingressi, come in equazione (4.2).

Nel primo caso un classificatore viene rappresentato da una generica funzione

$$f : \mathbb{R}^n \to \mathbb{R}^m \tag{4.1}$$

che permette di associare all'elemento $\mathbf{x}$ in ingresso, formato dalle n caratteristiche rappresentanti l'*esempio* da classificare, dei valore di confidenza rispetto alle possibili $\{y_1, \ldots, y_m\}$ classi di uscita (*categorie*):

$$f(\mathbf{x}) = (p(y_1|\mathbf{x}), \ldots, p(y_m|\mathbf{x}))$$

ovvero la probabilità che l'oggetto osservato sia proprio y_i data la quantità osservata $\mathbf{x}$.

A causa sia dell'infinità delle possibili funzioni sia della mancanza di ulteriori informazioni specifiche sulla forma del problema, la funzione f non potrà essere un funzione ben specifica ma verrà rappresentata da un modello a parametri nella forma

$$\mathbf{y} = f(\mathbf{x}, \boldsymbol{\beta})$$

dove $\mathbf{y} \in \mathbb{R}^m$ è lo spazio degli output, $\mathbf{x} \in \mathbb{R}^n$ spazio degli input mentre $\boldsymbol{\beta}$ sono i parametri del modello f da determinare nella fase di addestramento.

La fase di addestramento si basa su un insieme di esempi (*training set*) formato da coppie $(\mathbf{x}_i, \mathbf{y}_i)$ e attraverso questi esempi la fase di addestramento deve determinare i parametri $\boldsymbol{\beta}$ della funzione $f(\mathbf{x}, \boldsymbol{\beta})$ che minimizzino, sotto una determinata metrica (funzione di costo), l'errore sul *training set* stesso.

Per addestrare il classificatore bisogna pertanto individuare i parametri ottimi $\boldsymbol{\beta}$ che minimizzano l'errore nello spazio delle uscite: la classificazione è anche un problema di ottimizzazione. Per questa ragione *machine learning*, *fitting* di modelli e statistica risultano ambiti di ricerca strettamente legati. Le medesime considerazioni usate in Kalman o per Hough e tutto ciò detto nel capitolo di *fitting* di modelli ai minimi quadrati si possono usare per classificare e gli algoritmi specifici di classificazione possono essere usati ad esempio per adattare una serie di osservazioni affette da rumore a una curva.

Normalmente non è possibile produrre un insieme di addestramento completo: non è infatti sempre possibile ottenere qualsiasi tipo di associazione ingresso-uscita in modo da mappare in maniera sistematica tutto lo spazio degli *input* nello spazio degli *output* e, se ciò fosse anche possibile, risulterebbe comunque dispendioso disporre della memoria necessaria per rappresentare tali associazioni sotto forma di *Look Up Table*. Queste solo le principali ragioni dell'utilizzo di modelli nella classificazione.

Il fatto che il *training set* non possa coprire tutte le possibili combinazioni ingresso-uscita, combinato alla generazione di un modello ottimizzato verso tali dati incompleti, può provocare una non-generalizzazione dell'addestramento: elementi

non presenti nell'insieme di addestramento potrebbero essere classificati in maniera errata a causa dell'eccessivo adattamento al *training set* (problema dell'*overfitting*). Questo fenomeno è causato normalmente da una fase di ottimizzazione che si preoccupa più di ridurre l'errore sulle uscite piuttosto che di generalizzare il problema.

Tornando ai modi per vedere un classificatore, risulta spesso più semplice e più generalizzante ricavare direttamente dai dati in ingresso una superficie in $\mathbb{R}^n$ che separi le categorie nello spazio n-dimensionale degli ingressi. Si può definire una nuova funzione g che ad ogni gruppo di ingressi associ una ed una sola etichetta in uscita, nella forma

$$g : \mathbb{R}^n \to \mathbb{Y} = \{y_1, \ldots, y_m\} \tag{4.2}$$

Questo è il secondo modo di vedere un classificatore.

L'espressione (4.1) può essere sempre convertita nella forma (4.2) attraverso una votazione per maggioranza:

$$g(\mathbf{x}) = \arg\max_{y_i} p(y_i|\mathbf{x}) \tag{4.3}$$

Il classificatore, sotto questo punto di vista, è una funzione che restituisce direttamente il simbolo più somigliante all'ingresso fornito. Il *training set* ora deve associare a ogni ingresso (ogni elemento dello spazio) una ed una sola classe $y \in \mathbb{Y}$ in uscita. Solitamente questo modo di vedere un classificatore permette di ridurre la complessità computazionale e l'utilizzo di risorse.

Se la funzione (4.1) rappresenta effettivamente una funzione di trasferimento, una risposta, la funzione (4.2) può essere vista come un partizionamento dello spazio $\mathbb{R}^n$ dove a regioni, generalmente molto complesse e non contigue dello spazio degli ingressi, è associata un'unica classe.

Per le motivazioni addotte in precedenza non è fisicamente possibile realizzare un classificatore ottimo (se non per problemi di dimensioni molto contenute o per modelli semplici e conosciuti perfettamente) ma esistono diversi classificatori *general purpose* che a seconda del problema e delle performance richieste possono considerarsi sub-ottimi. Nel caso dei classificatori (4.2) il problema è quello di ottenere un partizionamento ottimo dello spazio e pertanto è richiesto un set di primitive veloci e tali da non usare troppa memoria nel caso di alti valori di n, mentre nel caso (4.1) è richiesta espressamente una funzione che modelli molto bene il problema evitando però specializzazioni.

Le informazioni (*features*) che si possono estrarre da una immagine per permetterne la classificazione sono molteplici. In genere usare direttamente i toni di grigio/colore dell'immagine è raramente usato in applicazioni pratiche perché tali valori sono normalmente influenzati dalla luminosità della scena e sopratutto perché rappresenterebbero uno spazio di ingresso molto vasto, difficilmente gestibile. È necessario pertanto estrarre dalla parte di immagine da classificare delle informazioni essenziali (*features*) che ne descrivano l'aspetto al meglio. Per questa ragione, tutta la teoria mostrata in sezione 6 è ampiamente usata in *machine learning*. Sono ampiamente usati infatti, sia le *feature di Haar* grazie alla loro velocità di estrazione o gli Istogrammi dell'Orientazione del Gradiente (*HOG*, sez. 6.2) per la loro accuratezza. Come compromesso, e loro generalizzazione, tra le due classi di *feature* di recente sono state proposte le Feature su Canali Integrali (*ICF*, sez. 6.3).

Per ridurre la complessità del problema di classificazione questo può essere diviso in più strati da affrontare in maniera indipendente: un primo strato trasforma lo spazio degli ingressi nello spazio delle caratteristiche, mentre un secondo livello esegue la classificazione vera e propria partendo dallo spazio delle caratteristiche.

Sotto questa considerazione le tecniche di classificazione si possono dividere in 3 categorie principali:

Rule-based learning In questo caso sia lo spazio delle caratteristiche che i parametri della funzione di classificazione sono decisi da un utente umano, senza sfruttare alcun insieme di dati o esempi di addestramento;

Machine Learning La trasformazione tra spazio di ingressi a spazio delle caratteristiche è scelta dall'utente tra un insieme finito di funzioni, mentre l'estrazione dei parametri del modello è lasciata all'elaboratore analizzando gli esempi forniti;

Representation learning Sia la trasformazione in spazio delle caratteristiche che l'estrazione dei parametri del modello sono attuati dal calcolatore.

Recentemente tecniche di *Representation learning* costruite con più strati in cascata tra loro (*Deep Learning*) hanno avuto molto successo a risolvere problemi di classificazione complesse.

Tra le tecniche per trasformare lo spazio di ingressi nello spazio delle caratteristiche è importante citare la PCA, tecnica lineare non supervisionata. La *Principal Component Analysis* (sezione 2.10.1) è una tecnica che permette di ridurre il numero di ingressi al classificatore, rimuovendo le componenti linearmente dipendenti o ininfluenti, riducendo pertanto la dimensione del problema cercando comunque di preservare al massimo l'informazione.

Per quanto riguarda i modelli e le tecniche di modellazione *general purpose* molto utilizzate sono

Regressione una regressione di dati ad un modello è di fatto un classificatore. Di conseguenza, tutta la teoria del capitolo 3 può e deve essere usata per classificare dati;Tra

Neural Network Le reti neurali permettono di generare funzioni di tipo (4.1) concatenando tra loro somme, moltiplicazioni e funzioni fortemente non lineari come le sigmoidi. Tecniche di regressione permettono di stimare i parametri di questo modello generico;

Classificatori Bayesiani è possibile usare il teorema di Bayes direttamente come classificatore o per unire insieme più classificatori in modo da massimizzare la probabilità *a posteriori* di individuare la classe corretta (sezione 4.2);

Albero di decisione dove i classificatori sono messi in cascata con altri classificatori (ed ogni nodo rappresenta un qualche attributo estratto dai dati in ingresso);

Decision Stump albero di decisione degenere (1 nodo), permette di partizionare lo spazio delle *features* usando una semplice soglia, diventando così il più semplice classificatore di tipo (4.2) ed esempio di classificatore debole;

Ensemble Learning Più classificatori deboli (*weak*) possono essere messi in relazione tra loro (*Ensemble Learning*, sezione 4.6) in modo da massimizzare qualche metrica globale (ad esempio il margine di separazione tra le classi). Di fatto non sono veri e propri classificatori ma sono tecniche per unire più classificatori semplici e generare un classificatore complesso (*ensemble*).

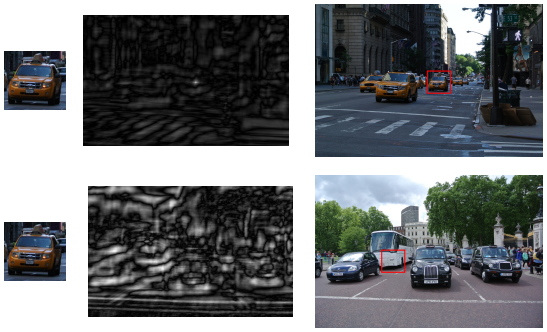


Figura 4.1: Esempio di *Template Matching*. L'approccio funziona bene sull'immagine di origine, ma non è possibile estenderlo ad altre immagini, soprattutto con variazioni di luminosità e scala sensibili.

4.1 Classificatori Binari

Un particolare caso, molto diffuso, di classificatore è quello di classificatore binario. In questo caso il problema consiste nel cercare una relazione che leghi il *training-set* $S = \{(\mathbf{x}_1, y_1) \dots (\mathbf{x}_l, y_l)\} \in (\mathbb{X} \times \mathbb{Y})$ dove $\mathbb{X} \subseteq \mathbb{R}^n$ è il vettore che raccoglie le informazioni da usare per l'addestramento e $\mathbb{Y} = \{+1, -1\}$ lo spazio delle classi associate.

Esempi di classificatori intrinsecamente binari sono:

LDA la *Linear Discriminant Analysis* (sezione 4.3) è una tecnica che permette di trovare il piano di separazione tra le classi che massimizza la distanza tra le distribuzioni;

Decision Stump Gli alberi di decisione a un solo livello hanno solo due possibili uscite;

SVM le Macchine a Vettori di Supporto *Support Vector Machines* (sezione 4.4) partizionano, massimizzando il margine, lo spazio delle *feature* usando iperpiani o semplici superfici.

Un particolare interesse ricoprono i classificatori lineari (LDA e SVM-Lineare) i quali, per risolvere il problema di classificazione binaria, individuano un iperpiano $(\mathbf{w}, b)$ di separazione tra le due classi.

L'equazione di un iperpiano, modificando leggermente la formula (1.49), è

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{4.4}$$

dove il vettore normale $\mathbf{w}$ può anche non essere di norma unitaria. Un iperpiano divide lo spazio in due sottospazi dove l'equazione (4.4) ha segno opposto. La superficie di separazione è un iperpiano che divide lo spazio in due sotto parti rappresentanti le due categorie della classificazione binaria.

Un classificatore lineare è basato su una funzione discriminante

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{4.5}$$

Il vettore $\mathbf{w}$ è chiamato *weight vector* ed il termine b è chiamato *bias*. I classificatori lineari assumono una certa importanza in quanto, attraverso la proiezione lungo l'asse $\mathbf{w}$, trasformano il problema da multidimensionale a scalare.

Il segno della funzione $f(\mathbf{x})$ rappresenta il risultato della classificazione. Un iperpiano di separazione equivale ad individuare una combinazione lineare degli elementi $\mathbf{x} \in \mathbf{X}$ in modo da ottenere

$$\hat{y} = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b) \tag{4.6}$$

4.2 Classificatori bayesiani

Il teorema di *Bayes*, associato alla Visione Artificiale, rappresenta una tecnica fondamentale per la classificazione di pattern, basata sull'esperienza (*training set*).

Per capire il teorema di Bayes è necessario fare un semplice esempio. Si supponga di voler classificare della frutta che viene mostrata a un osservatore (un elaboratore nel caso estremo). Per semplicità si supponga che i tipi di frutta (le categorie del classificatore) siano solo due, per esempio, arance e mele. Per gli esseri umani, ma allo stesso modo deve essere fatto per le macchine, determinare la tipologia di frutta che si stà osservando, avviene esaminando determinate caratteristiche (*feature*) estratte dall'osservazione della frutta, attraverso opportune tecniche.

Se i frutti vengono scelti in maniera totalmente casuale e non è possibile estrarre alcuna altra informazione da essi, l'approccio ottimo per classificarli sarebbe fornire una risposta totalmente a caso.

La teoria bayesiana di decisione svolge un ruolo importante solo quando risultano conosciute alcune informazioni *a priori* sugli oggetti.

Come primo passo, si supponga di non avere comunque conoscenza alcuna su come siano fatti i frutti, ma si sa che l'80% della frutta sono mele ed il resto siano arance. Se questa è l'unica informazione su cui basare la decisione, istintivamente si tenderà a classificare la frutta come mela (il classificatore ottimo): ogni frutto verrà classificato come mela in quanto, in mancanza di altre informazioni, è l'unico modo per minimizzare l'errore. Le informazioni *a priori* in questo caso sono le probabilità che il frutto scelto sia una mela o un'arancia.

Esaminiamo a questo punto il caso in cui sia possibile estrarre qualche informazione in più dalla scena osservata. Il concetto di Bayes applicato alla classificazione è molto intuitivo anche da questo punto di vista: se osservo una particolare caratteristica misurabile dell'immagine x (*features*) riesco a stimare la probabilità che tale immagine rappresenti una certa classe y_i *a posteriori* dell'osservazione. Sotto questo punto di vista i classificatori bayesiani forniscono esattamente la probabilità che il vettore di dati in ingresso rappresenti la determinata classe in uscita.

4.2.1 Il teorema di Bayes

La definizione di probabilità condizionata ci permette di ottenere immediatamente il seguente fondamentale

Teorema 2 (di Bayes) *Sia $\{\Omega, \mathcal{Y}, p\}$ uno spazio probabilizzato. Siamo gli eventi $y = y_i$ (abbreviato con y_i) con $i = 1..n$ un sistema completo di eventi di Ω e $p(y_i) > 0\ \forall i = 1..n$.*

In questo caso $\forall y_i \in \mathcal{Y}$ con $p(y_i) > 0$ si avrà che:

$$p(y_i|x) = \frac{p(y_i)p(x|y_i)}{\sum_{j=1}^{n} p(y_j)p(x|y_j)} \tag{4.7}$$

e questo $\forall i = 1..n$.

Il teorema di Bayes costituisce uno degli elementi fondamentali dell'approccio *soggettivista*, o *personale*, alle probabilità e all'inferenza statistica. Il sistema di alternative y_i con $i = 1..n$ viene spesso interpretato come un insieme di *cause* e il teorema di Bayes, note le probabilità iniziali delle diverse cause, permette di assegnare probabilità alle cause dato un effetto x. Le probabilità $p(y_i)$ con $i = 1..n$ possono essere interpretate come le conoscenze *a priori* (solitamente indicate con π_i), ossia quelle che si hanno prima di effettuare un *esperimento statistico*. Le probabilità $p(x|y_i)$ con $i = 1..n$ vengono interpretate come la *verosimiglianza* o informazione relativa a x acquisibile eseguendo un opportuno esperimento statistico. La formula di Bayes suggerisce dunque un meccanismo di *apprendimento dall'esperienza*: coniugando alcune conoscenze a priori sull'evento y_i date da $p(y_i)$ con quelle acquisibili da un esperimento statistico date da $p(x|y_i)$ si perviene ad una migliore conoscenza data da $p(x_i|y)$ dell'evento x_i detta anche *probabilità a posteriori* dopo aver eseguito l'esperimento.

Possiamo avere, per esempio, la distribuzione di probabilità per il colore delle mele, così come quella per le arance. Per usare la notazione introdotta in precedenza nel teorema, chiamiamo y_1 lo stato in cui la frutta sia una mela, y_2 la condizione in cui la frutta sia un'arancia e sia la x una variabile casuale che rappresenti il colore della frutta. Con questa notazione, $p(x|y_1)$ rappresenta la funzione densità per l'evento colore x subordinato al fatto che lo stato sia mela, $p(x|y_2)$ che sia arancia.

In fase di addestramento è possibile costruire la distribuzione di probabilità di $p(x|y_i)$ per i mela o arancia. Oltre a questa conoscenza sono sempre note le probabilità a priori $p(y_1)$ e $p(y_2)$, che rappresentano semplicemente il numero totale di mele contro il numero di arance.

Quello che stiamo cercando è una formula che dica quale è la probabilità di una frutta di essere mela o un'arancia, avendo osservato un certo colore x.

La formula di Bayes (4.7) permette proprio questo:

$$p(y_i|x) = \frac{p(x|y_i)p(y_i)}{p(x)} \tag{4.8}$$

date le conoscenze a priori, permette di calcolare la probabilità a posteriori che lo stato della frutta sia y_i data la *feature* misurata x. Pertanto, osservato un certo x sul nastro trasportatore, calcolati $p(y_1|x)$ e $p(y_2|x)$, si sarà inclini a decidere che

la frutta è una mela se il primo valore sarà maggiore del secondo (o viceversa):

$$p(y_1|x) > p(y_2|x)$$

ovvero:

$$p(x|y_1)p(y_1) > p(x|y_2)p(y_2)$$

In generale per n classi lo stimatore bayesiano si può definire come una *discrimant function*:

$$f(x) = \hat{y}(x) = \arg\max_i p(y_i|x) = \arg\max_i p(x|y_i)\pi_i \tag{4.9}$$

È anche possibile calcolare un indice, data la conoscenza a priori del problema, di quanto questo ragionamento sarà soggetto ad errori. La probabilità di compiere un errore data una *feature* osservata x sarà dipendente dal valore massimo delle n curve della distribuzione in x:

$$p(error|x) = 1 - \max\left[p(y_1|x), p(y_2|x), \ldots, p(y_n|x)\right] \tag{4.10}$$

4.2.2 Il classificatore bayesiano

Attraverso l'approccio bayesiano, sarebbe possibile costruire un classificatore *ottimo* se si conoscessero in maniera perfetta sia le probabilità a priori $p(y_i)$, sia le densità condizionate alla classe $p(x|y_i)$. Normalmente tali informazioni sono raramente disponibili e l'approccio adottato è quello di costruire un classificatore da un insieme di esempi (*training set*).

Per modellare $p(x|y_i)$ si utilizza normalmente un *approccio parametrico* e quando possibile, si fa coincidere tale distribuzione con quella di una gaussiana o con delle funzioni spline.

Le tecniche più usate per la stima sono la *Maximum-Likelihood (ML)* e la *Stima Bayesiana* che, sebbene differenti nella logica, portano a risultati quasi identici. La distribuzione gaussiana è normalmente un modello appropriato per la maggior parte dei problemi di *pattern recognition*.

Esaminiamo il caso abbastanza comune nel quale la probabilità delle varie classi è di tipo gaussiano multivariato di media $\boldsymbol{\mu}_i$ e matrice di covarianza $\boldsymbol{\Sigma}_i$. Il classificatore bayesiano ottimo è

$$\begin{aligned} \hat{y}(\mathbf{x}) = &\ \arg\max_i p(\mathbf{x}|y_i)\pi_i \\ = &\ \arg\max_i \log\left(p(\mathbf{x}|y_i)\pi_i\right) \\ = &\ \arg\min_i \left((\mathbf{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) - \log\det\boldsymbol{\Sigma}_i - 2\log\pi_i\right) \end{aligned} \tag{4.11}$$

usando la *negative log-likelihood* (sezione 2.8). Nel caso di probabilità a priori π_i uguali, l'equazione (4.11) coincide con il problema di cercare il minimo della distanza di Mahalanobis (sezione 2.4) tra le classi del problema.

4.2.3 Naive Bayes

Normalmente con una sola caratteristica estratta dall'oggetto da classificare non è possibile ottenere una precisione elevata di classificazione. Fortunatamente le caratteristiche che si possono estrarre da una immagine sono molteplici.

Siano indicate con x_j, con $j = 1, \ldots, m$ tali caratteristiche. È molto importante notare che gli eventi osservati x_j con cui costruire il classificatore bayesiano devono essere eventi indipendenti (indipendenza condizionale), altrimenti il teorema di Bayes non risulta più valido (uno dei limiti dei classificatori bayesiani): per esempio non si possono unire classificatori che analizzino parti dell'immagine in comune o non si può unire lo stimatore "è arancione" insieme a "è non rosso".

L'assunzione *Naive Bayes* (o *idiot Bayes*) sfrutta l'ipotesi semplificativa di indipendenza degli attributi (*feature*) osservati: in questo caso date m variabili osservate $x_1 \ldots x_m$ la probabilità che l'evento y_i si verifichi sarà:

$$p(x_1 \ldots x_m|y_i) = \prod_{j=1}^{m} p(x_j|y_i) \tag{4.12}$$

4.3 LDA

Un esempio di riduzione delle dimensioni del problema a scopo di classificazione è la Analisi di Discriminante Lineare *Linear Discriminant Analysis* (Fisher, 1936).

Se si analizza il funzionamento di PCA (sezione 2.10.1), questa tecnica si limita a massimizzare l'informazione non distinguendo tra loro le eventuali classi che compongono il problema: PCA non considera il fatto che i dati siano rappresentativi di diverse categorie. PCA non è un vero classificatore ma è una tecnica utile a semplificare il problema, riducendone le dimensioni. LDA cerca invece di massimizzare sia l'informazione discriminatoria tra le classi che l'informazione rappresentata dalla varianza.

Nel caso di un problema di due classi, il miglior classificatore bayesiano è quello che permette di individuare il margine di decisione (*decision boundary*) formato dall'ipersuperficie lungo la quale la probabilità condizionata delle due classi è uguale.

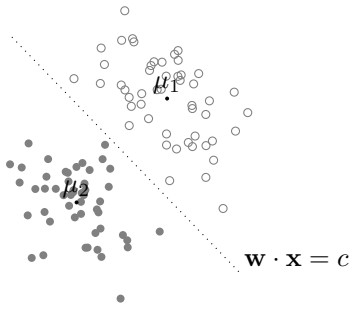


Figura 4.2: Analisi di Discriminante Lineare.

Se si forza l'ipotesi che le due classi del problema binario abbiano distribuzione gaussiana multivariata e uguale matrice di covarianza $\mathbf{\Sigma}$ è facile dimostrare che il margine di decisione bayesiano, equazione (4.11), diventa lineare.

In LDA viene fatta pertanto l'ipotesi di omoschedasticità e, sotto questa ipotesi, si vuole ottenere un vettore $\mathbf{w}$ che permetta di proiettare lo spazio n-dimensionale degli eventi in uno spazio scalare che però massimizzi la separazione tra le classi e permetta di separarle linearmente attraverso un margine di separazione del tipo

$$\mathbf{w}^\top \mathbf{x} = c \tag{4.13}$$

Per determinare questa superficie di separazione si possono usare diverse metriche. Sotto il termine LDA attualmente confluiscono diverse tecniche dove la Discriminante di Fisher (*Fisher's Linear Discriminant Analysis*) risulta la più diffusa in letteratura.

Si può dimostrare che la proiezione che massimizza la separazione tra le due classi dal punto di vista "statistico", ovvero l'iperpiano di decisione, si ottiene con

$$\mathbf{w} = \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{4.14}$$

e il valore di separazione ottimo si trova a metà strada tra le proiezioni delle due medie

$$c = \mathbf{w}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)/2 \tag{4.15}$$

nel caso in cui le probabilità a priori dei due insiemi siano identiche.

Questo margine di decisione è la soluzione alla massima verosomiglianza in caso di due classi con distribuzione uniforme e stessa covarianza.

4.4 SVM

LDA si pone come obiettivo quello di massimizzare la distanza statistica tra le classi ma non cerca di valutare quale sia l'effettivo margine di separazione tra di loro.

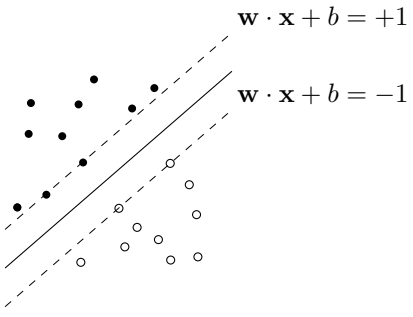


Figura 4.3: Iperpiano di separazione tra due classi ottenuto attraverso SVM. I punti sul margine (tratteggiato) sono i *Support Vectors*.

SVM [CV95], come LDA, permette di ottenere un classificatore lineare basato su una funzione discriminante nella stessa forma mostrata in equazione (4.5). SVM però va oltre: l'iperpiano ottimo in $\mathbb{R}^n$ viene generato in maniera tale da separare "fisicamente" (*decision boundary*) gli elementi del problema di classificazione binario ovvero si pone come obiettivo quello di massimizzare il margine di separazione tra le classi. Questo ragionamento premia molto per quanto riguarda la generalizzazione del classificatore.

Siano pertanto definite come classi di classificazione quelle tipiche di un problema binario nella forma $y_i = \{+1, -1\}$ e si faccia riferimento all'iperpiano di formula (4.4). Supponiamo che esistano dei parametri $(\mathbf{w}_0, b_0)$ ottimi tali che soddisfino il

vincolo

$$\begin{aligned} \mathbf{x}_i \cdot \mathbf{w}_0 + b_0 \geq +1 \quad \text{per } y_i = +1 \\ \mathbf{x}_i \cdot \mathbf{w}_0 + b_0 \leq -1 \quad \text{per } y_i = -1 \end{aligned} \tag{4.16}$$

ovvero, in forma più compatta:

$$y_i(\mathbf{x}_i \cdot \mathbf{w}_0 + b_0) - 1 \geq 0 \tag{4.17}$$

per ogni $(y_i, \mathbf{x}_i)$ campioni forniti durante la fase di addestramento.

Si può supporre che esistano, per ognuna delle categorie, uno o più vettori $\mathbf{x}_i$ dove le disuguaglianze (4.17) diventano uguaglianze. Tali elementi, chiamati *Support Vectors*, sono i punti più estremi della distribuzione e la loro distanza rappresenta la misura del margine di separazione tra le due categorie.

La distanza ρ punto-piano (cfr. eq.(1.52)) vale

$$\rho = \frac{\|\mathbf{w} \cdot \mathbf{x} + b\|}{\|\mathbf{w}\|} \tag{4.18}$$

Dati due punti di classe opposta che soddisfino l'uguaglianza (4.17), il margine può essere ricavato dall'equazione (4.18), e vale

$$\rho = \frac{2}{\|\mathbf{w}_0\|} \tag{4.19}$$

Per massimizzare il margine ρ dell'equazione (4.19) bisogna minimizzare la sua inversa, ovvero

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \tag{4.20}$$

sotto la serie di vincoli espressi dalla diseguaglianza (4.17). Questo è quello che viene definito come problema di ottimizzazione *primale* in forma standard dell'SVM.

Questa classe di problemi (minimizzazione con vincoli come disuguaglianze o *primal optimization problem*) si risolvono utilizzando l'approccio di Karush-Kuhn-Tucker che è il metodo dei moltiplicatori di Lagrange generalizzato a disuguaglianze. Attraverso le condizioni KKT si ottiene la funzione lagrangiana:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i \left(y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1\right) \tag{4.21}$$

da minimizzare in $\mathbf{w}$ e b e massimizzare in $\boldsymbol{\alpha}$. I pesi $\alpha_i \geq 0$ sono i moltiplicatori di Lagrange. Dall'annullamento delle derivate parziali si ottiene

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum y_i \alpha_i = 0 \tag{4.22}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \rightarrow \mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \tag{4.23}$$

Sostituendo tali risultati (le variabili primali) all'interno della lagrangiana (4.21) questa diventa funzione dei soli moltiplicatori, i *dual*, da cui la forma duale di Wolfe:

$$\Psi(\boldsymbol{\alpha}) = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{4.24}$$

sotto i vincoli $\alpha_i \geq 0$ e $\sum \alpha_i y_i = 0$. Il massimo della funzione Ψ calcolato su $\boldsymbol{\alpha}$ sono gli α_i associati a ogni vettore di addestramento $\mathbf{x}_i$. Tale massimo permette di trovare la soluzione del problema originale.

Su questa relazione sono valide le condizioni KKT tra le quali è di notevole importanza il vincolo, detto di *Complementary slackness*,

$$\alpha_i \left(y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1\right) = 0 \tag{4.25}$$

Questo vincolo dice che il massimo della lagrangiana o è sul bordo del vincolo ($\alpha_i \neq 0$) o è un minimo locale ($\alpha_i = 0$). Come conseguenza solo gli α_i sul limite sono non nulli e contribuiscono alla soluzione: tutti gli altri campioni di addestramento sono di fatto ininfluenti. Tali vettori, associati agli $\alpha_i > 0$, sono i *Support Vectors*.

Risolvendo il problema quadratico (4.24), sotto il vincolo (4.22) e $\alpha_i \geq 0$, i pesi che presentano $\alpha_i \neq 0$ saranno i *Support Vectors*. Tali pesi, inseriti nelle equazioni (4.23) e (4.25), porteranno a ricavare l'iperpiano di massimo margine.

Il metodo più usato per risolvere questo problema QP è il *Sequential Minimal Optimization* (*SMO*). Per una trattazione approfondita delle tematiche legate a SVM si può fare riferimento a [SS02].

4.4.1 Soft Margin SVM

In applicazioni reali non sempre esiste un margine, ovvero non sempre le classi sono linearmente separabili nello spazio delle *features* attraverso un iperpiano. Il concetto alla base del *Soft Margin* permette di ovviare a questo limite, introducendo una variabile ξ aggiuntiva per ogni campione, in modo da rilassare (*slack*) il vincolo sul margine

$$\begin{aligned} &y_i(\mathbf{w}\cdot\mathbf{x}_i+b)\geq 1-\xi_i \\ &\xi_i\geq 0, \forall i \end{aligned} \tag{4.26}$$

Il parametro ξ rappresenta la *slackness* associata al campione. Quando $0<\xi\leq 1$ il campione è correttamente classificato ma è all'interno dell'area di margine. Quando $\xi>1$ il campione entra nello spazio di decisione della classe opposta e perciò verrà classificato in maniera errata.

Per cercare ancora un iperpiano di separazione in qualche modo ottimo, la funzione costo da minimizzare deve considerare anche la distanza tra il campione e il margine:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum \xi_i \tag{4.27}$$

soggetta ai vincoli (4.26). Il parametro C è un grado di libertà del problema per indicare quanto un campione deve pagare il violare il vincolo sul margine. Quando C è piccolo, il margine è ampio, mentre quando C è prossimo a infinito si ricade alla formulazione *Hard Margin* di SVM vista in precedenza.

Ogni campione $\mathbf{x}_i$ può ricadere in uno di tre possibili stati:

- può stare oltre il margine $y_i(\mathbf{w}^\top\mathbf{x}_i+b)>1$ e di conseguenza non contribuire alla funzione;
- può stare sul margine $y_i(\mathbf{w}^\top\mathbf{x}_i+b)=1$ non partecipando direttamente alla minimizzazione ma solo come *support vector*,
- può infine cadere all'interno del margine ed essere penalizzato tanto quanto si discosta dai vincoli forti.

La lagrangiana del sistema (4.27), con i vincoli introdotti dalle variabili ξ, è

$$\mathcal{L}(\mathbf{w},b,\xi,\alpha)=\frac{1}{2}\|\mathbf{w}\|^2+C\sum_i\xi_i-\sum_i\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i+b)-1+\xi_i)-\sum_i\gamma_i\xi_i \tag{4.28}$$

Con l'aumento del numero di vincoli, le variabili duali sono sia $\boldsymbol{\alpha}$ che $\boldsymbol{\gamma}$.

Il risultato notevole è che, applicate le derivate, la formulazione duale di (4.28) diventa esattamente uguale alla duale del caso *Hard Margin*: le variabili ξ_i infatti non compaiono nella formulazione duale e l'unica differenza tra il caso *Hard Margin* e il caso *Soft Margin* è nel vincolo sui parametri α_i, in questo caso limitati tra

$$0\leq\alpha_i\leq C \tag{4.29}$$

invece che con la semplice diseguaglianza $\alpha_i\geq 0$. Il grande vantaggio di questa formulazione è proprio nella elevata semplicità dei vincoli e nel fatto che permetta di ricondurre il caso *Hard Margin* a un caso particolare ($C=\infty$) del *Soft Margin*. La costante C è un limite superiore al valore che gli α_i possono assumere.

4.4.2 SVM e funzioni kernel

Nonostante il Soft Margin, alcuni problemi sono intrinsecamente non separabili nello spazio delle feature. Tuttavia, dalla conoscenza del problema, è possibile intuire che una trasformazione non lineare $\phi: X\to F$ trasforma lo spazio delle feature di input $\mathcal{X}$ nello spazio delle feature $\mathcal{F}$ dove l'iperpiano di separazione permette di discriminare meglio le categorie. La funzione discriminante nello spazio $\mathcal{F}$ è

$$f(\mathbf{x})=\mathbf{w}^\top\phi(\mathbf{x})+b \tag{4.30}$$

Per permettere la separazione, normalmente lo spazio $\mathcal{F}$ è di dimensioni maggiori dello spazio $\mathcal{X}$. Questo aumento di dimensioni provoca un aumento della complessità computazionale del problema e la richiesta di risorse. I metodi Kernel risolvono questo problema.

Il vettore $\mathbf{w}$ è una combinazione lineare dei campioni di addestramento (i *support vector* nel caso *hard margin*):

$$\mathbf{w}=\sum_i\alpha_i\phi(\mathbf{x}_i) \tag{4.31}$$

La funzione discriminante assume pertanto la forma

$$\begin{aligned} f(\mathbf{x}) &= \textstyle\sum_i \alpha_i\phi(\mathbf{x}_i)^\top\phi(\mathbf{x})+b \\ &= \textstyle\sum_i \alpha_i k(\mathbf{x},\mathbf{x}_i)+b \end{aligned} \tag{4.32}$$

con la valutazione della funzione kernel $k(\mathbf{x}, \mathbf{x}')$.

Al momento della valutazione della funzione discriminante pertanto è richiesto l'utilizzo dei vettori di supporto (almeno quelli con un parametro α_i associato non trascurabile). Di fatto SVM con kernel individua alcuni campioni dell'insieme di addestramento come informazione utile per capire quanto vicino a loro è il campione di valutazione in esame.

Il *bias* si calcola istantaneamente dall'equazione (4.32), mediando

$$b = \mathrm{E}[y_j - \sum_i \alpha_i k(\mathbf{x}_j, \mathbf{x}_i)] \tag{4.33}$$

I kernel più diffusi, in quanto semplici da valutare, sono i kernel gaussiani nella forma

$$k(\mathbf{x}, \mathbf{x}') = e^{-\gamma \|\mathbf{x} - \mathbf{x}'\|^2} \tag{4.34}$$

con γ parametro da impostare, e i kernel polinomiali di grado d nella forma

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^d \tag{4.35}$$

e nel caso $d = 1$ la formulazione si riconduce al caso lineare.

L'utilizzo di funzioni kernel, unita alla possibilità di precalcolare tutte le combinazioni $k(\mathbf{x}_i, \mathbf{x}_j)$, permette di definire un'interfaccia comune tra gli addestramenti lineari e i non lineari, mantenendo di fatto lo stesso grado di prestazioni.

È da notare che le predizione $f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$ assume la forma

$$\phi(\mathbf{x}) = [k_1(\mathbf{x}, \mathbf{x}_1), \ldots, k_n(\mathbf{x}, \mathbf{x}_n)] \tag{4.36}$$

dove $\mathbf{x}_i$ rappresenta un sottoinsieme dell'addestramento. I modelli scritto in questa forma di fatto eseguono un *template matching* tra il campione $\mathbf{x}$ da valutare e i prototipi $\mathbf{x}_i$.

4.4.3 Minimizzazione Empirica del Rischio

Il vincolo *Soft Margin* (4.26) può essere riscritto come

$$y_i f(\mathbf{x}_i) \geq 1 - \xi_i \tag{4.37}$$

dove $f(\mathbf{x}_i)$ può essere anche la generica funzione kernel. Questa disequazione è equivalente a

$$\xi_i \geq \max(0, 1 - y_i f(\mathbf{x}_i)) \tag{4.38}$$

siccome $\xi_i \geq 0$. La funzione di perdita (4.38) è chiamata funzione perdita cardine (*Hinge Loss*)

$$\ell(y, \hat{y}) = \max(0, 1 - y\hat{y}) \tag{4.39}$$

e ha il vantaggio di essere convessa e non differenziabile solo in 1. La *hinge loss* è sempre maggiore della funzione perdita 0/1.

Il problema di addestramento di SVM nel caso non linearmente separabile è equivalente a un problema di ottimizzazione, non vincolato, su $\mathbf{w}$ del tipo

$$\min_{\mathbf{w} \in \mathbb{R}^d} \|\mathbf{w}\|^2 + C \sum_i^N \ell(y_i, f(\mathbf{x}_i)) \tag{4.40}$$

La funzione obiettivo continua ad essere descritta in due parti chiaramente distinte: la prima è la regolarizzazione di Tikhonov e la seconda è la minimizzazione del rischio empirica con la funzione di perdita Cardine. SVM può essere pertanto visto come un classificatore lineare che ottimizza la funzione di perdita Cardine con una regolarizzazione L2.

I dati di ingresso $\mathbf{x}_i$ possono cadere in 3 diverse categorie:

- $y_i f(\mathbf{x}_i) > 1$ sono i punti fuori dal margine e non danno nessun contributo alla funzione costo;
- $y_i f(\mathbf{x}_i) = 1$ sono i punti sul margine e non danno contributo al costo come nel caso "hard margin";
- $y_i f(\mathbf{x}_i) < 1$ sono i punti che violano il vincolo e contribuiscono al costo.

4.5 Classificazione multiclasse

SVM ritorna una funzione obiettivo $f(\mathbf{x}_i, \mathbf{W}, b) = \mathbf{W}\mathbf{x}_i + b$ il cui valore assoluto non ha un vero significato in quanto è un uscita non calibrata. L'estensione al caso multiclasse è difficile in quanto le differenti funzioni obiettivo per ogni classe non sono tra loro direttamente confrontabili.

Il concetto di *hinge loss* può però essere esteso al caso multiclasse. In questo caso viene definita una *SVM Loss* del tipo

$$\ell_i = \sum_{j \neq y_i} \begin{cases} 0, & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1, & \text{otherwise} \end{cases} = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \tag{4.41}$$

dove con $s_j = f_j(\mathbf{x}_i)$ è stato indicato, per semplicità, la funzione obiettivo associato alla classe j per il campione i-esimo.

Un'altra metrica simile è la *squared hinge loss*:

$$\ell_i = \sum_{j \neq y_i} \max\left(0, s_j - s_{y_i} + 1\right)^2 \tag{4.42}$$

Viene infine definita una funzione perdita sull'intero dataset come media

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \ell_i + \lambda R((W)) \tag{4.43}$$

con l'opzionale termine di regolarizzazione sui pesi.

Una metrica differente, estesa al caso multiclasse, è la funzione esponenziale normalizzata *Softmax*:

$$\ell_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} = -s_{y_i} + \log \sum_j e^{s_j} \tag{4.44}$$

La funzione obiettivo s_j può essere interpretata come una probabilità non normalizzata logaritmica per ogni classe e perciò si può sostituire la funzione perdita cardano con la funzione perdita entropia incrociata (*cross-entropy loss*). Un classificatore *Softmax* minimizza l'entropia incrociata tra le classi e siccome minimizza la *negative log likelihood* della classe corretta può essere visto come uno stimatore a massima verosimiglianza. Nel caso *Softmax* il termine di regolarizzazione $R((W))$ può essere visto, da un punto di vista statistico, come un *prior* sui pesi: in questo caso è una stima *Maximum a posteriori* (*MAP*).

4.6 Ensemble Learning

Il concetto di addestramento *Ensemble* richiama l'utilizzo di diversi classificatori, differenti, uniti in un certo modo per riuscire a massimizzare le prestazioni usando i punti di forza di ognuno e limitando le debolezze dei singoli.

Alla base del concetto di *Ensemble Learning* ci sono i classificatori deboli (*weak classifier*): un classificatore debole riesce a classificare almeno il $50\% + 1$ dei campioni di un problema binario. Sommati in un certo modo tra di loro, i classificatori deboli permettono di costruire un classificatore forte, risolvendo allo stesso tempo problemi tipici dei classificatori tradizionali (*overfitting* in primis).

L'origine dell'*Ensemble Learning*, del concetto di classificatore debole e in particolare il concetto di *probably approximately correct learning* (*PAC*) sono stati per primi introdotti da Valiant [Val84].

Di fatto le tecniche di *Ensemble Learning* non forniscono classificatori *general purpose*, ma indicano solo il modo ottimo per unire più classificatori tra loro.

Esempi di tecniche di *Ensemble Learning* sono

Decision Tree gli Alberi di Decisione, essendo costruiti da tanti *Decision Stump* in cascata sono un primo esempio di *Ensemble Learning*;

Bagging il *BootStrap AGGregatING* prova a ridurre i problemi di *overfitting* addestrando diversi classificatori su sottoparti del *training set* ed eseguendo infine una votazione per maggioranza;

Boosting Invece che prendere sottoparti del *training set* puramente casuali vengono, in parte, usati i campioni che rimangono non classificati correttamente;

AdaBoost l'*ADAptive BOOSTing* (sezione 4.6.2) è l'algoritmo di *Ensemble Learning* più conosciuto e progenitore della famiglia molto florida di classificatori *AnyBoost*;

Random Forest™ è un *BootStrap Aggregating* (*bagging*) su *Decison Tree*, *Ensemble Classifier* composto da diversi alberi di decisione, ognuno creato su un sottoinsieme dei dati di addestramento e delle caratteristiche da analizzare, che votano per maggioranza;

e molti altri ancora.

Esempi di classificatori deboli ampiamente usati in letteratura sono i *Decision Stump* [AL92] associati alle *feature* di Haar (sezione 6.1). Il *Decision Stump* è un classificatore binario nella forma

$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } pf(\mathbf{x}) > p\theta \\ -1 & \text{otherwise} \end{cases} \tag{4.45}$$

dove $f(\mathbf{x})$ è una funzione che estrae uno scalare dal campione da classificare, $p = \{+1, -1\}$ è una parità che serve per indicare la direzione della diseguaglianza e θ è la soglia di decisione (figura 4.4).

4.6.1 Alberi di Decisione

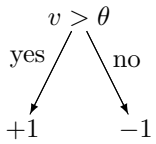


Figura 4.4: Esempio di *Decision Stump*. v è una caratteristica (*feature*) estratta dall'immagine e θ una soglia.

Un Albero di Decisione (*Decision Tree*) è un metodo molto semplice ed efficace per realizzare un classificatore e l'addestramento degli alberi di decisione è una delle tecniche attuali di maggior successo. Un albero di decisione è un albero di classificatori (*Decision Stump*) dove ogni nodo interno è associato ad una particolare "domanda" su una caratteristica (*feature*). Da questo nodo dipartono tanti archi quanti sono i possibili valori che la caratteristica può assumere, fino a raggiungere le foglie che indicano la categoria associata alla decisione. Particolare attenzione normalmente è posta per i nodi di decisione binaria.

Una buona "domanda" divide i campioni di classi eterogenee in dei sottoinsiemi con etichette abbastanza omogenee, stratificando i dati in modo da mettere poca varianza in ogni strato.

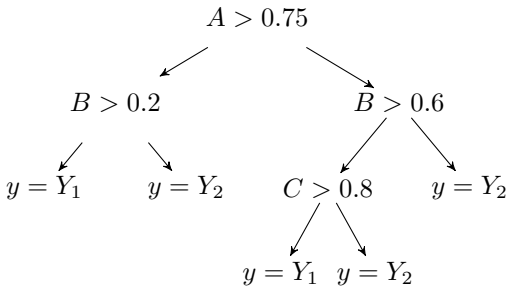


Figura 4.5: Esempio di *Decision Tree*.

Per permettere questo è necessario definire una metrica che misuri questa impurità. Definiamo X come un sottoinsieme di campioni di un particolare insieme di addestramento formato da m possibili classi. X è di fatto una variabile aleatoria, che assume solo valori discreti (il caso continuo è comunque uguale). È possibile associare ad ogni valore discreto x_i, che può assumere X, la distribuzione di probabilità $p(x_i) = p_i$. X è un *data set* formato da m classi e p_i è la frequenza relativa della classe i all'interno dell'insieme X.

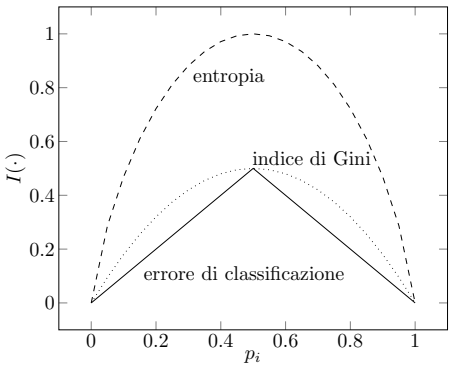


Figura 4.6: Confronto tra metriche di misura dell'impurità nel caso di problema di classificazione binario.

Data la definizione di X, negli alberi di decisione sono largamente usate le seguenti metriche:

Entropia Dalla teoria dell'informazione, l'entropia I_H di X vale:

$$I_H(X) = -\sum_{i=1}^{m} p_i \log_2 p_i \tag{4.46}$$

Indice di Gini L'indice di impurità di Gini è definito come

$$I_G(X) = 1 - \sum_{i=1}^{m} p_i^2 \tag{4.47}$$

Errore di Classificazione Dalla teoria bayesiana:

$$I_E(X) = 1 - \max_i p_i \tag{4.48}$$

Intuitivamente un nodo con distribuzione delle classi $(0, 1)$ ha impurità minima, mentre un nodo con distribuzione uniforme $(0.5, 0.5)$ ha impurità massima.

Una "domanda" $h_j(x)$, che ha k possibili risposte, divide l'insieme $\mathcal{E}$ nei sottoinsiemi $\mathcal{E}_1, \ldots, \mathcal{E}_k$.

Per testare quanto bene la condizione viene eseguita, bisogna confrontare il grado di impurità dei nodi figli con l'impurità del nodo padre: maggiore è la loro differenza, migliore è la condizione scelta.

Data una metrica $I(\cdot)$ che misuri l'impurità, il guadagno Δ è un criterio che può essere usato per determinare la bontà della divisione:

$$\Delta = I(\mathcal{E}) - \sum_{i=1}^{k} \frac{N(\mathcal{E}_i)}{N(\mathcal{E})} I(\mathcal{E}_i) \tag{4.49}$$

dove $N(\mathcal{E})$ è il numero di campioni nel nodo padre e $N(\mathcal{E}_i)$ è il numero di campioni nel nodo figlio i-esimo.

Se viene usata come metrica l'entropia, il guadagno Δ è conosciuto come *Information Gain* [TSK06].

Gli alberi di decisione inducono algoritmi che scelgono una condizione di test che massimizza il guadagno Δ. Siccome $I(\mathcal{E})$ è uguale per tutti i possibili classificatori e $N(\mathcal{E})$ è costante, massimizzare il guadagno è equivalente a minimizzare la somma pesata delle impurità dei nodi figli:

$$\hat{h} = \arg\min_{h_j} \sum_{i=1}^{k} N(\mathcal{E}_i) I(\mathcal{E}_i) \tag{4.50}$$

La miglior domanda $h_j(x)$ è quella che minimizza tale quantità.

Nel caso di classificatori binari, la metrica di Gini è ampiamente utilizzata, in quando il guadagno da minimizzare si riduce a

$$\frac{p_1 n_1}{p_1 + n_1} + \frac{p_2 n_2}{p_2 + n_2} \tag{4.51}$$

con p_1, n_1 numero di campioni positivi e negativi che il classificatore sposta nel ramo sinistro e p_2, n_2 numero di campioni nel ramo destro.

Gli alberi di decisione si adattano sia molto bene che velocemente ai dati di addestramento e conseguentemente, se non limitati, soffrono in maniera sistematica del problema di *overfitting*. Normalmente agli alberi viene applicato un algoritmo di raffinamento (*pruning*) per ridurre, ove possibile, il problema di *overfitting*. Gli approcci di pruning sono solitamente due: *pre-pruning* o *post-pruning*. Il *pre-pruning* si limita a fermare la creazione dell'albero sotto determinate condizioni per evitare una eccessiva specializzazione (esempio massima dimensione dell'albero). Il *post-pruning* invece raffina un albero già creato, eliminando quei rami che non soddisfano alcune condizioni su un *validation set* precedentemente selezionato.

Questa tecnica di creazione di un albero di decisione viene solitamente indicata come *Classification and regression trees* (CART) [B+84]. Infatti, nel caso reale in cui le caratteristiche analizzate sono grandezze statistiche, non si parla di creare un albero di classificazione, ma più propriamente di costruire un albero di regressione. Trovare la partizione ottima dei dati è un problema NP-completo, perciò normalmente si fa uso di algoritmi ingordi *greedy* come quello mostrato in sezione.

4.6.2 ADAptive BOOSTing

Uno dei classificatori *Ensemble* che ha attirato più interesse da parte dei ricercatori negli ultimi anni è sicuramente *AdaBoost* [FS95]. L'idea base di *AdaBoost* è quella di costruire una lista di classificatori assegnando, in maniera iterativa, un peso ad ogni nuovo classificatore considerando la sua capacità di riconoscere campioni non correttamente identificati dagli altri classificatori già coinvolti nell'addestramento. Tutti questi classificatori coinvolti voteranno con il peso loro assegnato e la scelta finale avverà per maggioranza.

Le tecniche di *Boosting* permettono di generare un classificatore nella forma di modello additivo:

$$F_T(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x}) + \ldots + f_T(\mathbf{x}) = \sum_{t=1}^{T} f_t(\mathbf{x}) \tag{4.52}$$

con $f_1, \ldots, f_T$ singoli classificatori.

Esaminiamo il caso di classificazione binaria, e sia $S = (\mathbf{x}_1, y_1) \ldots (\mathbf{x}_m, y_m) \in (\mathbb{X} \times \{-1, 1\})$ l'insieme degli m campioni disponibili per l'addestramento.

Una scelta abbastanza diffusa nell'ambito di fitting di modelli è quella di usare la regressione ai minimi quadrati (metrica ottima in caso di rumore gaussiano per esempio) per ottenere il modello additivo $F_T(\mathbf{x})$, minimizzando la quantità $\sum (y_i - F_T(\mathbf{x}_i))^2$. Tuttavia, a seguito di numerosi esperimenti, si è visto che la funzione costo quadratica non è la scelta ottima nei problemi di classificazione.

L'approccio di *AdaBoost* suggerisce invece che l'unione di tutti questi classificatori minimizzi una funzione costo differente, migliore, ovvero la funzione di perdita esponenziale (*exponential loss*):

$$\min_F \sum_{i=1}^{m} e^{-y_i F(\mathbf{x}_i)} \tag{4.53}$$

Siccome la minimizzazione globale della funzione (4.53) solitamente è impossibile, si può procedere in due modi

- ottimizzando un classificatore per volta in maniera ciclica fino a giungere a una situazione stabile (*generalized backfitting algorithm*);
- aggiungendo un nuovo classificatore per volta al modello additivo (*"greedy" forward stepwise approach*)

AdaBoost affronta il problema della classificazione attraverso il secondo approccio.

Sotto queste considerazioni l'obiettivo del processo di addestramento si riduce a individuare un classificatore addizionale $f(\mathbf{x})$ che minimizzi di volta in volta la quantità

$$f_{T+1} = \arg\min_f \sum_{i=1}^{m} e^{-y_i(F_T(\mathbf{x}_i)+f(\mathbf{x}_i))} = \arg\min_f \sum_{i=1}^{m} w_i e^{-y_i f(\mathbf{x}_i)} \tag{4.54}$$

avendo definito $w_i = e^{-y_i F_T(\mathbf{x}_i)}$ e sfruttando le proprietà dell'esponenziale.

AdaBoost è una tecnica che risponde a tutte queste esigenze.

Si supponga pertanto di avere a disposizione $\mathcal{H} = \{h_1, \ldots, h_T\}$ classificatori binari, ognuno dei quali, valutando la caratteristica $\mathbf{x}_i$, con $1 \leq i \leq m$, restituisca una opinione $y_i = \{-1, +1\}$.

Sia la funzione $F_T(\mathbf{x}; \boldsymbol{\alpha})$, definita come

$$F_T(\mathbf{x}; \alpha_1, \ldots, \alpha_T) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \tag{4.55}$$

una funzione il cui segno rappresenta l'ipotesi di classificazione e la sua magnitudine riflette la bontà della predizione. Il modello di equazione (4.55) è chiamato *Extended Additive Model* o *Adaptive Basis-Function Model*.

L'obiettivo è ottenere un classificatore forte $H(\mathbf{x}_i)$ come somma lineare pesata dei classificatori h_t, il cui segno determini l'ipotesi globale:

$$H(\mathbf{x}_i) = \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_i)\right) = \operatorname{sgn} F_T(\mathbf{x}_i; \boldsymbol{\alpha}) \tag{4.56}$$

Questa è una votazione per maggioranza: viene scelto come vincitrice l'ipotesi votata da più classificatori, ognuno con peso differente α_t. Sono proprio le costanti α_t, i pesi assegnati a ogni classificatore, il risultato fornito da questa tecnica di addestramento.

Per permettere di assegnare un voto al classificatore, è necessario che a ogni campione in ingresso x_i sia assegnato un certo peso w_i: più il peso è alto più il campione è stato classificato in maniera non corretta fino a questo punto dell'addestramento mentre più il peso è basso più è stato classificato correttamente. Alla prima iterazione, tutti i pesi sono posti uguali, pari a $w_i^{(0)} = 1/m$, in modo da avere una esatta distribuzione statistica. Varianti come l'*Asymmetric AdaBoost* assegnano pesi differenti alle diverse categorie coinvolte.

Sia $u_i = y_i h_t(\mathbf{x}_i)$ la funzione che esprime il successo $(+1)$ o il fallimento (-1) del classificatore h_t a valutare il campione x_i. Dati i pesi associati a ogni campione, è possibile per ogni classificatore calcolare W_{-1}, la somma dei pesi associati gli insuccessi, e W_{+1}, la somma dei pesi associati alle classificazioni corrette, ovvero attraverso la definizione di u_i, in forma compatta

$$W_b = \sum_{u_i = b} w_i \tag{4.57}$$

con $b = +1$, successo, e $b = -1$, insuccesso.

Sia ϵ_t la misura dell'errore del classificatore h_t calcolata come

$$\epsilon_t = \sum_{y_i \neq h_t(i)} w_i^{(t)} = \sum_{u_i=-1} w_i^{(t)} = W_- \tag{4.58}$$

somma dei pesi associati ai soli campioni classificati in maniera errata, e sia

$$r_t = W_+ - W_- = \sum_{i=1}^{m} w_i^{(t)} u_i \tag{4.59}$$

la media ponderata, usando i pesi w_i, delle performance u_i di classificazione.

Le iterazioni dell'algoritmo di *AdaBoost* sono le seguenti:

1. un *Oracolo* fornisce un classificatore h_t (la scelta è di fatto lasciata all'utente, cercando di selezionare il classificatore che minimizza l'errore ϵ_t, ma non è obbligatorio che debba essere per forza il migliore);
2. viene calcolato l'errore ϵ_t prodotto del classificatore h_t sui campioni in ingresso. Quando non si riesce a trovare un classificatore per il quale $\epsilon_t > 1/2$, l'addestramento non può proseguire e deve venire pertanto terminato;
3. dato l'errore, al classificatore h_t viene assegnato un peso α_t, calcolato come descritto in seguito;
4. ad ogni campione x_i la distribuzione associata $w_i^{(t+1)}$ viene aggiornata attraverso la funzione

$$w_i^{(t+1)} = \frac{1}{Z_t} w_i^{(t)} e^{-\alpha_t u_i} = \frac{1}{Z_t} w_i^{(t)} e^{-y_i f_t(\mathbf{x}_i)} \tag{4.60}$$

   Il peso associato ai campioni che hanno avuto successo nella classificazione viene diminuito di una quantità proporzionale a $e^{-\alpha_t}$, mentre ai campioni che sono stati classificati in maniera errata il peso è aumentato di e^{α_t}. Z_t è un fattore di normalizzazione scelto in modo tale che $\sum w_i^{(t)} = 1$ ma assume anche un significato importante come spiegato immediatamente sotto.

Il parametro di normalizzazione Z_t vale

$$Z_t = \sum_{i=1}^{m} w_i^{(t)} e^{-\alpha_t u_i} = e^{-\alpha_t} W_+ + e^{\alpha_t} W_- \tag{4.61}$$

e, risultato importante di *AdaBoost*, si può dimostrare che l'errore di classificazione viene limitato superiormente da

$$\frac{1}{m} \{i : H(x_i) \neq y_i\} \leq \prod_{t=1}^{T} Z_t \tag{4.62}$$

Per questo motivo Z_t è esattamente la quantità da minimizzare per ottenere il classificatore ottimo. Conseguenza diretta di questo risultato, si può vedere *AdaBoost* come uno schema che minimizza $\prod_t Z_t$.

La scelta ottima di α_t (e di riflesso quella di h_t) è quella dove la funzione (4.61) assume il minimo, ovvero

$$\alpha_t = \frac{1}{2} \log \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) = \frac{1}{2} \log \frac{W_+}{W_-} = \frac{1}{2} \log \left(\frac{1 + r_t}{1 - r_t} \right) \tag{4.63}$$

Con questa particolare scelta di α_t, Z_t assume il minimo e vale

$$Z_t = 2\sqrt{\epsilon_t (1 - \epsilon_t)} = 2\sqrt{W_- W_+} \tag{4.64}$$

Dall'equazione (4.64) si evince che Z_t viene minimizzato scegliendo il classificatore h_t che ha il minore valore di ϵ_t ovvero massimo W_+.

Scegliendo come peso quello di equazione (4.63) che minimizza Z_t, dopo ogni iterazione di *AdaBoost* i pesi associati a campioni identificati correttamente vengono diminuiti di un fattore $\exp(-\alpha_t)$ ovvero $\sqrt{W_-/W_+}$, mentre i pesi associati a campioni valutati erroneamente dall'ipotesi h_t vengono aumentati di un fattore $\exp(\alpha_t)$ ovvero $\sqrt{W_+/W_-}$.

Questo algoritmo è quello che viene definito in letteratura *AdaBoost.M1* o *Discrete AdaBoost* [FHT00]. Le ipotesi $h_t(\mathbf{x})$ usate da AdaBoost sono *feature* che possono assumere i soli valori $\{+1, -1\}$.

Il funzionamento intuitivo di *AdaBoost* è molto semplice: *AdaBoost* per ogni nuovo classificatore aggiunto alla serie si concentra sui *pattern* in ingresso che finora sono stati classificati peggio.

AdaBoost ha diverse interpretazioni: come classificatore che massimizza il margine, regressione logistica a un modello additivo, come minimizzatore a discesa del gradiente a passi discreti ma anche come regressione con tecnica di Newton.

AdaBoost, come SVM, ottiene come risultato quello di massimizzare il margine di separazione tra le classi, anche se con metriche differenti. In questo modo, entrambi, riescono ad essere meno sensibile a problemi come l'*overfitting*.

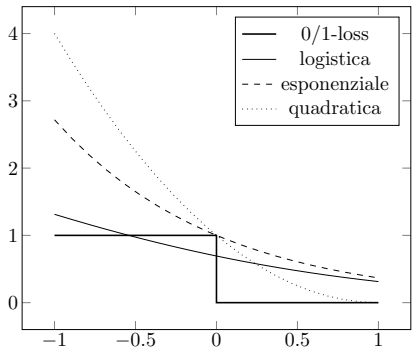


Figura 4.7: Confronto tra *loss function*: 0/1-loss, logistica, esponenziale e quadratica

4.6.3 AdaBoost e le sue varianti

Il problema di *Boosting* si può generalizzare e può essere visto come un problema dove è necessario cercare dei predittori $f_t(\mathbf{x})$ che minimizzino la funzione costo globale:

$$\sum_{i=1}^{m} \phi\left(y_i \left(f_1(\mathbf{x}_i) + \ldots + f_n(\mathbf{x}_i)\right)\right) \tag{4.65}$$

dove $\phi \in \mathcal{C}^1$ è una funzione convessa, non crescente con $\lim_{z\to\infty} \phi(z) = 0$.

Dal punto di vista analitico, *AdaBoost* è un esempio di ottimizzatore a discesa del gradiente (*coordinate-wise gradient descent*) che minimizza la *potential function* $\phi(z) = e^{-z}$, ottimizzando un coefficiente α_t per volta [LS10], come si vede dall'equazione (4.54).

Un elenco, non esaustivo ma che permette di fare luce su alcune peculiarità di questa tecnica, delle varianti di *AdaBoost* è:

AdaBoost con astensione

AdaBoost può essere esteso anche a casi di classificatori con astensione, dove le uscite possibili sono $h_j(x_i) \in \{-1, 0, +1\}$. Ampliando la definizione (4.57), per semplicità si indichino con W_- gli insuccessi, W_0 le astensioni e W_+ i successi del classificatore h_t.

Anche in questo caso Z_t assume il minimo con lo stesso valore di α_t del caso senza astensione, cfr. (4.63), e con tale scelta Z_t varrebbe

$$Z_t = W_0 + 2\sqrt{W_- W_+} \tag{4.66}$$

Tuttavia esiste una scelta più conservativa di α_t proposta da Freund e Shapire

$$\alpha_t = \frac{1}{2} \log\left(\frac{W_+ + 1/2W_0}{W_- + 1/2W_0}\right) \tag{4.67}$$

che permette di fissare un limite superiore a Z_t.

Real AdaBoost

Real AdaBoost generalizza il caso precedente ma soprattutto generalizza lo stesso modello additivo esteso [FHT00]. Invece che usare ipotesi dicotomiche $h_t(x)$ e associare ad esse un peso α_t si cerca direttamente la *feature* $f_t(x)$ che minimizza l'equazione (4.54).

Real AdaBoost permette di usare classificatori deboli che forniscono la distribuzione di probabilità $p_t(x) = P[y = 1|x, w^{(t)}] \in [0, 1]$, probabilità che la classe y sia effettivamente $+1$ data l'osservazione della caratteristica x.

Data una distribuzione di probabilità $p_t(x)$, la *feature* $f_t(x)$, che minimizza l'equazione (4.54), è

$$f_t(x) = \frac{1}{2} \log \frac{P[y = +1|x, w^{(t)}]}{P[y = -1|x, w^{(t)}]} = \frac{1}{2} \log \frac{p_t(x)}{1 - p_t(x)} \tag{4.68}$$

Tale risultato è pari a metà della della trasformazione logistica. Siccome l'obiettivo rimane sempre quello di minimizzare la funzione costo esponenziale, l'aggiornamento dei pesi rimane ancora quello di equazione (4.60).

Sia *Discrete* che *Real AdaBoost*, scegliendo un classificatore debole che rispetti l'equazione 4.68, fanno in modo che *AdaBoost* converga asintoticamente a

$$\lim_{T\to\infty} F_T(x) = \frac{1}{2} \log \frac{P[y = +1|x]}{P[y = -1|x]} \tag{4.69}$$

dimostrando come l'algoritmo di *AdaBoost* sia una procedura iterativa che combina diversi classificatori deboli per approssimare un classificatore Bayesiano.

Real AdaBoost può essere usato anche con un classificatore discreto come il *Decision Stump*. Applicando direttamente l'equazione (4.68) ai due possibili stati di uscita del *Decision Stump* (risulta comunque facile ottenere il minimo di Z_t per via algebrica) le risposte del classificatore devono assumere i valori

$$f(x) = \begin{cases} \frac{1}{2}\log\frac{W_{TP}}{W_{FP}} & x > \theta \\ \frac{1}{2}\log\frac{W_{FN}}{W_{TN}} & x \leq \theta \end{cases} \tag{4.70}$$

con i valori W_*, somma dei pesi associati ai Falsi Positivi (FP), Falsi Negativi (FN), Veri Positivi (TP) e Veri Negativi (TN). Con questa scelta di valori, Z_t assume come valore notevole

$$Z_t = 2\left(\sqrt{W_{TP}W_{FP}} + \sqrt{W_{FN}W_{TN}}\right) \tag{4.71}$$

metrica da usare per scegliere la miglior feature x e soglia θ.

Gentle AdaBoost

I pesi associati agli *outlier* in *Real AdaBoost* possono essere molto elevati a causa della presenza del logaritmo in equazione. Risulta in questo caso rendere più "gentile" la regressione.

Gentle AdaBoost generalizza ulteriormente il concetto di Ensemble Learning a modello additivo [FHT00] usando una regressione con passi tipici dei metodi di Newton:

$$F_{T+1}(x) = F_T(x) + f_t(x) = F_T(x) + \mathop{\mathrm{E}}_{w^{(t)}}[y|x] \tag{4.72}$$

L'ipotesi $f_t(x)$, da aggiungere al modello additivo all'iterazione t, viene scelta fra tutte le possibili ipotesi f_k come quella che ottimizza una regressione ai minimi quadrati pesata

$$f_t = \arg\min_{f_k} \sum_i w_i (y_i - f_k(x_i))^2 \tag{4.73}$$

ma per ogni iterazione viene usato l'aggiornamento dei pesi di *AdaBoost* (4.60), ovvero la funzione costo esponenziale.

Anche *Gentle AdaBoost* può essere usato con il *Decision Stump*. In questo caso il minimo di (4.73) dell'algoritmo di decisione assume una forma notevole in

$$f(x) = \begin{cases} \frac{W_{TP}-W_{FP}}{W_{TP}+W_{FP}} & x > \theta \\ \frac{W_{FN}-W_{TN}}{W_{TN}+W_{FN}} & x \leq \theta \end{cases} \tag{4.74}$$

LogitBoost

Per motivi storici, *AdaBoost* non manifesta esplicitamente un formalismo statistico. La prima cosa che si nota è che la risposta del classificatore di *AdaBoost* non è una probabilità, in quanto non limitata tra $[0, 1]$. Oltre a questo problema, parzialmente risolto da *Real AdaBoost*, minimizzare la *loss-function* (4.53) non sembra un approccio statistico come lo potrebbe essere invece massimizzare la verosimiglianza. È possibile tuttavia dimostrare che la funzione di costo di *AdaBoost* massimizza un funzione molto simile alla *Bernoulli log-likelihood*.

Per queste ragioni è possibile estendere *AdaBoost* alla teoria della regressione logistica, descritta in sezione 3.7.

La regressione logistica additiva assume la forma

$$\log\frac{P[y=+1|x]}{P[y=-1|x]} = F_T(x) = \sum_{t=1}^{T} f_t(x) \tag{4.75}$$

espressione interessante se confrontata con quella di AdaBoost di equazione (4.69). Invertendo l'equazione (4.75) si ottiene la relazione logistica

$$p(x) = P[y=+1|x] = \frac{e^{F_T(x)}}{1+e^{F_T(x)}} = \frac{1}{1+e^{-F_T(x)}} \tag{4.76}$$

che associa una stima della probabilità al modello additivo $F(x)$.

Il problema diventa quello di trovare una *loss function* adeguata a questa rappresentazione, ovvero individuare una variante di *AdaBoost* che massimizza esattamente la *Bernoulli log-likelihood* [FHT00].

Massimizzare la verosimiglianza di (4.76) equivale a minimizzare la *log-loss*

$$\sum_{t=1}^{T} \log\left(1 + \exp(-y_i F_T(x_i))\right) \tag{4.77}$$

LogitBoost per primo estende *AdaBoost* al problema dell'ottimizzazione logistica di una funzione $F_T(x)$ sotto la funzione costo $\phi(z) = \log(1 + e^{-z})$, massimizzando la *Bernoulli log-likelihood* usando iterazioni di tipo Newton.

I pesi associati a ogni campione derivano direttamente dalla distribuzione di probabilità

$$\begin{aligned} z_i &= \frac{y_i^* - p(x_i)}{p(x_i)(1-p(x_i))} \\ w_i &= \frac{p(x_i)}{1-p(x_i)} \end{aligned} \tag{4.78}$$

con $y^* = \{0, 1\}$ e scegliendo l'ipotesi $f_t(x)$ come regressione ai minimi quadrati di z_i a x_i usando i pesi w_i. La stima futura di $p(x_i)$ deriva direttamente dall'equazione (4.76).

Asymmetric-AdaBoost

Asymmetric-AdaBoost presenta una variante nella regola di aggiornamento dei pesi [VJ01]. Il problema di AdaBoost è che non permette un diretto controllo sul peso da assegnare agli errori di classificazione nelle diverse classi e non permette di minimizzare esplicitamente il numero di falsi positivi, ma solo l'errore di classificazione. Le varianti *Asymmetric-AdaBoost* modificano invece ad ogni iterazione t i pesi associati ai campioni positivi e negativi di un fattore di costo $c_+^{(t)}$ e $c_-^{(t)}$ rispettivamente.

Cascade

A prescindere dall'utilizzo i meno dei classificatori *Cascade* [VJ02], i pesi vengono modificati di un fattore $\beta_t = \epsilon_t/(1-\epsilon_t) = W_-/W_+$ solo nel caso di classificazione corretta, altrimenti i pesi rimangono invariati. Il peso associato a un classificatore viene assegnato come $\alpha_t = -\log \beta_t$, valore doppio rispetto al peso assegnato da *AdaBoost.M1*.

MAdaBoost

L'algoritmo *MAdaBoost* presenta un aggiornamento diverso dei pesi, per cercare di ridurre il contributo degli *outlier* (o esempi troppo complessi) nell'addestramento. Il peso $w_i^{(t)}$ massimo che può assumere un campione viene limitato superiormente dal valore $w_i^{(0)}$, valore che assume il peso all'inizio dell'algoritmo.

Questo comportamento può essere rappresentato da una funzione costo del tipo

$$\phi(z) = \begin{cases} 1 - z & z \leq 0 \\ e^{-z} & z > 0 \end{cases} \tag{4.79}$$

4.7 Reti Neurali

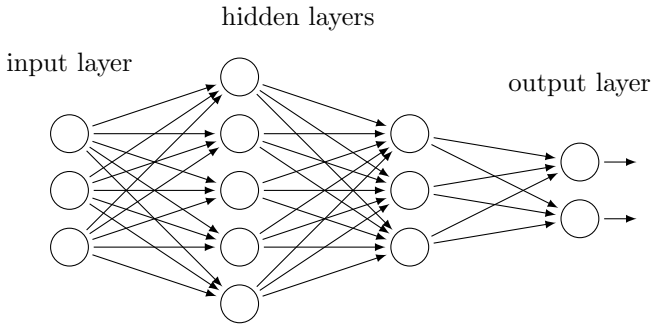


Figura 4.8: Esempio di topologia di una rete neurale.

La ricerca sul Machine Learning (e in generale la Visione Artificiale) ha sempre cercato di trarre spunto per lo sviluppo degli algoritmi dal cervello umano. Le reti neurali artificiali (*artificial neural networks* ANN) si basano sul concetto di "neurone artificiale" ovvero una struttura che, similarmente ai neuroni degli esseri viventi, applicano una trasformazione non lineare (detta funzione di attivazione) ai contributi pesati dei diversi ingressi del neurone:

$$y_k = f_k\left(\sum_i w_{k,i} x_{k,i} + b_k\right) \tag{4.80}$$

dove $x_{k,i}$ sono i vari ingressi relativi al k-esimo neurone a cui sono associati i pesi $w_{k,i}$, y_k è la rispota del neurone e la funzione di attivazione f, fortemente non lineare, normalmente è una funzione gradino, una sigmoide o una funzione logistica. Il bias b a volte viene simulato con un ingresso costante $x_k = +1$.

La rete neurale più semplice, composta da uno stadio di ingresso e uno stadio di uscita, è assimilabile al modello di perceptrone (*perceptron*) introdotto da Rosenblatt nel 1957. Similarmente al cervello degli esseri viventi, una rete neurale artificiale consiste nella connessione di diversi neuroni artificiali.

La geometria di una rete neurale *feedforward*, la topologia normalmente utilizzata in applicazioni pratiche, è quella del *MultiLayer Perceptron* MLP e consiste nella combinazione di molteplici strati nascosti di neuroni che collegano lo stadio degli ingressi con lo stadio delle uscite, stadio che sarà l'ingresso dello strato successivo. Un perceptrone multistrato è assimilabile a una funzione

La fase di addestramento consiste nello stimare i pesi w_i^k che minimizzano l'errore tra le etichette di addestramento e i valori predetti dalla rete $f_w(\mathbf{x})$:

$$S(\mathbf{w}) = \sum_i \|\mathbf{y}_i - f_{\mathbf{w}}(\mathbf{x}_i)\|^2 \tag{4.81}$$

La stima dei pesi w_i^k si può ottenere con tecniche note di ottimizzazione: di norma viene sfruttata la tecnica della *back propagation* che di fatto è una discesa del gradiente con la *chain-rule* per il calcolo delle derivate, essendo le MLP strutture stratificate.

4.7.1 Le funzioni di attivazione

La funzione di attivazione (*activation function*) del neurone artificiale trasforma i valori di ingresso nelle uscite. Funzioni di attivazione comuni sono

- gradino (Heaviside) $h(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$
- sigmoide $\sigma(x) = \frac{1}{1+e^{-x}}$
- softsign $s(x) = \frac{1}{1+|x|}$
- tangente iperbolica $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$
- rampa $r(x) = \max(0, x)$

Originariamente erano usate praticamente solo le funzioni di attivazioni sigmoidali (sigmoide e tangente iperbolica) ma di recente viene anche utilizzata la classe di funzioni rampa che, rispetto alle funzioni sigmoidali, sono funzioni non limitate. I Neuroni Artificiali con la funzione attivazione rampa sono definiti *Rectified Linear Units* (ReLUs) [KSH12a].

4.8 Apprendimento Profondo

Le reti neurali, e in particolare il loro addestramento tramite *backpropagation*, presentano diverse problematiche pratiche:

- richiedono dati di addestramento etichettati, mentre nei contesti basati su grandi moli di dati (*big data*) la maggior parte dei dati può non essere categorizzata;
- il tempo di addestramento scala male sia con l'aumentare della dimensione della rete sia con l'aumentare della quantità di dati;
- l'ottimizzazione può bloccarsi in minimi locali, rendendo la rete sub-ottima;
- l'addestramento supervisionato di modelli profondi (reti con molti strati nascosti) è un problema di ottimizzazione particolarmente complesso.

Per affrontare queste sfide, si è sviluppata una branca del *machine learning* nota come *deep learning*, che sfrutta architetture profonde e tecniche avanzate per migliorare l'efficacia dell'apprendimento.

Gli esseri umani affrontano problemi complessi suddividendoli in sotto-problemi e livelli multipli di rappresentazione astratta. Analogamente, il *deep learning* consente a un sistema di apprendere rappresentazioni gerarchiche dei dati, mappando direttamente funzioni complesse tra ingresso e uscita, senza dipendere da caratteristiche progettate manualmente.

Questo approccio permette di generare astrazioni di alto livello, spesso non esplicitabili dagli esseri umani, ma più gestibili dal calcolatore.

Con l'aumento della disponibilità di dati e delle applicazioni del *machine learning*, le tecniche di apprendimento automatico stanno evolvendo rapidamente. L'obiettivo del *deep learning* è costruire rappresentazioni di alto livello dei dati attraverso l'uso di strati multipli di operazioni non lineari, come nelle *Deep Neural Networks* (DNN).

4.8.1 Representation Learning

Il *Feature Learning* o *Representation Learning* rappresenta un ponte tra le tecniche tradizionali di *machine learning* e il *deep learning*. L'idea è quella di utilizzare tecniche di apprendimento non supervisionato per ridurre la dimensionalità del problema, conservando il più possibile l'informazione, e successivamente impiegare questi dati trasformati per la classificazione supervisionata.

Le prestazioni di un algoritmo di *machine learning* dipendono fortemente dalla rappresentazione dei dati in ingresso. Gran parte degli sforzi è dedicata alla progettazione di trasformazioni che convertano i dati grezzi in un formato ottimale per la classificazione.

Tra le tecniche di apprendimento non supervisionato utili per la rappresentazione troviamo:

- Segmentazione K-means
- Reti di Hopfield
- Sparse Coding
- Principal Component Analysis (PCA, sezione 2.10.1)
- Restricted Boltzmann Machines (RBM, sezione 4.8.2)
- AutoEncoder (sezione 4.8.3)

Queste tecniche mirano a ridurre la dimensionalità preservando l'informazione più rilevante, ovvero quella che meglio descrive i campioni di addestramento.

4.8.2 Restricted Boltzmann Machines

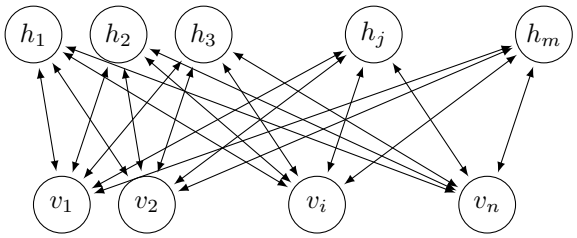


Figura 4.9: Restricted Boltzmann Machines.

Il punto di svolta tra le tecniche non profonde e le tecniche profonde di addestramento è considerato il 2006 quando Hinton e altri all'University of Toronto introducono le *Deep Belief Networks* (DBNs) [HOT06], un algoritmo che "avidamente" addestra una struttura a strati addestrando uno strato alla volta sfruttando un algoritmo di addestramento non-supervisionato. La peculiarità delle DBN consiste nel fatto che gli strati sono costituiti da *Restricted Boltzmann Machine* (RBM) [Smo86, FH94].

Sia $\mathbf{v} \in \{0,1\}^n$ una variabile stocastica binaria associata allo stato visibile e $\mathbf{h} \in \{0,1\}^m$ una variabile stocastica binaria associata allo stato nascosto. Dato uno stato $(\mathbf{v}, \mathbf{h})$ l'energia della configurazione degli strati visibili e nascosti è data da [Hop82]

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{n} a_i v_i - \sum_{j=1}^{m} b_j h_j - \sum_{i=1}^{n}\sum_{j=1}^{m} w_{i,j} v_i h_j \tag{4.82}$$

dove v_i e h_j sono gli stati binari dello strato visibile e dello strato nascosto rispettivamente mentre a_i, b_j sono i pesi e $w_{i,j}$ sono i pesi associati tra di loro. Una *Boltzmann Machine* è simile ad una rete di Hopfield, con la differenza che tutti gli output sono stocastici. Si può pertanto definire la *Boltzmann Machine* come un caso speciale di modello di Ising che a sua volta è un caso particolare di *Markov Random Field*. Allo stesso modo le RBM possono essere interpretate come reti neurali stocastiche dove i nodi e le connessioni corrispondono ai neuroni e alle sinapsi, rispettivamente.

La probabilità della configurazione congiunta $(\mathbf{a}, \mathbf{b}, \mathbf{W})$ è data dalla distribuzione di Boltzmann:

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z(\cdot)} e^{-E(\mathbf{v},\mathbf{h})} \tag{4.83}$$

dove la funzione di partizionamento Z è data da

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{4.84}$$

somma delle energie di tutte le possibili coppie di stati visibili e nascosti.

La parola *restricted* fa riferimento al fatto che non sono ammesse interazioni diretta tra le unità appartenenti allo stesso strato ma solo tra strati limitrofi.

Dato un input $\mathbf{v}$, lo stato binario nascosto h_j viene attivato con probabilità:

$$p(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_i v_i w_{i,j}\right) \tag{4.85}$$

dove $\sigma(x)$ è la funzione logistica $1/(1 + exp(-x))$. Allo stesso modo è facile ottenere lo stato visibile dato lo stato nascosto:

$$p(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_j h_i w_{i,j}\right) \tag{4.86}$$

Stimare i parametri del modello $(\mathbf{a}, \mathbf{b}, \mathbf{W})$ in modo da modellare correttamente la distribuzione dei dati di addestramento è un compito computazionalmente oneroso. Tuttavia, nel 2002 Hinton propose l'algoritmo di *Contrastive Divergence* (CD), che permette un addestramento molto più efficiente delle RBM, rendendole finalmente utilizzabili su larga scala. Una descrizione dettagliata e pratica dell'addestramento delle RBM si può trovare in [Hin12].

4.8.3 Auto-Encoders

Gli *Auto-Encoders* sono particolari tipi di reti neurali non supervisionate che mirano a codificare l'ingresso $\mathbf{x}$ in una rappresentazione compatta $\mathbf{c}(\mathbf{x})$, tale da permettere la ricostruzione dell'ingresso stesso. In questo senso, gli Auto-Encoders possono essere visti come un'alternativa alle RBM addestrate con l'algoritmo di *Contrastive Divergence*, con cui condividono l'obiettivo di apprendere rappresentazioni latenti significative.

La rete è composta da due parti:

- un *encoder* $\mathbf{c}(\cdot)$ che trasforma l'ingresso in una rappresentazione latente;
- un *decoder* $\mathbf{f}(\cdot)$ che cerca di ricostruire l'ingresso a partire dalla rappresentazione latente.

L'obiettivo è minimizzare la *negative log-likelihood* della ricostruzione:

$$-\log P\left(\mathbf{x}|\mathbf{c}(\mathbf{x})\right) \tag{4.87}$$

Nel caso in cui la distribuzione dei dati sia gaussiana, questa espressione si riduce alla classica regressione ai minimi quadrati (vedi sezione 2.8).

Se invece gli ingressi $\mathbf{x}_i$ sono binari (o seguono una distribuzione binomiale), la funzione costo diventa:

$$-\log P\left(\mathbf{x}|\mathbf{c}(\mathbf{x})\right) = \sum_i \mathbf{x}_i \log \mathbf{f}_i\left(\mathbf{c}(\mathbf{x})\right) + (1 - \mathbf{x}_i) \log\left(1 - \mathbf{f}_i(\mathbf{c}(\mathbf{x}))\right) \tag{4.88}$$

dove $\mathbf{f}(\cdot)$ è il decoder associato all'encoder $\mathbf{c}(\cdot)$.

La funzione $\mathbf{c}(\mathbf{x})$ rappresenta una compressione con perdita (*lossy compression*). Essa è efficace nel rappresentare i dati visti durante la fase di addestramento non supervisionato, ma può risultare sub-ottimale per dati non appartenenti al dominio di addestramento.

4.8.4 Reti Neurali Profonde

Nelle reti neurali tradizionali, l'ottimizzazione dei pesi avviene a partire da valori iniziali scelti casualmente. Questa scelta, sebbene semplice da implementare, comporta che all'aumentare della profondità della rete le prestazioni tendano a degradare, mentre architetture più superficiali (con uno o due strati nascosti) risultano generalmente più stabili e facili da addestrare.

Storicamente, l'addestramento di reti neurali multistrato (MLP) tramite discesa del gradiente ha incontrato due ostacoli principali:

- la presenza di numerosi minimi locali che ostacolano la convergenza;
- la comparsa di ampi plateau nella superficie di errore, che rallentano drasticamente l'ottimizzazione.

Di conseguenza, l'idea di utilizzare reti molto profonde per modellare problemi complessi è stata a lungo considerata impraticabile.

A partire dal 2012, grazie alla disponibilità di grandi quantità di dati (*Big Data*), alla crescente potenza di calcolo offerta dalle GPU e allo sviluppo di tecniche di ottimizzazione più efficaci (come *Adam*) 3.3.4, le reti neurali profonde hanno vissuto una vera e propria rinascita.

Un punto di svolta fondamentale fu la vittoria di *AlexNet* [KSH12b] alla competizione ImageNet Large Scale Visual Recognition Challenge (ILSVRC) del 2012: per la prima volta, una rete neurale convolutiva profonda superò nettamente gli approcci tradizionali, segnando l'inizio dell'era moderna del *deep learning*.

Da allora, le reti profonde sono diventate lo standard de facto in numerosi ambiti del *machine learning*. In particolare, l'elaborazione di dati strutturati come le immagini ha tratto enorme beneficio dalle *reti neurali convolutive* (CNN), che sfruttano la struttura spaziale del segnale visivo per apprendere in maniera più efficiente rappresentazioni gerarchiche e invarianti.

4.8.5 Reti Neurali Convolutive

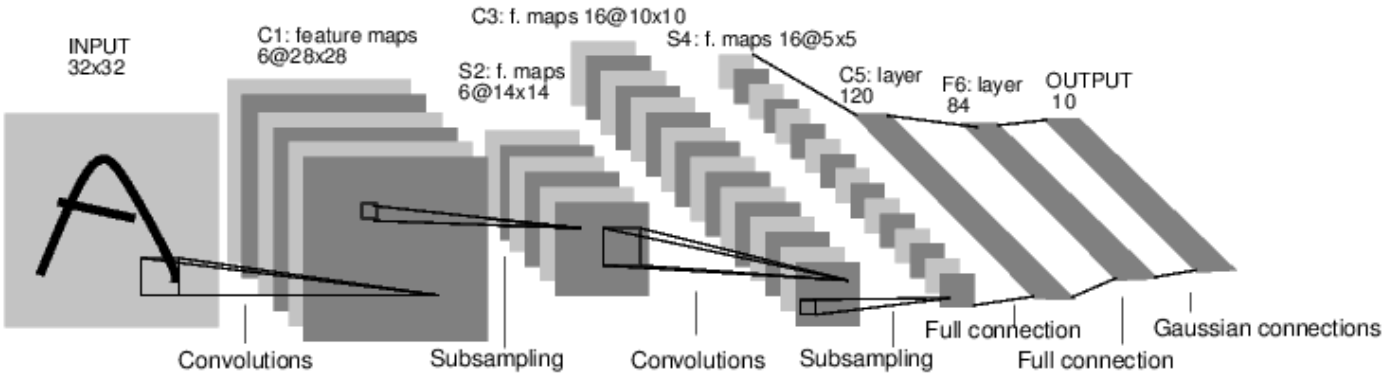


Figura 4.10: Esempio dell'architettura di una Rete Neurale Convolutiva, la LeNet5.

Le *Convolutional Neural Networks* (CNN) rappresentano una naturale evoluzione delle reti neurali profonde per il trattamento di dati strutturati spazialmente, come le immagini. A differenza dei classici *MultiLayer Perceptron* (MLP), le CNN sfruttano la correlazione locale dei pixel e la ridondanza spaziale attraverso strati convolutivi in cui i pesi sono condivisi. Questa architettura permette di apprendere automaticamente caratteristiche gerarchiche e invarianti rispetto alla traslazione, riducendo significativamente il numero di parametri da ottimizzare e migliorando l'efficienza dell'apprendimento. Nei paragrafi seguenti verranno descritti i principali componenti di una CNN, tra cui i layer convolutivi, le funzioni di attivazione, il pooling e le architetture tipiche di addestramento profondo.

Le CNN sono reti neurali multi-livello simili ai *MultiLayer Perceptron*, ma con una struttura peculiare: almeno uno degli strati è costituito da insiemi di neuroni che condividono i pesi, detti strati convolutivi (*convolutional layer*).

In uno strato convolutivo, l'attivazione di un neurone dipende dal prodotto scalare tra un kernel (o filtro) e una regione locale dell'ingresso:

$$a_{i,j} = \sum_k \sum_l \sum_m w_{k,l,m}\, x_{i+k,j+l,m} + b = \mathbf{w}^\top \mathbf{x}_{i,j} + b \tag{4.89}$$

dove $\mathbf{w}$ rappresenta i pesi del filtro, $\mathbf{x}_{i,j}$ la porzione locale dell'immagine centrata in (i,j) e b un eventuale bias.

La convoluzione può essere applicata con *stride* diversi da 1, ottenendo così un layer di attivazione sottocampionato rispetto all'ingresso. Poiché i filtri riducono progressivamente le dimensioni delle mappe di attivazione, è frequente introdurre un *padding* per mantenere costante la dimensione in uscita.

Le mappe di attivazione (*activation maps*) vengono trasformate da una funzione di attivazione non lineare, tipicamente una ReLU. Successivamente, uno strato di *pooling* viene spesso inserito per ridurre la dimensionalità e introdurre invarianza locale: il *max pooling* è la scelta più comune, mentre l'*average pooling* o l'*L2-norm pooling* erano più diffusi in passato.

Le CNN sono progettate per trarre vantaggio da ingressi bidimensionali multicanale. Una CNN prende in genere come ingresso un tensore di ordine 3 o 4; ad esempio, un'immagine $w \times h$ con 3 canali (R,G,B) rappresenta un tensore di ordine 3. Ogni strato convolutivo può contenere k kernel differenti, generando in uscita mappe di attivazione di dimensione $w \times h \times k$.

L'architettura di una rete profonda (*Deep Neural Network*, DNN) può includere:

- strati convolutivi (**C**);
- strati di pooling (**S**);
- strati di normalizzazione;
- strati completamente connessi (MLP);
- funzioni di perdita.

All'ultimo stadio (*loss layer*) viene applicato un metodo di classificazione tradizionale (MLP, AdaBoost o SVM) sulla rappresentazione ridotta dell'ingresso, conservando la maggior parte dell'informazione utile.

L'addestramento delle DNN avviene tipicamente tramite varianti della discesa stocastica del gradiente (vedi sezione 3.3.3), ottimizzando iterativamente i pesi dei filtri e degli strati completamente connessi sulla base dell'errore di classificazione.

4.8.6 Reti Ricorrenti e Transformers (per sequenze e immagini)

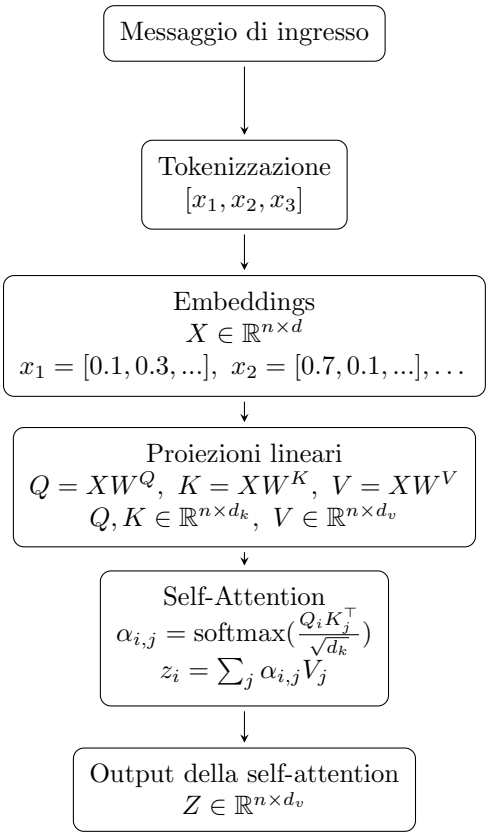


Figura 4.11: Schema concettuale del meccanismo di *self-attention* in un Transformer.

Le *Convolutional Neural Networks* (*CNN*) hanno rappresentato una svolta per l'elaborazione delle immagini statiche, grazie alla capacità di catturare gerarchie di caratteristiche locali attraverso filtri convolutivi e pooling. Tuttavia, le CNN tradizionali non sono adatte a gestire dati sequenziali come testo, segnali temporali o video, in cui l'ordine e la relazione temporale tra gli elementi è cruciale. Per affrontare questi problemi, sono state introdotte le *Reti Neurali Ricorrenti* (*RNN*), in cui i neuroni presentano connessioni ricorrenti che consentono di mantenere una memoria dello stato passato. Le RNN hanno trovato applicazione in compiti come il riconoscimento del parlato, la traduzione automatica, l'analisi di serie temporali e la descrizione automatica di immagini (*image captioning*). Tuttavia, le RNN tradizionali soffrono di difficoltà nell'apprendere dipendenze a lungo termine, a causa del problema del gradiente evanescente o esplodente. Per mitigare questi limiti sono state sviluppate architetture più sofisticate come le *Long Short-Term Memory* (LSTM) e le *Gated Recurrent Unit* (GRU), capaci di controllare quali informazioni mantenere o dimenticare attraverso meccanismi di *gating*.

Nonostante ciò, la prima generazione di reti neurali profonde sequenziali si basava su un paradigma *encoder-decoder*, in cui l'informazione della sequenza in ingresso veniva compressa in un tensore di dimensioni ridotte, che doveva preservare il maggior numero possibile di informazioni utili per il compito. Questo approccio ha tuttavia un limite intrinseco: elementi rilevanti dell'input possono essere trascurati o attenuati durante la compressione.

Un cambiamento fondamentale è arrivato con il "meccanismo di attenzione" (*attention mechanism*). L'idea alla base dell'attenzione è semplice ma potente: invece di comprimere tutte le informazioni in un unico vettore, il modello può "concentrarsi" dinamicamente sulle parti più rilevanti della sequenza di ingresso. I pesi di attenzione vengono calcolati in base alla rilevanza di ciascun elemento rispetto agli altri, permettendo di superare le limitazioni delle rappresentazioni compresse negli stati ricorrenti.

Sia $X \in \mathbb{R}^{n \times d}$ la matrice di input della sequenza, dove n è il numero di *token* (le singole unità in cui l'algoritmo divide la sequenza di ingresso, variabile da sequenza a sequenza) e d è la dimensione degli *embedding* (un vettore numerico che rappresenta ciascun token, fisso per il modello e capace di codificare informazioni semantiche e sintattiche).

Le matrici, chiamate di *query* (Q), *key* (K) e *value* (V), si ottengono tramite proiezioni lineari:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V \tag{4.90}$$

dove:

- $W^Q, W^K \in \mathbb{R}^{d\times d_k}$ e $W^V \in \mathbb{R}^{d\times d_v}$ sono matrici di pesi apprese durante l'addestramento;
- $Q, K \in \mathbb{R}^{n\times d_k}$ e $V \in \mathbb{R}^{n\times d_v}$.

Il meccanismo di *self-attention* può allora essere espresso in forma scalare (per singolo token) come:

$$\alpha_{i,j} = \text{softmax}\left(\frac{Q_i K_j^\top}{\sqrt{d_k}}\right), \quad z_i = \sum_{j=1}^{n} \alpha_{i,j} V_j \tag{4.91}$$

dove:

- $Q_i, K_j \in \mathbb{R}^{d_k}$ sono rispettivamente la *query* del token i e la *key* del token j;
- $V_j \in \mathbb{R}^{d_v}$ è il *value* del token j;
- $\alpha_{i,j}$ è lo scalare che indica quanto il token i presta attenzione al token j;
- $z_i \in \mathbb{R}^{d_v}$ è la nuova rappresentazione del token i risultante dalla combinazione pesata dei *value*.
- La funzione *softmax*, che normalizza i pesi $\alpha_{i,j}$ tra 0 e 1, è definita come:

$$\text{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}, \quad i = 1, \ldots, n \tag{4.92}$$

  dove $\mathbf{x}$ è il vettore di input (nel nostro caso $\mathbf{x} = \frac{Q_i K^\top}{\sqrt{d_k}}$ per la riga i).

In termini intuitivi, l'attenzione può essere vista come una generalizzazione dinamica dei metodi di ponderazione come *Bag of Words* o *TF-IDF*. Tuttavia, mentre TF-IDF assegna pesi statici ai termini, l'attenzione attribuisce pesi contestuali e dipendenti dal compito, consentendo al modello di focalizzarsi selettivamente sulle parti più rilevanti. Semanticamente, la matrice Z risultante ha la stessa lunghezza dell'ingresso (n token in, n token out), ma ogni token è arricchito con informazioni provenienti dal contesto globale della sequenza.

Il meccanismo di attenzione ha portato direttamente allo sviluppo dei *Transformers* [VSP+17], oggi lo standard *de facto* per la modellazione di sequenze in linguaggio naturale, visione artificiale e apprendimento multimodale. Nei Transformers, l'operatore centrale è il *self-attention*, che consente di modellare in modo parallelo e diretto le relazioni tra tutti gli elementi della sequenza. Rispetto alle RNN, i Transformers offrono vantaggi significativi in termini di parallelizzazione, stabilità numerica e capacità di apprendere dipendenze a lungo raggio.

Nel campo della visione artificiale, l'applicazione dei Transformers ha portato allo sviluppo dei *Vision Transformers* (ViT) [DBK+20], in cui un'immagine viene suddivisa in piccole regioni (*patch*) trattate come una sequenza, analogamente alle parole in un testo. Questi modelli hanno dimostrato prestazioni competitive o superiori rispetto alle CNN su vari compiti di classificazione, riconoscimento e segmentazione, soprattutto in presenza di grandi quantità di dati.

In applicazioni pratiche un solo modulo di *self-attention* non risulta sufficiente per estrarre abbastanza informazione dai *token* di ingresso. Il meccanismo di *multi-head self-attention* estende l'idea di *self-attention* permettendo al modello di guardare la sequenza sotto diverse prospettive contemporaneamente. In pratica:

- Si adottano h teste (*heads*) diverse. Per ciascuna head $p = 1, \ldots, h$, si hanno matrici di proiezione separate:

$$Q^{(p)} = XW^{Q,(p)}, \quad K^{(p)} = XW^{K,(p)}, \quad V^{(p)} = XW^{V,(p)}, \tag{4.93}$$

  dove ciascuna matrice $W^{Q,(p)}, W^{K,(p)}, W^{V,(p)} \in \mathbb{R}^{d\times d_k}$.

- Ogni head calcola una self-attention (scala, softmax, combinazione) nel suo sottospazio. Si usa la forma matriciale compatta, dove Attention è l'operatore *scaled dot-product attention*:

$$Z^{(p)} = \text{Attention}\left(Q^{(p)}, K^{(p)}, V^{(p)}\right) = \text{softmax}\left(\frac{Q^{(p)}(K^{(p)})^\top}{\sqrt{d_k}}\right) V^{(p)} \tag{4.94}$$

- Le uscite delle varie head vengono concatenate:

$$Z_{\text{concat}} = \left[Z^{(1)}\,;\, Z^{(2)}\,;\, \ldots\,;\, Z^{(h)}\right] \in \mathbb{R}^{n\times(h\cdot d_v)} \tag{4.95}$$

  e poi proiettate nuovamente in uno spazio di dimensione d (la dimensione del modello), tramite una matrice di uscita $W^O \in \mathbb{R}^{(h\cdot d_v)\times d}$:

$$Z = Z_{\text{concat}}\, W^O \tag{4.96}$$

Tabella 4.1: Esempi di geometria (*layers*, *embedding*, *heads*) per modelli Transformer noti.

Modello	**Layers**	**d**	**h**	$\mathbf{d_k} = d/h$
Transformer (base) [VSP+17]	6 encoder + 6 decoder	512	8	64
BERT-Base	12 encoder	768	12	64
BERT-Large	24 encoder	1024	16	64
GPT-3 (175B)	molti decoder	12288	96	128

- Spesso si sceglie $d_v = d_k$, e $d = h \cdot d_k$, così che la concatenazione seguita dalla proiezione restituisca una dimensione coerente con l'input del layer successivo.
- Questo permette di modellare diverse "tipologie" di attenzione (es. relazioni sintattiche, semantiche, dipendenze locali vs globali) nello stesso layer.

Oggi, RNN e Transformers rappresentano strumenti complementari: le prime restano utili in scenari con sequenze relativamente brevi o risorse limitate, mentre i secondi costituiscono la base delle architetture più avanzate del *deep learning* moderno. L'evoluzione dai meccanismi ricorrenti a quelli basati su attenzione ha segnato un cambio di paradigma: dall'idea di memoria compressa ad una rappresentazione dinamica e contestuale, dove il modello decide autonomamente "cosa guardare" per ogni elemento della sequenza.

4.9 Generalizzare l'addestramento

È importante osservare che le tecniche di *Machine Learning* sono molto di più di discorso di mera ottimizzazione. Uno degli obiettivi che ci si pone durante l'addestramento è fare in modo che il sistema sia capace di classificare nuovi campioni che ancora non ha visionato. Una via per combattere l'*overfitting* è la "regolarizzazione". Esistono diverse tecniche in letteratura per la regolarizzazione: le principali sono la regolarizzazione L1/L2 e l'uscita anticipata (*early-stopping*).

4.9.1 Regolarizzazione L1 ed L2

La regolarizzazione L1 ed L2 consiste nell'inserire un termine aggiuntivo alla funzione costo, penalizzante per alcune configurazioni. Regolarizzare, per esempio, la funzione costo

$$S(\boldsymbol{\beta}, \mathbf{X}) = -\sum_i \log P(Y = y_i | \mathbf{x}_i; \boldsymbol{\beta}) \tag{4.97}$$

significa aggiungere un termine, funzione solo di $\boldsymbol{\beta}$, in maniera da ottenere la nuova funzione costo del tipo

$$E(\boldsymbol{\beta}, \mathbf{X}) = S(\boldsymbol{\beta}, \mathbf{X}) + \lambda R(\boldsymbol{\beta}) \tag{4.98}$$

con $R(\boldsymbol{\beta})$ una funzione regolarizzante.

Una funzione regolarizzante molto diffusa è

$$R(\boldsymbol{\beta}) = \left(\sum_j |\beta_j|^p \right)^{1/p} \tag{4.99}$$

Valori comuni per p sono 1 o 2 (per questo viene chiamata regolarizzazione L1 o L2). Quando $p = 2$ può essere definita in letteratura anche come *weight decay*. Questo genere di funzioni di regolarizzazione penalizzano pertanto i parametri con valori troppi elevati.

4.9.2 Uscita anticipata

L'uscita anticipata combatte direttamente l'*overfitting* monitorando le prestazioni del modello su un insieme di esempi addizionale chiamato insieme di validazione (*validation set*). Quando la funzione obiettivo sul set di validazione smette di diminuire per un certo numero di iterazioni e anzi comincia a peggiorare l'addestramento viene interrotto in quanto c'è il sospetto di iniziare ad overfittare il problema.

4.10 Valutazione delle prestazioni

Dato un classificatore addestrato su un determinato insieme di addestramento (*Training Set*) è necessario valutarlo su un altro insieme (*Validation Set* o *Certification Set*). Da questo confronto è possibile estrarre degli indici che permettono di

valutare il classificatore e permettono di confrontare diversi classificatori tra loro. È assolutamente indispensabile che gli indici di prestazione vengano calcolati su un insieme di campioni non usati durante la fase di addestramento (il *validation set*) in modo da rilevare problemi come l'*overfitting* dei dati ovvero la mancata generalizzazione.

Fissati i parametri del classificatore si può creare la tabella di contingenza (*Confusion Matrix*):

		Valore Vero	
		p	n
Classificazione	p'	VP	FP
	n'	FN	VN

I Falsi Positivi (FP) vengono indicati anche come Falsi Allarmi. I Falsi Negativi (FN) come *miss*.

Dalla tabella vengono normalmente estratti alcuni valori di prestazione, come:

- *Accuracy* è il rapporto tra il Numero di Predizioni Corrette sul Numero totale di predizioni = (VP+VN)/(VP+VN+FN+FP);
- *Error Rate* è il Numero totale di predizioni errate sul Numero totale di predizioni = (FP+FN) / (VP+VN+FN+FP);
- *Precision* (o *specificity* o PPV) è la probabilità che un positivo ritornato dal classificatore sia corretto = VP / (VP+FP);
- *Recall* (o *hit-rate* o TPR) è la percentuale di positivi riconosciuti correttamente = VP / (VP+FN);
- *Miss-Rate* o FNR è l'opposto della Recall = 1-Recall = FN/(VP+FN).

Ogni classificatore ha uno o più parametri che, se modificati, cambiano il rapporto tra i riconoscimenti corrette e il numero di falsi positivi. Risulta pertanto difficile poter confrontare in maniera obiettiva due classificatori perché magari uno presenta, a parità di tale soglia, un numero di individuazioni corrette più elevato dell'altro ma magari un numero più alto di falsi positivi. Per confrontare pertanto le prestazioni di diversi classificatori binari ottenuti da diverse sessioni di addestramento si fa normalmente uso di curve al variare di questa soglia interna del classificatore.

Le curve di prestazioni che si possono trovare sono

- La curva ROC (*Receiver Operating Characteristic*) è un grafico cartesiano dove lungo le ascisse è indicato il numero di falsi positivi (percentuali *FPR*, per fotogramma o assoluti) e in ordinata è presente la percentuale di corretti riconoscimenti (*True Positive Rate TPR*), generati dal classificatore al variare della soglia. Ogni classificatore per essere definito tale deve avere una curva ROC migliore del classificatore casuale, ovvero la retta che congiunge i punti $(0, 0)$ e $(1, 1)$ del grafico ROC.
- *Precision-Recall* (*PRC*) che concentra l'analisi principalmente sui positivi. L'area sotto la PRC è chiamata *Average Precision* (*AP*). In problemi reali di detection il numero di Veri Negativi sono altissimi e perciò è necessario concentrarsi sui falsi positivi. La curva PRC ha il grosso vantaggio di nascondere la scala dei Falsi Positivi: normalmente infatti questi sono indicati per fotogrammi, per minuto, o su un altra unità di misura.
- *Detection error Tradeoff* (*DET*) che permette di rappresentare sugli assi gli errori negativi (miss) e gli errori positivi (false alarm). È una curva che concentra l'analisi puramente sugli errori.

È infine da notare che questi indici si riferiscono a qualunque classe di problemi che contempli il concetto di risultato corretto o sbagliato. È pertanto applicabile non solo ai classificatori, ma per esempio alle associazioni di punti caratteristici e altro ancora.

Recentemente, per poter fare un confronto più snello per le prestazioni di un classificatore, sono state proposte delle funzioni che applicate alle curve ROC permettono di ricavare un unico scalare rappresentante un voto della bontà di classificazione. Tali funzioni sono solitamente delle medie di campionamenti della curva ROC nelle sole zone di interesse pratico.

Capitolo 5

Punti Caratteristici

L'individuazione (estrazione) di punti chiave (*keypoint detection*), la loro caratterizzazione (*feature description*) e infine confronto (*matching*) sono tematiche strettamente legate all'interno della visione artificiale. Le applicazioni che fanno uso di punti chiave spaziano dalla creazione di immagini panoramiche alla ricostruzione tridimensionale, dall'odometria visuale all'inseguimento di oggetti e in moltissimi altri casi di utilizzo.

Il concetto di punto chiave richiama il fatto che non tutti ma solo alcuni punti dell'immagine hanno una probabilità elevata di essere individuati senza ambiguità durante un confronto. Sono punti notevoli, stabili, facilmente individuabili. Nell'ultima decade, come in quasi tutti i campi della Visione Computazionale, sono stati fatti grandi passi in avanti nello sviluppo di *local invariant features*, punti caratteristici che permettono alle applicazioni di definire una geometria locale dell'immagine e codificarla in maniera tale che sia invariante alle trasformazioni dell'immagine, quali traslazione, rotazione, scala e deformazioni affini.

In questo capitolo verranno trattate le tematiche più strettamente inerenti agli algoritmi di estrazione dei punti chiave. Il discorso invece della descrizione del punto sarà trattato nel capitolo seguente siccome è un argomento ortogonale tra quello di descrivere i punti e il concetto di classificazione.

Un elenco, non esaustivo, di algoritmi per individuare punti chiave è

Harris Corner Harris formalizza da un punto di vista matematico il concetto di bordo e, attraverso lo studio degli autovalori della matrice di covarianza nell'intorno di un punto, permette di ricavare la presenza o meno di uno spigolo. È invariante a cambiamenti di luminosità, a trasformazioni geometriche quali traslazioni e rotazioni, e minimamente a variazioni di scala (sezione 5.2);

KLT il Kanade-Lucas-Tomasi sfrutta una variante di Harris (Shi-Tomasi) come *corner detector* ed esegue il confronto sfruttando rappresentazioni piramidali della scena (dettagli in 7.2);

AST La classe degli *Advance Segment Test* (sezione 5.5) identifica un punto caratteristico osservando la differenza di luminosità dei punti su una circonferenza;

SIFT studia l'immagine in multisoluzione ed è invariante a trasformazioni simili (sezione 5.3);

SURF una variante di SIFT più performante basata sull'immagine integrale (sezione 5.4).

5.1 Individuatore Hessiano

Il problema dell'individuazione di punti notevoli che possano essere facilmente riconosciuti tra due immagini è stato inizialmente risolto spostando il problema verso quello di individuare punti angolari (*corner*) nell'immagine, ovvero scartando quelle porzioni dell'immagine senza tessitura o con solo bordi.

L'operatore Hessiano (*Hessian detector*) [Bea78], basato sulla matrice Hessiana derivata dall'espansione in serie di Taylor nell'intorno del punto da descrivere, cerca quelle parti dell'immagine che mostrano delle forti derivate nelle direzioni ortogonali. Tale algoritmo è basato sull'analisi della matrice delle derivate seconde ovvero l'Hessiana

$$\mathbf{H}(\mathbf{x}, \sigma) = \begin{bmatrix} I_{xx}(\mathbf{x}, \sigma) & I_{xy}(\mathbf{x}, \sigma) \\ I_{xy}(\mathbf{x}, \sigma) & I_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{5.1}$$

L'algoritmo calcola le derivate seconde dell'immagine I_{xx}, I_{xy}, I_{yy} per ogni punto dell'immagine e individua i punti nei quali il determinante dell'Hessiana

$$\det\left(\mathbf{H}(\mathbf{x}, \sigma)\right) = I_{xx}(\mathbf{x}, \sigma) I_{yy}(\mathbf{x}, \sigma) - I_{xy}^2(\mathbf{x}, \sigma) \tag{5.2}$$

diventa massimo. Questa ricerca è normalmente attuata sull'immagine del determinante dell'Hessiana a cui viene applicata una Non-Maxima Suppression su una finestra 3×3. Il massimo della risposta del deteminante Hessiano sono normalmente

localizzati sugli angoli e in zone dell'immagine con forte tessitura. L'utilizzo del determinante dell'Hessiana rende questo algoritmo invariante alla rotazione.

In applicazione pratiche non viene mai usata l'immagine originale ma una versione filtrata passa basso attraverso una gaussiana.

5.2 Förstner-Harris

L'algoritmo di Förstner-Harris [FG87, HS88] è stato esplicitamente disegnato per ottenere una elevata stabilità geometrica. Esso definisce punti caratteristici quei punti che hanno un massimo locale nel confronto ai minimi quadrati alla propria versione sottoposta a traslazione. Questo algoritmo ha avuto così tanto successo perchè permette di individuare le variazione dell'intensità dell'immagine nell'intorno di un punto usando la matrice di auto-correlazione tra le derivate prime dell'immagine.

Si definiscano le immagini dei gradienti (queste possono essere generate da un operatore differenziale, Sobel, Prewit o Roberts) $I_x(x,y)$ e $I_y(x,y)$ rispettivamente gradiente orizzontale e gradiente verticale dell'immagine da analizzare.

Da queste due immagini è possibile calcolare una funzione $\mathbf{C}(x,y)$, rappresentate la matrice di covarianza (autocorrelazione) delle immagini dei gradienti in un intorno di (x,y), definita come

$$\mathbf{C}(x,y) = \begin{bmatrix} \sum_{\boldsymbol{\delta}\in\Omega} I_x^2(\boldsymbol{\delta})w(\boldsymbol{\delta}) & \sum_{\boldsymbol{\delta}\in\Omega} I_x(\boldsymbol{\delta})I_y(\boldsymbol{\delta})w(\boldsymbol{\delta}) \\ \sum_{\boldsymbol{\delta}\in\Omega} I_x(\boldsymbol{\delta})I_y(\boldsymbol{\delta})w(\boldsymbol{\delta}) & \sum_{\boldsymbol{\delta}\in\Omega} I_y^2(\boldsymbol{\delta})w(\boldsymbol{\delta}) \end{bmatrix} \tag{5.3}$$

con $\boldsymbol{\delta} \in \Omega$ intorno di (x,y) e $w(\boldsymbol{\delta})$ un kernel opzionale, normalmente o una gaussiana centrata in (x,y), per permettere di pesare in maniera differente i punti nell'intorno, o una finestra costante su Ω. Originariamente $w(\boldsymbol{\delta})$ erano filtri molto piccoli ma, man mano che la potenza di calcolo è aumentata, si è passati a kernel gaussiani via via maggiori.

Di fatto in Harris si usano due filtri di convoluzione: uno derivativo per calcolare le immagini derivate e uno integrale per calcolare gli elementi della matrice. La dimensione di questi filtri e l'utilizzo di filtro gaussiano per pesare i punti rimanda alla lettura della sezione seguente riguardo la scala di individuazione delle *feature*.

La matrice $\mathbf{C}$ è la matrice dei momenti del secondo ordine. Per individuare punti caratteristici si possono analizzare gli autovalori λ_0 e λ_1 della matrice $\mathbf{C}$ (si legga per una trattazione più approfondita la sezione 2.10.1). Gli autovalori della matrice di auto-correlazione $\mathbf{C}$ permettono di caratterizzare il tipo di immagine contenuta nella finestra intorno al punto dato.

Se sono presenti due autovalori molto elevati il punto è un *corner*, se è presente un solo autovalore di valore elevato è un *edge*, altrimenti è una zona ragionevolmente piatta, ovvero in forma di funzione come

$$C = \min(\lambda_0, \lambda_1) \tag{5.4}$$

i cui massimi locali rappresentano i *corner* ricavati dall'algoritmo di Shi-Tomasi [ST94].

Per una matrice 2×2 gli autovalori si ottengono come soluzioni del polinomio caratteristico quadratico

$$p(x) = x^2 - \operatorname{trace}(\mathbf{C})x + \det(\mathbf{C}) \tag{5.5}$$

Harris, per evitare di calcolare esplicitamente gli autovalori di $\mathbf{C}$, introduce un operatore $H(x,y)$ definito come

$$H(x,y) = \det(\mathbf{C}) - \alpha \operatorname{trace}(\mathbf{C})^2 \tag{5.6}$$

dove α è un parametro compreso tra 0 e 0.25 e solitamente posto a 0.04.

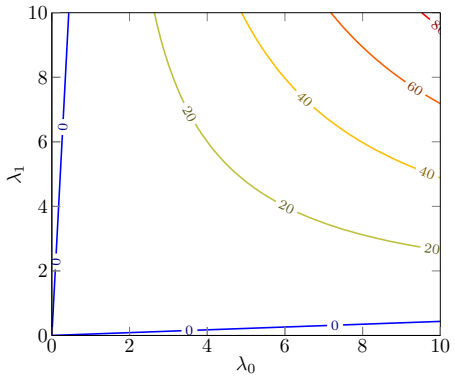


Figura 5.1: Risposta nel piano degli autovalori fornita dall'equazione di Harris a diversi valori della soglia avendo posto $\alpha = 0.04$. L'area interessata è molto simile a quella fornita dal metodo di Shi-Tomasi ma senza la necessità di calcolare esplicitamente gli autovalori.

Per Harris il punto (x,y) è un punto caratteristico (*corner*) se $H(x,y) > H_{thr}$, con H_{thr} soglia da definire. Il parametro α regola la sensitività del rilevatore di feature. Qualitativamente alzare α rimuove i bordi mentre alzare H_{thr} rimuove le zone piatte (figura 5.1).

5.3 Invarianza alla scala e alla rotazione

Harris è un individuatore di punti notevoli non invariante alle variazioni di scala. Per superare questa serie di limiti, Lindeberg [Lin94, Lin14] introduce il concetto di selezione automatica della scala, permettendo di individuare i punti caratteristici a un determinato livello di risoluzione. La rappresentazione piramidale della scena, algoritmo computazionalmente efficiente ampiamente usato in precedenza, diventa di fatto un caso particolare di questa rappresentazione scala-spazio.

Sia $G(x, y; t)$ la gaussiana bidimensionale di varianza $t > 0$, di equazione

$$G(x, y; t) = \frac{1}{2\pi t} e^{-\frac{x^2+y^2}{2t}} \tag{5.7}$$

(cfr. sezione 2.2).

La convoluzione $L(x, y; t)$ tra l'immagine $I(x, y)$ e la gaussiana $G(x, y; t)$

$$L(x, y; t) = G(x, y; t) * I(x, y) \tag{5.8}$$

genera la rappresentazione scala-spazio (*scale-space representation*) dell'immagine stessa. La varianza $t = \sigma^2$ del kernel gaussiano è chiamata parametro di scala (*scale parameter*). La rappresentazione dell'immagine alla scala degenere $t = 0$ è l'immagine originale stessa.

È da notare che applicare un filtro gaussiano a un immagine non crea nuove strutture: tutta l'informazione generata dal filtro era già contenuta nell'immagine originale.

Figura 5.2: Rappresentazione scala-spazio di una immagine 512×512: dall'immagine originale $t = 0$ alle scale 1, 4, 16, 64 e 256.

Il fattore di scala t è un numero continuo ma, per motivi computazionali, vengono usati passi discreti di questo valore, normalmente successioni esponenziali, come $t = 2^i$ o $t = \frac{1}{2}e^i$.

Applicare a una immagine scala-spazio un operatore derivata, per la proprietà commutativa tra la convoluzione e la derivata, è uguale ad eseguire la convoluzione dell'immagine originale con la derivata della gaussiana:

$$L_{x^\alpha}(\cdot; t) = \partial_{x^\alpha} L(\cdot; t) = (\partial_{x^\alpha} g(\cdot; t)) * f(\cdot) \tag{5.9}$$

con α notazione multi-indice della derivata. Allo steso modo è possibile estendere a un qualsiasi fattore di scala la definizione di tutti i filtri bordo o punti caratteristici. Attraverso il lavoro di Lindeberg è stato possibile estendere il concetto dei *Corner* di Harris a casi invarianti di scala (metodi Harris-Laplace e Hessian-Laplace [MS02]).

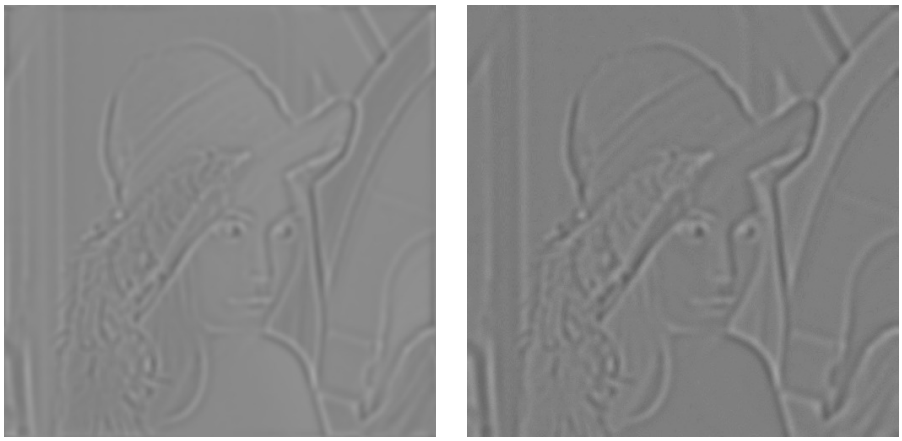
Alcuni operatori interessanti per trovare punti caratteristici sono per esempio il modulo del gradiente $|\nabla L|$, il laplaciano $\nabla^2 L$ e il determinante dell'hessiana $\det \mathcal{H}(L)$. Tutti questi operatori sono invarianti alle rotazioni, ovvero il punto di minimo/massimo esiste indipendentemente dalla rotazione che assume l'immagine.

Tra questi operatori, uno molto diffuso per individuare punti caratteristici è il Laplaciano della Gaussiana (LoG) normalizzato (*scale-normalized Laplacian operator*):

$$\nabla^2_n L(x,y,t) = t(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})G = -\frac{1}{t\pi}\left(1 - \frac{x^2+y^2}{2t}\right) e^{-\frac{x^2+y^2}{2t}} \tag{5.10}$$

Attraverso l'operatore LoG, è possibile individuare punti caratteristici come massimi o minimi locali nelle coordinate spaziali e scala.

Per esempio, un cerchio di raggio r ha la massima risposta al laplaciano al fattore di scala $\sigma = r/\sqrt{2}$.



(a) (b)

Figura 5.3: Confronto tra l'immagine LoG normalizzata *(a)* e DoG *(b)*

Lowe [Low04], nell'algoritmo *Scale-invariant feature transform* (*SIFT*), per aumentare le prestazioni, approssima il Laplaciano della Gaussiana (LoG) con una Differenza tra Gaussiane (DoG):

$$\begin{aligned} D(x,y;\sigma) &= (G(x,y;k\sigma) - G(x,y;\sigma)) * I(x,y) \\ &\approx L(x,y;k\sigma) - L(x,y;\sigma) \\ &\approx (k-1)\sigma^2 LoG(x,y;\sigma) \end{aligned} \tag{5.11}$$

Questo procedimento è più performante in quanto l'immagine gaussiana a scala $k\sigma$ può venire calcolata dall'immagine gaussiana σ applicando un filtro $(k-1)\sigma$, più piccolo e perciò nel complesso molto più veloce rispetto ad eseguire la convoluzione $k\sigma$ con l'immagine originale.

Se in LoG i punti caratteristici erano i minimi/massimi locali, sia nello spazio che nella scala, dell'immagine del laplaciano, in questo caso i punti caratteristici sono i punti minimo e massimo nell'immagine differenza tra le immagini scala $\sigma, k\sigma, \ldots, k^n\sigma$ attraverso le quali viene processata l'immagine (figura 5.4).

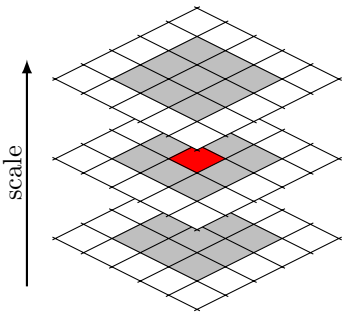


Figura 5.4: Individuazione di minimi e massimi locali: per ogni pixel e per ogni scala viene confrontato un intorno $3 \times 3 \times 3$.

Con l'introduzione del passo k, il dominio della variabile σ viene di fatto suddiviso in passi logaritmici discreti, raccolti in ottave, e ogni ottava viene suddivisa in S sottolivelli. In questo modo σ assume i valori discreti

$$\sigma(o,s) = \sigma_0 2^{o+\frac{s}{S}} \leftrightarrow k = 2^{\frac{1}{S}} \tag{5.12}$$

con σ_0 fattore base di scala.

I punti caratteristici, trovati come massimo/minimo in scala e spazio, entrambi discreti, vengono interpolati usando una regressione a una quadrica tridimensionale per trovare il punto caratteristico con precisione subpixel e subscala.

Tra un ottava e quella successiva l'immagine viene sottocampionata di un fattore 2: oltre all'analisi a scale multiple all'interno di ogni ottava, l'immagine viene processata nuovamente nell'ottava successiva dimezzando la dimensione orizzontale e verticale e tale procedimento viene ripetuto più volte.

La seconda fase di un algoritmo di individuazione e associazione di punti caratteristici consiste nell'estrarre un descrittore per eseguire i confronti, descrittore centrato nel punto caratteristico individuato. Di fatto, per essere invariante alla scala il descrittore deve essere estratto al medesimo fattore di scala associato al punto caratteristico.

Per essere invariante invece alla rotazione il descrittore deve essere estratto da una immagine che ha subito una qualche forma di normalizzazione rispetto alla direzione dominante estratta in intorno del punto valutato.

Da questa immagine ruotata alla scala del punto caratteristico è possibile estrare un descrittore che da importanza ai bordi nell'intorno per essere infine inviariante alla luminosità.

Tra le innumerevoli varianti va segnalato PCA-SIFT che usa PCA per ridurre le dimensioni del problema a un descrittore di soli 36 elementi. PCA viene usato in una fase precedente di addestramento.

5.4 SURF

L'algoritmo *Speeded Up Robust Features* [BETVG08] prende spunto dall'algoritmo SIFT e dalla teoria delle rappresentazioni scala-spazio per proporne una versione ottimizzata dove si sfruttano hessiane approssimate utilizzando l'immagine integrale, sia per individuare i punti caratteristici che per estrarne i descrittori.

SURF è invariante alla traslazione, scala e rotazione ma esiste una variante semplificata, indicata con "U-SURF", che è solo invariante a variazioni di traslazione e scala: in questo caso l'area intorno al punto individuato non viene normalizzata rispetto alla rotazione nel momento in cui viene estratto il descrittore.

In SURF i punti caratteristici vengono rilevati calcolando massimi locali sul determinante dell'immagine Hessiana definita come:

$$\mathcal{H}(x,y;t) = \begin{bmatrix} \frac{\partial}{\partial x^2}G(t) * I & \frac{\partial}{\partial xy}G(t) * I \\ \frac{\partial}{\partial yx}G(t) * I & \frac{\partial}{\partial y^2}G(t) * I \end{bmatrix} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{5.13}$$

immagine formata dalle convoluzioni tra le derivate di secondo ordine della gaussiana di varianza $t = \sigma^2$ e l'immagine nel punto (x, y). Per motivi di prestazioni le derivate delle gaussiane vengono quantizzate a numeri interi e approssimate a regioni rettangolari (*box filters*), ovvero alcune zone rettangolari intorno al punto vengono pesate positivamente, altre negativamente e la loro somma forma l'elemento della matrice $\mathcal{H}$.

La banda di questi filtri approssimati si può stimare come

$$\sigma = \frac{1.2}{9} l \tag{5.14}$$

con l della dimensione del filtro. Il filtro 9×9, il più piccolo possibile, per esempio approssima le derivate della gaussiana di varianza $\sigma = 1.2$.

L'immagine determinante viene calcolata come

$$\det(\mathcal{H}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{5.15}$$

dove w è un fattore che tiene conto della quantizzazione, cerca di compensare i vari errori di arrotondamento, e normalmente viene posto $w = 0.912$ costante. Il determinante infine viene normalizzato rispetto alla dimensione della scala coinvolta, in modo da poterlo confrontare a scale differenti.

L'immagine viene analizzata per più ottave (ogni ottava ha un fattore di scala doppio rispetto all'ottava precedente). Ogni ottava è divisa in un ugual numero di livelli di scala. Il numero di scale per ottava è limitato dalla natura strettamente quantizzata del filtro e le gaussiane approssimate non sono ben equispaziate come nel caso di SIFT. Di fatto 4 intervalli per ottava è l'unico numero di suddivisioni possibile.

All'interno di ogni ottava, al variare della scala s e della posizione, viene eseguita una *Non-Maxima Suppression* $3 \times 3 \times 3$ sull'immagine del determinante di $\mathcal{H}$. I minimi/massimi locali, interpolati attraverso una quadrica tridimensionale come per SIFT, sono i punti interessanti individuati da SURF. La scala è posta uguale alla varianza del filtro associato $s = \sigma$.

Dai punti di massimo così trovati, usando sempre l'immagine integrale, viene estratta l'orientazione dominante nell'intorno del punto (intorno di raggio $6s$ e campionato a passo s). Anche questo caso vengono usate feature di Haar di lato $4s$ e pesate con una gaussiana di distribuzione $\sigma = 2s$.

Attraverso l'informazione sull'orientazione viene generato un descrittore basato sulle direzioni dei gradienti campionando l'area in un intorno di $20s$, divisa in 4×4 regioni e pesando i punti con una gaussiana $\sigma = 3.3s$. All'interno di ogni regione vengono calcolati d_x, d_y, $|d_x|$ e $|d_y|$. Sia l'orientazione che l'istogramma dei gradienti sono estratti alla scala di rilevamento della *feature*.

5.5 AST

L'ultima classe di estrattori di punti caratteristici cade sotto il nome di *Accelerated Segment Test* sviluppate da Rosten. DI questo algoritmo esistono al momento tre versioni leggermente differenti.

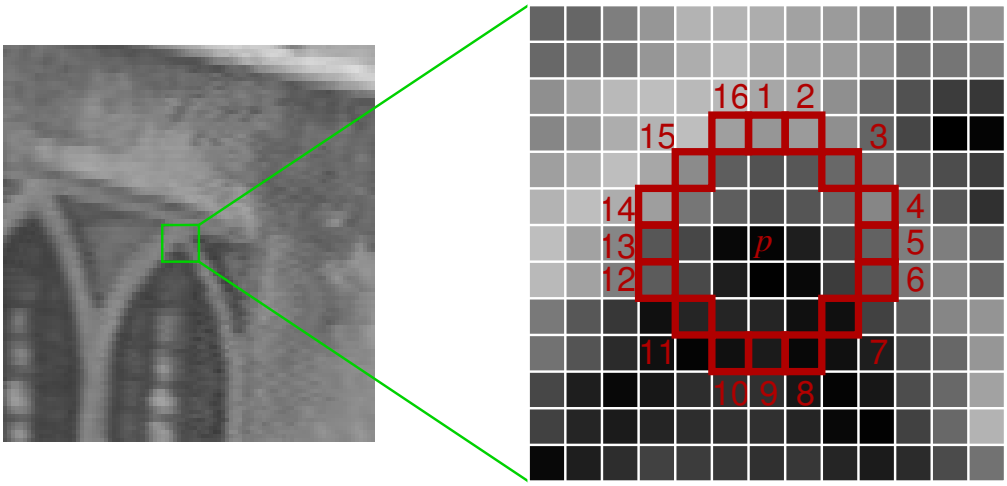


Figura 5.5: FAST: i 16 pixel sulla circonferenza di raggio 3 su cui eseguire il test di consecutività.

La prima versione di *Features from Accelerated Segment Test FAST* [RD05] è probabilmente quella più intuitiva: in questo caso sono indicati come caratteristici quei punti che hanno una sequenza continua di n pixel, lungo a una circonferenza di raggio dato, tutti più (o meno) luminosi del pixel centrale usato come riferimento per il tono di grigio. Nel caso, per esempio, di FAST-9 vengono analizzati i 16 pixels sulla circonferenza di raggio 3 e si verifica se sussistono 9 pixel contigui tutti sopra o tutti sotto una certa soglia rispetto al pixel centrale. Nelle versioni successive [RD06] ottimizza l'estrazione attraverso l'uso di alberi di decisione addestrati per individuare punti caratteristici che massimizzano la quantità locale di informazione. Tali alberi processano sempre i pixel sulla circonferenza.

Questo approccio è tipico degli ultimi anni quando, grazie all'abbondare di dataset pubblici, è stato fatto largo utilizzo di classificatori per costruire individuatori di punti caratteristici stabili. Di fatto, date delle primitive che descrivono l'intorno di un punto, l'utilizzo di una tecnica di ottimizzazione permette di individuare quelle che mostrano maggiore stabilità nel particolare compito. L'articolo di Rosten fra l'altro produce un ottimo *survey* sulle tecniche di estrazione di punti caratteristici precedenti.

Nell'ultima variante (FAST-ER) viene infine estesa l'area da analizzare non solo ai punti di una circonferenza, ma a tutti i pixel nell'intorno del punto centrale.

Capitolo 6

Descrittori

Un altro concetto che ha una collocazione trasversale tra le tematiche di visione artificiale è quello di *descrittore* (*Visual Descriptor*). Il descrittore infatti viene usato in diverse tematiche: viene usato per eseguire il confronto tra punti caratteristici o per generare la mappa di disparità nella visione stereoscopia, per fornire una rappresentazione compatta di una porzione dell'immagine per velocizzare la sua individuazione o ricerca, e grazie a questa soluzione compatta che però preserva gran parte dell'informazione, viene usata per generare lo spazio delle caratteristiche negli algoritmi di classificazione.

A seconda della trasformazione che subisce l'immagine da cui si vogliono caratterizzare i punti, il descrittore deve soddisfare alcuni principi di invarianza

raslazione É quella più facile e viene automaticamente risolta dall'estrattore di punti caratteristici;

scala É un'altra trasformazione che normalmente viene risolta dall'estrattore di punti caratteristici;

uminosità Le immagini possono subire una variazione di luminosità;

rotazione Le immagini possono rappresentare la stessa scena ruotata;

rospettiva I cambi di prospettiva deformano in maniera complessa la porzione di mondo osservata.

Prima che venisse introdotto il concetto di descrittore compatto, il modo universalmente diffuso per confrontare due punti caratteristici era la correlazione tra le aree intorno al punto:

$$d(\mathbf{p}_1, \mathbf{p}_2) = \sum_{\boldsymbol{\delta} \in \Omega} w_\delta (I_1(\mathbf{p}_1 + \boldsymbol{\delta}) - \bar{I}_1)(I_2(\mathbf{p}_2 + \boldsymbol{\delta}) - \bar{I}_2) \tag{6.1}$$

con Ω una finestra di dimensione fissa centrata nel punto delle due immagini e $\bar{I}_n$ il valor medio dell'immagine all'interno della finestra Ω. w_δ è un peso opzionale (ad esempio una gaussiana) per assegnare contributi diversi ai pixel vicini e lontani dal punto. La correlazione è invariante ai cambiamenti di luminosità ma richiede un elevato peso computazionale. In questo caso il descrittore è esattamente la porzione di immagine intorno al punto individuato [Mor80].

Un approccio simile alla correlazione, non invariante alla luminosità ma più performante dal punto di vista computazionale, è la SAD (*Sum of Absolute Differences*):

$$d(\mathbf{p}_1, \mathbf{p}_2) = \sum_{\boldsymbol{\delta} \in \Omega} |I_1(\mathbf{p}_1 + \boldsymbol{\delta}) - I_2(\mathbf{p}_2 + \boldsymbol{\delta})| \tag{6.2}$$

Per rendere la SAD invariante alla luminosità vengono normalmente eseguiti i confronti non sull'immagine originale, ma sulle immagini derivata orizzontale e derivata verticale. Questo ragionamento sembra molto semplice ma può essere ulteriormente generalizzato nel concetto di eseguire il confronto non sull'immagine originale, ma tra una o più immagini estratte attraverso l'ausilio di differenti *kernel*, kernel che provvedono a fornire al descrittore alcuni livelli di invarianza.

É altresì da notare che il confronto tra i pixel tra le immagini è comunque un algoritmo di tipo $O(n^2)$: eseguire questi confronti per punto richiede comunque un elevato peso computazionale e molteplici accessi in memoria. Soluzioni moderne vogliono superare questo limite prevedendo l'estrazione di un descrittore dall'intorno del punto di dimensione inferiore alla quantità di pixel rappresentati che però massimizzi l'informazione contenuta in essa.

Sia SIFT (sezione 5.3) che SURF (sezione 5.4) estraggono i loro descrittori sfruttando le informazioni sulla scala e sulla rotazione estratti dall'immagine (é possibile estrarre queste informazioni in maniera comunque indipendete e pertanto si possono applicare a qualunque classe di descrittori per renderli invarianti a scala e rotazione). I descrittori ottenuti da SIFT e SURF, sono differenti versioni del medesimo concetto, ovvero dell'istogramma dell'orientazione del gradiente (sezione 6.2), esempio di come comprimere in uno spazio di dimensioni ridotte la variabilità intorno al punto.

Tutti i descrittori usati attualmente non usano direttamente i punti dell'immagine come descrittore, ma è facile vedere che basta un sottoinsieme abbastanza ben distribuito dei punti per realizzare comunque una descrizione accurata del punto.

In [RD05] viene creato un descrittore con i 16 pixel presenti lungo la circonferenza discreta di raggio 3. Tale descrizione può essere resa ancora più compatta passando alla forma binaria dei *Local Binary Pattern* descritti in seguito o non vincolata alla circonferenza, come in *Census* o in *BRIEF*. Un altro approccio è campionare in maniera opportuna lo spazio dei kernel [GZS11], estraendo da m coordinate intorno al punto chiave, i valori che assumono convoluzioni dell'immagine originale (Sobel orizzontale e verticale), in modo da creare un descrittore di appena $2m$ valori.

É da notare che, per motivi prettamente computazionali di riutilizzo di risorse, spesso ad ogni particolare estrattore di punti caratteristici viene associato uno specifico estrattore di descrittori.

Da questa introduzione si capisce che descrivere un punto chiave con un insieme di dati inferiore ma allo stesso tempo sufficientemente descrittivo è un discorso che torna utile anche quando si parla di classificazione. Il concetto di descrittore nasce nel tentativo di estrarre informazioni locali dell'immagine che ne permettano di conservare buona parte dell'informazione. In questo modo è possibile eseguire confronti (relativamente) veloci tra punti tra immagini, o usare tali descrittori come caratteristiche su cui addestrare classificatori.

6.1 *Feature* di Haar

Le *Feature* di Haar (il nome deriva dalla somiglianza con le *wavelet* di Haar) indica una serie di filtri per immagini formati come sommatoria e sottrazioni di sottoparti puramente rettangolari dell'immagine stessa [PP99]. Esempi di *feature* di Haar sono mostrati in figura 6.1. Il valore risultante del filtro è la somma dei toni di grigio dei pixel sottesi alle aree in bianco, sottratto il valore dei pixel sottesi alle aree indicate in nero. Per loro natura tali filtri possono venire efficacemente implementati usando l'immagine integrale (sezione 1.14).

Le *Feature* di Haar vengono usate come approssimazione di convoluzioni per il calcolo di punti caratteristici nell'algoritmo di SURF, o come caratteristiche di ingresso ad alberi di decisione per ottenere classificatori deboli.

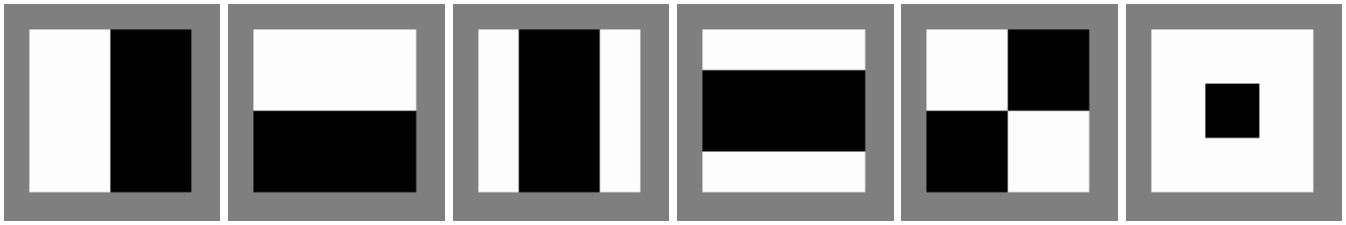


Figura 6.1: Esempi di *Feature* di Haar. Nelle aree chiare e nelle aree scure l'area sottesa viene sommata o sottratta rispettivamente.

Anche se la forma potrebbe essere potenzialmente qualsiasi, il numero di basi per le *feature* è normalmente limitato (si cerca se possibile di evitare *feature* troppo complesse e pesanti computazionalmente).

Oltre al tipo di *feature* è necessario selezionare la sotto-area di applicazione: da ogni sotto-finestra dell'area da analizzare infatti è possibile estrarre un valore a seguito dell'applicazione di una tra queste tante *feature*. Indicare quali sono le feature più discriminanti è lavoro dell'attività di addestramento (*Decision Stump* ordinati con *AdaBoost*) o attraverso tecniche come PCA.

6.2 Istogramma dell'Orientazione del Gradiente

L'istogramma dell'orientazione del gradiente *Histogram of Oriented Gradient* (*HOG*) è una delle tecniche che recentemente ha avuto più successo per descrivere in maniera efficace un area. Tale metodo è infatti usato per la prima volta con successo in SIFT per descrivere i punti caratteristici e insieme ad SVM per ottenere classificatori molto performanti [DT05].

Data la finestra all'interno della quale estrarre il descrittore, viene calcolato modulo e fase di un operatore gradiente (un filtro derivativo, Sobel, o qualsiasi altro) per ogni punto. La fase così estratta viene quantizzata: normalmente vengono calcolati da 6 a 9 *bin* e, opzionalmente, la fase viene calcolata con periodicità π ignorando pertanto il segno del gradiente.

Le idee alla base di HOG sono sia usare la fase del gradiente per avere un descrittore compatto ma invariante fortemente alla luminosità ma anche quella di scomporre la finestra in esame in sottoparti, chiamate celle, eventualmente sovrapposte e potenzialmente di qualunque forma e dimensione. Se normalmente le celle di HOG sono quadrate, é possibile trovare celle rettangolari in R-HOG o circolari in C-HOG.

Da ogni sottoparte in cui viene scomposta l'immagine viene estratto un pezzo di descrittore formato dall'istogramma del modulo del gradiente. Le versioni più usate di HOG cercano di normalizzare localmente la luminosità e il contrasto. Per fare questo, celle spazialmente limitrofe vengono raggruppate in blocchi. Per ogni blocco viene estratto un fattore di normalizzazione con il quale correggere il peso di ogni sotto cella.

Il *bin* dell'istogramma per ogni cella in cui è scomposta l'area rappresentano il descrittore, descrittore da usare nei confronti di punti o negli addestramenti per riconoscimento di oggetti.

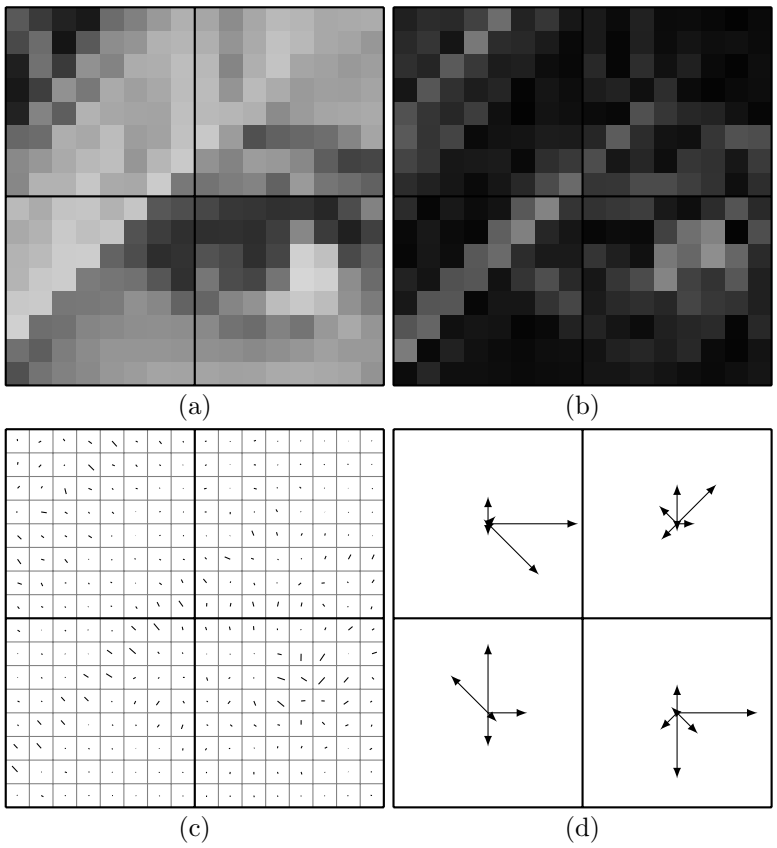


Figura 6.2: Calcolo dell'istogramma del gradiente: dalle celle, anche sovrapposte, in cui viene scomposta l'immagine *(a)* si calcolano modulo *(b)* e fase *(c)* dei gradienti e per ogni cella viene costruito un istogramma *(d)*.

6.3 Descrittore da Canale Integrale

Figura 6.3: Immagine dei canali usati da ICF. Da sinistra l'immagine originale e a seguire le immagini dei differenti canali: 8 canali per la fase del gradiente quantizzata, 1 canale con il modulo del gradiente e 3 canali per le componenti di LUV rispettivamente.

Le varianti di HOG presentano forme delle celle variabili e ci si è accorti che un modo per velocizzare il calcolo dell'istogramma del modulo delle fasi era sfruttare nuovamente l'immagine integrale: pertanto, a cavallo tra HOG e le *Feature* di Haar, recentemente hanno mostrato interessanti prestazioni le *Integral Channel Feature* essendo di fatto una generalizzazione di HOG che sfrutta l'immagine integrale.

I valori caratteristici che si possono estrarre derivano dalla sommatoria di aree calcolate non direttamente sull'immagine originale ma da differenti immagini secondarie, ottenute attraverso elaborazioni non-lineari dalla zona da caratterizzare. Tra le possibili elaborazioni quelle più diffuse sono i canali della fase del gradiente già visti in HOG, l'immagine della magnitudine del gradiente, l'immagine stessa a toni di grigio e, se disponibili, due canali aggiuntivi rappresentanti la crominanza. Per quanto riguarda il gradiente viene spesso calcolato con Sobel ma diversi esperimenti mostrano che il semplice filtro derivativo produce risultati comunque soddisfacenti. Anche in questo caso la fase di Sobel può essere utilizzata con o senza segno, a seconda delle particolari applicazioni.

Lo scalare rappresentante la caratteristica da estrarre è semplicemente la somma di un area rettangolare all'interno di uno dei canali calcolati.

6.4 Descrittori Binari

Uno dei problemi dei descrittori tradizionali è che sono formati da un vettore di un certo numero di valori, normalmente, a causa di eventuali normalizzazioni, a virgola mobile: come conseguenza sia per estrarre questo vettore che poi per eseguire la fase di confronto è necessario un elevato tempo computazionale.

Una delle alternative più promettenti è quella di estrarre come descrittore un vettore binario. Il vettore binario occupa meno spazio in memoria e per eseguire il confronto è sufficiente calcolare la distanza di Hamming tra le rispettive stringhe binarie. La distanza di Hamming si calcola in maniera molto efficiente calcolando lo XOR delle stringhe binarie e contando i bit attivi (POPCOUNT).

6.4.1 Trasformazione Census

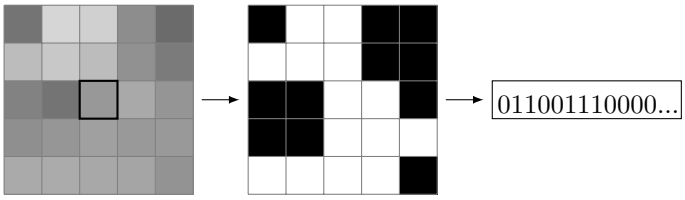


Figura 6.4: Calcolo della trasformazione di Census in un intorno di un punto: l'intorno di un punto viene binarizzato rispetto al valore del punto stesso e da questa sogliatura viene costruita una stringa binaria.

La trasformazione Census (*Census-Transformation*) [ZW94] consiste nel descrivere la porzione di un immagine intorno a un punto attraverso una stringa di bit. Per ogni pixel dell'immagine viene analizzata in maniera ordinata l'area circostante, di dimensioni e forma fissate, e ogni pixel di quest'area viene confrontato con il pixel generatore. Se il pixel ha una intensità di grigio maggiore viene associato un bit 1 mentre se ha intensità inferiore viene associato il bit 0:

$$\tau(\mathbf{x}) = \begin{cases} 1 & I(\mathbf{x}) < I(0) \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

Attraverso la scansione dell'area in maniera ordinata è possibile generare una stringa binaria. Un esempio della trasformazione Census di un area 5×5 è mostrata in figura 6.4: da quest'area viene generata una stringa binaria da 25-1 bit (il pixel centrale è di fatto ininfluente).

La trasformazione Census mostra il suo potenziale nel caso di confronti: due punti generici invece che essere confrontati attraverso una SAD dell'area circostante, vengono confrontati attraverso la distanza di Hamming tra le rispettive stringhe binarie della trasformata di Census.

Attraverso la costruzione della stringa binaria, sfruttando la differenza di tono di grigio, la trasformazione Census è abbastanza invariante alla luminosità.

6.4.2 Local Binary Pattern (LBP)

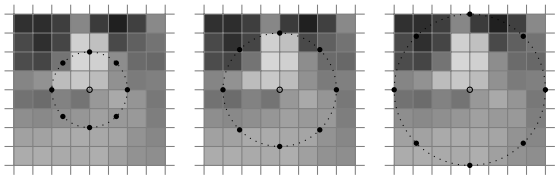


Figura 6.5: Pixel considerati durante l'estrazione di un descrittore LBP a 8 bit, al variare del raggio della circonferenza.

Nella prima versione di LBP [OPM02], il descrittore risultava indistinguibile alla trasformata di Census su una finestra 3×3: per ogni punto immagine vengono esaminati gli 8 pixel nel vicinato, sogliati attraverso il pixel centrale e in questo modo viene generata una stringa di 8 bit, equivalente pertanto a un descrittore intero compreso tra 0 e 255.

Questo concetto originale è stato in seguito esteso a n punti lungo una circonferenza di raggio ρ centrata nel pixel di cui si vuole calcolare la caratteristica (figura 6.5). Siccome il punto del raggio normalmente non cade esattamente su un pixel, si può eseguire una interpolazione bilineare per stimare il valore da sogliare per costruire la stringa binaria.

L'operatore LBP produce 2^n possibili valori per ogni punto dell'immagine. Nel caso in cui l'immagine venga ruotata, i valori dei pixel si muovono lungo la circonferenza e come conseguenza ruotano anche i bit all'interno della stringa binaria. È possibile ottenere un operatore LBP invariante alla rotazione, normalizzando la stringa attraverso una qualche trasformazione. Una di queste trasformazioni è, per esempio, eseguire n rotazioni sulla stringa binaria e prendere, tra tutte le risultati, quella di valore minimo:

$$\mathrm{LBP}_{r.i.} = \min_i \mathrm{ROR}_i(\mathrm{LBP}) \quad i \in [0, n-1] \tag{6.4}$$

6.4.3 BRIEF

La trasformata di Census non impone una forma specifica all'area su cui eseguire i confronti per generare la stringa binaria. Questo limite viene affrontato in [CLSF10], uno dei primi lavori che formalizza il problema della costruzione di descrittori binari discriminanti.

La maggior parte dei descrittori binari si ispira a Census, generalizzandolo: invece di confrontare ogni pixel con il solo centro dell'area, si effettuano confronti tra coppie di pixel selezionate in modo opportuno. La funzione di confronto è definita come:

$$\tau(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \tilde{I}(\mathbf{x}) < \tilde{I}(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

dove $\mathbf{x}$ e $\mathbf{y}$ sono coordinate di due pixel all'interno della patch attorno al punto da descrivere, e $\tilde{I}(\cdot)$ rappresenta l'intensità del pixel su una versione filtrata (tipicamente passa basso) dell'immagine originale.

La selezione delle coppie di pixel (la "maschera") avviene tramite un processo di addestramento su immagini campione, con l'obiettivo di massimizzare la capacità discriminante del descrittore.

In letteratura esistono numerosi approcci che affrontano il problema della scelta dei punti e del tipo di filtraggio da applicare all'immagine.

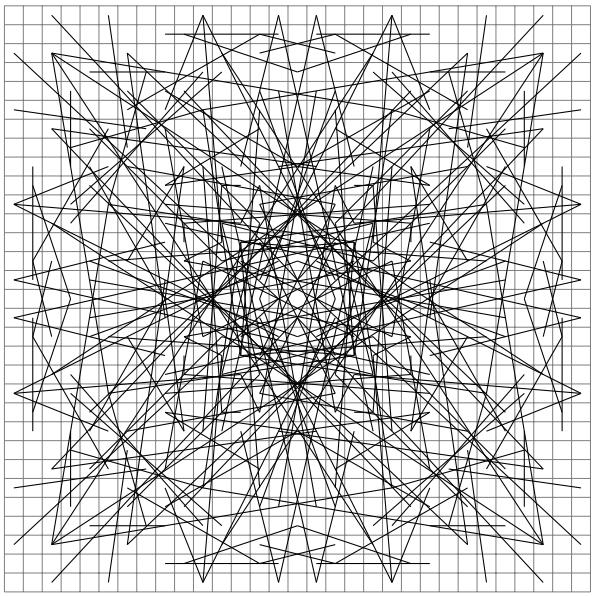


Figura 6.6: Esempio di descrittore BRIEF 256bit.

6.4.4 ORB

ORB (Oriented FAST and Rotated BRIEF) [RRKB11] nasce dalla combinazione tra il rilevatore di keypoint FAST e il descrittore BRIEF, modificato per essere invariante alla rotazione. Poiché FAST non fornisce informazioni sull'orientamento, ORB lo calcola determinando il centroide dell'intensità luminosa all'interno della patch attorno al keypoint: il vettore che collega il punto al centroide è rigido rispetto alle rotazioni e sufficientemente robusto alle variazioni di illuminazione.

Utilizzando questo angolo, ORB ruota il pattern BRIEF (impiegando pattern ruotati precalcolati, ad esempio quantizzando l'angolo giro in 30 settori) per generare un descrittore binario robusto alla rotazione, noto come rBRIEF. Inoltre, applicando FAST e BRIEF a diversi livelli di una piramide di immagini, si ottiene anche invarianza alla scala.

ORB offre prestazioni comparabili a quelle di SIFT e SURF in termini di accuratezza, ma con una velocità significativamente superiore e, soprattutto, senza vincoli di brevetto.

6.5 Machine Learning

Con l'evoluzione delle tecniche di Machine Learning il concetto di descrittore è andato via via sparendo lasciando spazio a una teoria molto più armonica riguardante il concetto di informazione. Vedere alcuni algoritmi come le RBN (sezione 4.8.2) e le CNN (sezione 4.8.5) in qualità di *auto-encoder* quando collegate a un *decoder* portano il concetto di descrittore al limite ovvero permettono di estrarre il numero minimo di caratteristiche che permettono la ricostruzione di una porzione dell'immagine (sotto una determinata metrica).

Capitolo 7

Tracking

A valle dell'estrazione dei punti caratteristici e del loro descrittore ci sta un discorso sull'associazione dei descrittori e sul concetto di flusso ottico denso.

7.1 Confronto e Associazione di descrittori

Come conclusione a questo capitolo è necessario spendere infine due parole sul discorso del confronto.

Siano I_1 e I_2 due immagini da analizzare e siano $\mathbf{p}_1$ e $\mathbf{p}_2$ due punti, probabilmente caratteristici, individuati rispettivamente nella prima e nella seconda immagine. Per sapere se questi due punti immagine rappresentano il medesimo punto, normalmente non osservato dallo stesso punto di vista e pertanto affetto da trasformazioni affini (traslazioni, cambi di scala, rotazioni), omografiche e probabilmente cambi di luminosità, è necessario definire una qualche forma di metrica $d(\mathbf{p}_1, \mathbf{p}_2)$ per eseguire tale confronto. Associato ad ogni descrittore è possibile definire una particolare metrica. In generale le metriche più diffuse sono la L1 (Manhattan, *SAD*) e la L2 (Euclidea, *SSD*).

Siccome i punti estratti dalle due immagini saranno sicuramente più di uno, deve essere eseguita una scansione e verrà associato a ogni punto della prima immagine solo quel punto della seconda che ha una distanza minima rispetto alla metrica selezionata:

$$\hat{\mathbf{p}_2} = \arg\min_i d(\mathbf{p}_1, \mathbf{p}_{2,i}) \tag{7.1}$$

Solitamente, per ridurre il numero di confronti errati, viene confermata l'associazione solo se la metrica è inferiore a una data soglia e il rapporto tra il miglior confronto e il secondo miglior confronto è inferiore a una seconda soglia di unicità.

Infine, dopo aver trovato $\mathbf{p}_2$, la miglior associazione del punto $\mathbf{p}_1$ sulla seconda immagine, si può verificare che $\mathbf{p}_2$ non abbia sulla prima immagine associazioni migliori.

7.2 Lucas-Kanade

Il metodo di stima del flusso ottico di Lucas-Kanade [LK81] è un metodo per stimare il movimento di caratteristiche interessanti in scene successive di un video. L'obiettivo è quello di associare un vettore movimento (u, v) ad ogni pixel "interessante" della scena confrontando due immagini consecutive.

L'algoritmo fa le seguenti assunzioni:

- La luminosità tra immagini consecutive non cambia;
- Le due immagini devono essere temporalmente vicine in maniera che gli oggetti non abbiano uno scostamento significativo (l'algoritmo lavora bene con oggetti in lento movimento);
- L'immagine contiene oggetti con sufficiente tessitura in scala di grigi (l'algoritmo originale non usa esplicitamente il colore) il cui gradiente cambia graduatamente.

Partendo dell'equazione del flusso ottico per ogni punto (x, y):

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t) \tag{7.2}$$

dove $I(t)$ è un immagine e $I(t + \delta t)$ la consecutiva. Con l'espansione in serie di taylor al primo ordine:

$$\begin{aligned} &I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t) \\ &I(x, y, t) + \tfrac{\partial I}{\partial x}(x, y, t)u\delta t + \tfrac{\partial I}{\partial y}(x, y, t)v\delta t + \tfrac{\partial I}{\partial t}(x, y, t)\delta t = I(x, y, t) \\ &\tfrac{\partial I}{\partial x}(x, y, t)u + \tfrac{\partial I}{\partial y}(x, y, t)v + \tfrac{\partial I}{\partial t}(x, y, t) = 0 \end{aligned} \tag{7.3}$$

L'algoritmo di Lucas-Kanade assume che il cambiamento di luminosità di un pixel della scena venga totalmente compensato dal gradiente della scena stessa ovvero

$$I_x u + I_y v + I_t = 0 \tag{7.4}$$

dato il gradiente temporale I_t e il gradiente spaziale (I_x, I_y).

Ovviamente il singolo pixel non contiene abbastanza informazione per risolvere questo problema. Per raccogliere più osservazioni viene assunto che un intorno del pixel abbbia lo stesso moto, ovvero

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_n) & I_y(\mathbf{p}_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_n) \end{bmatrix} \tag{7.5}$$

dove $\mathbf{p}_1 \dots \mathbf{p}_n$ sono i punti nell'intorno del punto da stimare. La soluzione può essere ottenuta attraverso il metodo delle *normal equations*

$$\begin{bmatrix} \sum I_x I_y & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} \tag{7.6}$$

Se si nota questa è anche la matrice dei punti caratteristici sfruttata poi da Shi-Tomasi o da Harris (vedi 5.2): i punti caratteristici di questa matrice sono punti che vengono facilmente tracciati con l'algoritmo di Lucas-Kanade.

Quando il moto è più grande di un pixel è necessario un algoritmo iterativo per risolvere il problema e un approccio *coarse-to-fine* per evitare i minimi locali: esisterà una scala per la quale il moto del pixel sarà inferiore ad un pixel.

Capitolo 8

Pin-Hole Camera

In questo capitolo viene affrontato il problema di descrivere il processo attraverso il quale la luce incidente sugli oggetti viene impressa su un sensore digitale. Tale concetto è fondamentale nell'elaborazione delle immagini in quando fornisce la relazione che lega i punti di un'immagine con la loro posizione nel mondo, ovvero permette di determinare la zona del mondo associata a un *pixel* dell'immagine o, viceversa, individuare l'area dell'immagine che raccoglie una determinata regione in coordinate mondo.

Il modello proiettivo universalmente accettato, detto della *Pin-Hole Camera*, è basato su semplici rapporti geometrici[1].

In figura 8.1 è mostrato uno schema molto semplificato di come avviene la formazione dell'immagine sul sensore. Il punto osservato $(x_i, y_i, z_i)^\top$, espresso in coordinate camera, viene proiettato su una cella del sensore $(\tilde{u}_i, \tilde{v}_i)^\top$. Tutti questi raggi passano per uno stesso punto: il punto focale (*pin-hole*).

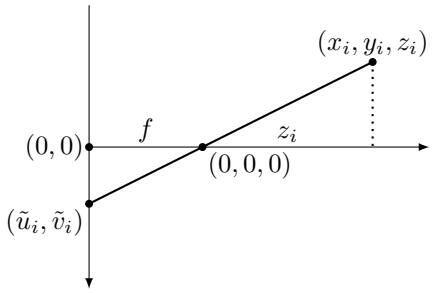


Figura 8.1: Il modello di camera *pin-hole*. Un punto mondo in coordinate camera viene proiettato sul piano immagine.

Analizzando la figura 8.1 si vede come i rapporti tra triangoli simili generati dai raggi ottici descrivono l'equazione che permette di proiettare un generico punto $(x_i, y_i, z_i)^\top$, espresso in coordinate *camera* (uno dei sistemi di riferimento in cui si può operare), in coordinate sensore $(\tilde{u}_i, \tilde{v}_i)^\top$:

$$\begin{bmatrix} \tilde{u}_i \\ \tilde{v}_i \end{bmatrix} = \frac{f}{z_i} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{8.1}$$

dove f è la distanza focale (distanza tra il *pin-hole* e il sensore). È da precisare che le coordinate $(x_i, y_i, z_i)^\top$, espresse in coordinate camera, in questo libro seguono la regola *della mano sinistra* (molto usata in *computer graphics*), contrapposta alla regola *della mano destra* (più usata in applicazioni robotiche) invece scelta per esprimere le coordinate mondo. L'utilizzo della coordinata z per esprimere la distanza è un obbligo puramente matematico a causa delle trasformazioni che verranno presentate fra poco.

Le coordinate sensore $(\tilde{u}_i, \tilde{v}_i)^\top$ non sono le coordinate immagine ma sono ancora delle coordinate "intermedie". È quindi necessario applicare una ulteriore trasformazione per ottenere le coordinate immagine:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} D_u \tilde{u}_i \\ D_v \tilde{v}_i \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \tag{8.2}$$

dove le coordinate (u_0, v_0) (*principal point*) tengono conto dello scostamento dell'origine delle coordinate nell'immagine memorizzata rispetto alla proiezione del punto focale sul sensore.

D_u e D_v sono fattori di conversione tra le unità del sistema di riferimento del sensore (metri) con quelle immagine (pixel) e tengono conto dei diversi fattori di conversione coinvolti. Con l'avvento dei sensori digitali normalmente $D_u = D_v$.

[1] La maggior parte dei modelli ricade nel modello pinhole inteso come modello dove tutti i fasci luminosi passano per uno stesso punto: esistono tuttavia modelli che generalizzano il modello pinhole a camere catadiottriche, per esempio il modello di Mei, o altri modelli diversi come il *double sphere*.

In mancanza di informazioni, reperibili dai vari *datasheet*, su f, D_u e D_v, c'è la tendenza ad accorpare queste variabili in due nuove variabili chiamate k_u e k_v, le lunghezze focali efficaci misurate in pixel, ottenibili in maniera empirica dalle immagini, come si vedrà nella sezione sulla calibrazione. Queste variabili, coinvolte nella conversione tra coordinate sensore e coordinate immagine, sono tra loro in relazione come

$$\begin{aligned} k_u &= D_u f = \frac{u_0}{\tan \alpha_u} \\ k_v &= D_v f = \frac{v_0}{\tan \alpha_v} \end{aligned} \tag{8.3}$$

con α_u e α_v angoli approssimabili alla semiampiezza dell'apertura della camera (orizzontale e verticale rispettivamente). Quando l'ottica non è distorta e il sensore ha pixel quadrati, k_u e k_v tendono ad assumere lo stesso valore.

A causa della presenza del rapporto, l'equazione (8.1) non è chiaramente rappresentabile in un sistema lineare. Tuttavia risulta possibile modificare tale scrittura, aggiungendo un incognita λ e un vincolo ulteriore, per poter rappresentare in forma di sistema lineare tale equazione. Per fare questo verrà sfruttata la teoria presentata in sezione 1.4 riguardante le coordinate omogenee. Grazie alle coordinate omogenee si mostra facilmente che il sistema (8.1) si può scrivere come

$$\begin{bmatrix} \lambda\, u_i \\ \lambda\, v_i \\ \lambda \end{bmatrix} = \lambda \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{8.4}$$

risolto per $\lambda = z_i$. Per questo motivo λ si sottointende e si usano invece le coordinate omogenee: per ottenere il punto in coordinate non omogenee bisogna infatti dividere le prime due coordinate per la terza, ottenendo l'equazione (8.1). L'utilizzo delle coordinate omogenee permette di rendere implicita la divisione per la coordinata z.

La matrice $\mathbf{K}$, unendo le trasformazioni (8.2) e (8.3), può essere scritta come:

$$\mathbf{K} = \begin{bmatrix} \frac{u_0}{\tan \alpha_u} & k_\gamma & u_0 \\ 0 & \frac{v_0}{\tan \alpha_v} & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} k_u & k_\gamma & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8.5}$$

Tale matrice non dipendendo, come vedremo successivamente, da fattori che non siano altri che quelli della camera stessa, è detta matrice dei fattori intrinseci. La matrice $\mathbf{K}$ è una matrice triangolare superiore, definita da 5 parametri.

Con i sensori digitali moderni e la costruzione di telecamere non manualmente ma con macchine a controllo numerico precise, è possibile porre lo *skew factor* k_γ, un fattore che tiene conto del fatto che l'angolo tra gli assi nel sensore non sia esattamente 90 gradi, a zero.

Ponendo $k_\gamma = 0$, l'inversa della matrice (8.5) si può scrivere come:

$$\mathbf{K}^{-1} = \begin{bmatrix} \frac{1}{k_u} & 0 & -\frac{u_0}{k_u} \\ 0 & \frac{1}{k_v} & -\frac{v_0}{k_v} \\ 0 & 0 & 1 \end{bmatrix} \tag{8.6}$$

La conoscenza di questi parametri (vedi sezione 8.5 riguardante la calibrazione) determina la possibilità di trasformare un punto da coordinate camera a coordinate immagine o, viceversa, generare la retta in coordinate camera sottesa a un punto immagine.

Con questa modellazione, in ogni caso, non si è tenuto conto dei contributi dovuti alla distorsione della lente. Il modello della pin-hole camera è infatti valido solamente se le coordinate immagine che si utilizzano si riferiscono a immagini senza distorsione.

8.1 Distorsione della lente

La totalità delle telecamere commerciali devia dal modello della *pin-hole* camera e tale deviazione è generalmente tanto maggiore quanto grande è il campo visivo della camera: siccome ogni ottica è composta da un certo numero di lenti, la distorsione deriva dalle non idealità nella fase di produzione e di assemblaggio dell'ottica. Ottenere infatti una lente non distorcente è un processo estremamente costoso e soprattutto nelle applicazioni a basso costo dove bisogna fare affidamento a ottiche economiche risulta un problema molto evidente.

Queste non idealità generano una distorsione non lineare difficilmente modellizzabile e, anche per il fatto che tale distorsione dipende dall'interazione tra la lente e il sensore, i produttori di lenti normalmente non forniscono, o non riescono a fornire, informazioni geometriche su come rappresentare tale distorsione.

È importante osservare che il modello della *pin-hole* camera è valido solamente se l'immagine su cui si lavora è non distorta pertanto calibrare, ovvero correggere la distorsione geometrica, è un prerequisito per ricostruire in maniera accurata la tridimensionalità della scena osservata.

Dal punto di vista del raggio ottico, la distorsione introdotta dalla lente si pone tra il mondo e il *pin-hole*. L'equazione della camera pin-hole modificata con la distorsione dell'ottica si trasforma in

$$\mathbf{p} = \mathbf{K} f_d([\mathbf{Rt}]\mathbf{x}) \tag{8.7}$$

Con questo formalismo la distorsione f_d trasforma un punto da coordinate non-distorte in coordinate distorte. Questa scelta, rispetto alla formulazione inversa, viene da considerazioni puramente pratiche: siccome l'obiettivo è quello di avere un immagine in uscita densa e non-distorta (si veda la discussione in sezione 1.12), è necessario calcolare quella funzione che trasforma appunto un punto non-distorto in un punto distorto.

In generale i contributi distorcenti della lente si dividono in radiali (diretti lungo la direttrice che unisce il punto al centro di distorsione) o tangenziali (che sono perpendicolari alla direttrice). I contributi tangenziali (e altri contributi qui non citati) sono normalmente piccoli mentre la distorsione radiale è sempre rilevabile e, man mano che la focale diventa corta, in generale aumenta di intensità.

Questa sezione si occupa di ricavare una relazione generale tra il punto ideale (x, y) e l'effettivo punto immagine distorto osservato $(\breve{x}, \breve{y})$.

In tutta l'immagine esiste un solo punto (x_d, y_d), definito centro di distorsione, dove la distorsione non produce effetti. Per questo punto $(x, y) = (\breve{x}, \breve{y})$.

Per definire la distorsione è necessario operare in una nuova serie di coordinate, relative al centro di distorsione:

$$\begin{aligned} \bar{x} &= x - x_d \\ \bar{y} &= y - y_d \end{aligned} \tag{8.8}$$

Il centro di distorsione è normalmente vicino a $(0, 0)$ ma non c'è nessuna garanzia che coincida con il *principal point*. In diversi articoli viene infatti proposto, come approssimazione, ignorare il centro di distorsione e far coincidere il centro di distorsione con il *principal point* o considerare solamente il termine di *decentering distortion*.

La formulazione classica di *Brown-Conrady* [Bro66] modella la distorsione della lente sotto forma di scostamento:

$$\begin{aligned} \breve{x} &= x + \delta_x(\bar{x}, \bar{y}) \\ \breve{y} &= y + \delta_y(\bar{x}, \bar{y}) \end{aligned} \tag{8.9}$$

Tali scostamenti possono essere suddivisi per contributi:

radial distortion Lo scostamento dovuto alla distorsione radiale ha equazione

$$\begin{aligned} \delta_x^r &= \bar{x} f_r(r) \\ \delta_y^r &= \bar{y} f_r(r) \end{aligned} \tag{8.10}$$

dove $f_r(r)$ è una funzione solo del raggio $r = \sqrt{\bar{x}^2 + \bar{y}^2}$, distanza euclidea tra il punto e il centro di distorsione, e con il vincolo $f_r(0) = 1$.

La funzione $f_r(r)$ della distorsione radiale non è un modello conosciuto ma può essere approssimata attraverso i primi termini dello sviluppo in serie:

$$f_r(r) = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots \tag{8.11}$$

La presenza delle sole potenze multiple di 2 è dovuta alla simmetria della funzione f_r.

thin prism distortion imperfezioni del costruttore e disallineamento tra il sensore e la lente, introducono ulteriori distorsioni asimmetriche. Si modella solitamente come

$$\begin{aligned} \delta_x^{(p)} &= s_1 r^2 + s_3 r^4 + \dots \\ \delta_y^{(p)} &= s_2 r^2 + s_4 r^4 + \dots \end{aligned} \tag{8.12}$$

Tali contributi sono spesso inadeguati tuttavia per descrivere gli effetti di decentramento dell'ottica.

decentering distortion È normalmente causata dall'assemblaggio errato della lente e dei diversi componenti che compongono l'ottica. Il modello di *Brown-Conrady* rappresenta il contributo di decentramento nella forma

$$\begin{aligned} \delta_x^{(t)} &= (p_1(r^2 + 2\bar{x}^2) + 2p_2\bar{x}\bar{y})(1 + p_3 r^2 + \dots) \\ \delta_y^{(t)} &= (p_2(r^2 + 2\bar{y}^2) + 2p_1\bar{x}\bar{y})(1 + p_3 r^2 + \dots) \end{aligned} \tag{8.13}$$

Questo contributo è costituito sia da una parte radiale che da una parte tangenziale.

Inserendo tutti questi contributi all'interno dell'equazione (8.9), il modello *Brown-Conrady* complessivo si scrive come

$$\begin{aligned}
\bar{x} &= x - x_d \\
\bar{y} &= y - y_d \\
r &= \sqrt{\bar{x}^2 + \bar{y}^2} \\
\breve{x} &= x + \bar{x}(k_1 r^2 + \ldots) + (p_1(r^2 + 2\bar{x}^2) + 2p_2\bar{x}\bar{y})(1 + p_3 r^2 + \ldots) + s_1 r^2 + \ldots \\
\breve{y} &= y + \bar{y}(k_1 r^2 + \ldots) + (2p_1\bar{x}\bar{y} + p_2(r^2 + 2\bar{y}^2))(1 + p_3 r^2 + \ldots) + s_2 r^2 + \ldots
\end{aligned} \tag{8.14}$$

Di fatto la distorsione radiale è dominante e, in buona parte delle applicazioni, i primi termini sono più che sufficienti.

Per esempio OpenCV modella la distorsione con il modello *R3P1*: 3 termini radiali (k_1, k_2, k_3) e il termine di decentramento di primo grado (p_1, p_2).

I coefficienti della distorsione sono ricavati con diverse tecniche disponibili in letteratura applicate ad immagini acquisite in ambiente strutturato (griglie di calibrazione). Normalmente viene usato un minimizzatore non lineare e, o si lavora su rette e si itera fino a che tutte le curve dell'immagine non diventino rette *plumb-line method* [DF01], o si forza che i punti su un piano di coordinate note rappresentino una omografia. Tali tecniche sono applicabili solamente se si opera in coordinate immagine (approccio 1).

Per calibrare la distorsione in coordinate camera normalizzate (approccio 2) bisogna che contemporaneamente distorsione e parametri intrinseci della camera vengano calcolati [Zha99]. Una stima iniziale dei parametri intrinseci può venire da un minimizzatore lineare ma la stima finale si ottiene solo attraverso un minimizzatore non lineare.

La tecnica ampiamente usata per stimare i parametri della distorsione è ottimizzare l'osservazione di punti caratteristici sull'immagine la cui posizione in coordinate mondo è conosciuta in modo da forzare una proiezione prospettica completa (sezione 8.5.6).

8.2 Coordinate Mondo e Coordinate Camera

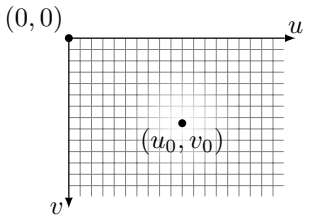


Figura 8.2: Coordinate Immagine (*Image coordinates*)

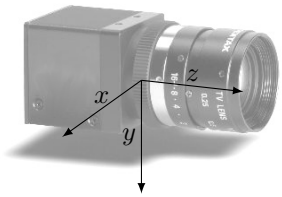


Figura 8.3: Coordinate Camera (*Camera coordinates*)

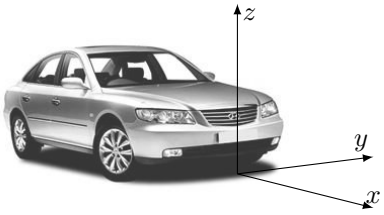


Figura 8.4: Esempio di coordinate "Veicolo" o "Mondo": Front-Left-Up o ISO 8855 (*World coordinates*)

Quando si opera su problemi pratici risulta necessario passare da un sistema di riferimento solidale con la camera, dove il punto $(0, 0, 0)^\top$ coincide con il fuoco (*pin-hole*), a un sistema di riferimento più generico, che meglio si adatti alle esigenze dell'utilizzatore, dove la camera è posizionata in un punto generico del "mondo" e orientata rispetto ad esso in modo arbitrario. Questo discorso si applica a qualsiasi sensore generico, anche non video, definendo delle relazioni che permettono di passare i punti da coordinate mondo a coordinate sensore e viceversa.

È necessario a questo punto fare una precisazione sulla terminologia legata ai sistemi di riferimento in questo libro: viene definito come sistema di riferimento "mondo" il sistema che di volta in volta è considerato assoluto e fisso, rispetto al quale viene posizionato il sensore. In figura 8.4 per esempio l'origine del sistema "mondo" è associato a un punto del veicolo (il

punto frontale per esempio). In questo caso il sistema "veicolo" (*body*) o "mondo" (*world*) sono sinonimi. Questa distinzione però viene meno nel caso in cui ci sia un veicolo in movimento rispetto a un "mondo" che nuovamente si può definire il sistema di riferimento fisso. In tal caso avremo le coordinate sensore, quelle locali del veicolo/body e infine quelle del mondo. Solitamente però il sistema di assi che contraddistingue sensore, veicolo e mondo viene mantenuto consistente.

Se in coordinate camera il ruolo speciale che assume la coordinata z è dovuto a motivazioni puramente matematiche, ovvero all'uso di coordinate omogenee che in fase di proiezione obbliga la divisione delle prime due componenti per la terza, in coordinate "sensore" questo limite viene meno. Anche se non vincolante in nessuna maniera, in questo libro è usato come sistema "sensore", "body" e "mondo" quello presentato in figura 8.4 (ISO 8855) che assegna all'asse z l'altezza del punto dal suolo.

Pertanto, per arrivare all'equazione definitiva della *pin-hole* camera si parte dall'equazione (8.4) e si applicano le seguenti considerazioni

- gli assi sono scambiati tra loro attraverso una permutazione $\mathbf{\Pi}$ (che è comunque una rotazione) per ottenere il sistema di riferimento finale;
- il sensore deve essere ruotato attraverso una trasformazione ${}^{w}\mathbf{R}_{b}$ e di conseguenza non coincide con gli assi del sistema di riferimento "mondo";
- il pin-hole non coincide ora con il punto $(0,0,0)^\top$ ma giace in un generico punto $\mathbf{t}_0 = (x_0, y_0, z_0)^\top$ espresso in coordinate mondo.

La conversione da coordinate "mondo" a coordinate "camera", essendo una composizione di rotazioni, è anche essa una rotazione di equazione $\mathbf{R} = {}^{c}\mathbf{R}_{w} = \mathbf{\Pi}\,{}^{w}\mathbf{R}_{b}^{-1}$.

Sia $(x_i, y_i, z_i)^\top$ un punto in coordinate "mondo" e $(\tilde{x}_i, \tilde{y}_i, \tilde{z}_i)^\top$ il medesimo punto in coordinate "camera". La relazione che lega questi due punti si può scrivere come

$$\begin{bmatrix}\tilde{x}_i\\ \tilde{y}_i\\ \tilde{z}_i\end{bmatrix} = \mathbf{R}\left(\begin{bmatrix}x_i\\ y_i\\ z_i\end{bmatrix} - \mathbf{t}_0\right) = \mathbf{R}\begin{bmatrix}x_i\\ y_i\\ z_i\end{bmatrix} + \tilde{\mathbf{t}}_0 \tag{8.15}$$

dove $\mathbf{R}$ è una matrice 3×3 che converte da coordinate mondo a coordinate camera, tiene conto delle rotazioni e della variazione del segno degli assi tra coordinate mondo e coordinate camera (vedi appendice A), mentre il vettore

$$\tilde{\mathbf{t}}_0 = -\mathbf{R}\mathbf{t}_0 \tag{8.16}$$

rappresenta la posizione del pin-hole $\mathbf{t}_0$ rispetto all'origine del sistema mondo, rappresentato però nel sistema di coordinate camera.

Va ricordato che le matrici di rotazione sono matrici ortonormali: hanno determinante 1, conservano pertanto distanze e aree, e l'inversa di una matrice di rotazione è la sua trasposta.

La matrice $\mathbf{R}$ e il vettore $\mathbf{t}_0$ possono venire accorpati in forma di matrice 3×4 sfruttando le coordinate omogenee. Grazie a questa rappresentazione, è possibile scrivere in maniera estremamente compatta la proiezione di un punto, espresso in coordinate mondo, omogenee a $(x_i, y_i, z_i)^\top$, in un punto di coordinate immagine, omogenee a $(u_i, v_i)^\top$:

$$\lambda\begin{bmatrix}u_i\\ v_i\\ 1\end{bmatrix} = \mathbf{K}[\mathbf{R}|\tilde{\mathbf{t}}_0]\begin{bmatrix}x_i\\ y_i\\ z_i\\ 1\end{bmatrix} \tag{8.17}$$

Da questa equazione risulta abbastanza esplicito che ad ogni punto dell'immagine (u_i, v_i) sono associati infiniti punti del mondo $(x_i, y_i, z_i)^\top$ che vivono su una retta al variare del parametro λ.

Sottointendendo λ e raccogliendo le matrici si ottiene l'equazione finale della *pin-hole* camera (che non tiene, né deve tener conto, della distorsione):

$$\begin{bmatrix}u_i\\ v_i\\ 1\end{bmatrix} = \mathbf{K}[\mathbf{R}|\tilde{\mathbf{t}}_0]\begin{bmatrix}x_i\\ y_i\\ z_i\\ 1\end{bmatrix} = \mathbf{P}\begin{bmatrix}x_i\\ y_i\\ z_i\\ 1\end{bmatrix} \tag{8.18}$$

avendo definito $\mathbf{P} = \mathbf{K}[\mathbf{R}|\tilde{\mathbf{t}}_0]$ matrice proiettiva (*camera matrix*) che verrà usata in seguito [Str87]. La matrice $\mathbf{P}$ è una matrice 3×4 ed, essendo rettangolare, non è invertibile.

È da sottolineare che ponendo un vincolo aggiuntivo sui punti, per esempio $z_i = 0$, la matrice $\mathbf{P}$ si riduce a una matrice 3×3, invertibile, che è esattamente la matrice omografica (vedi sezione 8.3.1) della trasformazione prospettica dei punti del suolo. La matrice $\mathbf{P}_{z=0}$ è un esempio di trasformazione IPM (*Inverse Perspective Mapping*) per ottenere una vista dall'alto (*Bird eye view*) della scena inquadrata [MBLB91].

La relazione inversa di quella di equazione (8.17), che trasforma punti immagine in coordinate mondo, si può scrivere come:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \lambda \mathbf{R}^{-1}\mathbf{K}^{-1} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} + \mathbf{t}_0 = \lambda \mathbf{v}(u_i, v_i) + \mathbf{t}_0 \tag{8.19}$$

dove risulta ben evidente che a ogni punto dell'immagine corrisponde una retta (al variare di λ) nel mondo che passa per il pin-hole ($\mathbf{t}_0$) e diretta nella direzione

$$\mathbf{v}(u_i, v_i) = \mathbf{R}^{-1}\mathbf{K}^{-1} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \tag{8.20}$$

con $\mathbf{v} : \mathbb{R}^2 \to \mathbb{R}^3$ funzione che associa a ogni punto immagine il vettore che congiunge il *pin-hole* con il punto sensore corrispondente.

Usando direttamente la *Camera Matrix* $\mathbf{P} = [\mathbf{P}_{3\times3}|\mathbf{p}_4]$ è possibile ottenere un risultato equivalente all'equazione (8.19) nella forma

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \lambda \mathbf{P}_{3\times3}^{-1} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} - \mathbf{P}_{3\times3}^{-1}\mathbf{p}_4 \tag{8.21}$$

in modo da non usare esplicitamente le matrici dei parametri intrinseci ed estrinseci. Le due formulazioni sono ovviamente equivalenti.

8.2.1 Proprietà della matrice di rotazione

La matrice di rotazione verrà spesso indicata nel testo, in modo da compattarne la scrittura, come array del linguaggio `C`:

$$\mathbf{R} = \begin{bmatrix} r_0 & r_1 & r_2 \\ r_3 & r_4 & r_5 \\ r_6 & r_7 & r_8 \end{bmatrix}$$

La matrice di rotazione è una matrice molto sovradimensionata: i suoi 9 parametri linearmente indipendenti sono di fatto generati da 3 variabili in modo non lineare (si veda appendice).

Senza esplicitare gli angoli da cui la matrice è generata, risulta possibile fornire qualche vincolo aggiuntivo. La matrice di rotazione ha la proprietà di non modificare le distanze essendo ortonormale e $\det(\mathbf{R}) = 1$. Ogni riga e ogni colonna devono avere modulo unitario, ed ogni riga e ogni colonna sono ortonormali tra loro, in quanto basi ortonormali dello spazio. Conoscendo pertanto due vettori riga o colonna della matrice $\mathbf{r}_1$, $\mathbf{r}_2$ è possibile determinare la terza base come prodotto vettoriale dei precedenti due:

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \tag{8.22}$$

Allo stesso modo il prodotto scalare tra due vettori riga o due vettori colonna deve dare valore nullo, in quanto ortogonali tra di loro. Sotto tali vincoli, esistono due soluzioni esatte, di cui una è:

$$\mathbf{R} = \begin{bmatrix} r_0 & r_1 & (1 - r_0^2 - r_1^2)^{\frac{1}{2}} \\ r_3 & r_4 & s(1 - r_3^2 - r_4^2)^{\frac{1}{2}} \\ (1 - r_0^2 - r_3^2)^{\frac{1}{2}} & s(1 - r_1^2 - r_4^2)^{\frac{1}{2}} & (r_0^2 + r_1^2 + r_3^2 + r_4^2 - 1)^{\frac{1}{2}} \end{bmatrix} \tag{8.23}$$

dove $s = sgn(r_1 r_4 + r_2 r_5)$, mentre l'altra soluzione ha esattamente i segni invertiti. Conoscendo una sottomatrice 2×2 è possibile ricavare gli altri elementi della matrice stessa a meno di un segno, basandosi sempre sul fatto che ogni riga e colonna hanno norma unitaria.

8.2.2 Risultati Notevoli

Possiamo usare la matrice di rotazione e l'equazione della pin-hole (8.18) per mostrare qualche risultato notevole. Definiamo, dal sistema, la funzione f_{pm} di $\mathbb{R}^3$ in $\mathbb{R}^2$ chiamata *perspective mapping* definita come:

$$f_{pm}(x, y, z) = \left(k_u \frac{r_0 x + r_1 y + r_2 z}{r_6 x + r_7 y + r_8 z} + u_0, k_v \frac{r_3 x + r_4 y + r_5 z}{r_6 x + r_7 y + r_8 z} + v_0 \right) \tag{8.24}$$

funzione del modello della *pin-hole camera* scritta in maniera esplicita. Per semplicità si è supposto il *pin-hole* coincidere con l'origine del sistema di riferimento.

I punti di fuga e calibrazione

Per ogni immagine esistono 3 punti di fuga, strettamente legati alla scelta degli assi di riferimento.

Prendiamo per esempio il primo asse. Nel nostro sistema di riferimento la coordinata x è la distanza (per le altre 2 coordinate il discorso è similare). Portiamo tale coordinata a infinito mantenendo le altre costanti. Quello che si ottiene è il punto

$$\lim_{x\to\infty} f_{pm}(x,y,z) = \left(k_u\frac{r_0}{r_6} + u_0, k_v\frac{r_3}{r_6} + v_0\right) \tag{8.25}$$

Usando le matrici omogenee è possibile ottenere lo stesso risultato, con un formalismo più compatto.

Prendendo la trasformazione prospettica (8.17) e mandando via via $x \to \infty$, $y \to \infty$ e $z \to \infty$, i punti immagine (in coordinate omogenee) che si ottengono, rappresentati i punti di fuga nelle 3 direzioni, sono esattamente le colonne della matrice $[\mathbf{v}_x\mathbf{v}_y\mathbf{v}_z] = \mathbf{K}\cdot\mathbf{R}$, ovvero :

$$\begin{aligned}\mathbf{v}_x &= \mathbf{K}\mathbf{r}_1\\ \mathbf{v}_y &= \mathbf{K}\mathbf{r}_2\\ \mathbf{v}_z &= \mathbf{K}\mathbf{r}_3\end{aligned} \tag{8.26}$$

avendo indicato con la sintassi $\mathbf{r}_i$ le colonne della matrice $\mathbf{R}$. Questo è un primo esempio di calibrazione della camera che sfrutta una conoscenza dell'immagine, ovvero la posizione dei punti di fuga.

In particolare, ponendosi nel caso semplificato $u_0 = 0$, $v_0 = 0$ e $k_\gamma = 0$, i punti di fuga si trovano in

$$\begin{aligned}\mathbf{v}_x &= \left(k_u\frac{r_0}{r_6}, k_v\frac{r_3}{r_6}\right)\\ \mathbf{v}_y &= \left(k_u\frac{r_1}{r_7}, k_v\frac{r_4}{r_7}\right)\\ \mathbf{v}_z &= \left(k_u\frac{r_2}{r_8}, k_v\frac{r_5}{r_8}\right)\end{aligned} \tag{8.27}$$

È da notare che siccome le 3 colonne di $\mathbf{R}$ sono ortonormali basta conoscere 2 punti di fuga per ottenere sempre il terzo (vedi sezione precedente).

Horizon Line

Se mandiamo a infinito non una variabile ma più di una otteniamo più di un punto. Per $x \to \infty$ ma con $y = mx$ il vanishing point degenera in una linea di equazione

$$k_v(r_3r_7 - r_4r_6)u + k_u(r_6r_1 - r_7r_0)v + k_uk_v(r_4r_0 - r_3r_1) = 0 \tag{8.28}$$

linea dell'orizzonte.

Punti e Linee degeneri

Come un punto nell'immagine proiettata degenera in una linea, una linea di equazione $au + bv + c = 0$ diventa nell'immagine proiettata

$$ak_u(r_0x + r_1y + r_2z) + bk_v(r_3x + r_4y + r_5z) + c(r_6x + r_7y + r_8z) = 0$$

ovvero

$$(ak_u\ r_0 + bk_vr_3 + cr_6)x + (ak_ur_1 + bk_vr_4 + cr_7)y + (ak_ur_2 + bk_vr_5 + cr_8)z = 0 \tag{8.29}$$

che rappresenta il piano degenere (con normale come da equazione) in tre dimensioni che passa per l'origine (il *pin-hole*).

8.3 Trasformazioni omografiche notevoli

È possibile fare un breve elenco di quali trasformazioni in visione artificiale si possono rappresentare attraverso un'omografia.

Le trasformazioni descritte in questa sezione permettono, data la conoscenza dell'orientazione della camera e dei parametri intrinseci, di ricavare la matrice $\mathbf{H}$ che determina la trasformazione e, viceversa, ottenendo la matrice omografica attraverso l'associazione di punti tra le due immagini, di ricavare alcuni parametri che legano tra loro le viste. È infatti importante far notare come, per tutte le trasformazioni dove è coinvolta una omografia (cambio di punto di vista, proiezione prospettica, IPM e rettificazione), quando è richiesta la conoscenza dei parametri necessari per generare la trasformazione, si possono comunque ricavare implicitamente le matrici rappresentative conoscendo come (almeno) 4 punti dell'immagine vengono trasformati (si veda per i dettagli la sezione 8.5.1). I parametri ottenuti dalla scomposizione della matrice omografica sono parametri ottenuti da una minimizzazione algebrica. La soluzione a massima verosimiglianza richiede una minimizzazione non lineare ma usa come punto di partenza il risultato ottenuto da questa fase.

8.3.1 Perspective Mapping e Inverse Perspective Mapping

Usando l'omografia è possibile realizzare la trasformazione di *inverse perspective mapping* (o *bird eye view*) invertendo semplicemente la matrice della *perspective mapping*.

La matrice omografica $\mathbf{H} = \mathbf{P}_Z$ della proiezione prospettica di un piano, *perspective mapping*, relativa a un piano z costante, dove normalmente $z = 0$ essendo il suolo il piano più importante, si può ricavare in maniera molto semplice in quanto:

$$\mathbf{P}_Z = \mathbf{K} \cdot \mathbf{R}_Z \tag{8.30}$$

dove $\mathbf{R}_Z$ è la matrice di rototraslazione di un piano che può essere espressa come

$$\mathbf{R}_Z = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & z\mathbf{r}_3 + \tilde{\mathbf{t}} \end{bmatrix} = \begin{bmatrix} r_0 & r_1 & r_2 z + \tilde{t}_x \\ r_3 & r_4 & r_5 z + \tilde{t}_y \\ r_6 & r_7 & r_8 z + \tilde{t}_z \end{bmatrix} \tag{8.31}$$

avendo indicato il vettore $\tilde{\mathbf{t}}$ come traslazione espressa in coordinate camera, come in equazione (8.16).

Questa matrice è molto importante e verrà discussa diffusamente nella sezione 8.5 della calibrazione.

La trasformazione (8.30) essendo un'omografia è invertibile. Quando trasforma in maniera densa tutti i punti immagine in punti mondo si chiama *Inverse Perspective Mapping*, mentre quando trasforma tutti i punti mondo in punti immagine si indica come *Perspective Mapping*. In entrambi i casi viene proiettato correttamente solo il piano z.

È sempre interessante notare come anche il modello più semplice della camera *pin-hole* a 9 parametri (6 estrinseci e 3 intrinseci) non è ricavabile dagli 8 parametri vincoli che la matrice omografica fornisce. Tuttavia, conoscendo i parametri intrinseci, è possibile ottenere una stima della rotazione e della posizione della camera (sezione 8.5), in quanto l'equazione 8.30 diventa invertibile:

$$\mathbf{R}_Z = \mathbf{K}^{-1}\mathbf{H} \tag{8.32}$$

8.3.2 Vanishing Point e linea dell'orizzonte

Proprio per il fatto di essere limitato a trasformazioni di piani, è possibile calcolare in maniera molto agevole il limite della coordinata x e y attraverso la trasformazione (8.30) come

$$\begin{aligned} \lim_{x\to\infty} \mathbf{H}(x, y, 1)^\top &= \left(\tfrac{h_0}{h_6}, \tfrac{h_3}{h_6}\right) \\ \lim_{y\to\infty} \mathbf{H}(x, y, 1)^\top &= \left(\tfrac{h_1}{h_7}, \tfrac{h_4}{h_7}\right) \end{aligned} \tag{8.33}$$

Questi limiti sono i *vanishing point* (cfr. sezione 8.2.2) dell'immagine.

8.3.3 Cambio di punto di vista

L'equazione generica che mette in relazione i punti immagine tra due punti di vista generici si può scrivere come

$$\begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} \equiv \mathbf{K}_2 \left(\mathbf{R}\mathbf{K}_1^{-1} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} + \mathbf{t} \right) \tag{8.34}$$

dove $\mathbf{t} = \mathbf{t}_1 - \mathbf{t}_2$ è il vettore che congiunge i due *pin-hole* e $\mathbf{R}$ è l'orientazione relativa tra le due viste come indicato in sezione1.9. Trattazione più accurata viene lasciata nel capitolo 9 sulla stereoscopia.

In genere non è possibile trasformare una vista generata da una camera nella vista generata da un altra. Ciò risulta possibile solo se si vuole rimappare correttamente solamente i punti di un determinato piano o quando le camere condividono lo stesso *pin-hole*.

Il secondo caso lo vedremo nella prossima sezione. Nel primo caso è possibile rimappare i punti da una visuale a quelli di un'altra sfruttando la combinazione di una *Perspective Mapping* seguita da una *Inverse Perspective Mapping* e sfruttando l'ipotesi che la scena osservata sia composta solamente da un piano (per esempio il suolo). I punti immagine vengono proiettati in coordinate mondo su una camera *1* e riproiettati di nuovo in coordinate immagine su una seconda camera *2* con parametri intrinseci ed estrinseci differenti. Siccome si riproietta sempre un piano, anche la composizione di questa trasformazione è ancora una omografia:

$$\mathbf{H} = \mathbf{H}_2 \cdot \mathbf{H}_1^{-1} \tag{8.35}$$

le trasformazioni omografiche infatti si combinano con la semplice moltiplicazione tra matrici. Espandendo l'equazione (8.35) con (8.30) si ottiene:

$$\mathbf{H} = \mathbf{K}_2 \cdot \mathbf{R}_{Z2} \cdot \mathbf{R}_{Z1}^{-1} \cdot \mathbf{K}_1^{-1} \tag{8.36}$$

Dal punto di vista teorico il fatto di dover forzare un piano z costante incide solamente se il vettore traslazione cambia. Nel caso in cui il vettore di traslazione venga modificato tra le due viste ed esistano punti non appartenenti al piano indicato

avviene una rimappatura errata tra le due viste (la trasformazione omografica non è più rispettata). La trasformazione (8.35) può servire per individuare anche ostacoli verticali all'interno di tecniche come il *Ground Plane Stereo* e il *Motion Stereo*.

Questa matrice omografica si può generalizzare conoscendo gli elementi della trasformazione delle due viste $(\mathbf{R}, \mathbf{t})$ e l'equazione del piano $(\mathbf{n}, d)$, dove $\hat{\mathbf{n}}$ è la normale al piano e d è la distanza tra la prima camera con il piano stesso. In questo caso infatti, un punto $\mathbf{x}_1$ della prima vista appartenente al piano soddisfa l'equazione

$$\hat{\mathbf{n}}^\top \mathbf{x}_1 = d \tag{8.37}$$

e questo punto è in relazione con il medesimo punto, visto però dalla seconda camera in accordo con l'equazione

$$\mathbf{x}_2 = \mathbf{R}\mathbf{x}_1 + \mathbf{t} \tag{8.38}$$

Unendo queste due equazioni si ottiene il vincolo omografico

$$\mathbf{H} = \mathbf{K}_2 \left(\mathbf{R} + \frac{1}{d} \mathbf{t} \hat{\mathbf{n}}^\top \right) \mathbf{K}_1^{-1} \tag{8.39}$$

Una omografia può essere sempre decomposta in $\left[\mathbf{R}, \frac{1}{d}\mathbf{t}, \hat{\mathbf{n}}\right]$ (esistono 4 decomposizioni possibili e va scelta quella che soddisfa i punti in ingresso).

8.3.4 Rettificazione

Il caso di pura rotazione è un caso particolare: una camera che ruota intorno al suo centro ottico acquisisce immagini di una scena 3D come se la scena fosse rappresentata su un piano infinitamente lontano dal pin-hole.

Nel caso in cui $\mathbf{t} = 0$, ovvero le coordinate dei due pin-hole delle due viste siano coincidenti $\mathbf{t}_1 = \mathbf{t}_2$, la trasformazione (8.34) si riduce di dimensione e si ottiene un'equazione compatibile con un'omografia e di conseguenza valida per qualunque punto dell'immagine indipendentemente dalla presenza o meno di un piano dominante. Pertanto, nel caso in cui tra le due viste il pin-hole sia in comune (pertanto pura rotazione o modifica dei parametri intrinseci), è possibile realizzare una trasformazione perfetta per tutti i punti dell'immagine. Tale processo di proiezione di punti da una camera a un altra modificando parametri intrinseci e rotazione è chiamato rettificazione.

Per rettificare un immagine, ovvero per generare un'immagine 1 densa partendo dai punti dell'immagine 2, è necessario utilizzare la matrice omografica

$$\mathbf{H}_{1,2} = \mathbf{K}_2 \mathbf{R}_2 \mathbf{R}_1^{-1} \mathbf{K}_1^{-1} \tag{8.40}$$

che permette di ricavare tutti i punti dell'immagine 1 dai punti dell'immagine 2, ovvero per ogni pixel (u_1, v_1) dell'immagine che si vuole generare si applica la trasformazione omografica $\mathbf{H}$ e si ricava il punto (u_2, v_2) dell'immagine sorgente da cui copiare il valore del pixel.

Attraverso la trasformazione (8.40) è possibile trasformare un'immagine acquisita da una camera di parametri $(\mathbf{K}_2, \mathbf{R}_2)$ in un'immagine di una camera virtuale di parametri $(\mathbf{K}_1, \mathbf{R}_1)$.

Discorso che si applica a tutte le omografie, un metodo per ottenere la matrice $\mathbf{H}$ senza la conoscenza dei parametri intrinseci ed estrinseci ma solo attraverso corrispondenze tra le viste delle due camere è mostrato in sezione 8.5.1. L'omografia può poi essere fattorizzata per riottenere i parametri che l'hanno generata.

8.4 Inverse Perspective Mapping

Si sono viste nelle sezioni precedenti esempi di prospettiva inversa: la possibilità di ricavare il punto 3D dato un punto immagine 2D e la conoscenza di un vincolo nel mondo sulla cui superficie il punto giace. É sempre possibile infatti creare un sistema tra il raggio ottico (8.19) è una varietà in $\mathbb{R}^3$:

$$\begin{cases} \mathbf{x} = \lambda \mathbf{V}\mathbf{p} + \mathbf{t} \\ f(\mathbf{x}) = 0 \end{cases} \tag{8.41}$$

avendo chiamato $\mathbf{V} = \mathbf{R}^{-1}\mathbf{K}$. Questa stessa formulazione si usa in grafica computazionale per indicare le tecniche di *RayTracing*. Generalizziamo in questa sezione diverse casistiche.

Intersezione raggio ottico e piano Un generico piano in $\mathbb{R}^3$ scritto nella forma

$$\hat{\mathbf{n}} \cdot \mathbf{x} + q = 0 \tag{8.42}$$

è un vincolo per permettere l'intersezione tra il raggio ottico (8.19) e il piano (8.42). Il sistema (8.41) è lineare e può essere risolto per λ e da λ inserita nella prima equazione determinare il punto 3D. É possibile realizzare anche una applicazione lineare associata all'intersezione piano retta nella forma $\mathbf{x} \equiv \mathbf{A}_{4\times3}\mathbf{p}$ avendo definito

$$\mathbf{A}_{4\times3} = \begin{bmatrix} \left(\hat{\mathbf{n}}^\top \mathbf{t} - q\right) \mathbf{I} + \mathbf{t}\hat{\mathbf{n}}^\top \\ \hat{\mathbf{n}}^\top \end{bmatrix} \mathbf{V} \tag{8.43}$$

Intersezione raggio ottico e una sfera La varietà ha equazione

$$\|\mathbf{x} - \mathbf{x}_0\|^2 = r^2 \tag{8.44}$$

che unita al sistema (8.41) permette di ottenere

$$\lambda^2 \|\mathbf{V}\mathbf{p}\|^2 + 2\lambda (\mathbf{V}\mathbf{p}) \cdot (\mathbf{t} - \mathbf{x}_0) + \|\mathbf{t} - \mathbf{x}_0\|^2 = r^2 \tag{8.45}$$

La soluzione dell'equazione di secondo grado può pertanto avere 0 (nessuna intersezione), 1 (raggio ottico tangente alla sfera) o 2 radici (il raggio ottico interseca la sfera).

8.5 Calibrazione

La fase di calibrazione della camera permette di ricavare (alcuni o tutti) i parametri che permettono al modello *pin-hole* di poter essere utilizzato per proiettare punti da coordinate mondo a coordinate camera. In inglese la calibrazione della camera, ovvero il ricavare i parametri intrinseci e/o estrinseci, si chiama *Camera resectioning* in quando il concetto di *Camera Calibration* si può riferire anche al problema della calibrazione fotometrica del sistema.

Le tecniche di calibrazione si possono dividere in due categorie a seconda di quale modello della camera *pin-hole* si vuole ricavare

implicita dove vengono estratti gli elementi della matrice proiettiva $\mathbf{P}$ o la matrice omografica $\mathbf{H}$ in modo da poter proiettare punti da un sistema di coordinate a un altro disinteressandosi della struttura interna del sensore;

esplicita dove vengono estratti i parametri fisici del sistema coinvolti nella proiezione prospettica.

La calibrazione implicita solitamente è un processo più veloce e con un numero sufficiente di punti rappresenta abbastanza bene la realtà anche se, come vedremo fra poco, la versione lineare che minimizza una quantità algebrica, non è lo stimatore a massima verosimiglianza. La calibrazione implicita infine ignora alcune non-linearità sempre presenti nei sistemi fisici. La calibrazione esplicita, oltre a rappresentare correttamente il modello con il numero minimo di parametri, permette più libertà di azione nell'uso dei parametri ottenuti, per poter fare operazioni sulle immagini o per poter variare dinamicamente alcuni parametri del sistema, e permette di implementare lo stimatore alla massima verosimiglianza.

Per permettere l'applicazione delle tecniche di calibrazione mostrate in questa sezione è necessario avere dei vincoli tra gli spazi proiettivi coinvolti, per esempio punti in coordinate immagine e i corrispondenti punti in coordinate mondo. Attraverso questa relazioni è possibile ricavare i parametri che rappresentano il modello proiettivo usato.

Il confine che separa la calibrazione implicita da quella esplicita tende a volte a venire meno: da una modalità è possibile sotto opportune condizioni passare all'altra.

- Con la *Direct Linear Transformation*, sezione 8.5.1, è possibile calibrare implicitamente il sistema, conoscendo la posizione di punti in coordinate mondo e in coordinate immagine, ricavando la matrice di proiezione $\mathbf{P}$, o la matrice di proiezione di un singolo piano $\mathbf{H}$, non esplicitando nessun parametro della camera. Usando poi l'equazione (8.32), si può ricavare la matrice $[\mathbf{Rt}]$ dei parametri estrinseci, avendo tuttavia informazione sui parametri intrinseci.
- Si è già accennato in precedenza (vedi sezione 8.2.2) come è possibile ricavare la matrice di rotazione data la conoscenza della matrice dei parametri intrinseci e dalla posizione dei punti di fuga.
- Se si conosce la matrice di rotazione $\mathbf{R}$ è possibile ottenere in forma esplicita il valore degli angoli che l'hanno generata (potrebbero esistere più soluzioni in questo caso).
- Se si riesce ad ottenere la matrice dei parametri intrinseci $\mathbf{K}$ è immediato ricavare in maniera esplicita i parametri intrinseci della camera.
- Zhang, sezione 8.5.4, propone un modo per ricavare i parametri intrinseci della camera se si conoscono le posizioni relative di punti appartenenti allo stesso piano, osservato però da più punti di vista.

8.5.1 Calibrazione implicita

L'idea base della *Direct Linear Transformation* proposta da Abdel-Aziz e Karara [AAK71] permette di calcolare direttamente i coefficienti delle matrici (8.50), (8.53) o della matrice (8.18) disinteressandosi completamente dei parametri e dalla struttura del modello della trasformazione prospettica. In tale articolo viene anche presentato un approccio per risolvere problemi sovradimensionati attraverso la tecnica della Pseudoinversa.

Dato il sistema (8.18) è necessario ricavare i 12 parametri della matrice proiettiva $\mathbf{P}$ per avere una calibrazione del sistema *implicita* ovvero dove non si conoscono i parametri (da 9 a 11 a seconda del modello) interni che han generato gli elementi della matrice stessa. Tale rappresentazione della camera *pin-hole* è ovviamente ideale (senza non-linearità dal modello).

La funzione prospettica scritta in forma implicita è

$$\begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \mathbf{P} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} = \begin{bmatrix} p_0 & p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 & p_7 \\ p_8 & p_9 & p_{10} & p_{11} \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \tag{8.46}$$

dove gli elementi $p_0 \dots p_{11}$ sono scritti in ordine *row-major*. È possibile rigirare il sistema (8.46) in modo da avere 2 coppie di vincoli lineari per ogni punto di cui si conoscono le sue coordinate in immagine e in coordinate mondo:

$$\begin{bmatrix} x_i & y_i & z_i & 1 & 0 & 0 & 0 & 0 & -u_i x_i & -u_i y_i & -u_i z_i & -u_i \\ 0 & 0 & 0 & 0 & x_i & y_i & z_i & 1 & -v_i x_i & -v_i y_i & -v_i z_i & -v_i \end{bmatrix} \begin{pmatrix} p_0 \\ \vdots \\ p_{11} \end{pmatrix} = 0 \tag{8.47}$$

Tale tecnica si chiama DLT (*direct linear transformation*). Siccome ogni punto fornisce 2 vincoli, per ottenere questi 12 parametri sono necessari almeno 6 punti non linearmente dipendenti ovvero che non appartengano allo stesso piano e tantomeno alla medesima retta.

Essendo un sistema omogeneo, la sua soluzione sarà il sottospazio nullo di $\mathbb{R}^{12}$, kernel della matrice dei termini noti. Per questo motivo la matrice $\mathbf{P}$ è conosciuta a meno di un fattore moltiplicativo e ne consegue che ha solo 11 parametri liberi (sono anche di meno considerando che una telecamera moderna ha solo 3-4 parametri intrinseci e i 6 estrinseci). Avendo rigirato il sistema la propagazione del rumore sui punti non è più lineare e questa soluzione non soddisfa la massima verosimiglianza. La matrice $\mathbf{P}$ ottenuta attraverso questo procedimento, anche se nasconde la struttura interna del sensore, permette di proiettare un punto da coordinate mondo a coordinate immagine e da un punto in coordinate immagine ricavare la retta che sottende tale punto nel mondo.

Il risultato è generalmente instabile usando solo 6 punti e perciò la stima viene normalmente effettuata processando più punti del minimo e si sfruttano tecniche come la pseudoinversa per determinare una soluzione che minimizzi gli errori di misura.

Generalizzazione nel caso di coordinate omogenee

L'equazione 8.46 può essere generalizzata al caso di punto "immagine" in coordinate omogenee (u_i, v_i, w_i):

$$\begin{pmatrix} u_i \\ v_i \\ w_i \end{pmatrix} \equiv \mathbf{P} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \tag{8.48}$$

Il problema è uguale a quello visto in precedenza, la soluzione omogenea esiste e l'equazione risolutiva omogenea (8.47) si generalizza in

$$\begin{bmatrix} w_i x_i & w_i y_i & w_i z_i & w_i & 0 & 0 & 0 & 0 & -u_i x_i & -u_i y_i & -u_i z_i & -u_i \\ 0 & 0 & 0 & 0 & w_i x_i & w_i y_i & w_i z_i & w_i & -v_i x_i & -v_i y_i & -v_i z_i & -v_i \end{bmatrix} \begin{pmatrix} p_0 \\ \vdots \\ p_{11} \end{pmatrix} = 0 \tag{8.49}$$

per ogni i.

Questa formulazione è utile quando il modello proiettivo non segue il modello pin-hole ma è sempre possibile ricavare le coordinate "camera" dei raggi ottici sottesi al pixel e pertanto disponibili in formato omogeneo.

Calcolo DLT dell'omografia

Solitamente per ridurre il numero di elementi della matrice $\mathbf{P}$ si può aggiungere il vincolo che tutti i punti coinvolti nel processo di calibrazione appartengano a un piano particolare (per esempio al terreno). Ciò significa porre la condizione $z_i = 0 \ \forall i$, che implica l'eliminazione di una colonna (relativa all'asse z) della matrice che si riduce alla dimensione 3×3, diventa invertibile e si può definire omografica (vedi sezione 1.10).

Definiamo pertanto la matrice $\mathbf{H} = \mathbf{P}_Z$ (crf. con (8.30)) come

$$\lambda \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \mathbf{H} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \tag{8.50}$$

Come si è visto nella sezione 8.3 tale matrice è molto utile perché permette fra le altre cose di rimuovere la prospettiva dall'immagine, sintetizzando una visuale fronto-parallela del piano, con una trasformazione dal nome di *rettificazione ortogonale*, *bird eye view* o *inverse perspective mapping*. Tale trasformazione vale pertanto sia che si voglia rimuovere la prospettiva (*perspective mapping* o *inverse perspective mapping*), sia che si voglia riproiettare un piano tra due immagini (*ground plane*

stereo), sia generare un immagine con differenti parametri (rettificazione, immagini panoramiche) attraverso l'utilizzo di un piano virtuale.

Come nel caso precedente è possibile trasformare la relazione non lineare (8.50) in modo da ottenere dei vincoli lineari:

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -u_i x_i & -u_i y_i & -u_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -v_i x_i & -v_i y_i & -v_i \end{bmatrix} \begin{pmatrix} h_0 \\ \vdots \\ h_8 \end{pmatrix} = 0 \tag{8.51}$$

Siccome anche questa matrice è definita a meno di un fattore moltiplicativo, ha solo 8 gradi di libertà e si può pertanto porre un vincolo ulteriore.

Se si dispone di un risolutore di sistemi lineari abbastanza moderno il vincolo aggiuntivo $|\mathbf{H}| = 1$ è automaticamente soddisfatto durante il calcolo del kernel della matrice dei termini noti (fattorizzazione QR o decomposizione SVD).

Un altro metodo più semplice ed intuivo consiste nel porre come vincolo aggiuntivo $h_8 = 1$: in questo modo, invece che risolvere un sistema omogeneo, si può risolvere un problema lineare tradizionale. Il sistema (8.50) si può anche in questo caso riarrangiare in modo da ottenere dei vincoli lineari nella forma:

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i u_i & -y_i u_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_i v_i & -y_i v_i \end{bmatrix} \begin{pmatrix} h_0 \\ \vdots \\ h_7 \end{pmatrix} = \begin{pmatrix} u_i \\ v_i \end{pmatrix} \tag{8.52}$$

Questo è un sistema (non omogeneo) di due equazioni in 8 incognite $h_0 \dots h_7$ e ogni punto, di cui si conoscono sia la posizione nel mondo su un piano sia la posizione nell'immagine, fornisce 2 vincoli.

L'avere imposto $h_8 = 1$ tuttavia implica che il punto $(0, 0)$ non può essere una singolarità dell'immagine (es. linea dell'orizzonte), e in generale non è una scelta ottima dal punto di vista della precisione della soluzione come già discusso in precedenza.

È importante notare che la soluzione dipende fortemente dalla normalizzazione scelta. La scelta $|H| = c$ può essere chiamata *standard least-squares*.

In entrambi i casi sono richiesti almeno 4 punti per ottenere un omografia $\mathbf{H}$ e ogni punto in più permette di ottenere una soluzione di errore inferiore. Questi sistemi, quando sovradimensionati, possono essere risolti usando il metodo della pseudoinversa 1.1.

La matrice $\mathbf{H}$ è definita da 4 parametri intrinseci e dai 6 parametri estrinseci. La separazione dei parametri intrinseci dai parametri estrinseci suggerisce di estrarre tali parametri in maniera indipendente in modo da irrobustire la calibrazione. Dopotutto i parametri intrinseci possono essere ricavati con un certo grado di precisione offline e valgono per tutti i possibili posizionamenti della camera (si veda poi 8.5.4).

Definiamo la matrice $\mathbf{R}_Z$ (cfr. con (8.31)) come

$$\lambda \begin{pmatrix} \tilde{u_i} \\ \tilde{v_i} \\ 1 \end{pmatrix} = \begin{bmatrix} r_0 & r_1 & p_x \\ r_3 & r_4 & p_y \\ r_6 & r_7 & p_z \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \mathbf{R}_Z \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \tag{8.53}$$

avendo indicato con $(\tilde{u_i}, \tilde{v_i})$ le cosidette coordinate immagine normalizzate (coordinante omogenee al punto $(\tilde{x}_i, \tilde{y}_i, \tilde{z}_i)^\top$ in coordinate camera).

La matrice $\mathbf{H}$ è definita a meno di un fattore di scala, mentre $\mathbf{R}_Z$ permette di definire la scala in quanto presenta ancora due colonne ortonormali. La conoscenza delle due colonne della matrice di rotazione permette di ricavare la terza e perciò tale calibrazione diventa valida per punti anche fuori dal piano $z = 0$.

Come è stato fatto in precedenza, un sistema non-lineare in 3 equazioni omogenee, quando opportunamente riarrangiato, fornisce due vincoli lineari:

$$\begin{aligned} &\mathbf{A}\mathbf{x} = 0 \\ &\mathbf{A} = \begin{bmatrix} x_i & y_i & 0 & 0 & -\tilde{u_i} x_i & -\tilde{u_i} y_i & 1 & 0 & -\tilde{u_i} \\ 0 & 0 & x_i & y_i & -\tilde{v_i} x_i & -\tilde{v_i} y_i & 0 & 1 & -\tilde{v_i} \end{bmatrix} \\ &\mathbf{x} = (r_0, r_1, r_3, r_4, r_6, r_7, p_x, p_y, p_z)^\top \end{aligned} \tag{8.54}$$

(Abdel-Aziz e Karara [AAK71]). È dunque possibile costruire un sistema di $2 \times N$ equazioni per tutti gli N punti di controllo, per cercar di ottenere le 9 incognite. La matrice è definita a meno di un fattore moltiplicativo, ma in questo caso la struttura interna della matrice $\mathbf{R}_Z$ può essere di aiuto per ricavare i parametri estrinseci (cfr. sezione 8.5.3). Di fatto le due colonne della matrice devono essere ortonormali:

$$\begin{aligned} r_0^2 + r_3^2 + r_6^2 &= 1 \\ r_1^2 + r_4^2 + r_7^2 &= 1 \\ r_0 r_1 + r_3 r_4 + r_6 r_7 &= 0 \end{aligned} \tag{8.55}$$

Questi vincoli non lineari aggiuntivi sono frutto del fatto che tale matrice è definita esplicitamente da solo 6 parametri (3 rotazioni e la traslazione).

Rappresentazione geometrica

Le equazioni (8.47) e (8.51) si possono ricavare anche da considerazioni puramente geometriche in quanto i vettori immagine e camera devono essere paralleli (il fattore $lamba_i$ è puramente moltiplicativo e sul vettore al massimo incide una trasformazione affine):

$$\mathbf{p} \times \mathbf{Px} = \mathbf{0} \qquad \mathbf{m}' \times \mathbf{Hm} = \mathbf{0} \tag{8.56}$$

Questa formulazione compatta è quella che normalmente viene indicata come *DLT* [HZ04] e si applica a tutte quelle trasformazioni lineari conosciute a meno di un fattore moltiplicativo per trasformare tale problema in un problema omogeneo.

8.5.2 Calcolo MLE dell'omografia

Per quanto riguarda il punto di vista computazionale, l'equazione (8.51) è mal condizionata in quanto ogni colonna rappresenta una quantità con un ordine di grandezza differente. Per ottenere dal punto di vista lineare una soluzione corretta è richiesta una fase precedente di normalizzazione. Hartley e Zisserman [HZ04] ricordano che la normalizzazione nella DLT è un passo essenziale e non si può ritenere puramente opzionale.

Il calcolo dell'omografia in equazione (8.52) tuttavia ha il difetto di non tenere conto dell'errore di misura sui punti. Di fatto la decomposizione SVD minimizza qualcosa che per puro caso assomiglia l'errore sul termine noto (cosa che invece proprio non viene fatta nel caso (8.51)) e in ogni caso non si riesce a valutare l'errore sulla matrice dei parametri. In questo caso specifico, dove si minimizza ai minimi quadrati un errore puramente matematico senza corrispondente geometrico, si parla di *algebraic least squares* (*ALS*).

Siccome la DLT minimizza un errore algebrico e non geometrico, anche se dal punto di vista computazionale la DLT normalizzata è migliore, potrebbe restituire risultati peggiori dal punto di fitting geometrico dei dati. La versione del sistema (8.51) normalizzato ai minimi quadrati viene indicato con *normalized algebraic least squares* (*NALS*).

Per superare il limite del calcolo sull'errore algebrico, è necessario tornare al problema originale e non cercare di trasformarlo in un problema lineare ma risolverlo, per esempio in maniera iterativa, attraverso un minimizzatore non lineare.

Se il rumore è presente solo su una delle due immagini, una funzione costo appropriata, con significato geometrico, è la distanza euclidea tra i punti misurati e i punti trasformati. Questo è chiamato normalmente errore di trasferimento (*transfer error*) e minimizza una funzione costo non lineare della forma

$$\arg\min_{\mathbf{H}} \sum \|\mathbf{m}'_i - \mathbf{Hm}_i\|^2 \tag{8.57}$$

dove $\mathbf{m}'_i$ è il punto immagine affetto da rumore gaussiano bianco, mentre il punto $\mathbf{m}_i$ è un punto perfettamente conosciuto. In tal caso la funzione che minimizza l'errore geometrico è anche quella che rappresenta la miglior stima del risultato dal punto di vista bayesiano (*Maximum Likelihood Estimator* o *MLE*).

Tuttavia quando entrambi i dati sono affetti da rumore la funzione costo (8.57) non è ottimale. Il modo più semplice per estendere la soluzione precedente consiste nel cercare di minimizzare l'errore di trasferimento diretto e l'errore di trasferimento inverso (*symmetric transfer error*):

$$\arg\min_{\mathbf{H}} \sum \|\mathbf{m}'_i - \mathbf{Hm}_i\|^2 + \|\mathbf{m}_i - \mathbf{H}^{-1}\mathbf{m}'_i\|^2 \tag{8.58}$$

In questo modo si tengono conto di entrambi i contributi nella soluzione del problema.

Questa tuttavia, non è ancora la soluzione ottima, almeno dal punto di vista statistico. Uno stimatore a massima verosomiglianza deve infatti considerare correttamente il rumore su entrambi i dati quando presente (quello che Hartley e Zisserman chiamano *Gold Standard*). La soluzione alternativa, di fatto quella più corretta, consiste nel minimizzare l'errore di Riproiezione.

Questa soluzione incrementa di molto la dimensione del problema in quanto si pone come obiettivo (o comunque richiede tra le incognite) anche quello di individuare i punti ottimi non affetti da rumore $\hat{\mathbf{m}}_i$ e $\hat{\mathbf{m}}'_i$:

$$\arg\min_{\mathbf{H}} \sum \|\mathbf{m}'_i - \hat{\mathbf{m}}'_i\|^2 + \|\mathbf{m}_i - \hat{\mathbf{m}}_i\|^2 \tag{8.59}$$

sotto il vincolo $\hat{\mathbf{m}'}_i = \mathbf{H}\hat{\mathbf{m}}_i$.

Nel caso ancora più generale con rumore di covarianza misurato per ogni singolo punto la metrica corretta è la distanza di mahalanobis (vedi sezione 2.4):

$$\|\mathbf{m} - \hat{\mathbf{m}}\|^2_\Gamma = (\mathbf{m} - \hat{\mathbf{m}})^\top \Gamma^{-1} (\mathbf{m} - \hat{\mathbf{m}}) \tag{8.60}$$

Nel caso in cui il rumore per punto sia costante la precedente espressione si riduce alla più intuitiva distanza euclidea.

Essendo una minimizzazione non lineare è richiesta tuttavia una soluzione iniziale da cui partire per trovare il minimo che soddisfa l'equazione costo: la soluzione lineare è ancora utile ed è usata come spunto iniziale per individuare un minimo sotto una metrica differente.

Lo stimatore MLE richiede l'uso di una variabile sussidiaria $\hat{\mathbf{m}}_i$ in più per ogni punto e tecniche iterative per risolvere il problema. È possibile usare come approssimazione della distanza geometrica, l'errore di Sampson, sezione 3.3.8. Il vincolo omografico (1.75) che lega i punti delle due immagini può essere scritto sotto forma di varietà $\mathcal{V}_H$ bidimensionale

$$\begin{aligned} h_0u_1 + h_1v_1 + h_2 - h_6u_1u_2 - h_7v_1u_2 - h_8u_2 = 0 \\ h_3u_1 + h_4v_1 + h_5 - h_6u_1v_2 - h_7v_1v_2 - h_8v_2 = 0 \end{aligned} \tag{8.61}$$

da cui lo Jacobiano

$$\mathbf{J}_{\mathcal{V}} = \begin{bmatrix} h_0 - h_6u_2 & h_1 - h_7u_2 & -h_6u_1 - h_7v_1 - h_8 & 0 \\ h_3 - h_6v_2 & h_4 - h_7v_2 & 0 & -h_6u_1 - h_7v_1 - h_8 \end{bmatrix} \tag{8.62}$$

da usare nel calcolo della distanza di Sampson [CPS05].

Propagazione dell'errore nel calcolo dell'omografia

Nel caso di errore su una singola immagine per calcolare come l'errore si propaga sulla matrice $\mathbf{H}$ è necessario calcolare lo Jacobiano della funzione costo (8.57). Esplicitando la trasformazione omografica si ottiene [HZ04]

$$\mathbf{J}_i = \frac{\partial r}{\partial \mathbf{h}} = \frac{1}{\hat{w}'} \begin{bmatrix} \mathbf{m}_i^\top & 0 & -\hat{u}_i'\mathbf{m}_i^\top/\hat{w}' \\ 0 & \mathbf{m}_i^\top & -\hat{v}_i'\mathbf{m}_i^\top/\hat{w}' \end{bmatrix} \tag{8.63}$$

con $\mathbf{m}_i = (u_i, v_i, 1)^\top$ e $\hat{\mathbf{m}}_i' = (\hat{u}_i', \hat{v}_i', \hat{w}_i')^\top = \mathbf{H}\mathbf{m}_i$. Attraverso la teoria mostrata in sezione 3.5 è possibile calcolare la matrice di covarianza dei parametri dell'omografia data la covarianza sui punti $\mathbf{m}_i'$. Siccome la matrice di covarianza totale $\mathbf{\Sigma}$ del rumore sui singoli punti sarà molto sparsa, in quando punti diversi si suppone che abbiano rumore indipendente, la covarianza $\mathbf{\Sigma}_h$ sui parametri ottenuti vale [HZ04]

$$\mathbf{\Sigma}_h = \left(\sum \mathbf{J}_i^\top \mathbf{\Sigma}_i^{-1} \mathbf{J}_i \right)^+ \tag{8.64}$$

con $\mathbf{\Sigma}_i$ matrice di covarianza del rumore sul singolo punto.

8.5.3 Calibrazione secondo Tsai

La calibrazione della camera per diverse applicazioni richiede la conoscenza completa dei parametri intrinseci ed estrinseci. Uno dei metodi più diffusi è sicuramente quello di Tsai [Tsa87] del 1985. Il pregio di Tsai è stato quello di dare ordine allo stato dell'arte discusso in precedenza e fornire una nomenclatura unica ed accettata per i parametri della camera come qui presentati.

Il modello della camera di Tsai è basato sulla proiezione prospettica della Pin-Hole Camera, ed è formato (nella sua forma classica) da 11 parametri:

f Lunghezza focale della camera

k Coefficiente di distorsione radiale di primo ordine

Cx,Cy Coordinate del centro ottico della lente

Sx Un fattore di scala orizzontale

Rx, Ry, Rz Angoli di rotazione per la trasformazione tra coordinate mondo e coordinate camera

Tx, Ty, Tz Vettore di traslazione per la trasformazione tra coordinate mondo e coordinate camera

Tsai esegue sia una analisi di tutte le tecniche sviluppate finora per la calibrazione, e infine propone un sistema a moduli, dove ogni modulo permette di ricavare una serie di questi parametri.

Principalmente fa notare che se la camera è distorta ma si pone il *principal point* coincidente con il *centro di distorsione* valgono i rapporti:

$$\frac{u_d}{v_d} = \frac{u_u}{v_u} \tag{8.65}$$

e di conseguenza è possibile creare vincoli sotto questa condizione usando le coordinate distorte piuttosto che quelle non distorte. Tale metodo pertanto è chiamato anche *radial alignment constraint* (RAC).

Inizialmente usando i parametri della camera forniti dal produttore calcola il vettore traslazione e rotazione da una griglia con punti coplanari $z_i = 0$ di coordinate note, sfruttando il vincolo

$$(r_0x_i + r_1y_i + \tilde{t}_x)u_i' = (r_3x_i + r_4y_i + \tilde{t}_y)v_i' \tag{8.66}$$

con (u'_i, v'_i) coordinate camera normalizzate usando i parametri della camera e della lente forniti dal produttore. Da questo vincolo si può creare un sistema lineare sovradimensionato di tipo

$$\begin{bmatrix} x_i u'_i & y_i u'_i & u'_i & -v'_i x_i & -v'_i y_i \end{bmatrix} \begin{bmatrix} \frac{r_0}{\tilde{t}_y} \\ \frac{r_1}{\tilde{t}_y} \\ \frac{t_x}{\tilde{t}_y} \\ \frac{r_3}{\tilde{t}_y} \\ \frac{r_4}{\tilde{t}_y} \end{bmatrix} = v'_i \tag{8.67}$$

avendo posto $\tilde{t}_y \neq 0$ (ovvero la griglia non deve passare per l'asse ottico). I rimanenti parametri della matrice $\mathbf{R}$ vengono ottenuti usando l'equazione (8.23).

Successivamente procede nel ricavare i parametri intrinseci corretti usando questi valori per la matrice di rotazione e traslazione.

8.5.4 Calibrazione con il metodo Sturm-Maybank-Zhang

Zhang [Zha99] e contemporaneamente Sturm e Maybank [SM99] individuano un metodo per ottenere una equazione lineare per ricavare i parametri della camera, eseguendo anche un aggiornamento delle tecniche di calibrazione (sempre valide, ma ormai relative agli anni 80) fatte principalmente da Tsai [Tsa87] e altri [WM94].

Questa tecnica sfrutta il calcolo di diverse matrici omografiche $\mathbf{H}$ ottenute dall'osservazione di un piano (per esempio una griglia di calibrazione con *marker* equispaziati) e da queste cerca di ricavare i parametri intrinseci della camera in maniera esplicita. Come già discusso in precedenza la matrice $\mathbf{H}$, trasformazione omografica di un piano, possiede 8 gradi di libertà ma non è possibile direttamente ricavare i 10 parametri espliciti che l'hanno generata. Metodi per ottenere la matrice omografica date le corrispondenze tra punti immagine e punti del piano sono discussi in sezione 8.5.1.

La matrice $\mathbf{H}$ e in particolare l'equazione (8.30) può essere esplicitata come

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = \lambda \mathbf{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \tag{8.68}$$

dove λ è indicato per sottolineare la presenza di un fattore moltiplicativo, incognito, nel calcolo della matrice omografica. Concentriamo l'attenzione sulla parte di matrice di rotazione formata dai vettori colonna $\mathbf{r}_1$ e $\mathbf{r}_2$ ortonormali tra loro.

Nonostante la presenza del fattore λ è infatti possibile esprimere delle relazioni basate sull'ortogonalità tra i vettori $\mathbf{r}_1$ e $\mathbf{r}_2$ in modo da forzare i seguenti due vincoli:

$$\begin{aligned} \mathbf{h}_1^\top \mathbf{W} \mathbf{h}_2 &= 0 \\ \mathbf{h}_1^\top \mathbf{W} \mathbf{h}_1 &= \mathbf{h}_2^\top \mathbf{W} \mathbf{h}_2 \end{aligned} \tag{8.69}$$

avendo definito $\mathbf{W}$, tralasciando lo *skew* per semplicità, come

$$\mathbf{W} = (\mathbf{K}^{-1})^\top \mathbf{K}^{-1} = \begin{bmatrix} \frac{1}{k_u^2} & 0 & -\frac{u_0}{k_u^2} \\ 0 & \frac{1}{k_v^2} & -\frac{v_0}{k_v^2} \\ -\frac{u_0}{k_u^2} & -\frac{v_0}{k_v^2} & \frac{u_0^2}{k_u^2} + \frac{v_0^2}{k_v^2} + 1 \end{bmatrix} \tag{8.70}$$

matrice simmetrica. Tale equazione è l'equazione di una conica ed è in effetti l'equazione della "conica assoluta" [LF97].

Le 4 (o 5 incognite non trascurando lo *skew*) della matrice $\mathbf{W}$ sotto i 2 vincoli (8.69) possono essere risolte usando almeno 2 (o 3) piani diversi, ovvero matrici $\mathbf{H}$ le cui colonne non siano linearmente dipendenti tra loro.

Ottenuta la matrice $\mathbf{W}$, con la decomposizione di Cholesky si può determinare infine la matrice originale. Alternativamente Zhang fornisce le equazioni per ottenere i parametri intrinseci della camera direttamente dalla matrice $\mathbf{W}$. Si può infatti trasformare $\mathbf{h}_i^\top \mathbf{W} \mathbf{h}_j = \mathbf{v}_{ij}^\top \mathbf{w}$, con opportuni valori del vettore $\mathbf{v}_{ij}$ e con $\mathbf{w}$, vettore da determinare, con i valori non nulli della matrice triangolare superiore di $\mathbf{W}$. In questo modo il sistema di equazioni (8.69) si trasforma nella soluzione di un sistema lineare omogeneo in $\mathbf{w}$.

Determinati i parametri intrinseci e la matrice $\mathbf{K}$, per ogni matrice omografica $\mathbf{H}$ usata nella fase di ottimizzazione è possibile stimare la rotazione e la traslazione:

$$\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} = \lambda \mathbf{K}^{-1} \mathbf{H} \tag{8.71}$$

Le colonne $\mathbf{r}_1$ e $\mathbf{r}_2$ sono normalmente sufficienti per ricavare gli angoli di rotazione. Da ogni griglia è possibile ricavare tutti i parametri estrinseci e misurare in questo modo l'errore di riproiezione.

Il sistema nel suo complesso è comunque mal condizionato e difficilmente si giunge a una soluzione stabile dopo ripetute prove. I valori ottenuti attraverso questa tecnica lineare servono però come punto di inizio in una fase di *Maximum Likelihood Estimation* per minimizzare gli errori di riproiezione (sezione 8.5.6).

Una sola nota: Zhang nel suo articolo fa coincidere il *Principal Point* con il centro di distorsione, cosa generalmente non esatta.

8.5.5 Il problemi P3P e PnP

La soluzione DLT è numericamente instabile mentre le tecniche di minimizzazione, non-lineari, ai minimi quadrati sono tecniche iterative che richiedono come punto di partenza della minimizzazione una soluzione vicina all'ottimo (per esempio la soluzione proposta da DLT). Attualmente non ci sono soluzioni definitive ed eleganti a questi problemi ma esistono diversi approcci possibili, tutti comunque migliori dell'approccio DLT. Queste tecniche cadono sotto il nome di PnP (*perspective-n-point*) e P3P (*perspective-3-point*) nel caso ottimo teorico in quanto bastano solo 3 punti per determinare la posa del sensore video.

Tra gli approcci interessanti da menzionare ci sono quelli descritti in [LFNP09] e in [HR11]. Sono approcci abbastanza lunghi e complessi ma forniscono stime della posa in maniera molto vicino a tecniche iterative che soffrono comunque di problemi di minimo locale.

8.5.6 Maximum Likelihood Estimation

Quando si esegue la fase di calibrazione per mettere in relazione punti immagine con punti mondo è facile ipotizzare che il punto in coordinate mondo abbia una precisione elevata mentre il valore del punto in coordinate immagine è conosciuto a meno di rumore gaussiano a media nulla.

Le tecniche viste in precedenza (in particolare la DLT) sono mere approssimazioni della soluzione ottima e devono essere usate come punto di partenza per una minimizzazione non lineare. Per ottenere la soluzione ottima è necessario minimizzare la somma degli errori al quadrato tra la posizione misurata affetta da rumore e la posizione predetta dal modello. Lo stimatore a massima verosimiglianza minimizza una funzione obiettivo del tipo

$$\min_{\beta} \|\mathbf{p}_i - f(\mathbf{x}_i, \beta)\|^2 \tag{8.72}$$

dove $\mathbf{x}_i$ è un punto in coordinate mondo e $\mathbf{p}_i$ è il corrispondente punto in coordinate immagine, affetto da rumore di osservazione dovuta all'algoritmo di individuazione del punto e alla quantizzazione spaziale in pixel che qualunque sensore applica ai raggi ottici. β sono i parametri della proiezione prospettica da stimare, preferibilmente quelli espliciti considerando anche la distorsione dell'ottica.

8.6 Modello Fish-Eye Camera

Volendo generalizzare il concetto di camera pin-hole si possono introdurre piu classi di modelli camera. Ricordo che le camere reali non sono mai camere ideali che si adattano perfettamente ad un determinato modelli di ottica. Esistono pertanto vari modelli base che cercano più o meno di approssimare l'equazione della lente a cui sopra vanno comunque aggiunti vari termini distorsivi.

In particolare le ottiche *fish-eye* sono lenti dove la distorsione barilotto è dominante e questo permette di ottenere, a parità di lunghezza focale, angoli di vista molto elevati, fino a 180 gradi o superiori.

Sia pertanto ϑ l'angolo incidente del raggio ottico del punto camera (x, y, z) rispetto all'asse ottico $(0, 0, 1)$. Questo angolo vale $\vartheta = \arctan r_i$ che si ottiene da

$$r_i = \sqrt{x_1^2 + y_1^2} = \frac{\sqrt{x^2 + y^2}}{z} \tag{8.73}$$

dove infatti $x_1 = x/z$ e $y_1 = y/z$. Attenzione che vale l'uguaglianza

$$\vartheta = \arctan \frac{\sqrt{x^2 + y^2}}{z} = \arccos \frac{z}{\sqrt{x^2 + y^2 + z^2}} \tag{8.74}$$

L'idea è quella di considerare tutti i modelli camera come manipolazione dell'angolo incidente ϑ in un angolo ϑ' o spesso viene definita una trasformazione $r(\vartheta) = f\vartheta'$ che trasforma l'angolo del raggio incidente in un raggio in pixel.

Le varie lenti *fish-eye* seguono equazioni leggermente differenti tra loro, tra le quali è possibile segnalare tra i modelli ideali (pertanto senza distorsione)

- lineare: $r = f\vartheta$
- ortografica: $r = f \sin \vartheta$
- angolo solido costante: $r = 2f \sin(\vartheta/2)$
- stereografica: $r = 2f \tan(\vartheta/2)$.

Il modello *pin-hole* si può vedere come caso particolare in quanto il rapporto tra l'angolo incidente di luce ϑ e la coordinate del pixel segue la regola $r = f \tan \vartheta$.

Avendo assorbito in r anche la focale (perciò è un raggio in pixel), le equazioni sopra permettono di trasformare l'angolo di luce incidente sull'ottica in un raggio proiettato sul sensore:

$$u = \frac{x}{\sqrt{x^2 + y^2}} r + u_0 \qquad v = \frac{y}{\sqrt{x^2 + y^2}} r + v_0 \tag{8.75}$$

siccome l'angolo di fase rimane costante tra raggio incidente e raggio proiettato in assenza di distorsione tangenziale.

La classe di modelli ideali sopra non tiene conto di eventuali non linearità dell'ottica. Il modello di Kannala-Brandt [KHB09] generalizza queste equazioni, parametrizzando una generica lente *fish-eye* nel termine $r(\vartheta)$ usando la classica espansione in serie di Taylor che permette di inglobare sia i diversi modelli di lente presentati che eventuali distorsioni introdotte dell'ottica.

Ricordo che anche nelle camere fish-eye i raggi ottici passano tutti per lo stesso punto (pertanto i raggi passano tutti sempre per un pin-hole) ma tutte le equazioni che si possono scrivere (e se ne vedranno diverse nel prossimo capitolo) in forma lineare sfruttando le coordinate omogenee con un modello fish-eye di quelli qua elencati, non si possono più usare e bisogna passare ad equazioni non lineari anche in coordinate omogenee.

Capitolo 9

Visione Multicamera

Questo capitolo tratta in generale gli algoritmi che coinvolgono l'analisi di immagini provenienti da più di una camera con particolare attenzione al caso di visione steroscopica.

La visione stereoscopica (stereopsi) è il processo attraverso il quale è possibile stimare distanze e posizioni di oggetti osservati da due sensori visivi e attraverso queste informazioni poter ricostruire la scena osservata. Tale discorso è facilmente estendibile al caso in cui la scena sia osservata non da due ma da più camere (*multiple view geometry*).

Queste viste possono essere temporalmente coincidenti (per esempio nel caso della coppia di camere che formano una stereocamera) o possono osservare la scena in punti dello spazio e del tempo differenti come accade per esempio quando si processano immagini della stessa camera che si sposta nello spazio (*motion stereo*, *structure from motion*).

L'analisi stereoscopica può essere implementata principalmente attraverso due tecniche:

- *Feature Matching* dove punti notevoli tra due immagini vengono confrontati senza vincoli, se non quelli che verranno in seguito mostrati, permettendo di individuare coppie omologhe di punti ma una ricostruzione *sparsa* della scena;
- *Rectified Stereo* dove i punti tra le immagini provenienti da camere allineate (in hardware o in software attraverso rettificazione) sono sulla stessa riga su entrambe le camere e questo permette di semplificare il problema della ricerca dei punti e ottenere ricostruzioni *dense* della scena osservata.

Condizione necessaria per attuare una ricostruzione tridimensionale completa della scena osservata, attraverso l'analisi di più immagini acquisite da punti di vista differenti, è la conoscenza dei parametri intrinseci delle camere coinvolte e la posa relativa tra di esse.

Se non si conosce la posa relativa questa può essere stimata attraverso l'analisi stessa delle immagini ma, come si vedrà in seguito, la distanza tra le camere sarà ricavata a meno di un fattore moltiplicativo e di conseguenza anche la ricostruzione tridimensionale sarà conosciuta a meno di tale fattore.

Se non si conoscono neanche i parametri intrinseci è sempre possibile mettere in relazione punti omologhi tra le due immagini e grazie a questo processo accelerare il confronto tra *KeyPoint* ma non sarà possibile dire nulla sulla ricostruzione tridimensionale della scena osservata (la ricostruzione è conosciuta a meno di una trasformazione affine).

9.1 Trasformazione di coordinate camera

Quando si parla di coordinate stereoscopiche è necessario fare una breve introduzione ai cambi di coordinate tra sistemi camera. Questa sezione di fatto è il proseguo della discussione generale relativa a sensori vista in sezione 1.9.

Quando i sensori coinvolti sono sensori video, le matrici di rotazione coinvolte nelle equazioni della camera sono matrici che convertono da coordinate mondo a coordinate camera e non, come indicato in precedenza, a coordinate sensore.

Nel caso della camera *pin-hole*, il generico punto mondo $\mathbf{x}$ viene rototraslato nel punto ${}^{i}\mathbf{m}$, espresso in coordinate camera del sensore i-esimo, attraverso la relazione:

$$ {}^{i}\mathbf{m} = {}^{i}\mathbf{R}\left(\mathbf{x} - \mathbf{t}_i\right) = {}^{i}\mathbf{R}\mathbf{x} - {}^{i}\mathbf{R}\mathbf{t}_i \tag{9.1} $$

che coinvolge la matrice di rotazione e permutazione ${}^{i}\mathbf{R}$, espressa nella forma della camera *pin-hole* ovvero matrice che converte da coordinate mondo a coordinate camera. L'inversa di questa trasformazione esiste sempre e vale

$$ \mathbf{x} = {}^{i}\mathbf{R}^{-1}\,{}^{i}\mathbf{m} + \mathbf{t}_i \tag{9.2} $$

Pertanto, dato un punto in coordinate camera ${}^{1}\mathbf{m}$ osservato nel sistema di riferimento del primo sensore video, tale punto viene rappresentato nel sistema di riferimento della seconda camera in

$$ \begin{aligned} {}^{2}\mathbf{m} &= {}^{2}\mathbf{R}\left({}^{1}\mathbf{R}\right)^{-1}{}^{1}\mathbf{m} - {}^{2}\mathbf{R}(\mathbf{t}_2 - \mathbf{t}_1) \\ &= {}^{2}\mathbf{R}_1\,{}^{1}\mathbf{m} + {}^{2}\mathbf{t} \end{aligned} \tag{9.3} $$

avendo definito

$$\begin{aligned} {}^2\mathbf{R}_1 &= {}^2\mathbf{R}\left({}^1\mathbf{R}\right)^{-1} \\ {}^2\mathbf{t} &= -{}^2\mathbf{R}(\mathbf{t}_2 - \mathbf{t}_1) \end{aligned} \tag{9.4}$$

con le matrici ${}^1\mathbf{R}$ e ${}^2\mathbf{R}$ ancora definite come nel modello *pin-hole*. In questo caso la matrice $\mathbf{R}$ è una matrice che converte un punto dalle coordinate camera del primo sistema di riferimento nelle coordinate camera del secondo sistema. Le discussioni in questo capitolo si rifaranno a sistemi camera e pertanto i parametri delle pose relative saranno quelli indicati in equazione (9.4).

Le relazioni che legano punti tra due sensori dipendono solamente dalla loro posa relativa e di conseguenza le coordinate ${}^1\mathbf{m}$ e ${}^2\mathbf{m}$ del medesimo punto mondo $\mathbf{x}$, osservato nei due sensori video, dovranno soddisfare sempre l'equazione (9.3).

9.1.1 Posa relativa in coordinate camera e in coordinate sensore

Come già ricordato più volte, per trasformare un sistema di riferimento camera in uno sensore è sufficiente applicare la trasformazione

$${}^c\mathbf{R}_w = {}^c\mathbf{\Pi}_b \, {}^w\mathbf{R}_b^{-1} \Leftrightarrow {}^w\mathbf{R}_b = {}^c\mathbf{R}_w^{-1} \, {}^c\mathbf{\Pi}_b \tag{9.5}$$

dove ${}^c\mathbf{R}_w = \mathbf{R}$ è la matrice di rotazione che viene usata nelle equazioni della camera *pin-hole*.

Per esempio, attraverso queste relazioni, è possibile ottenere le relazioni che legano le pose relative espresse in coordinate camera con quelle espresse in coordinate sensore di equazione (1.66):

$$\begin{aligned} {}^{2b}\mathbf{R}_{1b} &= {}^c\mathbf{\Pi}_b^{-1} \, {}^2\mathbf{R} \, {}^1\mathbf{R}^{-1} \, {}^c\mathbf{\Pi}_b \\ {}^{2b}\mathbf{t}_{2,1} &= -{}^c\mathbf{\Pi}_b^{-1} \, {}^2\mathbf{R}\left(\mathbf{t}_2 - \mathbf{t}_1\right) \end{aligned} \tag{9.6}$$

con i parametri ${}^1\mathbf{R}$, $\mathbf{t}_1$, ${}^2\mathbf{R}$ e $\mathbf{t}_2$ definiti come nel modello *pin-hole* ovvero matrici che trasformano da mondo a camera e vettori espressi in coordinate mondo. Come si vede, questo risultato è totalmente compatibile con quello ottenuto in equazione (9.4).

Queste relazioni permettono di ottenere i parametri della posa relativa $(\mathbf{R}, \mathbf{t})$ partendo dai parametri della camera *pin-hole*, relazioni che permettono di convertire coordinate sensore 1 nelle coordinate sensore 2 come in equazione 1.64.

9.2 Il piano epipolare

Nei capitoli precedenti è stato fatto più volte notare che da una sola immagine non è possibile ottenere le coordinate mondo dei punti che compongono l'immagine, senza informazioni addizionali.

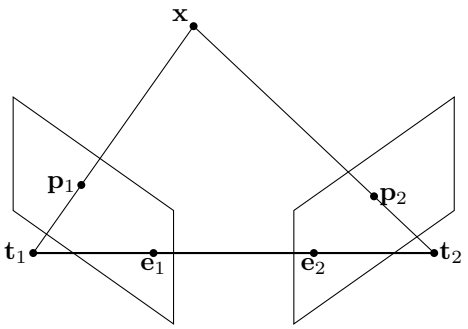


Figura 9.1: Geometria epipolare tra due camere: $\mathbf{t}_1$ e $\mathbf{t_2}$ sono i *pin-hole*, $\mathbf{e}_1$ e $\mathbf{e}_2$ sono gli epipoli e il punto mondo $\mathbf{x}$ viene proiettato nei due punti immagine $\mathbf{p}_1$ e $\mathbf{p}_2$ rispettivamente. Tutti i punti coinvolti appartengono al medesimo piano.

L'unica cosa che un generico punto dell'immagine $\mathbf{p}$ può fornire, data l'equazione (8.19) della camera pin-hole, è una relazione tra le (infinite) coordinate mondo $\mathbf{x}$ sottese al punto immagine ovvero il luogo delle coordinate mondo che proiettate darebbero esattamente quel particolare punto immagine. Tale relazione è l'equazione di una retta passante per il *pin-hole* $\mathbf{t}$ e per il punto sul sensore corrispondente al punto $\mathbf{p}$ immagine. Riportando nuovamente l'equazione (8.19), è facile vedere qual'è la dipendenza tra i parametri della camera i-esima, il punto immagine $\mathbf{p}_i$ e la retta che rappresenta tutti i possibili punti mondo $\mathbf{x}$ sottesi a $\mathbf{p}_i$:

$$\mathbf{x} = \lambda(\mathbf{K}_i\mathbf{R}_i)^{-1}\mathbf{p}_i + \mathbf{t}_i = \lambda\mathbf{v}_i(\mathbf{p}_i) + \mathbf{t}_i \tag{9.7}$$

dove $\mathbf{v}_i$ ha lo stesso significato che aveva in equazione (8.20), vettore direzione tra il *pin-hole* e il punto sensore. Come si evince sia dall'esperienza che dalla relazione lineare che lega tali punti, si può dire che il punto sotteso $\mathbf{x}$ è conosciuto a meno di un fattore di scala λ.

Nel caso della visione stereo abbiamo due sensori e pertanto dobbiamo definire due sistemi di riferimento con parametri $\mathbf{K}_1\mathbf{R}_1$ e $\mathbf{K}_2\mathbf{R}_2$ rispettivi e posizione dei pin-hole $\mathbf{t_1}$ e $\mathbf{t_2}$ espressi sempre in coordinate mondo.

La retta (9.7), luogo dei punti mondo associabili al punto immagine $\mathbf{p}_1$ visto nel primo sistema di riferimento, può essere proiettata nella vista della seconda camera:

$$\begin{aligned}\mathbf{p}_2 &= \lambda \mathbf{K}_2\mathbf{R}_2(\mathbf{K}_1\mathbf{R}_1)^{-1}\mathbf{p}_1 - \mathbf{K}_2\mathbf{R}_2(\mathbf{t}_2 - \mathbf{t}_1) \\ &= \lambda \mathbf{K}_2\mathbf{R}\mathbf{K}_1^{-1}\mathbf{p}_1 + \mathbf{K}_2\mathbf{t} \\ &= \lambda \mathbf{K}_2\mathbf{R}\mathbf{K}_1^{-1}\mathbf{p}_1 + \mathbf{e}_2\end{aligned} \tag{9.8}$$

dove compare una parte variabile, che dipende dal punto considerato e dal valore λ, e un vettore $\mathbf{e}_2$ sempre costante che non dipende dal punto considerato.

Questo punto costante è l'epipolo. L'epipolo è il punto di intersezione di tutte le linee epipolari e rappresenta la proiezione del pin-hole di una camera nell'immagine dell'altra ovvero il "punto di fuga" delle linee epipolari.

Date due camere le proiezioni delle coordinate dei *pin-hole* $\mathbf{t}_1$ e $\mathbf{t}_2$ sull'opposta immagine sono

$$\begin{aligned}\mathbf{e}_1 &= \mathbf{P}_1\mathbf{t}_2 = \mathbf{K}_1\mathbf{R}_1(\mathbf{t}_2 - \mathbf{t}_1) \\ \mathbf{e}_2 &= \mathbf{P}_2\mathbf{t}_1 = \mathbf{K}_2\mathbf{R}_2(\mathbf{t}_1 - \mathbf{t}_2)\end{aligned} \tag{9.9}$$

dove $\mathbf{P}_1$ e $\mathbf{P}_2$ sono le matrici proiettive. I punti $\mathbf{e}_1$ e $\mathbf{e}_2$ sono gli *epipoli*. Se nell'equazione (9.9) inseriamo le definizioni di posa relative espresse in (9.4), le coordinate immagine degli epipoli, intese come la proiezione su una immagine del *pin-hole* dell'altra camera, sono

$$\begin{aligned}\mathbf{e}_1 &= \mathbf{K}_1\mathbf{R}^\top\mathbf{t} \\ \mathbf{e}_2 &= \mathbf{K}_2\mathbf{t}\end{aligned} \tag{9.10}$$

sole funzioni della posa relativa tra le due camere.

La matrice $\mathbf{R}$, per costruzione, converte da coordinate camera 1 a coordinate camera 2 e $\mathbf{t}$ rappresenta la posizione del pin-hole della camera 1 espresso nel sistema di riferimento della camera 2.

Le rette generate dai punti della prima immagine passano tutte per uno stesso punto formato dalla proiezione del pin-hole $\mathbf{t}_1$ sulla seconda immagine: di fatto il punto in coordinate mondo e i due epipoli creano un piano (il piano epipolare) dove vivono le possibili soluzioni, i punti in coordinate camera, del problema della ricostruzione tridimensionale (figura 9.1).

La geometria epipolare è la geometria che lega due immagini acquisite da due punti di vista differenti. Le relazioni che intercorrono tra le immagini tuttavia non dipendono dalla scena osservata ma dipendono solamente dai parametri intrinseci delle camere e dalle pose relative.

Per ogni punto osservato, il piano epipolare è il piano formato dal punto in coordinate mondo e dai 2 centri ottici.

La linea epipolare è l'intersezione tra il piano epipolare e il piano immagine nella seconda immagine. Di fatto il piano epipolare interseca in entrambe le immagini il piano nelle rette epipolari e definisce le corrispondenze tra le linee.

Nelle prossime sezioni verrà discusso sia come ricavare la retta lungo la quale un punto appartenente ad una immagine deve trovarsi in un'altra immagine, sia come dati due (o più) punti omologhi ottenere il punto tridimensionale corrispondente.

9.3 Ricostruzione tridimensionale

L'obiettivo della visione stereoscopica (e in generale la visione multi-oculare) è quella di poter eseguire la ricostruzione tridimensionale della scena osservata. Dall'individuazione del medesimo punto mondo proiettato tra diversi punti di vista tra i quali è conosciuta la posa relativa è possibile eseguire la ricostruzione tridimensionale del punto osservato.

9.3.1 Triangolazione

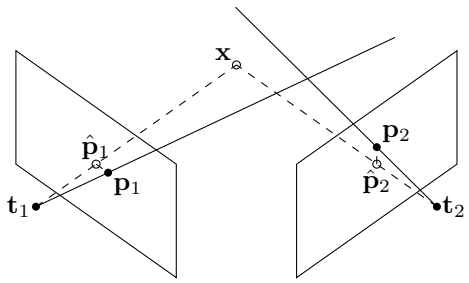


Figura 9.2: Esempio di triangolazione. Con la conoscenza della calibrazione delle camere, il punto mondo $\mathbf{x}$ può essere ricavato dall'osservazione della sua proiezione su almeno due immagini ($\mathbf{p_1}, \mathbf{p_2}, \ldots$). Tuttavia a causa del rumore le rette risultanti non passano per il punto $\mathbf{x}$ e possono non intersecarsi tra loro. La soluzione alla massima verosimiglianza richiede di minimizzare la somma al quadrato degli errori tra il punto osservato $\mathbf{p}_i$ e il punto predetto $\hat{\mathbf{p}}_i$.

Osservando la figura 9.2 è facile intuire che la soluzione del problema della triangolazione è il punto di incontro delle rette epipolari generate dalle due immagini. Tale problema può essere facilmente esteso al caso di n camere dove la posa relativa

tra di loro è conosciuta. In mancanza della conoscenza della posa assoluta questa potrebbe essere ottenuta direttamente dalle immagini stesse usando tecniche come la matrice Essenziale (sezione 9.4).

A causa delle imprecisioni di individuazione dei punti omologhi (un discorso a parte si potrebbe fare per gli errori di calibrazione) le rette formate dai raggi ottici sono in genere sghembe. In tal caso è necessario ricavare la soluzione più vicina sotto qualche funzione di costo: la soluzione ai minimi quadrati è possibile sempre con $n \geq 2$, sia con tecniche come la *Forward Intersections* o la *Direct Linear Transfer* DLT.

Ogni raggio ottico sotteso al pixel immagine (u_i, v_i), con $i = 1, \ldots, n$ la vista i-esima, deve soddisfare l'equazione (9.7). Il punto di intersezione (*Forward Intersections*) di tutti questi raggi è la soluzione di un sistema lineare, potenzialmente sovradimensionato, con $3 + n$ incognite in $3n$ equazioni:

$$\begin{cases} \mathbf{x} & = \lambda_1 \mathbf{v}_1 + \mathbf{t}_1 \\ \ldots & \\ \mathbf{x} & = \lambda_n \mathbf{v}_n + \mathbf{t}_n \end{cases} \tag{9.11}$$

dove con $\mathbf{v}_i = \mathbf{R}_i^{-1}\mathbf{K}_i^{-1} \begin{pmatrix} u_i & v_i & 1 \end{pmatrix}^\top$ si è indicata la direttrice del raggio ottico in coordinate mondo. Le incognite sono il punto mondo da stimare $\mathbf{x}$ e le distanze lungo l'asse ottico λ_i.

La soluzione in forma chiusa, limitata al caso di sole due rette, è disponibile in sezione 1.5.8. Questa tecnica si può applicare al caso di una camera allineata con gli assi e la seconda posizionata relativamente alla prima in accordo con la relazione (9.3).

Sfruttando le proprietà del prodotto vettoriale, si può arrivare alla stessa espressione usando le matrici di proiezione prospettica e i punti immagine, espressi sotto forma di coordinate omogenee:

$$\begin{cases} [\mathbf{p}_1]_\times \mathbf{P}_1 \mathbf{x} = 0 \\ \ldots \\ [\mathbf{p}_n]_\times \mathbf{P}_n \mathbf{x} = 0 \end{cases} \tag{9.12}$$

con $[\cdot]_\times$ il prodotto vettoriale scritto in forma matriciale. Ognuno di questi vincoli fornisce tre equazioni ma solo 2 sono tra loro linearmente indipendenti.Tutti questi vincoli possono essere infine riarrangiati in forma di sistema omogeneo nella forma

$$\mathbf{A}\mathbf{x} = 0 \tag{9.13}$$

dove $\mathbf{A}$ è una matrice $2n \times 4$ con n il numero di viste nelle quali il punto $\mathbf{x}$ è osservato. La soluzione del sistema omogeneo (9.13) si può ottenere con la decomposizione a valori singolari. Tale approccio è chiamato *Direct Linear Transform* (DLT) per similitudine con la tecnica di calibrazione.

La minimizzazione in coordinate mondo tuttavia non è ottima dal punto di vista della minimizzazione del rumore. In mancanza di ulteriori informazioni sulla struttura della scena osservata, la stima ottima (*Maximum Likelihood Estimation*) è sempre quella che minimizza l'errore in coordinate immagine (*reprojection*) ma richiede un maggiore peso computazione e utilizzo di tecniche non lineari, in quanto la funzione di costo da minimizzare è

$$\arg\min_{\mathbf{x}} \sum_{i=1}^{n} \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|^2 \tag{9.14}$$

con $\hat{\mathbf{p}}_i \equiv \mathbf{P}_i \mathbf{x}$ dove $\mathbf{P}_i$ è la matrice di proiezione dell'immagine i-esima (figura 9.2).

È un problema non-lineare non-convesso: sono presenti potenzialmente vari minimi locali e la soluzione lineare deve essere usata come punto di inizio della minimizzazione.

Una ulteriore classe di tecniche, che sfruttano l'informazione ricavata dai vincoli epipolari e attraverso questa permettono di stimare la posizione dei punti non affetti da rumore senza dover ricavare il punto tridimensionale, è mostrato in sezione 9.4.4.

9.3.2 Ricostruzione con camere rettificate

Se i punti (omologhi) tra le due immagini di una coppia stereo fossero sulla stessa riga dell'immagine (ovvero la stessa coordinata v) è possibile sfruttare codice altamente ottimizzato per cercare le corrispondenze [LZ99] e ottenere immagini di disparità dense.

Esiste una configurazione particolare di due camere in cui tale condizione viene rispettata ovvero quando i parametri intrinseci sono uguali e gli assi ottici orientati perpendicolarmente al vettore che congiunge i pin-hole. Per esempio, nel caso in cui il vettore che congiunge i pin-hole giaccia sull'asse y, la configurazione della coppia stereo, che permette di ottenere punti omologhi sulla stessa riga, è quella che ha gli angoli di rotazione $\rho = 0$ e $\gamma = 0$ e angolo di *pitch* uguale.

Il procedimento software per ottenere questa configurazione, quando in hardware tale vincolo non è rispettato, consiste nella rettificazione (vedi 8.3.4) ovvero, partendo da una immagine acquisita con un insieme di parametri (hardware), si ottiene una nuova vista della stessa scena ma con parametri intrinseci, *yaw*, *pitch* e *roll* desiderati.

Attraverso questa considerazione, il problema della ricostruzione tridimensionale si può sempre ricondurre a una coppia di camere perfettamente allineate tra di loro e tra gli assi e a una rototraslazione per trasformare le coordinate mondo da questo sistema sensore all'effettivo sistema reale.

Nelle sezioni successive verrà mostrato il caso particolare sia di camere perfettamente allineate rispetto agli assi, sia di camere allineate ma inclinate (angolo di *pitch* non nullo), sia di camere arbitrariamente orientate.

9.3.3 Camere allineate

Nel caso di camere perfettamente allineate rispetto agli assi e aventi parametri intrinseci uguali (stessa focale e stesso *principal point*) le equazioni per la ricostruzione tridimensionale si semplificano enormemente.

In questa condizione le equazioni della proiezione prospettica si riducono a

$$\begin{aligned} u_i &= -k_u \frac{y - y_i}{x - x_i} + u_0 \\ v_i &= -k_v \frac{z - z_i}{x - x_i} + v_0 \end{aligned} \tag{9.15}$$

con (x, y, z) un punto in coordinate "mondo" (si veda la sezione successiva) e (u_i, v_i) coordinate del punto proiettato sull'immagine i-esima. Il punto (u_0, v_0) è il *principal point* che deve essere il medesimo per tutte le camere coinvolte e le camere si è supposto che siano tutte perfettamente allineate con la terna di assi.

Limitiamoci ora al solo caso stereoscopico: per semplicità verrà indicato con il pedice *1* la camera sinistra e *2* la camera destra. I vincoli di allineamento impongono $x_1 = x_2 = 0$, $y_1 = b$, $y_2 = 0$ e $z_1 = z_2 = 0$ avendo posto, senza perdita di generalità, la camera destra al centro del sistema di riferimento. La quantità $b = y_1 - y_2$ è definita *baseline*.

La differenza $d = u_1 - u_2$ delle coordinate orizzontali delle proiezioni un medesimo punto visto nelle due immagini della coppia stereo si definisce *disparità*. Tale valore si ottiene inserendo i vincoli di allineamento nell'equazione (9.15) e risulta

$$u_1 - u_2 = d = k_u \frac{b}{x} \tag{9.16}$$

Invertendo questa semplice relazione e sostituendola in equazione (9.15) è possibile ricavare il punto in coordinate mondo (x, y, z) corrispondente a un punto (u_2, v_2) della camera destra con disparità d:

$$\begin{aligned} x &= k_u \frac{b}{d} \\ y &= -(u_2 - u_0)\frac{b}{d} \\ z &= -(v - v_0)\frac{k_u}{k_v}\frac{b}{d} \end{aligned} \tag{9.17}$$

È chiaro che deve essere $d \geq 0$ per punti mondo posti davanti alla coppia stereo.

Come è possibile notare, ogni elemento è determinato attraverso il fattore moltiplicativo b della *baseline*, vero fattore di scala della ricostruzione, e dall'inverso della disparità $1/d$.

Triangolazione in coordinate mondo

Le coordinate (x, y, z) così ottenute sono coordinate sensore, riferite a una configurazione stereoscopica particolare dove orientazione e posizionamento sono allineati e coincidenti con gli assi del sistema. Per passare da coordinate sensore al caso generico di coordinate mondo, con camere arbitrariamente orientate, bisogna applicare una trasformazione che porti le coordinate da sensore a mondo, ovvero la matrice di rotazione ${}^{w}\mathbf{R}_b$ e la traslazione $(x_i, y_i, z_i)^\top$ coordinata del *pin-hole*, in modo da poter scrivere

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = {}^{w}\mathbf{R}_b \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} + \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{9.18}$$

Unendo l'equazione (9.17) con l'equazione (9.18), è possibile definire una matrice $\mathbf{M}$ in modo che la conversione tra punto immagine-disparità (u_i, v, d) e coordinata mondo (x, y, z) si possa scrivere in forma molto compatta come

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{1}{d}\mathbf{M} \begin{bmatrix} 1 \\ u_i - u_0 \\ v - v_0 \end{bmatrix} + \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{9.19}$$

dove i può rappresentare indistintamente la camera sinistra o la destra.

9.3.4 Camere allineate e triangolazione in coordinate camera

Le equazioni espresse in precedenza si riferiscono a un sistema di riferimento "sensore" o "mondo". Per completezza, e per introdurre relazioni che verranno usate in seguito, vengono ora riportate anche le equazioni nel caso di sistema di riferimento "camera".

Per mantenere il segno della *baseline* positivo, sia ora $b = \tilde{x}_2 - \tilde{x}_1$, $\tilde{y}_1 = \tilde{y}_2 = 0$ e $\tilde{z}_1 = \tilde{z}_2 = 0$. In questo caso è la camera sinistra (con pedice *1*) ad essere al centro del sistema di riferimento.

In coordinate camera le relazioni tra le due immagini si scrivono come

$$d = u_1 - u_2 = k_u \frac{b}{\tilde{z}} \tag{9.20}$$

per la disparità e

$$\begin{aligned} \tilde{x} &= (u_1 - u_0)\frac{b}{d} \\ \tilde{y} &= (v - v_0)\frac{k_u}{k_v}\frac{b}{d} \\ \tilde{z} &= k_u \frac{b}{d} \end{aligned} \tag{9.21}$$

per l'equazione del punto tridimensionale proiettato sul punto della camera sinistra (u_1, v) di disparità d.

9.3.5 Ricostruzione tridimensionale e omografia

L'equazione (9.21) è facilmente esprimibile in forma omogenea. La matrice che permette dalle coordinate immagine-disparità di ricostruire direttamente le coordinate del punto tridimensionale espresso nel sistema di riferimento camera è

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{k_u} & 0 & 0 & -\frac{u_0}{k_u} \\ 0 & \frac{1}{k_v} & 0 & -\frac{v_0}{k_v} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{k_u b} & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} = \mathbf{Q} \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} \tag{9.22}$$

mentre la sua inversa

$$\begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & 0 & u_0 & 0 \\ 0 & k_v & v_0 & 0 \\ 0 & 0 & 0 & k_u b \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ 1 \end{bmatrix} = \mathbf{Q}^{-1} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ 1 \end{bmatrix} \tag{9.23}$$

è la matrice che permette di proiettare un punto da coordinate camera a coordinate immagine-disparità (sono matrici conosciute a meno di un fattore moltiplicativo, perciò possono essere espresse in diverse forme). La ricostruzione tridimensionale del punto immagine-disparità nel sistema di riferimento mondo, equazione (9.17), è equivalente. La matrice $\mathbf{Q}$ è chiamata *reprojection matrix* [FK08].

In condizioni reali, essendo la camera rototraslata rispetto alle condizioni ideali, è sufficiente moltiplicare la matrice $\mathbf{Q}$ per la matrice 4×4, rappresentante la trasformazione tra coordinate camera a mondo, per ottenere una nuova matrice che permette di passare da coordinate disparità a coordinate mondo e viceversa.

L'utilizzo di tale formalismo permette di trasformare punti disparità acquisiti da coppie di camere posizionate in punti di vista differenti (ad esempio una coppia stereo che si sposta nel tempo o due coppie stereo rigidamente connesse tra loro). In questo caso la relazione che lega punti disparità acquisiti nei due punti di vista è anche rappresentata da una matrice 4×4:

$$\mathbf{H}_{2,1} = \mathbf{Q}_1^{-1} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \mathbf{Q}_2 = \mathbf{Q}_1^{-1} \begin{bmatrix} {}^1\mathbf{R}_2 & {}^1\mathbf{t}_{2,1} \\ 0 & 1 \end{bmatrix} \mathbf{Q}_2 \tag{9.24}$$

che permette di trasformare (u_2, v_2, d_2) in (u_1, v_1, d_1) (è una trasformazione omografica in 4 dimensioni del tutto simile a quelle in 3 dimensioni viste finora). Da notare che si è usato come posa $(\mathbf{R}, \mathbf{t})$ la sintassi di equazione (1.64) volendo esprimere il punto dal sistema di riferimento 2 nel sistema 1. Siccome tutti i punti coinvolti sono espressi in coordinate camera se si hanno trasformazioni tra sensori espresse in coordinate mondo, come normalmente accade, occorre aggiungere il cambio di sistema di riferimento. Tale classe di trasformazioni vengono normalmente indicate come *3D Homographies*.

9.3.6 Camere inclinate rispetto a un piano

Esaminiamo il caso particolare in cui le camere sono allineate rispetto agli assi, hanno parametri intrinseci uguali, rotazione relativa nulla e siano inclinate dell'angolo di beccheggio *pitch* rispetto al piano $z = 0$.

In questa particolare condizione la matrice di proiezione si semplifica leggermente assumendo la forma

$$\mathbf{KR} = \begin{bmatrix} u_0 \cos\vartheta & -k_u & -u_0 \sin\vartheta \\ -k_v \sin\vartheta + v_0 \cos\vartheta & 0 & -k_v \cos\vartheta - v_0 \sin\vartheta \\ \cos\vartheta & 0 & -\sin\vartheta \end{bmatrix} \tag{9.25}$$

È da notare che è stato usato il sistema di angoli *RPY* (appendice A.1) per la matrice $\mathbf{R}$.

La coordinata orizzontale u di un generico punto (x, y, z) in coordinate mondo vale di conseguenza:

$$u = u_0 - \frac{k_u(y - y_0)}{\cos\vartheta(x - x_0) - \sin\vartheta(z - z_0)} \tag{9.26}$$

Con le ipotesi di camere rettificate viste in precedenza, ovvero stessa orientazione e parametri intrinseci uguali, condizione che si può sempre ottenere con la rettificazione o considerando righe opportune dell'immagine, la matrice proiettiva (9.25) risulta essere la stessa nei due sistemi di riferimento differenti e, osservando l'equazione (9.26), l'unica differenza tra camere differenti risulta essere il solo numeratore a causa della differente posizione del *pin-hole* lungo l'asse delle y. Ne consegue che la differenza delle coordinate u nelle due immagini $d = u_1 - u_2$ (disparità) vale

$$d = u_1 - u_2 = \frac{k_u b}{\cos\vartheta(x - x_0) - \sin\vartheta(z - z_0)} \tag{9.27}$$

avendo definito nuovamente $b = y_1 - y_2$. Usando la relazione (9.26) nell'equazione (9.27) si ottiene il risultato notevole

$$u_i = u_0 - d\frac{y - y_i}{b} \tag{9.28}$$

da cui infine si ricava la coordinata y del punto

$$y = -b\frac{u_i - u_0}{d} + y_i \tag{9.29}$$

Nel caso in cui le camere siano allineate perfettamente, l'unico parametro di calibrazione che incide sulla coordinata y risulta essere la sola b.

La coordinata v del punto si può scrivere invece come

$$v - v_0 = -\frac{k_v}{bk_u}(\sin\vartheta(x - x_0) + \cos\vartheta(z - z_0))d \tag{9.30}$$

Da cui il sistema di equazioni:

$$\begin{aligned} \cos\vartheta(x - x_0) - \sin\vartheta(z - z_0) &= \frac{bk_u}{d} \\ \sin\vartheta(x - x_0) + \cos\vartheta(z - z_0) &= -\frac{v - v_0}{k_v}\frac{bk_u}{d} \end{aligned} \tag{9.31}$$

la cui soluzione che permette di ottenere le restanti due coordinate tridimensionali del punto dato è

$$\begin{aligned} x - x_0 &= \frac{bk_u}{d}\left(\cos\vartheta - \frac{v - v_0}{k_v}\sin\vartheta\right) \\ z - z_0 &= -\frac{bk_u}{d}\left(\frac{v - v_0}{k_v}\cos\vartheta + \sin\vartheta\right) \end{aligned} \tag{9.32}$$

V-Disparity

Un caso particolare di disparità è quando si osserva un piano, quello del terreno, che, per numero di punti, è preponderante sull'immagine. Nel caso in cui la *baseline* sia lungo l'asse y, la disparità del piano $z = 0$ è solo funzione di v e tale equazione risulta essere quella di una retta.

La relazione della disparità dalla coordinata v si può ricavare dal valore di x dalla seconda e sostituendolo nella prima delle equazioni (9.31):

$$\begin{aligned} x - x_0 &= \tan\vartheta(z - z_0) + \frac{k_u}{d\cos\vartheta}b \\ v - v_0 &= -k_v\tan\vartheta - d\frac{k_v}{k_u}\frac{z - z_0}{b\cos\vartheta} \end{aligned} \tag{9.33}$$

Dalla prima delle equazioni (9.33), si vede che l'espressione della disparità dipende solamente dalla distanza x se l'altezza z è fissata (ad esempio sul suolo), e dalla seconda si vede che la disparità d cresce linearmente con la coordinata v seguendo un coefficiente angolare noto

$$d = \cos\vartheta\frac{b}{z_0}(v - v_{d=0}) \tag{9.34}$$

nel caso classico in cui $k_u \approx k_v$ (pixel quadrato). Il punto di disparità nulla $v_{d=0}$, sopra menzionato, si trova in

$$v_{d=0} = v_0 - k_v\tan\vartheta \tag{9.35}$$

e dipende solo dall'apertura verticale e dal *pitch* (è ovviamente la stessa coordinata del *vanishing point*).

9.4 Matrice Essenziale e matrice Fondamentale

Nel 1981, Christopher Longuet-Higgins [Lon81] osserva per primo che un generico punto espresso in coordinate mondo, i rispettivi punti in coordinate camera e i *pin-hole* devono essere tra loro coplanari. La derivazione geometrica delle relazioni che intercorrono tra i punti viene tralasciata ma viene presentata direttamente quella analitica.

È stato più volte ripetuto che un punto su un'immagine sottende una retta nel mondo, e la retta nel mondo proiettata su un altra immagine, acquisita da una camera posta in un punto di vista differente, rappresenta la retta epipolare dove giace l'omologo del punto della prima immagine. Tale equazione, che lega punti di un'immagine con rette nell'altra, può essere espressa attraverso una forma matriciale.

Per seguire il ragionamento di Higgins, la matrice dei parametri intrinseci verrà sottintesa e le coordinate usate saranno quelle camera normalizzate.

Senza perdere generalità, si consideri pertanto un sistema costituito da due camere, la prima posizionata e orientata rispetto alla seconda con matrice di proiezione $\mathbf{P}_1 = [\mathbf{R}|\mathbf{t}]$ mentre la seconda è posta nell'origine del sistema di riferimento allineata con gli assi ovvero con matrice di proiezione $\mathbf{P}_2 = [\mathbf{I}|\mathbf{0}]$: si può giungere al medesimo risultato partendo da due generiche camere calibrate, arbitrariamente orientate e posizionate rispetto ad un sistema terzo, attraverso le relazioni $\mathbf{R} = {}^2\mathbf{R}_1 = \mathbf{R}_2^{-1}\mathbf{R}_1$ e $\mathbf{t}$, ovvero posizione della camera 1 rispetto al sistema 2.

Un generico punto $\mathbf{x} \in \mathbb{R}^3$ ha coordinata $\mathbf{x}_1$ e $\mathbf{x}_2$ nei due diversi sistemi di riferimento e viene proiettato sui sensori 1 e 2 nei punto in coordinate camera $\mathbf{m}_1$ e $\mathbf{m}_2$ rispettivamente.

Questi punti immagine sappiamo che sottendendo un sottospazio di $\mathbb{R}^3$ di equazione per esempio $\lambda\mathbf{m}_2$ passante per il pin-hole del secondo sensore (qui imposto ad essere in $\mathbf{0}$) ovvero

$$\mathbf{x}_2 = \lambda_2\mathbf{m}_2 + \mathbf{0} \tag{9.36}$$

Un generico punto $\mathbf{x}_1 = \lambda_1\mathbf{m}_1$ espresso in coordinate del sensore 1 e osservato da quel sensore può venire proiettato in coordinate del sensore 2 in accordo con l'equazione

$$\mathbf{x}_2 = {}^2\mathbf{R}_1\mathbf{x}_1 + \mathbf{t} \tag{9.37}$$

L'equazione della retta epipolare, retta in coordinate camera nel secondo sensore e luogo dei punti dove deve giacere $\mathbf{m}_2$, associata al punto $\mathbf{m}_1$ (osservato e pertanto espresso in coordinate camera del primo sensore), risulta essere

$$\lambda_2\mathbf{m}_2 = \lambda_1\,{}^2\mathbf{R}_1\mathbf{m}_1 + \mathbf{t} \tag{9.38}$$

Il luogo dei punti omogenei $\mathbf{m}_2$ si ottiene facendo variare il parametro λ_1 e questa retta in $\mathbb{R}^3$ risulta essere una retta anche in $\mathbb{R}^2$. Se due punti sono effettivamente omologhi, il sistema è risolvibile ed è possibile ricavare i parametri λ_1 e λ_2 (questo è un esempio di ricostruzione tridimensionale tramite triangolazione come si è visto in sezione 9.3.1).

Esiste tuttavia una relazione che mette in relazione i punti delle due camere cancellando i parametri λ ma soprattutto permette di fare il ragionamento inverso ovvero quello di ricavare la posa relativa tra le due camere $(\mathbf{R}, \mathbf{t})$ dato un elenco di punti omologhi.

Se si moltiplicano entrambi i lati dell'equazione (9.38), prima vettorialmente per $\mathbf{t}$, poi scalarmente per $\mathbf{m}_2^\top$, si ottiene infatti

$$\lambda_2\mathbf{m}_2^\top(\mathbf{t} \times \mathbf{m}_2) = \lambda_1\mathbf{m}_2^\top(\mathbf{t} \times \mathbf{R}\mathbf{m}_1) + \mathbf{m}_2^\top(\mathbf{t} \times \mathbf{t}) \tag{9.39}$$

Su tale relazione è possibile applicare le proprietà del prodotto vettoriale $\mathbf{t} \times \mathbf{t} = \mathbf{0}$ e scalare $\mathbf{m}_2 \cdot (\mathbf{t} \times \mathbf{m}_2) = 0$.

Questo passaggio ha un significato fisico: vengono per prima di tutto inseriri i vincoli di coplanarità (tutti espressi per esempio nel reference 2) tra i punti $\mathbf{0}$ (pinhole della camera 2), $\mathbf{m}_2$, ${}^2\mathbf{R}_1\mathbf{m}_1 + \mathbf{t}$, $\mathbf{x}_2 = {}^2\mathbf{R}_1\mathbf{x}_1 + \mathbf{t}$ e $\mathbf{t}$ (pinhole della camera 1 nel sistema 2), e uniti con il fatto che il corpo è rigido.

Attraverso questa formula, è possibile esprimere le relazioni che intercorrono tra i punti omologhi $\mathbf{m}_1$ e $\mathbf{m}_2$, rappresentati sotto forma di coordinate camera omogenee, in una forma molto compatta

$$\mathbf{m}_2^\top(\mathbf{t} \times \mathbf{R}\mathbf{m}_1) = 0 \tag{9.40}$$

Indicando infine con $[\mathbf{t}]_\times$, matrice antisimmetrica, il prodotto vettoriale in forma matriciale (sezione 1.7), è possibile raccogliere i diversi contributi sotto forma di matrice

$$\mathbf{E} = [\mathbf{t}]_\times\mathbf{R} = \mathbf{R}\left[\mathbf{R}^\top\mathbf{t}\right]_\times \tag{9.41}$$

dove va ricordato bene il significato delle matrici confrontando con l'equazione (9.37). Si può definire in questo modo una relazione lineare che lega i punti camera delle due viste

$$\mathbf{m}_2^\top\mathbf{E}\mathbf{m}_1 = 0 \tag{9.42}$$

La matrice $\mathbf{E}$ è definita *Matrice Essenziale*.

Bisogna infine stare molto attenti agli indici perché non c'è una convenzione unica per indicare i punti 1 e 2: supponendo la convenzione (9.41) soddisfatta quello che bisogna ricordarsi è che matrice **E** codifica la posa relativa della camera dei punti a destra (nel nostro caso $\mathbf{m}_1$) della matrice **E** rispetto alla camera dei punti a sinistra (nel nostro caso $\mathbf{m}_2$) della matrice.

La matrice **E**, mettendo in relazioni punti omogenei, è anch'essa omogenea e pertanto definita a meno di un fattore moltiplicativo.

La matrice Essenziale ha le seguenti proprietà:

- la trasposta della matrice Essenziale della coppia ordinata di camere (1,2) è la matrice Essenziale della coppia (2,1);
- **E** è una matrice di rango 2 con 5 gradi di libertà (rappresenta infatti una posa relativa, perciò 3 angoli e la direzione tra gli epipoli, ovvero 2 gradi di libertà);
- i due valori singolari della matrice **E** devono essere uguali e il terzo deve essere zero.

La matrice Essenziale crea delle relazioni in coordinate camera e pertanto, per poterla utilizzare da un punto di vista pratico, è necessario avere a disposizione punti espressi in questo particolare sistema di riferimento, ovvero è necessario conoscere i parametri intrinseci delle camere coinvolte.

L'equazione

$$\mathbf{m}_2^\top (\mathbf{E}\mathbf{m}_1) = 0 \tag{9.43}$$

può essere interpretata come equazione anche di un piano nello spazio 2 passante per $\mathbf{0}$, ovvero il piano epipolare formato dai due epipoli e dal punto mondo, piano dove il punto $\mathbf{m}_2$ deve appartenere.

È tuttavia possibile introdurre una ulteriore relazione tra i punti delle immagini, trascurando completamente i parametri intrinseci delle camere stesse.

Se si applica la definizione di coordinate camera omogenee $\mathbf{p} = \mathbf{K}\mathbf{m}$ nella relazione (9.42) si ottiene

$$\mathbf{m}_2^\top \mathbf{E}\mathbf{m}_1 = \mathbf{p}_2^\top \mathbf{K}_2^\top \mathbf{F}\mathbf{K}_1\mathbf{p}_1 = \mathbf{p}_2^\top \mathbf{F}\mathbf{p}_1 \tag{9.44}$$

La matrice Fondamentale (*Fundamental matrix*) è definita (Faugeras e Hartley, 1992) come:

$$\mathbf{p}_2^\top \mathbf{F}\mathbf{p}_1 = 0 \tag{9.45}$$

dove $\mathbf{p}_1$ e $\mathbf{p}_2$ sono coordinate, sempre omogenee, dei punti omologhi rispettivamente sulla prima e sulla seconda immagine.

Se due punti sulle due immagini della coppia stereoscopica rappresentano lo stesso punto nel mondo, l'equazione (9.45) deve essere soddisfatta.

La matrice fondamentale permette di restringere l'intervallo di ricerca di corrispondenze tra le due immagini in quanto, per il dualismo punto-retta, dalla relazione (9.45) si può esplicitare il luogo dei punti nella seconda immagine dove cercare i punti della prima. Infatti l'equazione di una linea dove i punti $\mathbf{m_2}$ ed $\mathbf{m_1}$ devono vivere è descritta da

$$\begin{aligned} \mathbf{l}_2 &= \mathbf{F}\mathbf{m}_1 \\ \mathbf{l}_1 &= \mathbf{F}^\top \mathbf{m}_2 \end{aligned} \tag{9.46}$$

dove $\mathbf{l}_1$ ed $\mathbf{l}_2$ sono i parametri della retta epipolare, appartenente alla prima e seconda immagine rispettivamente, scritta in forma implicita.

La relazione che intercorre tra la matrice Fondamentale e la matrice Essenziale risulta essere, equazione (9.44),

$$\mathbf{E} = \mathbf{K}_2^\top \mathbf{F}\mathbf{K}_1 \tag{9.47}$$

o viceversa

$$\mathbf{F} = \mathbf{K}_2^{-\top} \mathbf{E}\mathbf{K}_1^{-1} = \mathbf{K}_2^{-\top} [\mathbf{t}]_\times \mathbf{R}\mathbf{K}_1^{-1} \tag{9.48}$$

La matrice Essenziale raccoglie in sxE9 le pose relativa tra le camere, mentre la matrice Fondamentale nasconde sia i parametri intrinseci che la posa relativa.

La matrice Essenziale introduce vincoli uguali a quelli della matrice Fondamentale ma, anche se introdotta storicamente prima della matrice Fondamentale, ne è un caso particolare perchxE9 esprime le relazioni rispetto a coordinate camera.

F è una matrice 3×3 di rango 2 e per essere determinata bastano 7 punti, in quanto i gradi di libertà risultano essere appunto solamente 7 (un fattore moltiplicativo e il determinante nullo riducono la dimensione del problema). La relazione che lega la matrice Fondamentale ai 7 gradi di libertà è una relazione non lineare (relazione non facilmente esprimibile attraverso una qualche rappresentazione algebrica). Con (almeno) 8 punti si riesce invece ad ottenere una stima lineare della matrice, come descritto nella sezione successiva.

La matrice Fondamentale ha le seguenti proprietà:

- la trasposta della matrice Fondamentale della coppia ordinata di camere (1,2) è la matrice Fondamentale della coppia (2,1);

- $\mathbf{F}$ è una matrice di rango 2 con 7 gradi di libertà (la matrice omogenea $\mathbf{F}$ ha 8 gradi di libertà a cui va aggiunto il vincolo $\det \mathbf{F} = 0$);
- $\mathbf{l}_2 = \mathbf{F}\mathbf{p}_1$ e $\mathbf{l}_1 = \mathbf{F}^\top \mathbf{p}_2$ sono le rette epipolari rispettivamente nell'immagine 2 di un punto dell'immagine 1 e nell'immagine 1 di un punto dell'immagine 2;
- siccome gli epipoli devono soddisfare la relazione $\mathbf{F}\mathbf{e}_1 = 0$ e $\mathbf{F}^\top \mathbf{e}_2 = 0$, rispettivamente, consegue che essi sono i kernel "sinistro" e "destro" della matrice $\mathbf{F}$;
- $\mathbf{F}$ è una "quasi correlazione" ovvero una trasformazione che trasforma punti in linee ma non è invertibile.

Figura 9.3: La matrice Fondamentale permette di individuare le rette epipolari, immagine destra, su cui vivono i punti omologhi dei punti dell'immagine sinistra.

Le matrici Fondamentale ed Essenziale possono essere usate per restringere il campo di ricerca dei punti omologhi tra due immagini e/o filtrare via eventuali *outlier* (ad esempio in RANSAC). La matrice Essenziale, se decomposta, permette di ricavare la posa relativa tra le due camere e in quanto tale dare una idea, approssimata, del movimento che ha subito una camera che si sposta nel mondo (*motion stereo*) o della posa relativa di due camere in una coppia stereoscopica (Auto-Calibrazione).

L'uso della matrice Essenziale permette di ricavare la posa relativa tra due viste. Non è però possibile conoscere la lunghezza della *baseline* che unisce i due *pin-hole*, ma solo la sua direzione. Tuttavia, avendo a disposizione la matrice Essenziale, è sempre possibile eseguire una ricostruzione tridimensionale della scena osservata a meno di un fattore moltiplicativo: i rapporti tra le distanze sono conosciuti, ma non il loro valore assoluto. Questo permette però, quando si osserva la medesima scena da più di due viste differenti, una ricostruzione tridimensionale coerente, dove per tutte le viste il fattore moltiplicativo sconosciuto rimane sempre lo stesso, permettendo perciò di fondere tutte le singole ricostruzioni in un'unica ricostruzione conosciuta a meno del medesimo fattore di scala.

9.4.1 Determinazione delle matrici

La matrice Essenziale si può ricavare in forma chiusa conoscendo le pose relative tra i sensori e, con la conoscenza dei parametri intrinseci delle camere coinvolte, è possibile ottenere la matrice Fondamentale.

L'applicazione però più diffusa della matrice Essenziale (o Fondamentale) è quella di ricavare la posa relativa tra le camere, data la conoscenza di un elenco di punti omologhi: conoscendo i parametri intrinseci si può ricavare la matrice Essenziale (e da questa ricavare la matrice Fondamentale) o senza nessuna conoscenza dei parametri delle camere poter ricavare la matrice Fondamentale.

Algoritmo degli 8 punti

Il criterio per ottenere la matrice $\mathbf{F}$ si può formalizzare come una minimizzazione di una funzione costo

$$\min_{\mathbf{F}} \sum_{i=1}^{n} \left(\mathbf{p}_{2,i}^\top \mathbf{F} \mathbf{p}_{1,i}\right)^2 \tag{9.49}$$

sotto ulteriori vincoli che riguardano questa volta la struttura di $\mathbf{F}$.

Si può notare che il vincolo epipolare (9.45) può essere riscritto anche come

$$(\mathbf{p}_1 \otimes \mathbf{p}_2)^\top \mathrm{vec}(\mathbf{F}) = 0 \tag{9.50}$$

avendo sfruttato la sintassi compatta offerta del prodotto di Kronecker $\otimes$ o alternativamente

$$\mathbf{u}_i \mathbf{f} = 0 \tag{9.51}$$

in forma più esplicita definendo

$$\begin{aligned}(\mathbf{p}_1 \otimes \mathbf{p}_2)^\top = &\quad \mathbf{u}_i = (x_1x_2, y_1x_2, x_2, x_1y_2, y_1y_2, y_2, x_1, y_1, 1)\\ \mathrm{vec}(\mathbf{F}) = &\quad \mathbf{f} = (f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}, f_{3,1}, f_{3,2}, f_{3,3})^\top\end{aligned} \tag{9.52}$$

con $\mathbf{p}_{1,i} = (x_1, y_1)$ e $\mathbf{p}_{2,i} = (x_2, y_2)$. Con questo formalismo è possibile mettere in evidenza una tecnica che permette di ricavare gli elementi di $\mathbf{F}$ come soluzione di un sistema lineare omogeneo formato da vincoli come in equazione (9.51).

Raccolti tutti i vincoli $\mathbf{u}_i$, si ottiene un sistema omogeneo del tipo

$$\mathbf{Uf} = 0 \tag{9.53}$$

di n equazioni in 9 incognite.

Per ricavare la matrice Essenziale il discorso è analogo ed è soluzione di un sistema $\mathbf{n}_i\mathbf{e} = 0$ nella forma

$$\begin{aligned}\mathbf{n}_i &= (x_1x_2, y_1x_2, z_1x_2, x_1y_2, y_1y_2, z_1y_2, x_1z_2, y_1z_2, z_1z_2)\\ \mathbf{e} &= (e_{1,1}, e_{1,2}, e_{1,3}, e_{2,1}, e_{2,2}, e_{2,3}, e_{3,1}, e_{3,2}, e_{3,3})\end{aligned} \tag{9.54}$$

con $\mathbf{m}_{1,i} = (x_1, y_1, z_1)$ e $\mathbf{m}_{2,i} = (x_2, y_2, z_2)$. Con tutti i vincoli $\mathbf{n}_i$, questo è un sistema omogeneo del tipo

$$\mathbf{Ne} = 0 \tag{9.55}$$

Di fatto, usando le coordinate omogenee, i sistemi (9.52) e (9.54) sono equivalenti dal punto di vista algoritmico.

Ai vincoli espressi in questi sistemi omogenei, ne va sempre aggiunto uno ulteriore, per esempio $\|\mathbf{f}\| = 1$, normalmente già soddisfatto dai risolutori lineari di sistemi omogenei. Questo algoritmo è pertanto chiamato *eight-point algorithm* in quanto la soluzione del problema richiede almeno 8 punti per essere determinata. Ulteriori vincoli, per raggiungere gli effettivi gradi di libertà delle matrici, non sono invece esprimibili sotto forma lineare.

Rafforzamento dei vincoli

A causa del rumore, normalmente le matrici ottenute dal sistema lineare non soddisfano il requisito che siano di rango 2 (e nel caso di matrice Essenziale, perciò con un ampio numero di gradi di libertà, che non appartengano proprio al sottospazio delle matrici Essenziali). Una possibile soluzione a questo problema è quella di cercare la matrice più vicina a quella restituita dal sistema lineare che soddisfa però il vincolo sul rango. Tale risultato si ottiene per esempio usando una decomposizione SVD seguita da una composizione, come suggerito da Tsai, Huang e Hartley:

$$\begin{aligned}\mathbf{F} &= \mathbf{U}\,\mathrm{diag}(r, s, t)\mathbf{V}^\top\\ \mathbf{F}' &= \mathbf{U}\,\mathrm{diag}(r, s, 0)\mathbf{V}^\top\end{aligned} \tag{9.56}$$

Questo procedimento è chiamato *constraint enforcement*. La matrice Essenziale rispetto a quella Fondamentale ha in più il vincolo di avere i 2 valori singolari non nulli uguali:

$$\begin{aligned}\mathbf{E} &= \mathbf{U}\,\mathrm{diag}(r, s, t)\mathbf{V}^\top\\ \mathbf{E}' &= \mathbf{U}\,\mathrm{diag}(1, 1, 0)\mathbf{V}^\top\end{aligned} \tag{9.57}$$

Se i valori singolari (in seguito a una SVD) della matrice sono 1, la matrice si dice matrice Essenziale normalizzata (*normalized essential matrix*). La matrice Essenziale ottenuta ponendo $\mathbf{D}' = \mathrm{diag}(1, 1, 0)$ è la matrice essenziale normalizzata più vicina a quella data, in accordo con la norma di Frobenius. La matrice Essenziale generata attraverso l'equazione (9.57) soddisfa la *cubic trace-constraint* (Demazure, 1988)

$$\mathbf{E}\mathbf{E}^\top\mathbf{E} - \frac{1}{2}\,\mathrm{trace}\left(\mathbf{E}\mathbf{E}^\top\right)\mathbf{E} = 0 \tag{9.58}$$

Tale vincolo è condizione necessaria perchxE9 la matrice in analisi sia effettivamente Essenziale.

Le matrici ottenute attraverso questo procedimento di rafforzamento soddisfano tutti i requisiti per essere matrici Fondamentali o Essenziali, ma non rappresentano una minimizzazione algebrica, ne tanto mento geometrica, dei vincoli originali.

Algoritmo dei 7 punti

Gli algoritmi che sfruttano meno di 8 punti per estrarre una matrice Essenziale o Fondamentale si basano più o meno tutti sullo stesso principio: viene estratto il kernel multidimensionale di $\mathbf{U}$ o $\mathbf{N}$, visto che la matrice Fondamentale o Essenziale deve appartenere a un elemento di questo spazio, e vengono forzati alcuni vincoli tipici del problema in questione.

In maniera non lineare è relativamente facile poter ottenere una matrice Fondamentale con soli 7 punti, considerando il fatto che la matrice $\mathbf{U}$, formata dagli elementi di equazione (9.52), deve essere di rango 7, in quanto 7 sono in effetti i gradi di libertà della matrice Fondamentale. Risolvendo il sistema (9.52) formato da (almeno) 7 punti si ottiene un sottospazio di dimensione 2, formato da due basi $\mathbf{f}_1$ e $\mathbf{f}_2$, a cui sono associate due matrici $\mathbf{F}_1$ e $\mathbf{F}_2$: nello spazio delle possibili soluzioni è necessario trovare una matrice $\mathbf{F} = \alpha\mathbf{F}_1 + (1 - \alpha)\mathbf{F}_2$ tale che abbia rango 2 ovvero imponendo $\det\mathbf{F} = 0$, equazione non lineare di terzo grado in α. In questo caso le soluzioni reali di α possono essere 1 o 3: nel caso di 3 soluzioni reali, vanno tutte e 3 valutate sui dati per individuare quella più plausibile.

Algoritmo dei 5 punti

Con meno di 7 punti esistono solo algoritmi per determinare la matrice Essenziale. La matrice Essenziale infatti è formata da soli 5 gradi di libertà e in linea teorica può essere stimata attraverso l'analisi di corrispondenze fra solo 5 punti [Nis04]. L'algoritmo dei 5 punti è di fatto lo standard per la stima della matrice essenziale, tuttavia la sua implementazione è estremamente complessa.

Sfruttando solo 5 corrispondenze, la matrice $\mathbf{N}$ del sistema (9.55) ha un difetto 4 di rango. La matrice Essenziale deve essere pertanto formata come combinazione lineare delle ultime 4 colonne della matrice V ottenuta dalla SVD, ovvero:

$$\mathbf{E} = x\mathbf{X} + y\mathbf{Y} + z\mathbf{Z} + \mathbf{W} \tag{9.59}$$

con $(\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{W})$ ultime 4 colonne di autovettori della matrice V e (x, y, z) incognite. Per ottenere tali incognite è necessario soddisfare i vicoli

$$\begin{aligned} &\det \mathbf{E} = 0 \\ &\mathbf{E}\mathbf{E}^\top\mathbf{E} - \tfrac{1}{2}\operatorname{trace}\left(\mathbf{E}\mathbf{E}^\top\right)\mathbf{E} = 0 \end{aligned} \tag{9.60}$$

equivalente a un problema di 10 polinomi di terzo grado nelle 3 incognite.

La richiesta comunque di risolvere un sistema non-lineare fa diminuire i vantaggi rispetto alle soluzioni proposte in sezione 9.4.2.

Condizionamento del sistema

La generazione, attraverso tecnica SVD, delle matrici Essenziale e Fondamentale e in seguito l'irrobustimento di queste, forzando i valori singolari ad essere uguali, è un processo molto sensibile al rumore.

La matrice (9.52) è mal condizionata: questo accade quando si cerca di risolvere un sistema lineare i cui termini noti sono formati da numeri con ordini di grandezza differenti. Il metodo proposto da Hartley [Har95] propone di migliorare la soluzione normalizzando le coordinate dei punti.

Le coordinate $\mathbf{p}_1$ e $\mathbf{p}_2$ vengono traslate separatamente in modo da avere centroide nullo e riscalate in modo da avere come valor medio 1 (o $\sqrt{2}$ valor medio del modulo) nel nuovo sistema di coordinate $\tilde{\mathbf{p}}_1$ e $\tilde{\mathbf{p}}_2$ rispettivamente. Definiamo pertanto due matrici di trasformazione $\mathbf{T}_1$ e $\mathbf{T}_2$ tali che

$$\begin{aligned} \tilde{\mathbf{p}}_1 &= \mathbf{T}_1\mathbf{p}_1 \\ \tilde{\mathbf{p}}_2 &= \mathbf{T}_2\mathbf{p}_2 \end{aligned} \tag{9.61}$$

in questo modo è possibile determinare la matrice fondamentale compatibile $\tilde{\mathbf{F}}$

$$\mathbf{p}_2^\top\mathbf{F}\mathbf{p}_1 = \tilde{\mathbf{p}}_2^\top\mathbf{T}_2^{-\top}\mathbf{F}\mathbf{T}_1^{-1}\tilde{\mathbf{p}}_1 = \tilde{\mathbf{p}}_2^\top\tilde{\mathbf{F}}\tilde{\mathbf{p}}_1 = 0 \tag{9.62}$$

da cui poi ricavare la matrice originale $\mathbf{F} = \mathbf{T}_2^\top\tilde{\mathbf{F}}\mathbf{T}_1$.

9.4.2 Stima alla Massima Verosimiglianza

Quando si usa la decomposizione SVD per irrobustirne i vincoli, la matrice Fondamentale (o Essenziale) che si ottiene soddisfa pienamente i requisiti per essere Fondamentale (o Essenziale) ma tuttavia è solo una matrice più simile sotto una particolare norma (in questo caso Frobenius) a quella ottenuta dal sistema lineare. Neanche questa soluzione pertanto è ottima perchxE9 non tiene comunque conto di come avviene la propagazione dell'errore dai punti in ingresso all'interno della trasformazione: è di fatto ancora una soluzione algebrica e non geometrica.

Una prima tecnica, che minimizza l'errore geometrico, consiste nel sfruttare la distanza tra i punti e le rette epipolari generate attraverso la matrice Fondamentale (*epipolar distance*).

Anche solo intuitivamente, la distanza tra un punto $\mathbf{p}_2$ e la retta epipolare $\mathbf{F}\mathbf{p}_1$ può essere usata come metrica per stimare l'errore geometrico:

$$d\left(\mathbf{p}_2, \mathbf{F}\mathbf{p}_1\right) = \frac{\left|\mathbf{p}_2^\top\left(\mathbf{F}\mathbf{p}_1\right)\right|}{\sqrt{\left(\mathbf{F}\mathbf{p}_1\right)_1^2 + \left(\mathbf{F}\mathbf{p}_1\right)_2^2}} \tag{9.63}$$

dove con $(.)_i$ è indicata la componente i-esima del vettore (si veda la sezione 1.5.3 per l'equazione della distanza punto-retta). Più la distanza è bassa, più la matrice $\mathbf{F}$ è a tutti gli effetti la matrice che mette in relazione i punti omologhi.

Siccome è possibile calcolare sia per la prima immagine che per la seconda questo errore, è corretto minimizzare entrambi i contributi insieme. Attraverso questa metrica è possibile definire una funzione costo che minimizzi l'errore in maniera simmetrica (*symmetric transfer error*) tra le due immagini:

$$\min_{\mathbf{F}} \sum_i \left(d\left(\mathbf{p}_{1,i}, \mathbf{F}\mathbf{p}_{2,i}\right)^2 + d\left(\mathbf{p}_{2,i}, \mathbf{F}^\top\mathbf{p}_{1,i}\right)^2 \right) \tag{9.64}$$

Anche in questo caso si può cercare una soluzione a 8 incognite ma, per cercare una soluzione robusta, risulta necessario vincolare $\mathbf{F}$ ad essere di rango 2.

Alternativamente al *Symmetric Transfer Error* in letteratura viene spesso usata l'approssimazione al primo grado della distanza tra i punti e la funzione (*Sampson-error*, sezione 3.3.8). È possibile definire così una distanza approssimata tra i punti immagine omologhi $(\mathbf{p}_1, \mathbf{p}_2)$ e la varietà $\hat{\mathbf{p}}_2^{\top}\mathbf{F}\hat{\mathbf{p}}_1 = 0$ attraverso la metrica

$$r(\mathbf{p}_1, \mathbf{p}_2, \mathbf{F}) \approx \frac{\left|\mathbf{p}_2^{\top}\mathbf{F}\mathbf{p}_1\right|}{\sqrt{(\mathbf{F}\mathbf{p}_1)_1^2 + (\mathbf{F}\mathbf{p}_1)_2^2 + (\mathbf{F}^{\top}\mathbf{p}_2)_1^2 + (\mathbf{F}^{\top}\mathbf{p}_2)_2^2}} \tag{9.65}$$

dove con $(.)_i$ è indicata nuovamente la componente i-esima del vettore. Attraverso questa metrica approssimata, mantenendo sempre il vincolo aggiuntivo $\det \mathbf{F} = 0$, è possibile minimizzare

$$\min_{\mathbf{F}} \sum_{i=1}^{n} r(\mathbf{p}_{1,i}, \mathbf{p}_{2,i}, \mathbf{F})^2 \tag{9.66}$$

Sia il *Symmetric Transfer Error* che la distanza di Sampson, metriche comunque migliori della stima algebrica, non sono lo stimatore ottimo. La stima di massima verosimiglianza *Maximum Likelihood Estimation* per la matrice Fondamentale si otterrebbe infatti usando una funzione costo del tipo

$$\min_{\mathbf{F}} \sum_{i} \|\mathbf{p}_{1,i} - \hat{\mathbf{p}}_{1,i}\|^2 + \|\mathbf{p}_{2,i} - \hat{\mathbf{p}}_{2,i}\|^2 \tag{9.67}$$

avendo indicato con $\hat{\mathbf{p}}_{1,i}$ e $\hat{\mathbf{p}}_{2,i}$ i punti esatti e con $\mathbf{p}_{1,i}$, $\mathbf{p}_{2,i}$ i corrispondenti punti misurati affetti da rumore gaussiano bianco a media nulla. La funzione costo (9.67) va minimizzata sotto il vincolo

$$\hat{\mathbf{p}}_{2,i}^{\top}\mathbf{F}\hat{\mathbf{p}}_{1,i} = 0 \tag{9.68}$$

e con i vincoli aggiuntivi dovuti alla natura di $\mathbf{F}$. In questo caso i punti esatti $\hat{\mathbf{p}}_{1,i}$ e $\hat{\mathbf{p}}_{2,i}$ diventano parte del problema (variabili ausiliarie, *subsidiary variables*). Tuttavia, introdurre i punti $\hat{\mathbf{p}}_{1,i}$ e $\hat{\mathbf{p}}_{2,i}$ come incognite, rende il problema non risolvibile, in quanto sarebbero presenti sempre più incognite che vincoli.

Per risolvere questo problema bisogna unire il problema del calcolo della matrice Essenziale o Fondamentale con quello della ricostruzione tridimensionale e porre come variabile ausiliaria direttamente la coordinata tridimensionale del punto osservato $\hat{\mathbf{x}}_i$.

La matrice Essenziale può essere ottenuta data la conoscenza dei parametri intrinseci dei due sensori. In questo caso infatti è possibile sfruttare il sistema non lineare che proietta la variabile ausiliaria $\hat{\mathbf{x}}_i$ sulle rispettive due osservazioni sui due rispettivi sensori:

$$\begin{aligned} \hat{\mathbf{p}}_{1,i} &\equiv \mathbf{K}_1\hat{\mathbf{x}}_i \\ \hat{\mathbf{p}}_{2,i} &\equiv \mathbf{K}_2(\mathbf{R}\hat{\mathbf{x}}_i + \mathbf{t}) \end{aligned} \tag{9.69}$$

dove la matrice $\mathbf{R}$ può essere espressa tramite una parametrizzazione a 3 variabili (si veda la sezione A) mentre il vettore $\mathbf{t}$ deve venire rappresentato tramite una parametrizzazione a 2 (la scala rimane infatti sempre un fattore sconosciuto). Inserendo i vincoli (9.69) nell'equazione (9.67), l'obiettivo di ricavare la matrice Essenziale viene trasformato in quello di ricavare direttamente i parametri relativi tra i due sensori. Se infine richiesta, ottenuta la posa relativa tra i sensori, è possibile ottenere la matrice Essenziale applicando direttamente la definizione (9.41).

Quando i parametri intrinseci non sono disponibili, nel caso della stima della matrice Fondamentale, non è possibile effettuare una vera ricostruzione tridimensionale della scena proprio per la mancanza di questi parametri. È possibile sfruttare tuttavia delle proiezioni prospettiche fittizie ponendo $\mathbf{K}_1 = \mathbf{I}$ e ottenendo dei vincoli del tipo:

$$\begin{aligned} \hat{\mathbf{p}}_{1,i} &\equiv \hat{\mathbf{x}}_i \\ \hat{\mathbf{p}}_{2,i} &\equiv \mathbf{P}\hat{\mathbf{x}}_i \end{aligned} \tag{9.70}$$

usando direttamente la variabile ausiliaria $\hat{\mathbf{x}}_i$, coordinata che verrà pertanto conosciuta a meno di una trasformazione affine $\mathbf{K}_1^{-1}$ ovvero i parametri intrinseci della camera 1.

Inserendo i vincoli (9.70) nell'equazione (9.67) anche questa volta l'obiettivo di ricavare la matrice Fondamentale viene trasformato in quello di ricavare i parametri della matrice proiettiva $\mathbf{P}$. Attraverso la matrice camera $\mathbf{P} = [\mathbf{R}'|\mathbf{t}']$, matrice camera fittizia, è infine possibile ricavare $\mathbf{F}$ applicando direttamente la definizione (9.41) dove tuttavia la matrice $\mathbf{R}'$ non è una matrice di rotazione.

La stima a massima Verosimiglianza della matrice Fondamentale, corretta dal punto di vista probabilistico, richiede comunque una gran quantità di risorse: oltre alle 12 incognite globali necessarie per stimare $\mathbf{P}$ (rispetto alle 5 della matrice Essenziale) per ogni coppia di punti da minimizzare vengono inserite nel problema 3 ulteriori incognite.

Infine, come avvertenza finale, per la stima ottima delle matrici in presenza di eventuali *outlier* nella scena sono ampiamente sfruttate tecniche come RANSAC (sezione 3.12).

9.4.3 Fattorizzazione della Matrice Essenziale

Nelle sezioni precedenti si è visto come, sfruttando almeno 5 corrispondenze tra punti omologhi tra due immagini, è possibile ottenere la matrice Essenziale che codifica la posa relativa tra le due camere. La matrice Essenziale può essere nuovamente fattorizzata in rotazione e traslazione. In questo modo è possibile ottenere i parametri relativi delle camere coinvolte e, attraverso questa informazione, riuscire ad eseguire una ricostruzione tridimensionale della scena osservata.

Come suggerito da Trivedi, dalla definizione di matrice essenziale (9.41) è facile mostrare che la matrice simmetrica $\mathbf{E}\mathbf{E}^\top$ è indipendente dal vettore rotazione:

$$\mathbf{E}\mathbf{E}^\top = [\mathbf{t}]_\times [\mathbf{t}]_\times^\top = \begin{bmatrix} t_y^2 + t_z^2 & -t_x t_y & -t_x t_z \\ -t_y t_x & t_z^2 + t_x^2 & -t_y t_z \\ -t_z t_x & -t_z t_y & t_x^2 + t_y^2 \end{bmatrix} \tag{9.71}$$

Dalla matrice $\mathbf{E}\mathbf{E}^\top$ si può ricavare il vettore traslazione $\mathbf{t}$, ricordando sempre che tale vettore è conosciuto a meno di un fattore moltiplicativo (e pertanto di segno), con cui poi ricavare $\mathbf{R}$.

La matrice Essenziale può essere anche fattorizzata direttamente attraverso la Decomposizione a Valori Singolari. Sia $\mathbf{U}\mathbf{D}\mathbf{V}^\top$, dove $\mathbf{D} = \text{diag}(1, 1, 0)$, la SVD di $\mathbf{E}$ (se così non fosse, è comunque possibile proiettare la matrice $\mathbf{E}$ nello spazio delle matrici Essenziali, come descritto in sezione 9.4.1). Attraverso questa decomposizione si possono estrarre i componenti generatori di $\mathbf{E}$:

$$[\mathbf{t}]_\times = \mathbf{U}\left(\mathbf{R}_z^\top \mathbf{D}\right)\mathbf{U}^\top \quad \mathbf{R} = \mathbf{U}\mathbf{R}_z\mathbf{V}^\top | \mathbf{U}\mathbf{R}_z^\top\mathbf{V}^\top \tag{9.72}$$

dove

$$\mathbf{R}_z^\top \mathbf{D} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{R}_z = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.73}$$

con $\mathbf{R}_z$ rotazione intorno all'asse z di un angolo di $\frac{\pi}{2}$. È da notare che $[\mathbf{t}]_\times \mathbf{t} = 0$ per ogni possibile $\mathbf{t}$. Si può dimostrare che questo è possibile solo quando $\mathbf{t} = \mathbf{U}(0, 0, 1)^\top = \mathbf{u}_3$, ultima colonna della matrice $\mathbf{U}$.

La matrice di rotazione $\mathbf{R}$ presenta così due possibili soluzioni ruotate di 180° tra loro rispetto all'asse che congiunge i due pin-hole. Siccome il vettore $\mathbf{t}$ è conosciuto a meno di un fattore moltiplicativo e il vincolo $|\mathbf{t}| = 1$ non permette di ricavare il segno della traslazione, esistono anche due ulteriori alternative per la fattorizzazione dovute a una ambiguità sul segno che può assumere $\mathbf{t}$. Esistono pertanto 4 differenti fattorizzazioni, tutte plausibili, di una matrice Essenziale e fra queste va scelta quella che proietta tutti i punti (o la maggior parte) frontalmente rispetto ad entrambe le camere.

9.4.4 Chiralità e ricostruzione con pose relative

Dalla decomposizione della matrice Essenziale, a meno di un fattore moltiplicativo, esistono pertanto 4 possibili configurazioni (le due matrici di rotazione e gli associati vettori di traslazione) che ricombinate permettono di ottenere nuovamente la matrice Essenziale di origine. Per determinare quale decomposizione è quella corretta l'unico modo è trovare la configurazione che ricostruisce la maggioranza dei punti tridimensionali in maniera opportuna ovvero, più semplicemente, la configurazione che porta la maggioranza dei punti ad avere la coordinata camera z ad essere positiva.

Sia $(\mathbf{R}, \mathbf{t})$ una decomposizione della matrice Essenziale, siano $\mathbf{m}_1$, $\mathbf{m}_2$ le coordinate camera di due punti omologhi, e si definiscano $\tilde{\mathbf{m}}_1 = (\tilde{u}_1, \tilde{v}_1, 1)$ e $\tilde{\mathbf{m}}_2 = (\tilde{u}_2, \tilde{v}_2, 1)$ tali che

$$\tilde{\mathbf{m}}_1 = 1/z_1 \mathbf{m}_1 \qquad \tilde{\mathbf{m}}_2 = 1/z_2 \mathbf{m}_2 \tag{9.74}$$

coordinate camera normalizzate di una coppia di punti omologhi, ovvero

$$\tilde{u}_1 = x_1/z_1 \qquad \tilde{v}_1 = y_1/z_1 \qquad \tilde{u}_2 = x_2/z_2 \qquad \tilde{v}_2 = y_2/z_2 \tag{9.75}$$

L'obiettivo è quello di ricavare le coordinate z_1 e z_2 attraverso le quali, valutandone la positività, si può supporre che i punti omologhi siano frontali rispetto all'osservatore e da questo dedurre la correttezza della decomposizione della matrice Essenziale. Sfruttando il formalismo (9.74), l'equazione (9.38) diventa

$$z_2 \tilde{\mathbf{m}}_2 = z_1 \mathbf{R} \tilde{\mathbf{m}}_1 + \mathbf{t} \tag{9.76}$$

Svolendo l'equazione con z_1 come incognita si ottiene infine

$$z_1 = \frac{t_x - \tilde{u}_2 t_z}{(\tilde{u}_2 \mathbf{r}_3 - \mathbf{r}_1) \cdot \tilde{\mathbf{m}}_1} = \frac{t_y - \tilde{v}_2 t_z}{(\tilde{v}_2 \mathbf{r}_3 - \mathbf{r}_2) \cdot \tilde{\mathbf{m}}_1} \tag{9.77}$$

avendo indicato con $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ le 3 righe della matrice $\mathbf{R}$. La prima equazione sfrutta la coordinata $\tilde{u}_2$ per ricavare z_1 mentre la seconda sfrutta la coordinata $\tilde{v}_2$. In questo modo si ottiene la coordinata $\mathbf{m}_1$ da cui ottenere attraverso l'equazione (9.38) immediatamente $\mathbf{m}_2$ e in particolare

$$z_2 = \mathbf{r}_3 \cdot \mathbf{m}_1 + t_z \tag{9.78}$$

per poter valutare la frontalità dell'altro elemento della coppia.

È da notare che la soluzione del problema si poteva ottenere risolvendo direttamente il sistema (9.76) come se fosse un sistema lineare sovradimensionato di 2 incognite in 3 equazioni (approccio simile a quello che si è visto in sezione 9.3.1).

In entrambi i casi viene ottimizzata una quantità algebrica e pertanto non sarà la stima alla massima verosimiglianza del punto tridimensionale: diversamente dagli algoritmi discussi in sezione 9.3.1, questo è un approccio in effetti poco adatto per ricavare le coordinate mondo precise ma sufficiente per verificare che la scelta della decomposizione sia quella corretta.

Va sempre ricordato che essendo il vettore $\mathbf{t}$ estratto dalla matrice Essenziale conosciuto a meno di un fattore moltiplicativo i punti così stimati sono conosciuti a meno di un fattore moltiplicativo.

É da notare che questo discorso è chiaramente generico e può essere applicato al caso di ricostruzione tridimensionale conoscendo la posa relativa tra sensori.

9.5 Rimozione rumore sotto vincoli epipolari

Come si è visto in sezione 9.3.1, triangolare punti affetti da rumore porta a rette non incidenti la cui intersezione non minimizza il residuo in coordinate immagine (per esempio sotto la metrica della distanza euclidea). Abbiamo anche visto che la miglior stima dei punti non affetti da rumore minimizza la quantità di equazione 9.67 sotto il vincolo epipolare 9.68. Finora però, data la conoscenza della matrice Essenziale/Fondamentale, questa minimizzazione ha richiesto il punto tridimensionale come variabile ausiliaria e una tecnica (iterativa) di ottimizzazione inizializzata per esempio sfruttando la triangolazione con rette sghembe dei punti affetti da rumore.

Esiste una tecnica globale non lineare che permette di ottenere la triangolazione ottima (la stima dei punti immagine) attraverso un metodo polinomiale [HS97] che richiede di trovare le radici di un polinomio di 6° grado. Come più chiaramente discusso in [Lin10], la triangolazione ottima può essere vista come il seguente problema di minimizzazione:

$$\min \left(\delta\mathbf{m}_2^\top \delta\mathbf{m}_2 + \delta\mathbf{m}_1^\top \delta\mathbf{m}_1 \right) \tag{9.79}$$

soggetto al vincolo epipolare

$$\hat{\mathbf{m}}_2^\top \mathbf{E} \hat{\mathbf{m}}_1 = (\mathbf{m}_2 - \mathbf{S}^\top \delta\mathbf{m}_2)^\top \mathbf{E} (\mathbf{m}_1 - \mathbf{S}^\top \delta\mathbf{m}_1) = 0 \tag{9.80}$$

avendo definito $\delta\mathbf{m}_1 = \mathbf{S}(\mathbf{m}_1 - \hat{\mathbf{m}}_1)$ e $\delta\mathbf{m}_2 = \mathbf{S}(\mathbf{m}_2 - \hat{\mathbf{m}}_2)$ dove

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{9.81}$$

serve ad estrarre le sole componenti non omogenee dal vettore. Come visto in precedenza i punti $(\hat{\mathbf{m}}_1, \hat{\mathbf{m}}_2)$ sono la stima dei punti non affetti da rumore mentre $(\mathbf{m}_1, \mathbf{m}_2)$ sono i punti osservati.

Questo problema di minimizzazione vincolata si può risolvere attraverso l'ausilio dei moltiplicatori di Lagrange:

$$\mathcal{L}(\delta\mathbf{m}_1, \delta\mathbf{m}_2, \lambda) = \delta\mathbf{m}_1^\top \delta\mathbf{m}_1 + \delta\mathbf{m}_2^\top \delta\mathbf{m}_2 - 2\lambda (\mathbf{m}_2 - \mathbf{S}^\top \delta\mathbf{m}_2)^\top \mathbf{E} (\mathbf{m}_1 - \mathbf{S}^\top \delta\mathbf{m}_1) \tag{9.82}$$

Il gradiente della Lagrangiana si annulla in

$$\begin{aligned} &\hat{\mathbf{m}}_2^\top \mathbf{E} \hat{\mathbf{m}}_1 = (\mathbf{m}_2 - \mathbf{S}^\top \delta\mathbf{m}_2)^\top \mathbf{E} (\mathbf{m}_1 - \mathbf{S}^\top \delta\mathbf{m}_1) = 0 \\ &\delta\mathbf{m}_1 = \lambda \mathbf{S}\mathbf{E}^\top \left(\mathbf{m}_2 - \mathbf{S}^\top \delta\mathbf{m}_2 \right) = \lambda \mathbf{S}\mathbf{E}^\top \hat{\mathbf{m}}_2 = \lambda \mathbf{n}_1 \\ &\delta\mathbf{m}_2 = \lambda \mathbf{S}\mathbf{E} \left(\mathbf{m}_1 - \mathbf{S}^\top \delta\mathbf{m}_1 \right) = \lambda \mathbf{S}\mathbf{E}\hat{\mathbf{m}}_1 = \lambda \mathbf{n}_2 \end{aligned} \tag{9.83}$$

da cui si ottengono 5 vincoli in 5 incognite (le coordinate differenziali e λ). Tali vincoli si possono parametrizzare in funzione di una variabile ausiliaria dalla quale ottenere la famosa equazione di grado 6. Tale approccio è esattamente il medesimo sia se si usano punti immagine e la matrice Fondamentale che punti camera con la matrice Essenziale.

Sempre in [Lin10] vengono proposte anche tecniche sub-ottime, iterative, dal basso costo computazionale, dove ad ogni iterazione il vincolo epipolare viene comunque soddisfatto.

Ottenuti i punti immagine non affetti da rumore, con qualsiasi tecnica di triangolazione (rette sghembe di sezione 1.5.6 o la DLT di sezione 9.3.1) è possibile ricavare il punto tridimensionale. Una formulazione alternativa [KK95], dati due punti omologhi espressi in coordinate camera $\hat{\mathbf{m}}'$ e $\hat{\mathbf{m}}$ il punto tridimensionale formato dall'intersezione dei raggi ottici è

$$\mathbf{x} = \frac{(\mathbf{t} \times \mathbf{R}\hat{\mathbf{m}}') \cdot \mathbf{z}}{\|\mathbf{z}\|^2} \tag{9.84}$$

dove $\mathbf{z} = \hat{\mathbf{m}} \times \mathbf{R}\hat{\mathbf{m}}'$.

9.6 Punti omologhi in spazi immagine alternativi

Attraverso una opportuna *Warp-Table* è possibile trasformare l'immagine in ingresso in una forma alternativa che preservi comunque la possibilità di ricostruire tridimensionalmente la scena. Volendo preservare anche il concetto di disparità, ovvero avere punti omologhi lungo la stessa coordinata verticale tra le due immagini della coppia stereoscopica, sono necessari alcuni vincoli ulteriori: è necessario che la coordinata immagine orizzontale sia funzione della coordinata x camera mentre la coordinata verticale dell'immagine non sia funzione della coordinata x. La funzione della coordinata orizzontale deve essere monotona (normalmente crescente) in x mentre la funzione della coordinata verticale monotona crescente in y.

Una parametrizzazione diffusa è quella polare dove, per evitare commistioni tra gli assi, si è scelta come equazione

$$\begin{aligned} x' &= \operatorname{atan} \frac{x}{\sqrt{y^2+z^2}} \\ y' &= \operatorname{atan} \frac{y}{z} \end{aligned} \tag{9.85}$$

che proietta un punto camera (x, y, z) su un punto immagine (x', y') (per confronto ricordo che l'equazione prospettica ha come equazioni $x' = x/z$, $y' = y/z$). Data questa proiezione è possibile scrivere l'equazione inversa come

$$\begin{aligned} x &= \sin(x') \\ y &= \cos(x')\sin(y') \\ z &= \cos(x')\cos(y') \end{aligned} \tag{9.86}$$

Questa parametrizzazione permette di proiettare in uno spazio immagine tutte le coordinate di una semisfera (fino a 180 gradi) cosa che in effetti il modello pin-hole non permette. Questa parametrizzazione pertanto è comoda per rimappare camere Fish-Eye. Attraverso questa parametrizzazione è possibile scrivere una equazione simile a quella di (9.21) per triangolare due punti immagine.

9.7 Odometria Visuale e Bundle Adjustment

La *Visual Odometry* si pone come obiettivo quello di ricavare la posa relativa che ha assunto una camera (o una coppia stereoscopica) che si muove nello spazio analizzando due immagini in sequenza. Il problema dell'odometria visuale per una sola telecamera si risolve normalmente con il calcolo della matrice essenziale e la sua successiva decomposizione. In questo caso, come già indicato in precedenza, non è possibile conoscere la scala del movimento, ma solo mettere in relazione tra loro i vari movimenti. Discorso differente nel caso in cui si ha a disposizione una coppia stereoscopica.

Data una serie di osservazioni temporali di punti mondo ricavati dalla ricostruzione tridimensionale $(\mathbf{x}_i, \mathbf{x}'_i)$ è possibile ricavare in maniera lineare una trasformazione di rototraslazione $(\mathbf{R}, \mathbf{t})$ che trasforma i punti del mondo all'istante di tempo t all'istante di tempo t' in modo da poter essere espressi con una equazione del tipo:

$$\mathbf{x}'_i = \mathbf{R}\mathbf{x}_i + \mathbf{t} \tag{9.87}$$

Tale approccio è generale e non dipende dal particolare sensore utilizzato per ricavare i punti.

La rototraslazione eseguita dalla coppia di sensori può essere ricavata minimizzando la quantità:

$$\sum_i \|\mathbf{x}'_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}\|^2 \tag{9.88}$$

La soluzione a 12 parametri, lineare da dati sovradimensionati, troverà un minimo assoluto ma non è lo stimatore ottimo, in quanto minimizza una quantità algebrica e in ogni caso non garantisce che la matrice di rotazione sia ortonormale. Partendo dalla soluzione lineare, l'utilizzo di un minimizzatore non-lineare (per esempio Levenberg-Marquardt, sezione 3.3.6) sulla funzione costo di equazione (9.88) permette di determinare i 6 parametri (3 rotazioni e 3 traslazioni) in modo più preciso. Questa è algoritmo è indicato come *3D-to-3D* perchxE9 ricava il movimento partendo da coppie di punti tridimensionali. Come alternativa alla soluzione lineare è possibile anche una soluzione in forma chiusa [Hor87].

L'approccio mostrato ora è generale ma mal si adatta al caso di punti mondo ottenuti da una ricostruzione tridimensionale da immagini. La funzione costo mostrata, infatti, ottimizza quantità in coordinate mondo e non in coordinate immagine: il rumore sui punti dell'immagine si propaga in maniera non lineare durante la fase di triangolazione e perciò solo in coordinate immagine è possibile supporre che il rumore di individuazione dei punti sia gaussiano a media nulla. Non è pertanto possibile realizzare uno stimatore a massima verosimiglianza sfruttando solamente i punti in coordinate mondo. Un approccio più raffinato è quello indicato come *3D-to-2D* dove si cerca di minimizzare la riproiezione di un punto del passato in coordinate immagine:

$$\sum_i \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|^2 \tag{9.89}$$

dove $\hat{\mathbf{p}_1}$ è la proiezione, rototraslata, del punto tridimensionale $\mathbf{x}_i$ ottenuto dal fotogramma precedente. Questo problema è anche conosciuto come *perspective from n points* (*PnP*) in quanto molto simile al problema già visto in precedenza della calibrazione di una camera in ambiente statico.

Chiaramente anche questo approccio è inficiato dal fatto che il punto tridimensionale $\mathbf{x}_i$ non è un dato del problema ma è conosciuto con una certa quantità di errore. Per questa ragione è necessario fare un ulteriore passo minimizzando entrambi gli errori in coordinate immagine (è la *Maximum Likelihood Estimation*):

$$\sum_i \|\mathbf{p}_1 - \hat{\mathbf{p}_1}\|^2 + \|\mathbf{p}_2 - \hat{\mathbf{p}_2}\|^2 + \|\mathbf{p}'_1 - \hat{\mathbf{p}'_1}\|^2 + \|\mathbf{p}'_2 - \hat{\mathbf{p}'_2}\|^2 \tag{9.90}$$

avendo imposto $\hat{\mathbf{p}_1} = \mathbf{K}_1\mathbf{R}_1(\hat{\mathbf{x}}_i - \mathbf{t}_1)$, $\hat{\mathbf{p}_2} = \mathbf{K}_2\mathbf{R}_2(\hat{\mathbf{x}}_i - \mathbf{t}_2)$, $\hat{\mathbf{p}'_1} = \mathbf{K}_1\mathbf{R}_1(\hat{\mathbf{x}'}_i - \mathbf{t}_1)$ e $\hat{\mathbf{p}'_2} = \mathbf{K}_2\mathbf{R}_2(\hat{\mathbf{x}'}_i - \mathbf{t}_2)$ a cui va aggiunto il vincolo di equazione (9.87), mantenendo l'incognita sull'effettiva posizione del punto $\hat{\mathbf{x}}_i$ nei due sistemi di riferimento. In questo modo viene sia minimizzato lo spostamento che eseguono le camere, sia la coordinata tridimensionale di ogni singola *feature* nel mondo. Anche in questo caso la soluzione alla massima verosimiglianza richiede di risolvere un problema non lineare di dimensioni notevoli. Nel caso di una coppia stereo rettificata, la funzione costo può essere di molto semplificata.

L'odometria visuale è un algoritmo di dead-reckoning e pertanto è affetto da deriva. È possibile estendere questi ragionamenti al caso in cui non siano solo due gli istanti di tempo coinvolti nella minimizzazione ma molteplici. In questo caso si entra in un discorso complicato per cercare di ridurre il più possibile gli errori di deriva nel comporre le diverse trasformazioni. Un tutorial che affronta queste tematiche è [SF11].

Quando si vuole affrontare il problema dal punto di vista bayesiano, sfruttando l'equazione (9.90), e si intendono processare contemporaneamente tutti i fotogrammi, invece che odometria visuale si preferisce parlare di *Bundle Adjustment*.

Il concetto di *Bundle Adjustment*, introdotto dalla fotogrammetria e poi acquisito dalla *Computer Vision* (si veda l'ottimo survey [TMHF00]), indica una minimizzazione multivariabile in modo da ottenere contemporaneamente una ricostruzione tridimensionale, le pose relative della camere in una sequenza di immagini ed eventualmente i parametri intrinseci delle camere stesse.

Si tratta di una estensione alle tecniche non-lineari che stimano i parametri attraverso la minimizzazione di una funzione di costo adeguata basata sugli errori di riproiezione dei punti individuati, nella stessa forma di equazione (9.90).

Siccome la stessa *feature* può essere vista da diverse immagini, il processo di stima condiziona tutte le pose e di conseguenza il problema non si può scomporre in n problemi separati di odometria visuale: tutte le immagini della sequenza devono essere minimizzate contemporaneamente. Per questo motivo il problema della *Bundle Adjustment* è un problema dimensionalmente elevato, sicuramente non-convesso, che richiede una ottimizzazione non semplice e fa ricorso a minimizzazione sparsa per preservare memoria e migliorare la precisione.

Un approccio alternativo al *Bundle Adjustment*, sicuramente non il miglior stimatore alla massima verosimiglianza ma che introduce un numero minore di incognite, è quello del *Pose Graph Optimization* [GKSB10] che, sfruttando informazione della medesima posa ottenuta da più percorsi ovvero avendo individuato dei *Loop*, permette di ottimizzare solamente le pose rispetto a quelle ottenute dall'odometria visuale. Sia $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ un vettore di parametri dove l'elemento $\mathbf{x}_i$ rappresenta la posa del nodo i-esimo. Siano $\mathbf{z}_{ij}$ e $\mathbf{\Omega}_{ij}$ la misura e la matrice di precisione dell'osservazione *virtuale* della posa relativa tra i nodi i e j. L'obiettivo è ottenere una stima dei parametri $\mathbf{x}$ date le osservazioni virtuali $\mathbf{z}_{ij}$. Siccome le pose relative vengono ottenute come confronto di due pose assolute, parametri da ottenere, si può definire la funzione costo

$$\mathbf{e}_{ij} = \mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}(\mathbf{x}_i, \mathbf{x}_j) \tag{9.91}$$

misura dell'errore tra la posa relativa virtuale misurata $\mathbf{z}_{ij}$ e quella predetta $\hat{\mathbf{z}}_{ij}(\mathbf{x}_i, \mathbf{x}_j)$ date le configurazioni $\mathbf{x}_i$ e $\mathbf{x}_j$ da valutare rispettivamente per i nodi i e j. Sfruttando l'informazione sulla precisione della stima della singola posa relativa è possibile definire una funzione costo globale

$$F(\mathbf{x}) = \sum_{<i,j>} \mathbf{e}_{ij}^\top \mathbf{\Omega}_{ij} \mathbf{e}_{ij} \tag{9.92}$$

di fatto somma delle singole distanze di Mahalanobis tra tutte le coppie (i, j) su cui è stata fatta una misura di posa relativa. La funzione $F(\mathbf{x})$, minimizzata rispetto a $\mathbf{x}$, fornisce la miglior stima delle pose assolute del problema, tutto senza coinvolgere i singoli elementi di cui è composta la singola osservazione reale.

9.8 Ricostruzione, rappresentazione e disegno di ambienti tridimensionali

Le nuvole di punti (*pointclouds*) o *mesh* sono le primitive tridimensionali più diffuse per la rappresentazione in memoria di ambienti e oggetti. Questi approcci avevano il pregio o il difetto di separare nettamente la parte di ricostruzione da quella di rendering.

Recentemente sia i metodi *Neural Radiance Field* (NeRF) [MST+20] ma soprattutto i *3D Gaussian Splatting* hanno dato una notevole spinta in questo settore uniformando la parte di ricostruzione con quella di rendering.

9.8.1 Armoniche Sferiche

Un punto nodale della rappresentazione del colore viene dall'uso delle Armoniche Sferiche (*Spherical Harmonics SH*). Le armoniche sferiche sono soluzioni dell'equazione di Laplace in coordinate sferiche, ortogonali e formanti una base completa

per le funzioni definite su una sfera, ovvero qualsiasi funzione $L(\theta, \phi)$ può essere espansa in una serie di armoniche sferiche:

$$L(\mathbf{d}) = L(\theta, \phi) = \sum_{l=0} \sum_{m=-l}^{m=l} k_l^m Y_l^m (\theta, \phi) \tag{9.93}$$

Tutta questa classe di funzioni può essere generata da una unica formula, scegliendo con $l \geq 0$ per il grado dell'armonica e per l'ordine $-l \leq m \leq l$:

$$Y_l^m(\theta, \phi) = \frac{(-1)^l}{2^l l!} \sqrt{\frac{(2l+1)(l+m)!}{4\pi(l-m)!}} e^{im\phi} P_l^m (\cos\theta) \tag{9.94}$$

dove $P_l^m (\cos\theta)$ sono i polinomi di Legendre associati [YLT+21]. Nel caso di $l = 0$, la prima armonica è una costante sulla sfera e vale $Y_0^0 = \frac{1}{2}\sqrt{\frac{1}{\pi}} \approx 0.282$.

In grafica computazionale sono usate per rappresentare l'informazione sull'illuminazione in modo compatto ed efficiente. Le Armoniche Sferiche decompongono la luce incidente in un insieme di coefficienti, ognuno associato ad una armonica differente. Questi coefficienti catturano le caratteristiche della luce, come intensità e colore, lungo differenti direzioni della superficie sferica.

L'idea è quella di selezionare un grado massimo di l e ogni componente del colore (rossa, verde, blue) scriverla come combinazione lineare delle armoniche sferiche usando come parametro (θ, ϕ) il raggio ottico che collega il punto all'osservatore.

9.8.2 Rendering

Per molti aspetti i point-based α-blending, i rendering volumetrici *NeRF-style* e i *Gaussian Splatting* condividono la parte di basso livello per disegnare la scena: il colore in un pixel dell'immagine viene approssimato integrando i campioni lungo il raggio che attraversa questo pixel. Il colore finale è una somma ponderata dei colori dei punti 3D campionati lungo questo raggio, ponderata in base alla trasmittanza.

Il colore C di un pixel è l'integrale delle varie densità che si incontrano lungo il raggio ottico $\mathbf{r} = \mathbf{o} + t\mathbf{d}$ nell'intervallo $[t_n, t_f]$:

$$C = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\,\mathbf{c}(\mathbf{r}(t), \mathbf{d})\,dt \tag{9.95}$$

dove

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))\,ds\right) \tag{9.96}$$

La funzione $T(t)$ denota la trasmittanza accumulata lungo il raggio da da t_n a t, cioè la probabilità che il raggio viaggi da t_n a t senza colpire qualsiasi altra particella.

Gli integrali possono essere trasformati in sommatorie dei vari contributi. Il colore C di un pixel può essere visto pertanto come sommatoria di vari contributi:

$$C = \sum_{i=1}^{N} T_i \left(1 - \exp(-\sigma_i \delta_i)\right) \mathbf{c}_i \tag{9.97}$$

dove $T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$ dove $\delta_i = t_{i+1} - t_i$.

Questa rappresentazione si riduce al classico α-blending usando $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$. In questo modo anche la trasmittanza si può scrivere come $T_i = \prod_{j=1}^{i-1}(1 - \alpha_i)$.

Nei tipici approcci basato sui punti neurali, il colore C di un pixel si ricava fondendo gli $\mathcal{N}$ punti ordinati che sottendono il pixel stesso:

$$C = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{9.98}$$

9.8.3 Rappresentazione neurale e campi di radianza

9.8.4 Splattering 3D di Gaussiane

L'idea dello splattering 3D di Gaussiane è quello di rappresentare l'immagine come miscela di gaussiane tridimensionali. Le gaussiane 3D si basano sulla estensione tridimensionale delle gaussiane monodimensionali. Le gaussiane tridimensionali sono definite da una matrice di covarianza Σ (in coordinate mondo) e centrate nel punto (media) μ:

$$G(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1} (\mathbf{x}-\mu)} \tag{9.99}$$

Per essere disegnata questa gaussiana deve prima essere trasportata in coordinate camera attraverso una rototraslazione $\mathbf{W}$ e infine proiettata in coordinate immagine. Tuttavia si può pensare ad una approssimazione, disegnando una gaussiana bidimensionale nello spazio immagine. In spazio 2D la covarianza Σ' diventa

$$\Sigma' = \mathbf{JW}\Sigma\mathbf{W}^\top\mathbf{J}^\top \tag{9.100}$$

dove $\mathbf{W}$ è la sola parte rotazionale della trasformazione e usando, come approssimazione, il Jacobiano $\mathbf{J}$ della proiezione prospettica calcolato nel punto rototraslato in camera $(x, y, z)^\top$. Per esempio nel caso di proiezione camera pinhole:

$$\mathbf{J} = \begin{bmatrix} k_u/z & 0 & -\frac{k_u x}{z^2} \\ 0 & k_v/z & -\frac{k_v y}{z^2} \end{bmatrix} \tag{9.101}$$

La matrice Σ' pertanto ha dimensionalità 2×2 [ZPvBG01] ed è equiparabile alla matrice di ad una gaussiana 2D.

In [KKLD23] si fa un passo ulteriore: siccome risulta difficile parametrizzare una matrice di covarianza (semi definita positiva) si parte dal fatto che la matrice Σ rappresenta un ellissoide e pertanto si può avere una minima parametrizzazione invece di usare tutti i termini della matrice come incognita. L'idea infatti è quella di usare una matrice di scala $\mathbf{S}$ (3 DOF) e una matrice di rotazione $\mathbf{R}$ (altri 3 DOF ma normalmente rappresentata da un quaternione, vedi sezione A.3):

$$\Sigma = \mathbf{RSS}^\top\mathbf{R}^\top \tag{9.102}$$

parametrizzando pertanto ogni gaussiana con 6 DOF. Da notare che $\mathbf{SS}^\top = \text{diag}\left(s_x^2, s_y^2, s_z^2\right)$.

Associato infine ad ogni punto ci può essere un colore RGB o delle armoniche sferiche (*Spherical Harmonics SH*) oltre ovviamente al parametro di opacità α simile a quello di NeRF. Dal punto di vista pratico le gaussiane sono renderizzate dalle più vicine alle più lontane fino a saturazione dell'opacità.

9.8.5 Splattering 2D di Gaussiane

Lo Splattering di Gaussiane 2D può essere visto come alternativa più semplice alle Gaussiane 3D e da un punto di vista storico erano già state introdotte precedententemente.

Le gaussiane 2D sono rappresentate da un punto centrale $\mathbf{p}_k$, da due vettori unitari tangenti $(\mathbf{t}_u, \mathbf{t}_v)$ e un fattore di scala $\mathbf{S} = (s_u, s_v)$ che controlla la varianza in 2 dimensioni della gaussiana.

Si può organizzare l'orientazione della gaussiana 2D in una matrice 3×3 di rotazione $\mathbf{R} = [t_u, t_v, t_w]$ (parametrizzabile come classica rotazione in 3D) avendo definito $t_w = t_u \times t_v$ e i fattori di scala in una matrice diagonale $\mathbf{S} = \text{diag}(s_u, s_v, 0)$.

La gaussiana 2D Ã¨ pertanto definita su un piano tangente di equazione

$$P(u, v) = \mathbf{p}_k + s_u\mathbf{t}_u u + s_v\mathbf{t}_v v = \mathbf{H}(u, v, 1, 1)^\top \tag{9.103}$$

avendo definito la matrice omografica $\mathbf{H} = \begin{bmatrix} s_u\mathbf{t}_u & s_v\mathbf{t}_v & 0 & \mathbf{p}_k \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{RS} & \mathbf{p}_k \\ 0 & 1 \end{bmatrix}$. Ad ogni punto (u, v) in coordinate del piano è chiaramente associata una gaussiana di equazione $e^{\frac{u^2+v^2}{2}}$.

Appendice A

Matrici di Rotazione

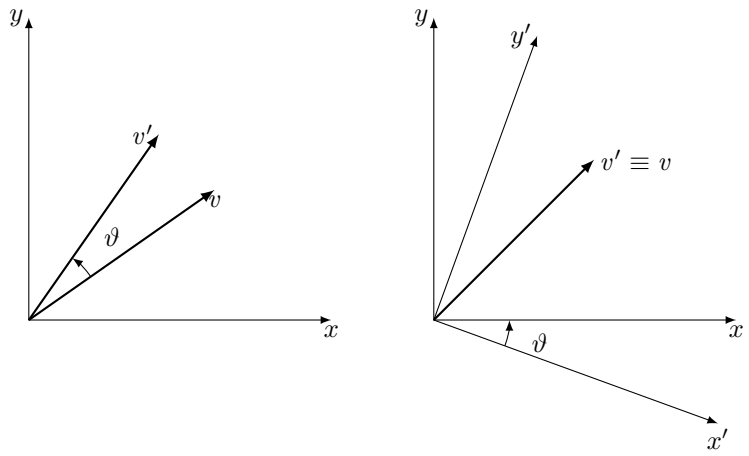


Figura A.1: Rappresentazione dell'applicazione di una rotazione ad un vettore e a un sistema di assi: a sinistra una *Inner/Active/Alibi Transformation*, a destra una *Outer/Passive/Alias Transformation*.

Le rotazioni sono trasformazione isometriche dello spazio euclideo ovvero che trasformano vettori preservandone la lunghezza e lasciano un luogo di punti inalterato nello spazio pari ad un iperpiano (il *centro di rotazione* nel caso bidimensionale o un *asse di rotazione* nel caso tridimensionale).

L'insieme di tutte le Matrici di Rotazione $SO(n)$ in $\mathbb{R}^n$ è definito come *Speciale Ortogonale*

$$SO(n) = \left\{\mathbf{R} \in \mathbb{R}^n : \mathbf{R}\mathbf{R}^\top = \mathbf{R}^\top\mathbf{R} = \mathbf{I}, \det \mathbf{R} = +1\right\} \tag{A.1}$$

ovvero $\mathbf{R}^{-1} = \mathbf{R}^\top$.

Esistono due possibili convenzioni per definire una matrice di rotazione: alcuni autori preferiscono scrivere la matrice che trasforma da coordinate sensore a coordinate mondo, altri invece l'opposto. La matrice stessa di rotazione ha la duplice veste di indicare una rotazione all'interno di un sistema di riferimento (*Active* o *Alibi*), o la trasformazione di coordinate da un sistema di riferimento a un secondo sistema di riferimento (*Passive* o *Alias*).

In questo libro le matrici sono prevalentemente usate per rappresentare cambi di base e, quando possibile, il sistema di riferimento sorgente e quello di destinazione sono ben evidenziati.

Per discutere delle matrici di rotazione e fare qualche considerazione interessante risulta comodo partire ad analizzare il caso bidimensionale, schematizzato in figura A.1.

Si può verificare che $SO(2)$ ha un solo grado di libertà. La matrice $\mathbf{R}_\vartheta$, che rappresenta una rotazione bidimensionale, può essere scritta nella forma

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{bmatrix} \tag{A.2}$$

Come si può vedere dalla figura A.1 quando si parla di una rotazione di un angolo ϑ la stessa trasformazione può essere vista in modi differenti, a seconda di quale sistema di riferimento l'osservatore si posizioni solidale. La matrice $\mathbf{R}_\theta$ permette di ruotare un vettore in senso antiorario (rispetto all'origine del sistema di riferimento) di un angolo ϑ (figura a sinistra in A.1) [1]. La matrice di forma (A.2) oltre che a ruotare un vettore in senso antiorario permette anche di ottenere le cosidette coordinate "mondo" di un punto conoscendo la coordinate "sensore" e sapendo che tale sensore è ruotato di un angolo ϑ

[1] Come già accennato all'inizio, bisogna fare attenzione perché la trasformazione inversa/trasposta, ovvero la matrice generata dall'angolo $-\vartheta$, in letteratura può essere indicata come "matrice di rotazione" e indicata anch'essa con la lettera $\mathbf{R}$.

(legge della mano destra) nel sistema di riferimento "mondo". La matrice (A.2) permette perciò di passare da coordinate "sensore" a coordinate "mondo", mentre l'inversa di questa matrice permette di passare da coordinate "mondo" a coordinate "sensore".

La distinzione tra *Inner/Active/Alibi Transformation* e *Outer/Passive/Alias Transformation* è un altro modo per descrivere la differenza tra rotazioni. Questi termini sono spesso utilizzati in contesti matematici e fisici per chiarire se una trasformazione agisce sul sistema di riferimento stesso (il sistema di riferimento viene ruotato o traslato, mentre gli oggetti rimangono fissi nello spazio, perciò *alias* o *passive*) o sugli oggetti all'interno di un sistema di riferimento fisso (gli oggetti vengono ruotati o traslati, mentre il sistema di riferimento rimane invariato, perciò *alibi* o *active*).

La matrice di rotazione viene anche chiamata Matrice dei Coseni Direzionali (*Direction Cosine Matrix*, *DCM*) in quanto le colonne della matrice di trasformazione corrispondono alle matrici dei coefficienti dei vecchi vettori base espressi rispetto alla nuova base.

In questo libro, lavorando di fatto con sensori (e non con bracci robotici), tutte le matrici sono di fatto matrici di cambiamento di base in quanto si vuole principalmente conoscere la coordinata di un punto di visto da un sensore nel sistema di riferimento superiore o viceversa.

Passando al caso tridimensionale il discorso si complica ulteriormente: esistono infinite parametrizzazioni per esprimere una rotazione partendo da 3 parametri $\mathbf{so}(3)$.

È per esempio possibile definire una rotazione come composizione di 3 rotazioni elementari intorno a uno dei 3 assi ma essendo la moltiplicazione tra matrici non commutativa esistono comunque 24 modi per comporre tra loro queste 3 matrici. Le combinazioni di matrici vengono indicate come sequenze di Eulero seguite da 3 numeri per indicare l'ordine di combinazione delle rotazioni: 1 per l'asse x, 2 per l'asse y e 3 per l'asse z. In ambito robotico sono ampiamente diffuse la rappresentazione di Angoli di Eulero (sequenza ZYZ) o quella degli angoli di Tait-Bryan (sequenza di Eulero 321 o ZYX) e si veda la sezione seguente A.1 per i dettagli[2]. Nella letteratura italiana i sei gruppi (XYZ, YZX, ZXY, XZY, ZYX, YXZ) vengono definiti angoli di Cardano.

Questo sistema di angoli tuttavia presenta alcune singolarità che ne limitano l'utilizzo. Alternativamente la sintassi proposta da Rodrigues (sezione A.2) o i quaternioni (sezione A.3) possono essere usati per superare questo problema.

Sempre per il fatto che il prodotto delle matrici non è commutativo, nello spazio tridimensionale c'è addirittura un ulteriore livello di ambiguità dato dall'ordine con cui le rotazioni con angoli di eulero vengono descritte, in quanto le rotazioni possono essere definite estrinseche o intrinseche:

Rotazioni Estrinseche Si riferiscono a rotazioni attorno ad assi fissi che coincidono con il sistema di riferimento iniziale dell'oggetto in rotazione. Gli assi di riferimento rimangono invariati durante le rotazioni. Le sequenze di rotazione estrinseche sono specificate da notazioni come x-y-z.

Rotazioni Intrinseche Si riferiscono a rotazioni attorno ad assi mobili che sono attaccati all'oggetto in rotazione. Questi assi cambiano posizione dopo ogni rotazione. Le sequenze di rotazione intrinseche sono specificate da notazioni come x-y'-z", dove gli apici indicano i nuovi sistemi di riferimento successivi.

Una rotazione estrinseca equivale ad una rotazione intrinseca (e viceversa) ma con l'ordine di composizione delle trasformazioni invertito (e.g. una trasformazione estrinseca z-y-x equivale ad una intrinseca x-y'-z").

Indipendentemente dal significato geometrico a cui si vuole dare la matrice di rotazione è possibile comunque fare diverse considerazioni.

Come già detto in precedenza, la definizione della matrice $\mathbf{R}$ nell'equazione della pin-hole camera è stata definita, sia per comodità che per tradizione, in modo tale da non ruotare un vettore (sarebbe stata ovvero una conversione da coordinate "sensore" a coordinate "mondo") ma all'opposto rimuove la rotazione di punti del mondo conoscendo l'orientazione della camera stessa ovvero permette di convertire da coordinate "mondo" a coordinate "camera".

Ricavare una espressione della matrice $\mathbf{R}$ nella forma espressa nel modello della pin-hole camera vuol dire trovare una matrice che trasforma un punto da coordinate "mondo" a coordinate "camera" ovvero bisogna sempre usare la matrice inversa delle matrici di rotazioni che si possono trovare nelle sezioni seguenti.

Sia pertanto una generica rotazione ${}^{w}\mathbf{R}_{b}$ che trasforma da coordinate locali, mobili, "sensore" (*body coordinates* nel caso generico) a coordinate globali, fisse, "mondo": la matrice $({}^{w}\mathbf{R}_{b})^{-1} = {}^{b}\mathbf{R}_{w}$ sarà pertanto la matrice che converte da coordinate mondo a coordinate sensore.

Siccome però il sistema di riferimento camera/immagine è un sistema *Left-Bottom-Front* (X crescente verso destra, Y crescente verso il basso, Z la profondità come in figura 8.3) che è diverso dal sistema di riferimento *Front-Left-Up* sensore/mondo (Z crescente verso l'alto, X profondità e Y crescente verso sinistra come in figura 8.4) tipico dell'ambiente *automotive*, è necessario definire una matrice

$$ {}^{c}\mathbf{\Pi}_{b} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \tag{A.3} $$

[2]Gli angoli Roll-Pitch-Yaw XYZ o semplicemente Roll-Pitch-Yaw sono spesso utilizzati in robotica e aeronautica; tuttavia per creare confusione questa è una rappresentazione intrinseca e sono equivalenti a quelli che a volte vengono chiamati angoli di Eulero ZYX dove invece viene usata una rappresentazione estrinseca.

matrice di permutazione degli assi. La matrice di permutazione ha determinante +1 pertanto è ancora una rotazione che preserva la chiralità dello spazio (trasforma sistemi destrorsi in sistemi destrorsi).

Lavorando in ambito areonautico o navale potrebbe invece essere necessario passare dal sistema camera/immagine a un sistema *Front-Right-Down* (ad esempio il NED). In questa situazione la matrice di permutazione è

$$ {}^{c}\mathbf{\Pi}_{b'} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{A.4} $$

Sotto queste considerazioni, la matrice $\mathbf{R}$ che converte da "mondo" a "camera", formalismo usato normalmente nell'equazione della camera *pin-hole*, ha come espressione

$$ \mathbf{R} = {}^{c}\mathbf{R}_{w} = {}^{c}\mathbf{\Pi}_{b} \left({}^{w}\mathbf{R}_{b} \right)^{-1} \tag{A.5} $$

A.1 Tait-Bryan Angles

Un modo per definire la matrice di rotazione in 3 dimensioni consiste nel comporre tra loro rotazioni rispetto ai 3 assi principali del sistema di riferimento.

Definiamo ϑ l'angolo di beccheggio *pitch*, γ l'angolo di imbardata *yaw* e ρ l'angolo di rollio *roll*, angoli di orientazione del sensore rispetto al sistema di riferimento mondo[3] Tali angoli e tale nomenclatura sono definiti come *Tait-Bryan Angles*, *Cardan Angles* (da Girolamo Cardano) o *nautical angles*.

Di seguito saranno mostrate le matrici (come riferimento per esempio [LaV06]) che convertono un vettore da coordinate sensore a coordinate mondo attraverso angoli che rappresentano l'orientazione del sensore rispetto al mondo stesso e sono le medesime matrici che ruotano un vettore in senso antiorario (*counterclockwise rotation of axes*) rispetto ai vari assi del sistema di riferimento.

Gli assi di tale sistema di riferimento sono quelli mostrati in figura 8.4. Si faccia comunque attenzione perché per i veicoli terrestri e per le navi viene prediletto un sistema di riferimento diverso da quelli aeronautico.

La matrice di rotazione dell'angolo *roll* ρ (asse X):

$$ \mathbf{R}_{x} = \mathbf{R}_{\rho} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\rho & -\sin\rho \\ 0 & \sin\rho & \cos\rho \end{bmatrix} \tag{A.6} $$

La matrice di rotazione dell'angolo *pitch* ϑ (asse Y):

$$ \mathbf{R}_{y} = \mathbf{R}_{\vartheta} = \begin{bmatrix} \cos\vartheta & 0 & \sin\vartheta \\ 0 & 1 & 0 \\ -\sin\vartheta & 0 & \cos\vartheta \end{bmatrix} \tag{A.7} $$

La matrice di rotazione dell'angolo *yaw* γ (asse Z):

$$ \mathbf{R}_{z} = \mathbf{R}_{\gamma} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.8} $$

(La Valle [LaV06], pag. 80-81).

Come si è detto nella sezione precedente, la composizione di rotazioni non è commutativa ed è necessario fare una scelta.

In campo aeronautico viene suggerita come convenzione *Roll-Pitch-Yaw* (RPY). Sotto questa particolare convenzione la matrice di cambiamento di base (alias) si costruisce come ${}^{w}\mathbf{R}_{b} = \mathbf{R}_{z}\mathbf{R}_{y}\mathbf{R}_{x}$[4] ovvero, eseguendo le moltiplicazioni,

$$ \begin{bmatrix} \cos\gamma\cos\theta & \cos\gamma\sin\theta\sin\rho - \sin\gamma\cos\rho & \cos\gamma\sin\theta\cos\rho + \sin\gamma\sin\rho \\ \sin\gamma\cos\theta & \sin\gamma\sin\theta\sin\rho + \cos\gamma\cos\rho & \sin\gamma\sin\theta\cos\rho - \cos\gamma\sin\rho \\ -\sin\theta & \cos\theta\sin\rho & \cos\theta\cos\rho \end{bmatrix} \tag{A.9} $$

Va ricordato che tale matrice trasforma punti dalle coordinate mobili "sensore" (*body coordinates* nel caso generico) alle coordinate fisse "mondo".

[3] attenzione che non esiste neanche una notazione accettata univocamente sulle lettere greche da associare ai 3 angoli. Si può trovare per esempio ϕ per l'angolo di *yaw* e ψ per l'angolo di *roll*.

[4] La sequenza z-y'-x" intrinsica (l'uso degli apici sottolinea questo tipo di trasformazione) generebbe invece $\mathbf{R} = \mathbf{R}_{x}\mathbf{R}_{y}\mathbf{R}_{z}$. Per creare ulteriore confusione la sequenza x-y'-z" è conosciuta come Roll-Pitch-Yaw (o Roll-Pitch-Yaw XYZ), mentre la sequenza z-y'-x" (intrinseca) è comunemente conosciuta come Yaw-Pitch-Roll (o Roll-Pitch-Yaw ZYX).

Nel caso specifico in cui il sensore fosse una camera pin-hole, usando questa convenzione e considerando l'equazione (A.5), la matrice di rotazione $\mathbf{R}$ della camera pin-hole che converte da coordinate "mondo" Front-Left-Up a coordinate "camera", si può esprimere come prodotto di

$$ {}^{c}\mathbf{R}_{w} = {}^{c}\mathbf{\Pi}_{b}\mathbf{R}_{\rho}^{-1}\mathbf{R}_{\vartheta}^{-1}\mathbf{R}_{\gamma}^{-1} \tag{A.10} $$

ovvero

$$ \begin{bmatrix} -\cos\gamma\sin\theta\sin\rho + \sin\gamma\cos\rho & -\sin\gamma\sin\theta\sin\rho - \cos\gamma\cos\rho & -\cos\theta\sin\rho \\ -\cos\gamma\sin\theta\cos\rho - \sin\gamma\sin\rho & -\sin\gamma\sin\theta\cos\rho + \cos\gamma\sin\rho & -\cos\theta\cos\rho \\ \cos\gamma\cos\theta & \sin\gamma\cos\theta & -\sin\theta \end{bmatrix} \tag{A.11} $$

Va ribadito che la matrice ${}^{c}\mathbf{R}_{w}$, espressa come nella formula (A.10), è la matrice che "rimuove" la rotazione di un sensore avete quei particolari angoli di posizionamento e pertanto trasforma da coordinate "mondo" a coordinate "camera" mentre normalmente in letteratura si tende a indicare come matrice di rotazione quella matrice che converte da coordinate "sensore" a coordinate "mondo".

È interessante notare che da un punto di vista puramente grafico, le colonne della matrice inversa/trasposta della matrice (A.11), la quale, quest'ultima, permette di trasformare punti da coordinate camera a coordinate mondo, permettono facilmente di disegnare gli assi e così rappresentare graficamente l'orientazione della camera.

A.1.1 Angoli di Eulero

La sequenza di Eulero (ZYZ) si basa sulla successione di tre rotazioni elementari:

$$ \mathbf{R}_{ZYZ}(\rho, \vartheta, \gamma) = \mathbf{R}_{z}(\rho)\mathbf{R}_{y}(\vartheta)\mathbf{R}_{z}(\gamma) \tag{A.12} $$

A.2 Parametrizzazione Asse-Angolo

Ogni rotazione è equivalente a una rotazione intorno a un asse (di rotazione) di una certa quantità angolare. Da questo presupposto parte la formulazione di una rotazione di Rodrigues o Parametrizzazione Asse-Angolo. La formulazione di Rodrigues cerca di risolvere i problemi di singolarità intrinseci delle formulazioni di Tait-Bryan e Eulero (diverse combinazioni di valori rappresentano la stessa matrice di rotazione), oltre a fornire una formulazione geometrica e concisa della rotazione.

La formula della rotazione proposta da Rodrigues è formata da un versore $\mathbf{k}$ e da un angolo ϑ i quali permettono di rappresentare una rotazione dei punti dello spazio di un angolo ϑ, intorno all'asse formato dal vettore $\mathbf{k}$, con verso positivo nel senso della regola della mano destra.

È possibile convertire asse e angolo in una matrice di rotazione attraverso una equazione compatta proposta da Rodrigues:

$$ \mathbf{R} = \mathbf{I} + \sin\vartheta[\mathbf{k}]_{\times} + (1 - \cos\vartheta)(\mathbf{k}\mathbf{k}^{\top} - \mathbf{I}) \tag{A.13} $$

(questa è una delle molteplici rappresentazioni disponibili in letteratura) che equivale, esplicitando i termini, alla matrice di rotazione

$$ \mathbf{R} = \begin{bmatrix} c + k_x^2(1-c) & k_xk_y(1-c) - k_zs & k_ys + k_xk_z(1-c) \\ k_zs + k_xk_y(1-c) & c + k_y^2(1-c) & -k_xs + k_yk_z(1-c) \\ -k_ys + k_xk_z(1-c) & k_xs + k_yk_z(1-c) & c + k_z^2(1-c) \end{bmatrix} \tag{A.14} $$

avendo dichiarato $s = \sin\vartheta$ e $c = \cos\vartheta$. Quando $\vartheta = 0$, ovvero in assenza di rotazione, la matrice si riduce all'identità.

La formulazione inversa è anch'essa estremamente compatta e vale:

$$ \begin{aligned} \vartheta &= \cos^{-1}\left(\frac{\operatorname{trace}\mathbf{R} - 1}{2}\right) \\ \mathbf{k} &= \frac{1}{2\sin\vartheta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \end{aligned} \tag{A.15} $$

Siccome $\mathbf{k}$ e ϑ sono di fatto 4 parametri, solitamente si usa un vettore $\mathbf{w} = \vartheta\mathbf{k}$ generico per rappresentare una rotazione nella formulazione di Rodrigues e si attuano le sostituzioni:

$$ \begin{aligned} \mathbf{k} &= \frac{\mathbf{w}}{\|\mathbf{w}\|} \\ \vartheta &= \|\mathbf{w}\| \end{aligned} \tag{A.16} $$

in modo da rappresentare correttamente la trasformazione da $\mathbf{so}(3)$ a $SO(3)$.

A.2.1 Rotazioni infinitesimali

La definizione compatta $\mathbf{w} = \vartheta\mathbf{k}$ unita alla formula di Rodrigues permette di esprimere rotazioni infinitesimali in maniera molto semplice da calcolare.

Se si manda infatti ϑ a infinitesimi, la formula (A.13) si può approssimare a

$$\mathbf{R} \approx \mathbf{I} + \sin\vartheta[\mathbf{k}]_\times \approx \mathbf{I} + [\mathbf{w}]_\times = \begin{bmatrix} 1 & -w_z & w_y \\ w_z & 1 & -w_z \\ -w_y & w_z & 1 \end{bmatrix} \tag{A.17}$$

A.3 Quaternioni

Son: Well, Papa, can you multiply triplets?
Father: No [sadly shaking his head], I can only add and subtract them. (William Rowan Hamilton, Conversation with his sons (1843))

I quaternioni sono un tentativo di estensione dei numeri complessi a una dimensione maggiore. Tale formulazione è stata proposta per la prima volta da Sir William Rowan Hamilton. Sono rappresentati da un vettore di $\mathbb{R}^4$ nella forma

$$\mathbf{q} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} = q_w + q_x i + q_y j + q_z k \tag{A.18}$$

a volte indicato anche $\mathbf{q} = \begin{pmatrix} q_0 & q_1 & q_2 & q_3 \end{pmatrix} = \begin{pmatrix} q_1 & q_2 & q_3 & q_4 \end{pmatrix}$. I quaternioni hanno differenti proprietà rispetto agli ordinari vettori quadridimensionali (come per esempio lo sono anche le coordinate omogenee). Il quaternione (A.18) può essere visto come composto da una parte vettoriale $\mathbf{v} \in \mathbb{R}^3$ e da una parte scalare q_w:

$$\mathbf{q} = \begin{bmatrix} q_w \\ \mathbf{v} \end{bmatrix} \tag{A.19}$$

q_w è definita parte scalare (o componente reale) mentre q_x, q_y, q_z sono le componenti vettoriali (o immaginarie). Un quaternione con solo la parte scalare è chiamato *reale* mentre un quaternione con solo la parte vettoriale *puro*.

Il prodotto tra quaternioni per esempio non è commutativo (ma è comunque associativo).

È possibile creare un vettore aumentato (*augmented vector*) di un vettore $\mathbf{r} \in \mathbb{R}^3$ nello spazio dei quaternioni come:

$$\bar{\mathbf{r}} = \begin{bmatrix} 0 \\ \mathbf{r} \end{bmatrix} \tag{A.20}$$

Il complesso coniugato di un quaternione $\mathbf{q}^*$ è

$$\mathbf{q}^* = \begin{bmatrix} q_w \\ -\mathbf{v} \end{bmatrix} \tag{A.21}$$

La norma $|\mathbf{q}|$ è

$$|\mathbf{q}| = \sqrt{\mathbf{q}^*\mathbf{q}} = \sqrt{q_w^2 + \mathbf{v}^2} \tag{A.22}$$

Un quaternione $|\mathbf{q}| = 1$ è chiamato quaternione *unità*. L'inverso di un quaternione unità è il suo complesso coniugato $\mathbf{q}^{-1} = \mathbf{q}^*$.

La proprietà più importante di un quaternione è che esso rappresenta una rotazione in $\mathbb{R}^3$.

Una rotazione $\mathbf{R} = e^{\vartheta\hat{\mathbf{u}}}$, espressa in rappresentazione asse/angolo, può essere scritta sotto forma di quaternione

$$\mathbf{q} = \exp(\vartheta\hat{\mathbf{u}}) = \begin{bmatrix} \cos(\vartheta/2) \\ \hat{\mathbf{u}}\sin(\vartheta/2) \end{bmatrix} \tag{A.23}$$

con ϑ un angolo di rotazione e $\hat{\mathbf{u}}$ un versore tridimensionale. In questo caso è un quaternione unità e rappresenta la rotazione di un angolo ϑ intorno all'asse $\hat{\mathbf{u}}$. Si noti che una rotazione di $-\vartheta$ rispetto a $-\hat{\mathbf{u}}$ fornisce lo stesso quaternione che la rotazione di ϑ intorno a $\hat{\mathbf{u}}$ risolvendo la singolarità della rappresentazione asse/angolo. Allo stesso modo è possibile definire il "logaritmo" di un quaternione:

$$\vartheta\hat{\mathbf{u}} = \log\mathbf{q} = \begin{cases} 2\dfrac{\mathbf{v}}{\|\mathbf{v}\|}\arccos q_w & \mathbf{v} \neq 0 \\ 0 & \mathbf{v} = 0 \end{cases} \tag{A.24}$$

che ritorna la consueta rappresentazione asse-angolo di una rotazione dato un quaternione.

Le rotazioni sono rappresentate da quaternioni di lunghezza unitaria $\mathbf{q}^\top\mathbf{q} = 1$.

È possibile ruotare un punto usando direttamente i quaternioni $\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}$, o un quaternione unitario può essere convertito in una matrice di rotazione (*directional cosine matrix*):

$$\mathbf{R} = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2q_xq_y - 2q_wq_z & 2q_xq_z + 2q_wq_y \\ 2q_xq_y + 2q_wq_z & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2q_yq_z - 2q_wq_x \\ 2q_xq_z - 2q_wq_y & 2q_yq_z + 2q_wq_x & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{A.25}$$

o in maniera equivalente:

$$\mathbf{R} = \begin{bmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_xq_y - q_wq_z) & 2(q_xq_z + q_wq_y) \\ 2(q_xq_y + q_wq_z) & 1 - 2(q_x^2 + q_z^2) & 2(q_yq_z - q_wq_x) \\ 2(q_xq_z - q_wq_y) & 2(q_yq_z + q_wq_x) & 1 - 2(q_x^2 + q_y^2) \end{bmatrix} \tag{A.26}$$

in modo da calcolare poi $\mathbf{p}' = \mathbf{R}\mathbf{p}$.

È da notare che $\mathbf{q}$ e $-\mathbf{q}$ rappresentano la medesima matrice di rotazione $\mathbf{R}$.

Viceversa dalla matrice di rotazione è possibile ricavare il quaternione attraverso per esempio

$$\begin{aligned} q_w^2 &= (r_{11} + r_{22} + r_{33} + 1)/4 \\ q_x &= (r_{32} - r_{23})/(4q_w) \\ q_y &= (r_{13} - r_{31})/(4q_w) \\ q_z &= (r_{21} - r_{12})/(4q_w) \end{aligned} \tag{A.27}$$

(operativamente si cerca la componente maggiore e si calcolano gli altri componenti rispetto a quella).

Il prodotto tra due quaternioni rappresenta infine la composizione di rotazioni:

$$\mathbf{q} \times \mathbf{t} = \begin{bmatrix} t_wq_w - t_xq_x - t_yq_y - t_zq_z \\ t_wq_x + t_xq_w - t_yq_z - t_zq_y \\ t_wq_y + t_xq_z + t_yq_w - t_zq_x \\ t_wq_z - t_xq_y + t_yq_x + t_zq_w \end{bmatrix} \tag{A.28}$$

Appendice B

Nomenclatura

In questa sezione è riportata la nomenclatura usata in questo libro e in generale nei problemi di Visione Artificiale.

K Matrice dei parametri Intrinseci (vedi 8.5), anche indicata con **A** da diversi autori;

R Matrice di Rotazione (vedi eq. (8.15));

E Matrice Essenziale (vedi eq. (9.41));

F Matrice Fondamentale (vedi eq. (9.45));

P Matrice Proiettiva (vedi eq. (8.18));

Π Matrice di Permutazione (vedi eq. (A.3));

k_u, k_v Lunghezza focale in pixel (vedi eq. (8.3));

k_γ Fattore di *Skew*, raramente usato;

W, H Dimensione dell'immagine;

u_0, v_0 *Principal Point*;

ϑ Angolo di Beccheggio;

γ Angolo di Imbardata;

ρ Angolo di Rollio.

In this section nomenclature commonly used in artificial vision are reported.

K Matrix of Intrinsic Parameters (see eq. (8.5)), and sometimes it is referred as **A**;

R Rotation Matrix (see eq. (8.15));

E Essential Matrix (see eq. (9.41));

F Fundamental Matrix (see eq. (9.45));

P Camera Matrix (see eq. (8.18));

Π Permutation Matrix (see eq. (A.3));

k_u, k_v Horizontal and Vertical focal lenghts in pixel dimension (see eq. (8.3));

k_γ Skew Factor, rarely used;

W, H Image size in pixel unit;

u_0, v_0 Principal Point (the orthogonal projection of the optical center onto the image plane) coordinates in pixel unit;

ϑ Pitch angle;

γ Yaw angle;

ρ Roll angle.

Bibliografia

[AAK71] Y.I. Abdel-Aziz and H.M. Karara. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. In *Proc. ASP/UI Symp. on Close-Range Photogrammetry*, pages 1–18, Urbana, Illinois, January 1971.

[AHB87] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987.

[AL92] Wayne Iba Ai and Pat Langley. Induction of one-level decision trees. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 233–240. Morgan Kaufmann, 1992.

[B+84] Leo Breiman et al. *Classification and Regression Trees*. Chapman & Hall, New York, 1984.

[Bea78] P. R. Beaudet. Rotationally invariant image operators. In *International Conference on Pattern Recognition*, 1978.

[BETVG08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110:346–359, June 2008.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[BR96] Michael J Black and Anand Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International Journal of Computer Vision*, 19(1):57–91, 1996.

[Bro66] Duane C Brown. Decentering distortion of lenses. *Photogrammetric Engineering*, 32(3):444–462, 1966.

[Che03] Zhe Chen. Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond. Technical report, McMaster University, 2003.

[CKY09] Sunglok Choi, Taemin Kim, and Wonpil Yu. Performance evaluation of ransac family. In *Proceedings of the British Machine Vision Conference*, pages 81.1–81.12. BMVA Press, 2009. doi:10.5244/C.23.81.

[CLSF10] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 778–792, Berlin, Heidelberg, 2010. Springer-Verlag.

[CM09] S.L. Campbell and C.D. Meyer. *Generalized inverses of linear transformations*. Society for Industrial Mathematics, 2009.

[CPS05] Ondra Chum, Tomás Pajdla, and Peter Sturm. The Geometric Error for Homographies. *Computer Vision and Image Understanding*, 97(1):86–102, January 2005.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995. 10.1007/BF00994018.

[DBK+20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[DF01] Frederic Devernay and Olivier D. Faugeras. Straight lines have to be straight. *Machine Vision and Applications*, 13(1):14–24, 2001.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA, 2005. IEEE Computer Society.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[FB87] Martin A. Fischler and Robert C. Bolles. Readings in computer vision: issues, problems, principles, and paradigms. chapter Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, pages 726–740. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[FG87] W. Förstner and E. Gülch. A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centres of Circular Features, 1987.

[FH94] Yoav Freund and David Haussler. Unsupervised learning of distributions on binary vectors using two layer networks. Technical report, Santa Cruz, CA, USA, 1994.

[FHT00] J. Friedman, T. Hastie, and R. Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. *The Annals of Statistics*, 38(2), 2000.

[FK08] Robert B. Fisher and Kurt Konolige. Range sensors. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 521–542. Springer, 2008.

[FPF99] Andrew Fitzgibbon, Maurizio Pilu, and Robert B. Fisher. Direct least square fitting of ellipses. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(5):476–480, May 1999.

[FS95] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, London, UK, 1995. Springer-Verlag.

[GKSB10] G. Grisetti, R. Kuemmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *Intelligent Transportation Systems Magazine, IEEE*, 2(4):31–43, 2010.

[gre84] Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives. *Journal of the Royal Statistical Society: Series B (Methodological)*, 46(2):149–170, 1984.

[GVL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)(3rd Edition)*. The Johns Hopkins University Press, 3rd edition, October 1996.

[GZS11] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV)*, 2011.

[Har95] R.I. Hartley. In defence of the 8-point algorithm. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 1064–1070, June 1995.

[Her08] Christoph Hertzberg. A framework for sparse, non-linear least squares problems on manifolds, 2008.

[Hin12] Geoffrey E. Hinton. A practical guide to training restricted boltzmann machines. In Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade (2nd ed.)*, volume 7700 of *Lecture Notes in Computer Science*, pages 599–619. Springer, 2012.

[Hop82] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554–2558, apr 1982.

[Hor87] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[HOT06] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[Hou59] P. V. C. Hough. Machine Analysis of Bubble Chamber Pictures. In *International Conference on High Energy Accelerators and Instrumentation*, CERN, 1959.

[HR11] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (dls) method for pnp. In *2011 International Conference on Computer Vision*, pages 383–390, Nov 2011.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[HS97] Richard I Hartley and Peter Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997.

[Hub96] P.J. Huber. *Robust statistical procedures.* CBMS-NSF regional conference series in applied mathematics. Society for Industrial and Applied Mathematics, 1996.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[JU97] S.J. Julier and J.K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, volume 3, page 26, 1997.

[Kab76] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[KHB09] Juho Kannala, Janne Heikkilä, and Sami S. Brandt. Geometric camera calibration. In *In: Wah BW (ed.) Encyclopedia of Computer Science and Engineering.*, volume 3, pages 1389–1400. Wiley, Hoboken, NJ, 2009.

[KK95] Yasushi Kanazawa and Kenichi Kanatani. Reliability of 3-d reconstruction by stereo vision. *IEICE Transactions*, 78-D(10):1301–1306, 1995.

[KKLD23] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering, 2023.

[KSH12a] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[KSH12b] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[LaV06] S. M. LaValle. *Planning Algorithms.* Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[LF97] Q.-T. Luong and O. D. Faugeras. Self-calibration of a moving camera from pointcorrespondences and fundamental matrices. *Int. J. Comput. Vision*, 22(3):261–289, 1997.

[LFNP09] V. Lepetit, F.Moreno-Noguer, and P.Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal Computer Vision*, 81(2), 2009.

[Lin94] Tony Lindeberg. *Scale-Space Theory in Computer Vision.* Kluwer Academic Publishers, Norwell, MA, USA, 1994.

[Lin10] Peter Lindstrom. Triangulation made easy. In *CVPR*, pages 1554–1561. IEEE Computer Society, 2010.

[Lin14] Tony Lindeberg. Scale selection. In Katsushi Ikeuchi, editor, *Computer Vision*, pages 701–713. Springer US, 2014.

[LK81] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[Lon81] Longuet. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, Sep. 1981.

[Lou05] M I A Lourakis. A brief description of the levenberg-marquardt algorithm implemened by levmar. *Matrix*, 3:2, 2005.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[LS10] Philip M. Long and Rocco A. Servedio. Random classification noise defeats all convex potential boosters. *Mach. Learn.*, 78(3):287–304, March 2010.

[LZ99] Charles Loop and Zhengyou Zhang. Computing rectifying homographies for stereo vision. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:1125, 1999.

[Mah36] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55, April 1936.

[MBLB91] H.A. Mallot, H.H. Bülthoff, JJ Little, and S. Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological cybernetics*, 64(3):177–185, 1991.

[MBT04] Kaj Madsen, Hans Bruun, and Ole Tingleff. *Methods for Non-Linear Least Squares Problems (2nd ed.)*. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2004.

[MK04] Gerard Medioni and Sing Bing Kang. *Emerging Topics in Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[Mor80] Hans Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. In *tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & doctoral dissertation, Stanford University*, number CMU-RI-TR-80-03. September 1980.

[MS02] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, pages 128–142. Springer, 2002. Copenhagen.

[MST+20] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *CoRR*, abs/2003.08934, 2020.

[Nie99] H. B. Nielsen. Damping parameter in marquardt's method. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, apr 1999.

[Nis04] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(6):756–777, June 2004.

[OPM02] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[PIK92] John Princen, John Illingworth, and Josef Kittler. A formal definition of the hough transform: Properties and relationships. *Journal of Mathematical Imaging and Vision*, pages 153–168, 1992.

[PP99] Constantine Papageorgiou and Tomaso Poggio. Trainable pedestrian detection. In *ICIP (4)*, pages 35–39, 1999.

[RD05] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1508–1511, October 2005.

[RD06] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[Rou84] Peter J. Rousseeuw. Least Median of Squares Regression. *Journal of the American Statistical Association*, 79(388):871–880, December 1984.

[RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[Rud16] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2016.

[SF11] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE Robot. Automat. Mag.*, 18(4):80–92, 2011.

[SM99] Peter Sturm and Steve Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Fort Collins, USA*, pages 432–437, Juin 1999.

[Smo86] P. Smolensky. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Information Processing in Dynamical Systems: Foundations of Harmony Theory, pages 194–281. MIT Press, Cambridge, MA, USA, 1986.

[SS02] B. Schölkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. Mit Press, 2002.

[SSM06] G. Sibley, G. Sukhatme, and L. Matthies. The iterated sigma point kalman filter with applications to long range stereo. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[ST94] J. Shi and C. Tomasi. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 593–600, Seattle, United States, June 1994.

[Str87] Thomas M. Strat. Readings in computer vision: issues, problems, principles, and paradigms. chapter Recovering the camera parameters from a transformation matrix, pages 93–100. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[Sze10] Richard Szeliski. Computer vision : Algorithms and applications. *Computer*, 5:832, 2010.

[TMHF00] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, pages 298–372, London, UK, 2000. Springer-Verlag.

[Tsa87] R. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *Robotics and Automation, IEEE Journal of*, 3(4):323–344, August 1987.

[TSK06] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[Ume91] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.

[Val84] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27:1134–1142, November 1984.

[VHV91] Sabine Van Huffel and Joos Vandewalle. *The Total Least Squares Problem.* Society for Industrial and Applied Mathematics, 1991.

[VJ01] Paul Viola and Michael Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In *Advances in Neural Information Processing System 14*, pages 1311–1318. MIT Press, 2001.

[VJ02] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2002.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[WB95] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995.

[WM94] G. Q. Wei and S. D. Ma. Implicit and explicit camera calibration: Theory and experiments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(5):469–480, 1994.

[YLT+21] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021.

[ZB17] Christopher Zach and Guillaume Bourmaud. Iterated lifting for robust cost optimization. In Gabriel Brostow Tae-Kyun Kim, Stefanos Zafeiriou and Krystian Mikolajczyk, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 86.1–86.11. BMVA Press, September 2017.

[Zha99] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the Seventh IEEE International Conference on Computer Vision.*, volume 1, pages 666–673 vol.1, 1999.

[ZPvBG01] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 371–378, New York, NY, USA, 2001. Association for Computing Machinery.

[ZW94] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *ECCV (2)*, pages 151–158, 1994.