Fundamentals of Computer Vision Analysis

Paolo Medici
Department of Information Engineering, University of Parma / VISLAB srl

October 22, 2025

This book aims to provide a reasonably concise introduction to the fundamentals of geometry, algebra, and statistics necessary for understanding and applying the more advanced techniques in computer vision. As you will see, it includes several elements not strictly related to image processing, but that prove useful for anyone interested in developing complex applications based on image analysis, involving concepts such as tracking and high-level sensor fusion.

To keep the exposition concise, I have generally avoided delving into the proofs of the various theorems, but, with the aim of sparking curiosity, I have left them for the reader to explore." The original goal of this book has never been to provide a rigorous and exhaustive treatment, which often leads to getting lost in calculations and proofs, risking to tire the reader and distract attention from the truly important concepts. Similarly, I did not aim to cover every possible topic related to image processing and computer vision; instead, I have focused only on those subjects directly related to the experiments I have personally conducted in my research activities areas in which I feel most confident and where I can offer some meaningful contribution. The writing of this book has been strongly influenced by my research areas, which focus primarily on applications of computer vision to robotic perception and the development and control of autonomous vehicles.

Computer Vision is an extremely stimulating field of science, even for those outside the discipline. The very fact that geometry, statistics, and optimization are so closely "intertwined" in computer vision makes it a comprehensive area of study, worthy of interest even for outsiders. However, this broad interconnection among topics has not made the task of dividing this book into chapters any easier, and consequently, as will be seen, cross-references between chapters are widespread.

The citations included in the text are very limited; I refer only to works that I consider fundamental, and whenever possible, I have cited the original authors who first proposed the ideas underlying the theory. Reading the articles listed in the bibliography is highly recommended.

For the organization of this volume, I drew inspiration from several books, which I recommend reading, including *Multiple View Geometry"* [HZ04] by Hartley and Zisserman, *"Pattern Recognition and Machine Learning"* [Bis06], and *"Emerging Topics In Computer Vision"* [MK04] edited by Medioni and Kang. For topics more closely related to image processing, an excellent book, also available online, is *"Computer Vision: Algorithms and Applications"* by Szeliski [Sze10].

Lastly, I would like to note that this book has been written over the past 20 years, and some content may be quite outdated (particularly in the wake of the machine learning revolution that took place in the middle of the last decade) but I have chosen to retain it for historical reasons.

The mathematical notation used in this book is minimalist:

- Matrices are denoted by uppercase bold letters $\mathbf{A}$, while vectors are denoted by lowercase bold letters $\mathbf{x}$;
- The transpose of the inverse of a matrix $\mathbf{A}$, that is, $\left(\mathbf{A}^{-1}\right)^{\top} = \left(\mathbf{A}^{\top}\right)^{-1}$, is written as $\mathbf{A}^{-\top}$;
- In statistical contexts, the notation $\hat{x}$ indicates the estimated value of the quantity x;
- Unless otherwise specified, the notation ${}^{b}\mathbf{R}_{a}$ denotes a change-of-basis matrix that transforms reference frame a into reference frame b, while ${}^{a}\mathbf{v}$ denotes a vector expressed in reference frame a.

Finally, refer to Appendix B for a brief overview of the meaning of the symbols used in this book.

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/.

Chapter 1

Elements

This first chapter presents various topics in mathematical analysis and analytic geometry necessary for understanding and utilizing the algorithms of algebra, statistics, and, of course, machine vision that will be discussed in the subsequent chapters.

1.1 Overdetermined linear systems

Analyzing real systems, it is easy to encounter the problem of deriving the "solution" of an overdetermined linear system.

The importance of this topic is evident: when observations are made on a real system, it is naturally affected by observation noise. This noise clearly compromises the result of the single observation, but fortunately, it is usually possible to acquire many more observations than unknowns, thus obtaining an overdetermined system.

Under these conditions, to obtain a solution to the problem that minimizes the error, the use of a numerical regression technique is required, for example, least squares. In this first section, widely used mathematical techniques throughout the book are presented: for further details regarding these techniques, one can refer to Chapter 3 which is entirely focused on this subject.

We therefore have an *overdetermined* linear system

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

where $\mathbf{A}$ is a rectangular matrix $m \times n$ and with $m \geq n$. As this matrix is rectangular, it does not have an inverse; however, it is still possible to define for every possible solution $\mathbf{x} \in \mathbb{R}^n$ a value of the error, also known as the residual, that this potential solution would entail. There is no general solution for an overdetermined system, but only solutions that minimize the residual under a particular metric.

We define[1] the error metric as the norm of the residual

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

The so-called "least squares" solution of a linear system (1.1) is represented by the vector $\mathbf{x}$ that minimizes the Euclidean distance of the residual (1.2), meaning that finding the optimal solution of the system, in terms of least squares regression, is equivalent to locating the minimum of this error function with respect to $\mathbf{x}$.

If the equation (1.1) is multiplied by $\mathbf{A}^\top$, a "traditional" linear system is obtained, which has the solution

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

This is a first solution method for overdetermined systems and is referred to in the literature as the technique of *normal equations*: the original problem is transformed into a classical linear system where the coefficient matrix is square and therefore invertible using classical techniques.

However, the solution proposed in equation (1.3) is numerically unstable since cond $\approx \mathbf{A}^2$. Further details on the conditioning of matrices and the propagation of disturbances in the solution of well-posed or over-determined linear systems will be presented in section 2.7.

If the system is well-conditioned, the most stable technique for solving a problem involving the *normal equations* is Cholesky factorization.

It can be shown that a solution $\mathbf{x}$, better conditioned and that minimizes the function (1.2), exists and is given by:

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

[1] the motivation for this choice will be discussed in detail in the chapter on model analysis.

By construction, $\mathbf{x}$ is a solution to the system (1.1) and is also the vector that minimizes the function (1.2). The pseudoinverse matrix of $\mathbf{A}$ is denoted as $\mathbf{A}^+$ and is given by

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

This solution of the system is called the Moore-Penrose pseudoinverse.
The pseudoinverse has the following properties:

- The pseudoinverse of a matrix exists if the inverse of $\mathbf{A}^\top\mathbf{A}$ exists;
- The pseudoinverse of a square matrix coincides with its inverse;
- The pseudoinverse of a matrix, if it exists, is unique.

It is important to clarify from the outset that in minimizing the quantity (1.2), no assumptions have been made regarding the distribution of noise within the various components of which the matrix is composed: without such information, there is no guarantee that the solution will be optimal from a statistical point of view. Without assumptions about the noise distribution, the solution obtained through this minimization is indeed a purely algebraic solution that minimizes an algebraic error.

It is possible to obtain a slightly better solution from a statistical perspective when the noise is zero-mean white Gaussian and the variance of the noise for each observation is known. In this case, it is possible to assign different weights to each equation of the system by multiplying each row of the system by an appropriate weight, thus weighting each acquired data point differently.

A more in-depth discussion on this topic can be found in section 3.2, and in general, in chapter 2, the general case will be addressed where the way in which the error in the observed data affects the estimation of the parameters is known.

Stable techniques exist instead, based on factorizations that allow for deriving the solution directly from the matrix $\mathbf{A}$.

Using, for example, QR factorization, a method that is well-known for its numerical stability, the original problem (1.1) transforms into the problem $\mathbf{QRx} = \mathbf{y}$, and the solution can be derived from $\mathbf{Rx} = \mathbf{Q}^\top\mathbf{y}$, leveraging the orthogonality of matrix $\mathbf{Q}$. In QR factorization, the relationship $\mathbf{R}^\top\mathbf{R} = \mathbf{A}^\top\mathbf{A}$ holds, meaning that $\mathbf{R}$ is the Cholesky factorization of $\mathbf{A}^\top\mathbf{A}$: through this relationship, the pseudoinverse can ultimately be obtained explicitly.

On the other hand, using the Singular Value Decomposition *Singular Value Decomposition* (SVD), the oversized matrix $\mathbf{A}$ is decomposed into three matrices with interesting properties. Let $\mathbf{A} = \mathbf{USV}^*$ be the singular value decomposition (SVD) of $\mathbf{A}$. $\mathbf{U}$ is a unitary matrix of dimensions $m \times n$ (depending on the formalism used, *complete SVD* or *economic SVD*, the dimensions of the matrices may vary, and $\mathbf{U}$ may become $m \times m$), $\mathbf{S}$ is a diagonal matrix that contains the singular values (the eigenvalues of matrix $\mathbf{AA}^\top$, with dimensions, depending on the formalism, $n \times n$ or $m \times n$), and $\mathbf{V}^*$ is an orthonormal matrix, conjugate transposed, of dimensions $n \times n$.

Through a purely mathematical procedure, it is obtained that the pseudoinverse of $\mathbf{A}$ is equivalent to

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

where the pseudoinverse of a diagonal matrix $\mathbf{S}^+$ is equivalent to its inverse, that is, a diagonal matrix composed of the reciprocals of the corresponding values.

In summary, the ways to solve an oversized linear system are

- Using the normal equations $(A^\top A)x = A^\top b$ or the Moore-Penrose pseudoinverse $x = (A^\top A)^{-1}A^\top b$ (since $(A^\top A)$ is positive semidefinite, Cholesky can be used as a solver);
- The QR decomposition $A = QR \rightarrow x = R^{-1}Q^\top b$;
- The SVD decomposition $A = U\Sigma V^\top \rightarrow x = A^+ b = V\Sigma^+ U^\top b$;
- The Cholesky decomposition $A^\top Ax = U^\top Ux = A^\top b \rightarrow x = U^{-1}\left(U^{-\top}A^\top b\right)$ (as U is the upper triangular matrix U^{-1} and $U^{-\top}$ are easy to compute);

From a numerical computing perspective, the condition number of the matrix to be inverted can be reduced by scaling the columns of A.

Further details on the Moore-Penrose pseudoinverse can be found in many books, for example in [CM09] or in the foundational text on numerical analysis [GVL96].

Let us now examine the case in which the linear system to be solved is homogeneous.

A homogeneous linear system has the form

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

and typically the obvious solution $\mathbf{x} = 0$, which is obtained through equation (1.4), is not useful for the problem at hand. In this case, it is necessary to find, still under the terms of a least squares regression, a $\mathbf{x} \in \mathbb{R}^n$, which is non-zero, representing a vector subspace, specifically the *kernel* of $\mathbf{A}$. The generating vector of the subspace is known up to one or more multiplicative

factors (depending on the dimension of the null space). To obtain a unique solution, an additional constraint must be imposed, for instance $|\mathbf{x}| = 1$, allowing us to formalize

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

with the constraint $|\mathbf{x}| = 1$, thus performing a constrained minimization.

In this case, the SVD proves to be an extremely effective and computationally stable technique: the bases of the *kernel* of $\mathbf{A}$ are indeed exactly the columns of $\mathbf{V}$ associated with the zero singular values (eigenvalues) of the diagonal matrix $\mathbf{S}$. Generally, due to the presence of noise, there will not be an exactly zero singular value, but the column associated with the minimum singular value must be chosen.

The eigenvectors associated with null singular values of the matrix $\mathbf{S}$ thus represent the *kernel* of the matrix itself, and the number of null eigenvalues represents the dimension of the *kernel*. It should be noted that in equation (1.6), the presence of zeros in the diagonal matrix $\mathbf{S}$ was problematic: it is now understood that this presence is indicative of the fact that one of the components of the problem is completely uncorrelated with the solution and, as such, could be neglected. This result will indeed be used in section 2.10.1 in the discussion of the PCA algorithm.

The solution of the subspace of $\mathbf{A}$ is therefore

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

where $\mathbf{v}_i$ are the columns of the matrix $\mathbf{V}$, "right" singular vectors of $\mathbf{A}$ corresponding to N singular values, eigenvalues of 0 of the matrix $\mathbf{A}^\top\mathbf{A}$.

The SVD decomposition is one of the most stable and versatile techniques developed in recent years for solving linear systems, and throughout this book, this technology will be widely used.

1.2 Eigenvalues and Eigenvectors

Eigenvalues and eigenvectors were introduced in the previous section. In this section, a minimal discussion will be provided to use them effectively.

Definizione 1 *Given a square matrix* $\mathbf{A}$ *of order* n*, a number (real or complex)* λ*, and a non-zero vector* $\mathbf{x}$ *are said to be an* eigenvalue *and an* eigenvector *of* $\mathbf{A}$ *if the following relation holds:*

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

$\mathbf{x}$ *is also referred to as the* eigenvector *associated with the* eigenvalue λ.

An eigenvector is a vector $\mathbf{x} \neq 0$ that does not change direction due to the geometric linear transformation (application) $\mathbf{A}$ but only changes its magnitude by a factor λ known as the eigenvalue.

Rewriting the system (1.10) using the identity matrix $\mathbf{I}$, it follows that the eigenvalue and its associated eigenvector are obtained as the solution of the homogeneous system:

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

If $\mathbf{x}$ is an eigenvector of $\mathbf{A}$ associated with the eigenvalue λ and $t \neq 0$ is a number (real or complex), then $t\mathbf{x}$ is also an eigenvector of λ.

In general, the set of vectors $\mathbf{x}$ associated with an eigenvalue λ of $\mathbf{A}$ forms a subspace of $\mathbb{R}^n$ called the *eigenspace*.

The dimension of this subspace is called the geometric multiplicity of the eigenvalue.

Definizione 2 *The* characteristic polynomial *of* $\mathbf{A}$ *in the variable* x *is defined as follows:*

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

From the definition (1.11), it follows that λ is an eigenvalue if and only if $p(\lambda) = 0$. The roots of the characteristic polynomial are the eigenvalues of $\mathbf{A}$, and consequently, the characteristic polynomial has a degree equal to the dimension of the matrix. The matrices 2×2 and 3×3 have notable characteristic polynomials.

Properties of Eigenvalues and Eigenvectors

- $\mathbf{A}$ and $\mathbf{A}^\top$ have the same eigenvalues;
- If $\mathbf{A}$ is nonsingular, and λ is one of its eigenvalues, then λ^{-1} is an eigenvalue of $\mathbf{A}^{-1}$;
- If $\mathbf{A}$ is orthogonal, then $|\lambda| = 1$;
- $\lambda = 0$ is an eigenvalue of $\mathbf{A}$ if and only if $\det(\mathbf{A}) = 0$;

- The eigenvalues of diagonal and triangular matrices (upper and lower) are the elements of the main diagonal;
- The sum of the diagonal elements is equal to the sum of the eigenvalues, that is, $\text{trace}\,\mathbf{A} = \sum \lambda_i$;
- The determinant of a matrix is equal to the product of its eigenvalues, that is, $\det \mathbf{A} = \prod \lambda_i$;
- Symmetric matrices have real eigenvalues and orthogonal eigenvectors, thus the corresponding set of unit eigenvectors $\{\mathbf{v}_i\}$ is a set of orthonormal vectors;
- A square matrix $\mathbf{A}$ can be expressed in terms of the eigenvalue matrix $\mathbf{D} = \text{diag}\,(\lambda_1, \ldots, \lambda_n)$ and the eigenvalues $\mathbf{V}$ in the form
$$\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1} \tag{1.13}$$
- A symmetric square matrix $\mathbf{A}$ can be expressed in terms of its real eigenvalues λ_i and eigenvalues $\{\mathbf{v}_i\}$ in the form
$$\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^{\top} = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^{\top} \tag{1.14}$$
known as the spectral decomposition of $\mathbf{A}$.

1.3 Alternative Parametrizations in Space and Manifolds

Points in various spaces $\mathbb{R}^n$ can be described using coordinates different from Cartesian ones. In this section, we present some useful parametrizations that will be used throughout the book.

We introduce the following definition:

Definizione 3 *S^n is the unit sphere in $\mathbb{R}^n$ such that:*
$$S^n := \left\{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|^2 = 1\right\} \tag{1.15}$$

Since a generic parametrization in S^n will have $n+1$ components and only 1 constraint, it must possess n degrees of freedom (DOF) by definition.

In case $n = 0$, the "unit sphere" $S^0 = \{-1, +1\}$ consists of only 2 points and therefore is not a connected manifold.

With $n = 1$, the manifold is exactly the same as $SO(2)$ that is with the parametrization $S^1 = \left\{ \begin{bmatrix} \cos\alpha \\ \sin\alpha \end{bmatrix} ; \alpha \in \mathbb{R} \right\}$.

The sphere S^2 (the surface of the sphere or a direction in $\mathbb{R}^3$) is instead a 2-manifold that lacks group structure. It can be parametrized by two parameters (for example, polar coordinates as we will see shortly) but will always exhibit singularities.

1.3.1 Polar Coordinates

Assuming Cartesian coordinates, this section introduces polar coordinates and in particular shows the relationships that connect Cartesian coordinates to polar ones.

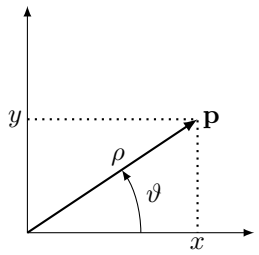


Figure 1.1: Correspondence between polar and Cartesian coordinates.

For a point in two-dimensional space, the relationship that links these two coordinate systems is written as:
$$\begin{aligned} x &= \rho\cos\vartheta \\ y &= \rho\sin\vartheta \end{aligned} \tag{1.16}$$

The inverse transformation, from Cartesian to polar coordinates, is
$$\begin{aligned} \rho &= \sqrt{x^2 + y^2} \\ \vartheta &= \text{atan2}(y, x) \end{aligned} \tag{1.17}$$

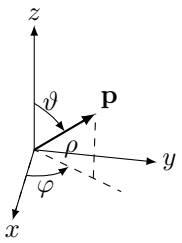


Figure 1.2: Polar coordinates in three dimensions: spherical coordinates.

A point on a sphere does not have a unique representation: for the same reason, as will be emphasized multiple times in the appendix, there are infinite representations of a rotation in three-dimensional space.

A very common choice is spherical coordinates (*spherical coordinate system*).

With this convention, the relationship between Cartesian and polar coordinates can be written as

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

where ϑ is defined as *zenith* while φ is called *azimuth*.

The inverse transformation, from Cartesian to polar coordinates, is obtained as

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

1.3.2 Stereographic Projection

A fairly common alternative to parameterize the sphere S^n is to use stereographic projection to transform coordinates from the manifold space $\mathbb{R}^n$ (parameter space) to $\mathbb{R}^{n+1}$ (Cartesian space) and vice versa.

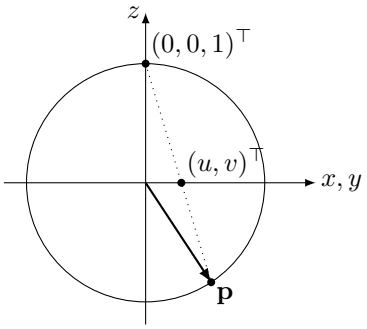


Figure 1.3: Stereographic projection.

In the three-dimensional sphere S^2, a function φ_{+3} can be defined as the stereographic projection from space $U_{+3} := S^2/\ [0,0,1]^\top$ to $\mathbb{R}^2$:

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

along with its inverse

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

where $[0,0,1]^\top$ denotes the "north pole" of the sphere. φ_{+3} and φ_{+3}^{-1} are continuous in U_{+3}, thus the stereographic projection is a homeomorphism. In this projection, a relationship is established between the point $(u,v,0)$ on the plane $z=0$ and the point on the sphere (x,y,z), the intersection between the projective line connecting the origin $(0,0,1)^\top$ (the only point of singularity) with the point on the plane and the unit sphere, as shown in figure 1.3.

Similarly, spaces $U_{\pm i} := S^2/\ \pm e_i$ can be defined where the e_i are the unit vectors within which to define 6 similar parameterizations (each with its own distinct singularity), allowing the selection of the most appropriate parameterization to operate at the point furthest from the singularity of that specific formula.

1.4 Homogeneous Coordinates

This section introduces homogeneous coordinates, a mathematical tool that proves very useful for discussing the problem of projective geometry as well as various formalisms addressed in different chapters of this book.

Homogeneous coordinates of a point on the plane $\mathbf{p} = (x, y) \in \mathbb{R}^2$ will be any ordered triplet $\tilde{\mathbf{p}} = (x', y', w') \in \mathbb{R}^3$ of real numbers such that $w' \neq 0$, $\frac{x'}{w'} = x$, and $\frac{y'}{w'} = y$. Similarly, homogeneous coordinates of a point $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ will be a quadruplet of numbers $\tilde{\mathbf{p}} = (x', y', z', w') \in \mathbb{R}^4$ such that $w' \neq 0$ and $\frac{x'}{w'} = x$, $\frac{y'}{w'} = y$, and $\frac{z'}{w'} = z$.

The point $\tilde{\mathbf{p}}$ expressed in homogeneous coordinates corresponds to the *real* point $\mathbf{p}$ (*inhomogeneous*):

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

The vector $(x, y, 1)$ is called the *augmented vector*.

Homogeneous coordinates have the following properties:

- Homogeneous coordinates are defined up to a proportionality constant. For example, the triplet $(x, y, 1)$ and any of its multiples $\lambda \neq 0$, namely $(x, y, 1) \cong (\lambda x, \lambda y, \lambda)$, are homogeneous coordinates of the same point in space (x, y);
- Points in homogeneous coordinates with coordinate $w = 0$ are referred to as **improper**, *points at infinity* or *ideal points*, and have no geometric meaning in Cartesian space, but can represent a point at infinity in the direction of the vector (x, y).

In homogeneous coordinates, there is a distinction between a vector ($w = 0$) and a point ($w \neq 0$), which is not the case with Euclidean coordinates. The set composed of all non-zero triplets/quadrupoles forms a two-dimensional/three-dimensional projective space.

Homogeneous coordinates allow for the representation of points at infinity and enable the expression of all geometric coordinate transformations used in computer vision in matrix form. The use of homogeneous coordinates is prevalent in *computer graphics* due to their ability to represent, just like in the Cartesian case, affine transformations through the use of matrices and also allows representing perspective projections with the same formalism.

1.5 Lines, Planes, and Hyperplanes

This section provides a brief summary of the equations for lines and, by extension, hyperplanes. A line is a set of points that separates the Cartesian plane into two parts; the plane is the set of points that separates three-dimensional space into two parts, and, generalizing, the hyperplane is that set of points that separates space $\mathbb{R}^n$ into two parts. This definition will be useful when discussing classification later.

1.5.1 Line

There are several formulations to express the concept of a line.

In the most general case, the multidimensional case, a line, the locus of points $\mathbf{x} \in \mathbb{R}^n$ of dimension 1, takes the form

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

where $\mathbf{p} \in \mathbb{R}^n$ is a generic point of origin, $\mathbf{v} \in \mathbb{R}^n$ is the direction vector, and $t \in \mathbb{R}$ is a scalar. In this case, it is referred to as a parametric ray.

In most applications, the line is a typical concept in two-dimensional space. In this space, ignoring the equation of the line written in explicit form $y = mx + q$ as it presents singularities, we focus on the line written in implicit form. The equation of the line written in implicit form is:

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

This representation is very useful because it allows considering both horizontal and vertical lines without any singularities. The parameter c equals zero when the line passes through the origin and, of course, the line passes through a point (x', y') when $c = -ax' - by'$.

In the two-dimensional case, the equation of the parametric ray (1.22) reduces to the equation of the implicit line with parameters

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

From the first of the equations (1.24), it is seen that the vector formed by the parameters (a, b) and the direction vector are orthogonal to each other. The generator vector of the line is indeed proportional, for example, to $\mathbf{v} \propto (-b, a)$ or $\mathbf{v} \propto (\frac{1}{a}, -\frac{1}{b})$. The vector $\mathbf{v}'$ orthogonal to the given line is simply $\mathbf{v}' \propto (a, b)$ and the line orthogonal to the given one has an implicit equation written in the form

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

where c' is obtained by selecting the point on the original line through which the perpendicular must pass.

The parameters of the line written in implicit form are homogeneous (the equation (1.23) is indeed called the homogeneous equation of the line), meaning they represent a vector subspace of $\mathbb{R}^3$: any multiple of those parameters represents the same line. Thus, these parameters are defined up to a multiplicative factor. This consideration suggests another way to represent a line and a generic hyperplane.

Lines, written in homogeneous implicit form, must satisfy the equation (dot product):

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

with $\mathbf{x} \in \mathbb{R}^3$ being the point in homogeneous coordinates and $\mathbf{l} = (a, b, c)^\top$ the parameters of the line. For homogeneous coordinates see the previous section 1.4 while for the implications of this notation, on the point-line duality, see subsection 1.5.7.

Since the implicit line is known up to a multiplication factor, there are infinitely many ways to express the same line. One possible normalization of the line is obtained by dividing the parameters by the length $\sqrt{a^2 + b^2}$. In this case, a particular solution of the line is obtained as the parameters correspond to those of a line written in polar coordinates in the same form as equation (1.46), and consequently, with this normalization, the parameter c represents the minimum distance between the line and the origin of the axes.

Finally, since the line is a hyperplane in 2 dimensions, its equation can be written as shown in equation (1.49).

1.5.2 Line passing through two points

For two points (x_0, y_0) and (x_1, y_1) in Cartesian space $\mathbb{R}^2$, there exists an implicit line with the equation

$$(y_1 - y_0)x - (x_1 - x_0)y - y_1x_0 + x_1y_0 = 0 \tag{1.27}$$

where it is clearly visible that there are no singularities and all values are admissible.

Indicating with (d_x, d_y) the difference between the two points, the line passing through a point (x_0, y_0) and directed along the vector (d_x, d_y) has the equation

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

Generalizing to the n-dimensional case, the equation of the line in $\mathbb{R}^n$, passing through two points $\mathbf{p}$ and $\mathbf{q}$ written in homogeneous form, is the locus of points $\mathbf{x} \in \mathbb{R}^n$ such that

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

the equation of the parametric ray with $t \in \mathbb{R}$ scalar value. The values of $\mathbf{x}$ associated with values $t \in [0, 1]$ are internal points to the segment $(\mathbf{p}, \mathbf{q})$.

Using instead the homogeneous coordinates, limited to the two-dimensional Cartesian case, the following remarkable result is obtained: the line with parameters $\mathbf{l} = (a, b, c)^\top$, passing through the points $\mathbf{x}_1$ and $\mathbf{x}_2$, is obtained as

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

since any point $\mathbf{x}$, to belong to the line, must satisfy the equation (1.26).

1.5.3 Point-Line Distance

The distance of a point (x', y') from a straight line (*line-point distance*), understood as orthogonal distance, that is, the distance between the given point and the closest point on the line, is given by:

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

In the n-dimensional case, the point $\mathbf{x}$ on the line described by equation (1.22) that is closest to a point $\mathbf{m}$ is the point for which the scalar t assumes the value

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

scalar projection onto the direction $\mathbf{v}$ of the segment $\mathbf{m} - \mathbf{p}$.

This version becomes very interesting when measuring the distance between a point $\mathbf{m}$ and a segment $(\mathbf{p}, \mathbf{q})$ utilizing the line generated as in equation (1.29). In this case, a value of t between $[0, 1]$ indicates that the closest point to $\mathbf{m}$ lies within the segment, as the scalar projection of segment $(\mathbf{p}, \mathbf{m})$ onto segment $(\mathbf{p}, \mathbf{q})$.

Finally, in section 1.5.10, in equation (1.54), it will be shown how to find the point on a hyperplane that is closest to a generic point. This formulation can also be applied to lines expressed in hyperplane form, and consequently, the point (x, y) on line (a, b, c) that is closest to point (x', y') is

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

1.5.4 Point-Segment Proximity

In various problems, it is necessary to know the distance between a point $\mathbf{p}$ and a polyline formed by multiple segments. The computational weight of this problem grows linearly with the number of points that make up the line: in order to perform these analyses, it is essential that the comparison with the single point is therefore very fast.

In this section, a segment will be defined as that part of the line bounded by the points $\mathbf{a}$ and $\mathbf{b}$. The point $\mathbf{p}$ and the segment can relate in 3 ways: the closest point is $\mathbf{a}$, the closest point is $\mathbf{b}$, or the closest point is a point between the two endpoints.

From a strictly computational point of view, calculating these 3 distances would require 9 multiplications, 6 additions, and one division, in addition to the necessary 3 comparisons. This section shows how the comparison can be computationally improved by using the dot product.

Without loss of generality, one can assume that $\mathbf{a} = (0, 0)^\top$. From the definition of the dot product

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

and the length of the orthogonal projection of $\mathbf{p}$ onto $\mathbf{b}$

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

it is possible to compute the point-segment distance more efficiently. The nearest point to $\mathbf{p}$ is $\mathbf{a}$ if and only if $\alpha > \pi/2$, that is $\mathbf{p} \cdot \mathbf{b} < 0$, while the nearest point is $\mathbf{b}$ if and only if the projection of $\mathbf{p}$ onto $\mathbf{b}$ is greater than $\|\mathbf{b}\|$, that is $\mathbf{p} \cdot \mathbf{b} > \|\mathbf{b}\|^2$. In this case, to obtain just the information about proximity, 4 multiplications and 2 additions are sufficient. If and only if the nearby point is found to be internal, one can proceed with the traditional point-line distance.

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

One can view a generic line in a space $\mathbb{R}^n$ as the interpolation of two points in the same space:

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

In the specific case of $\mathbb{R}^3$, these equations require 6 parameters to estimate (a "bounded 3D line" indeed has 6 degrees of freedom).

A line in the space $\mathbb{R}^n$ can be seen as a point plus a direction vector:

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

In the specific case of $\mathbb{R}^3$, these equations require 5 parameters (since a direction vector can be described by only 2 variables). In this case, the locus of points can be derived by multiplying by $\times\hat{\mathbf{v}}$:

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

The vector $\mathbf{x}_0 \times \hat{\mathbf{v}}$ obviously describes a vector orthogonal to the other two, but whose length is important. This representation is identical to that obtained using the PlÃ¼cker coordinate system.

In the space $\mathbb{R}^3$, the line is the locus of points at the intersection of 2 planes (one of which may potentially pass through the origin). Again, we are discussing at least 5 parameters to estimate.

However, in $\mathbb{R}^3$, lines have only 4 degrees of freedom: we can see that every line is tangent to a sphere of radius r, intersecting at point $m = (r, \theta, \phi)$ in spherical coordinates. The last parameter is a rotation angle γ around vector m to indicate the direction of the line (this part requires a couple of additional conditions to avoid singularities).

1.5.6 Intersection of Two Lines

Let ℓ_1 and ℓ_2 be two lines with parameters $\mathbf{l}_1$ and $\mathbf{l}_2$ intersecting at the point $\mathbf{x}$ expressed in homogeneous coordinates. To obtain the point of intersection, it is necessary to solve a homogeneous system in the form

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

The system, of type $\mathbf{A}\mathbf{x} = 0$, can also be extended to the case of n intersecting lines, with $n > 2$, resulting in an overdetermined system that can be solved using the SVD or QR decomposition technique. The solution to the overdetermined problem, affected by noise, represents the point that minimizes the algebraic residue of equation (1.39).

In the case of only two lines, the system (1.39) directly provides the solution.

The intersection between two lines $\mathbf{l}_1$ and $\mathbf{l}_2$, written in implicit form (1.23), is the point $\mathbf{x} = \mathbf{l}_1 \times \mathbf{l}_2$ expressed in homogeneous coordinates, where $\times$ is the cross product.

It is noteworthy that, since homogeneous coordinates can represent points at infinity, this particular formalism also accommodates the case where the two lines are parallel.

1.5.7 Principle of Duality

A concept that will be useful later is the principle of duality point-line. This principle is based on the commutative property of the scalar product applied to the equation of the line written in implicit form, where the positions of the points on the line are expressed in homogeneous coordinates:

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

It is therefore possible to obtain dual formations when the parameters of a line $\mathbf{l}$ are replaced with those of a point $\mathbf{x}$.

From this consideration arises the principle of duality (*Duality Principle*) which guarantees that the solution of the dual problem, where the meanings of line and point are swapped, is also a solution to the original problem.

For example, as seen in the previous sections, given two points $\mathbf{p}$ and $\mathbf{q}$, it is possible to define a line $\mathbf{l} = \mathbf{p} \times \mathbf{q}$ passing through them, while given two lines $\mathbf{l}$ and $\mathbf{m}$, it is possible to define a point $\mathbf{x} = \mathbf{l} \times \mathbf{m}$ as their intersection.

1.5.8 Distance between lines in $\mathbb{R}^3$

In space $\mathbb{R}^3$, and generally in all higher-dimensional spaces, two lines $\mathbf{l}_1$ and $\mathbf{l}_2$ may not intersect at any point even if they are not parallel. Such lines are defined as skew lines (*skew lines*). For these particular lines, a parameter of interest is their minimum distance, and consequently, the points on the two lines that represent this minimum.

Let two lines formed by points $\mathbf{x}_1$ and $\mathbf{x}_2$ have the equations

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

where $\mathbf{p}_1$ and $\mathbf{p}_2$ are two generic points belonging to the lines, $\mathbf{v}_1$ and $\mathbf{v}_2$ are the direction vectors, and $t_1, t_2 \in \mathbb{R}$ are scalar values, the unknowns of the problem.

The "distance" between two generic points on the two lines is

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

having defined $\mathbf{r} = \mathbf{p}_2 - \mathbf{p}_1$. The quantity to minimize is $\|\mathbf{d}\|^2$, a function of t_1 and t_2, whose gradient vanishes at

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

This is a linear system in t_1 and t_2 that can be easily solved, and with this solution, one can derive the two minimum points $\mathbf{p}_1$ and $\mathbf{p}_2$.

There is also an alternative formulation to solving the linear system that arrives at the same result through purely geometric considerations. It can be demonstrated that the distance between the two lines in $\mathbb{R}^3$ is given by

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

defining $\mathbf{n} = \mathbf{v}_1 \times \mathbf{v}_2$. The vector $\mathbf{n}$, the cross product, is by definition orthogonal to both lines, and the distance is the projection of the segment $\mathbf{r}$ along that vector. It is clear that when the lines are parallel ($\mathbf{n} = \mathbf{0}$), it is not possible to establish a reasonable value for triangulation.

The plane formed by the translation of the second line along $\mathbf{n}$ intersects the first line at the point of minimum distance

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

Regardless of the chosen formalism, substituting these values into equations 1.41 yields the three-dimensional coordinates of the closest points between the lines.

1.5.9 Line in Polar Coordinates

The lines presented so far tend to have an oversized representation concerning the degrees of freedom. The line in the plane $\mathbb{R}^2$ indeed has only 2 degrees of freedom, while the line expressed in implicit form depends on as many as 3 parameters, known up to a multiplicative factor and without a clearly visible geometric meaning. On the other hand, the explicit equation of the line with two parameters $y = mx + q$ exhibits the singularity of vertical lines.

A solution to the problem is to change the parametrization and exploit polar coordinates. Using polar coordinates, it is possible to express a line in a two-dimensional space without singularities and using only 2 parameters:

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

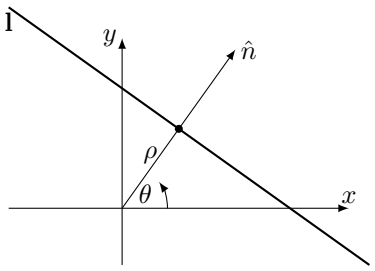


Figure 1.4: Line expressed in polar coordinates.

where ρ is the distance between the line and the point $(0, 0)$ and θ is the angle formed by this distance segment (orthogonal to the line) and the x-axis (figure 1.4). One should compare this representation with that expressed in equation (1.49). Under this formulation, the relationship between these two parameters and the equation of the line becomes non-linear.

This equation is commonly used in the Hough transform for lines (section 3.11) in order to exploit a two-dimensional and bounded parameter space.

With this particular form, the distance between a point in space (x_i, y_i) and the line is written in a very compact way as

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

1.5.10 Plans

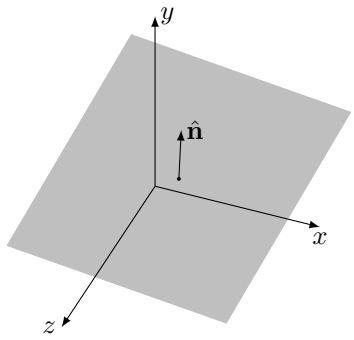


Figure 1.5: An example of plane in $\mathbb{R}^3$.

It is possible to generalize the discussion of lines to planes and hyperplanes in space $\mathbb{R}^n$. As with lines, there exists an implicit and homogeneous form of the equation of a plane understood as the locus of points expressed by the coordinate $\tilde{\mathbf{x}} \in \mathbb{R}^{n+1}$ homogeneous to $\mathbf{x} \in \mathbb{R}^n$:

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

The scalar product between homogeneous coordinates always encodes hyperplanes.

Homogeneous coordinates are known up to a multiplicative factor, and therefore an optional constraint can be imposed: similar to lines, one can think that the first n parameters of the homogeneous coordinate form a unit-length vector.

A generic plane, or hyperplane, is thus the set of points $\mathbf{x} \in \mathbb{R}^n$ that satisfy the condition

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

where $\mathbf{n} \in \mathbb{R}^n$ is the normal to the plane and $\rho = 0$ if and only if the plane passes through the origin. In the case of $\rho \neq 0$, an alternative representation of the plane is

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

and another expression of equation (1.49) that can be found in the literature is

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

with $\mathbf{x}_0 \in \mathbb{R}^n$ a generic point of the plane from which the correspondence $\rho = \mathbf{x}_0 \cdot \mathbf{n}$ can be derived.

It should be remembered that the degrees of freedom are always and only n.

When introduced, the normalization constraint $|\hat{\mathbf{n}}| = 1$ represents a particular case: under this condition, as in the case of lines, ρ assumes the meaning of the minimum Euclidean distance between the plane and the origin.

If the plane (or hyperplane) is normalized, the distance between a generic point $\mathbf{p}$ and the plane is measured as

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

otherwise, as in the case of lines, it is necessary to divide the distance by $\|\mathbf{n}\|$.

The point $\mathbf{x}$ closest to a generic point $\mathbf{p}$ belonging to the hyperplane is found at the intersection of the line directed by $\mathbf{n}$ passing through $\mathbf{p}$ and the plane itself:

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

or in

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

This formulation is also applicable to lines, as already seen.

As for the various methods for generation, in section 3.6.3 it will be shown how to obtain the least squares regression of a set of points to the plane equation.

As in the case of the line, the parameters of the plane in $\mathbb{R}^3$ can also be expressed using 3 polar coordinates (azimuth, zenith, and ρ):

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

the equation of the plane expressed in spherical polar coordinates (1.18).

1.5.11 The Division of the Plane

The line (hyperplane) separates the plane (space) into two parts, and within each of these parts, the function $\mathbf{m}^\top\mathbf{x}$ assumes the same sign. Through this consideration, it is easy to determine whether a set of points lies entirely on the same side of a line/hyperplane or not.

For example, in the case of the line, depending on how the generating vector is oriented, one can understand in which of the two half-planes (left, right) a generic point falls by studying $s = ax_i + by_i + c$: when $s < 0$ the point is to the left of the line, $s > 0$ the point is to the right, and finally when $s = 0$ the point is on the line.

This consideration, namely that the equation of a plane efficiently determines in which half-space a generic point falls, will be used in the chapter on classifiers: the simple equation of a line or a plane can be used as a classifier if the category space, generated by appropriate n measurable features of the image, is separable by a linear surface.

1.6 Conics

The conic is an algebraic curve that is the locus of points obtainable as the intersection between a circular cone and a plane. The equation of a conic written in implicit form is

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

It is worth noting that the parameters of the conic are known up to a multiplicative factor.

Equation (1.56) shows the equation of the conic written in traditional, non-homogeneous Cartesian coordinates. The use of homogeneous coordinates allows the writing of quadratic equations in matrix form.

If homogeneous coordinates are used instead of Cartesian coordinates, applying the substitutions $x = x_1/x_3$ and $y = x_2/x_3$, the equation of the conic can be expressed in homogeneous form:

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

In this way, it is possible to represent equation (1.56) in matrix form

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

where $\mathbf{C}$ is the symmetric matrix 3×3 of the parameters and $\mathbf{x}$ is the locus of points (expressed in homogeneous coordinates) of the conic. Since it is expressed by homogeneous ratios, this matrix is defined up to a multiplicative factor. The conic is defined by 5 degrees of freedom, that is, by the 6 elements of the symmetric matrix minus the scale factor.

Due to the point-line duality, the line $\mathbf{l}$ tangent to a conic $\mathbf{C}$ at the point $\mathbf{x}$ is simply $\mathbf{l} = \mathbf{C}\mathbf{x}$.

Writing the conic in equation (1.58) takes the form of a curve defined by a locus of points and is therefore also called a *point conic* because it defines the equation of the conic using points in space. Using the duality theorem, it is also possible to express a conic $\mathbf{C}^* \propto \mathbf{C}^{-1}$, dual to $\mathbf{C}$, in terms of lines this time: a tangent line $\mathbf{l}$ to the conic $\mathbf{C}$ satisfies $\mathbf{l}^\top\mathbf{C}^*\mathbf{l} = 0$.

In section 3.6.7, techniques for estimating the parameters that encode a conic given the points will be presented.

1.7 Cross Product

In space $\mathbb{R}^3$, it is possible to transform the vector product operator into a linear application, that is, to give a matrix representation to the cross product, such that $[\mathbf{x}]_\times \mathbf{y} = \mathbf{x} \times \mathbf{y}$.

In the text, the matrix 3×3 associated with the cross product will be denoted by $[\mathbf{x}]_\times$. The form of this matrix, which is antisymmetric, is

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

where $\mathbf{x} = (x_0, x_1, x_2)^\top$. This matrix has zero determinant and maximum rank 2.

1.8 Geometric Transformations

Geometric transformations of points in the plane are bijective transformations that associate one and only one point in the plane to each point in the same plane.

Geometric transformations can be classified into

Affinity In the Cartesian plane, the affine transformation is a bijective mapping that associates the point $\mathbf{p}$ to the point $\mathbf{p}'$ through a function of the form

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

An affinity enjoys the following properties:

- transforms lines into lines;
- preserves collinearity among points;
- preserves parallelism and incidence among lines;
- generally does not preserve shape or angles.

Being bijective, the affine transformation is invertible, and the inverse is also an affine transformation with parameters

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

Similarity A similarity is an affine transformation that preserves the ratio between dimensions and angles.

The form of the equation is identical to that of the affine transformation (1.60) but can only represent changes in scale, reflections, rotations, and translations. Depending on the sign of the determinant of $\mathbf{A}$, similarities are divided into *direct* (positive determinant) that preserve orientation or *inverse* (negative determinant) where the orientation is reversed.

Isometry Isometries are similar transformations that preserve distances:

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

for every $x, y \in \mathbb{R}^n$.

Isometries between Euclidean spaces are written as in equation (1.60) where, however, $\mathbf{A}$, a necessary and sufficient condition for it to be an isometry, must be an *orthogonal* matrix.

Since the matrix $\mathbf{A}$ is orthogonal, it must have a determinant of ± 1. As with similarities, if $\det \mathbf{A} = 1$ the isometry is said to be *direct*, while if $\det \mathbf{A} = -1$ the isometry is *inverse*.

Examples of isometries include

- translations;
- rotations;
- central and axial symmetries.

1.9 Relative Positioning Between Sensors

We introduce for nomenclature the relationships that exist between various reference systems, which will be used throughout this book. Further nomenclature regarding coordinate systems will be found in the upcoming section 8.2, which should be referred to for some terms.

Let ${}^{w}\mathbf{x} \in \mathbb{R}^3$ be a point expressed in "global" coordinates, "world" (*world coordinates*), and let ${}^{s}\mathbf{x}$ be the same point expressed in "local" coordinates, "sensor" (or *body* in the generic case of a system in motion relative to another). The coordinate ${}^{s}\mathbf{x}$ represents spatial information captured by the mere sensor: this coordinate therefore does not possess knowledge of how the sensor is positioned and oriented in world space. Although representing the same physical point, the coordinates expressed by the two points are indeed different as they are represented in two different reference systems: one represents a position understood as absolute (the world), while the second represents the point as seen from the sensor, where the sensor is at the center of the reference system aligned with respect to the axes.

Definition 1 *The relation that links world coordinates to sensor coordinates is*

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

with ${}^{w}\mathbf{R}_s$ *being the rotation matrix that allows the transformation of a point from sensor coordinates to world coordinates and* $\mathbf{t}$ *representing the position of the sensor with respect to the origin of the reference system.*

Let us now denote by the numbers 1 and 2 two generic sensors connected to the common reference frame world w through the parameters $({}^{w}\mathbf{R}_1, {}^{w}\mathbf{t}_1)$ and $({}^{w}\mathbf{R}_2, {}^{w}\mathbf{t}_2)$ respectively, expressed as in definition 1.

Let $({}^{1}\mathbf{R}_2, {}^{1}\mathbf{t}_{2,1})$ be the "relative" pose of sensor 2 with respect to sensor 1, a pose that allows for the conversion of a point from the reference frame of sensor 2 to the reference frame of sensor 1:

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

The matrix ${}^{1}\mathbf{R}_2$, which represents the orientation of sensor 2 with respect to sensor 1, transforms the sensor coordinates, while ${}^{1}\mathbf{t}_{2,1}$ is the pose of sensor 2 relative to sensor 1 expressed in the reference frame 1.

The parameters of the relative pose are obtained from the poses of the individual sensors, poses expressed with respect to a third reference system (the world frame), through the relationships:

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

From now on, to simplify the notation, we will implicitly understand the world reference frame w and therefore, when not specified, the coordinates are referred to this system and the change of basis also leads to this one.

The inverse relative pose $({}^{2}\mathbf{R}_1, {}^{2}\mathbf{t}_{1,2})$, which transforms from system 2 to system 1, can be obtained from $({}^{1}\mathbf{R}_2, {}^{1}\mathbf{t}_{2,1})$ as follows:

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

Given the knowledge of the relative pose between the sensors and the absolute pose of one of the two (for simplicity, sensor 1), it is possible to derive the absolute pose of the second sensor through the transformation:

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

1.9.1 Estimation of Rotations and Translations

There are several techniques to obtain an optimal estimate of the rigid rotation and translation transformation between points belonging to the same space. An optimal estimate refers to the maximum likelihood estimate, where the observation noise ξ_i is completely additive in the space $\mathbb{R}^n$ in which the points reside.

Said are therefore two sets of points $\{{}^{1}\mathbf{x}_i\}$ and $\{{}^{2}\mathbf{x}_i\}$ such that they are related by

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

as in equation (1.64) but with the noise vector ξ_i.

To solve the optimal problem $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$ that transforms all points from $\mathbf{x}^1$ to $\mathbf{x}^2$, a least squares criterion is required that optimizes a cost function of the form

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

where w_i are any priors associated with each sample. The least squares solution in non-linear form clearly faces issues of local minima.

A notable result concerning the translation vector is obtained by applying the classic derivatives $0 = \partial S/\partial \mathbf{t}$ to equation (1.69), which is null at

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

and therefore, given $\mathbf{R}$, it is immediate to find the translation in optimal form or vice versa; it suffices to derive only the rotation $\hat{\mathbf{R}}$ to solve the pose problem. This result can also be reached from an intuitive perspective: the centroid of the two sets of points following a rigid transformation must comply with the relationship of equation (1.68).

The optimal rotation $\mathbf{R}$ must therefore minimize

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

having defined $\mathbf{p}_i = \mathbf{x}_i - \bar{\mathbf{x}}$. Therefore, any pose registration problem in n dimensions always reduces to a problem with n DOF.

An initial technique for calculating the rotation makes use of principal components (see section 2.10.1). The principal components extracted from each set of points *separately* form a basis of the space. It is possible to determine a rotation that aligns these bases since the matrices of the column eigenvectors $\mathbf{R}_1$ and $\mathbf{R}_2$ yield $\mathbf{R} = \mathbf{R}_2 \left(\mathbf{R}_1\right)^\top$ directly. However, there may exist multiple solutions (each of which should be verified), and due to noise, deriving the axes through PCA can become extremely unreliable (for example, if the resulting distribution were circular, any estimation would be impossible).

The best approach to minimize (1.71) is to minimize or rather maximize:

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

minimizing the system of equations (1.69) is equivalent to maximizing the trace of the matrix $\hat{\mathbf{R}}\mathbf{H}$ where $\mathbf{H}$ is the correlation matrix between the two point clouds defined as

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

It is demonstrated that the matrix $\hat{\mathbf{R}}$ that maximizes the trace of $\hat{\mathbf{R}}\mathbf{H}$ is

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

having decomposed using singular value decomposition $\mathbf{H} = \mathbf{U}\Sigma\mathbf{V}^\top$. This algorithm is first described in [Kab76], rediscovered in [AHB87], but certainly improved in [Ume91] (particularly case $\det \hat{\mathbf{R}} = -1$) and is commonly referred to as the Kabsch-Umeyama algorithm.

This solution, much more stable than the PCA-based one and always valid for $n > 3$, requires particular attention only in two-dimensional and three-dimensional cases to handle potential reflections (in such cases, the determinant of the resulting matrix may be negative).

The "disadvantage" of the SVD-based technique compared to the PCA-based one is the requirement that the associations between the points of the two distributions be correct.

The combination of the two techniques along with an iterative approach is called *Iterative Closest Point* (ICP).

1.10 Homographic Transformations

The homogeneous coordinates (section 1.4) allow for the representation of a very broad spectrum of transformations, unifying under the same formalism both linear transformations (affine, rotations, translations) and perspective transformations.

Given two distinct planes Π_i and Π_j, they are said to be related by a homographic transformation (*homographic transformation*) when there exists a one-to-one correspondence such that:

- to each point or each line of Π_i corresponds exactly one point and one line of Π_i
- to each bundle of lines of Π_j corresponds a projective bundle on Π_j

Let the plane Π be observed from two different views. In the space $\mathbb{R}^2$, the homography (the projective transformation) is represented by equations of the type:

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

where (u_i, v_i) are coordinates of points belonging to the plane Π_i, while (u_j, v_j) are points of the plane Π_j.

Due to its particular form, this transformation can be described through a linear transformation using homogeneous coordinates (section 1.4):

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

having defined

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

In space $\mathbb{R}^2$, homographies are encoded by matrices 3×3 (2D homographies): in the same way, projective transformations can be defined for higher-dimensional spaces. For compactness and to maintain reference to a memory representation in *row-major*, as in `C`, the matrix $\mathbf{H}_{ij}^{\Pi}$ has been expressed using coefficients $h_0 \dots h_8$ rather than the classical syntax to indicate the elements of the matrix.

The homographic matrix $\mathbf{H}_{ij}^{\Pi}$ is defined as the matrix that converts homogeneous points $\mathbf{x}_i$ belonging to the plane Π_i of the image i into homogeneous points $\mathbf{x}_j$ of the image j with the relationship

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

Since this is a relationship between homogeneous quantities, the system is defined up to a multiplicative factor: any multiple of the parameters of the homographic matrix defines the same transformation because any multiple of the input or output vectors also satisfies the relationship (1.75). Consequently, the degrees of freedom of the problem are not 9, as in a general affine transformation in $\mathbb{R}^3$, but 8 since it is always possible to impose an additional constraint on the elements of the matrix. Commonly used examples of constraints are $h_8 = 1$ or $\|\mathbf{H}\|_F = 1$. It is worth noting that $h_8 = 1$ is generally not an optimal constraint from a computational perspective because the order of magnitude of h_8 can be very different from that of the other elements of the matrix itself and could lead to singularities, in addition to the edge case where h_8 could be zero. The alternative constraint $\|\mathbf{H}\|_F = 1$, which is satisfied for free using solvers based on SVD or QR factorizations, is computationally optimal.

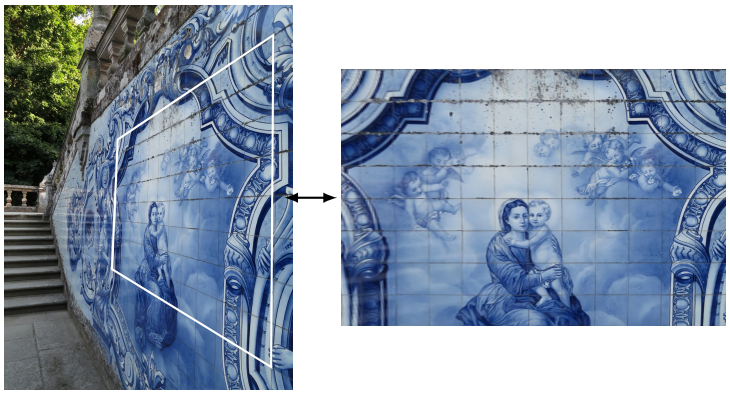


Figure 1.6: Example of homographic transformation: the homography relates planes in perspective with planes not in perspective.

The applications involving homographic transformations are numerous. They will be discussed in detail in chapter 8 regarding the pin-hole camera, but in summary, such transformations allow for the removal of perspective from image planes, the projection of planes in perspective, and the association of points of planes observed from different viewpoints.

Un way to obtain perspective transformations is to relate points between the planes to be transformed and thereby determine the parameters of the homographic matrix (1.75), even in an over-dimensional manner, for example through the method of least squares. One way to derive the coefficients will be shown in equation (8.49). It should be noted that this transformation, which links points of planes between two perspective views, holds true only for the points of the considered planes: the homography relates points of planes to one another, but only those. Any point not belonging to the plane will be reprojected incorrectly.

It is easy to see that every homography is always invertible and the inverse of the transformation is also a homographic transformation:

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

Figure 1.7: Example of homographic transformation: the homography relates 'virtual' planes to each other.

A possible form for the inverse of the homography (1.75) is

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

and, being known up to a multiplicative factor, no division has been used in deriving the parameters of the inverse transformation (*unnormalized inverse homographic matrix*).

It is worth noting that when the two planes related are parallel, then $h_6 = 0 \wedge h_7 = 0$, the homographic transformation reduces to an affine transformation represented by the classic equations

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

previously encountered.

1.10.1 Homography and Lines

There are interesting applications of homography in various fields.

A homographic transformation generally transforms lines into lines. In special cases, however, it can transform lines into points, such as in the perspective projection of elements on the horizon: homogeneous coordinates indeed represent points and vectors differently, and when a line reduces to a point, its homogeneous coordinate becomes 0.

The homographic transformation applied to a line (effect of the point-line duality) is exactly the inverse transformation of the one that transforms the corresponding points between the spaces: the transformation $\mathbf{H}_{ij}$ that transforms points $\mathbf{x}_i$ from image i to points $\mathbf{x}_j$ of image j transforms the equations of the lines from image j to image i:

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

Examining points and lines at infinity (for example, at the horizon), one can see how a point at infinity has coordinates $(x, y, 0)^{\top}$. Therefore, there exists a special line $\mathbf{l}_{\infty} = (0, 0, 1)^{\top}$ that connects all these points.

The principle of duality allows for explaining how, given a transformation $\mathbf{M}$ (projective or homographic), the transformation that transforms a point $\mathbf{x}$ into $\mathbf{x}'$ can be expressed as

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

while the transformation that transforms a line $\mathbf{l}$ instead becomes

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

1.10.2 Homography and Conics

A conic is transformed through a homographic transformation $\mathbf{x}' = \mathbf{H}\mathbf{x}$ into a conic. Indeed, it follows that

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

which is still a quadratic form $\mathbf{C}' \equiv \mathbf{H}^{-\top}\mathbf{C}\mathbf{H}^{-1}$. The use of the equivalence symbol $\equiv$ is necessary since the conic is defined up to a multiplicative factor.

This notable result allows us to demonstrate that a conic viewed in perspective is still a conic.

1.11 Points Inside Triangles and Quadrilaterals

Consider the problem of determining whether a point is inside relatively simple polygons such as triangles or quadrilaterals. The treatment of the case of generic polygons is more complex and is left to the reader (and optionally reduced to the triangle/quadrilateral case).

In the case of triangles and quadrilaterals, the approaches presented are very efficient when it is necessary to perform comparisons between a single polygon and a large number of points, as such techniques allow the use of precalculations: the fundamental idea is to exploit a transformation of the polygon's coordinate space into a space where comparisons are easier to perform.

For example, a parallelogram formed by two generating vectors can always be transformed into a unit square $(0,0)-(1,1)$ through an affine transformation $\mathbf{p} \mapsto (X, Y)$. A point $\mathbf{p}$ lies inside the parallelogram if $0 < X(\mathbf{p}) < 1$ and $0 < Y(\mathbf{p}) < 1$. The same transformation applies to triangles formed by the same generating vectors, but with the comparison $0 < X(\mathbf{p}) < 1$ and $0 < Y(\mathbf{p}) < 1 - X(\mathbf{p})$. To determine if a point lies inside the parallelogram, only 4 multiplications and 6 additions are necessary. The cost of creating the affine transformation is higher, but if the number of comparisons is large, this constant overhead becomes negligible.

To transform generic quadrilaterals into a unit square $(0,0)-(1,1)$, one must instead use the homographic transformation (see section 1.10). Despite the high initial cost of creating the transformation, the check for whether a point belongs to the geometric figure is relatively simple

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

thus limited to 6 multiplications, 6 sums, and 4 comparisons.

1.12 Transformations between images and Look Up Table

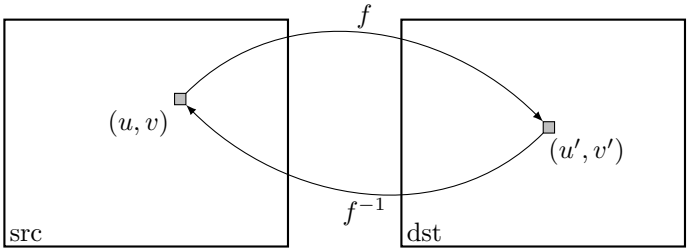


Figure 1.8: Direct and inverse transformation between images.

Being a somewhat delicate topic that could lead to some ambiguities, it is advisable to dedicate a section to how transformations between images are practically applied.

Let f be a generic bijective transformation

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

that transforms the point $(u, v)^\top$ belonging to the source image into the point $(u', v')^\top$ of the destination image, that is

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

This transformation will be called *Forward Warping*.

Since images are not continuous but quantized into pixels, the transformation f cannot be used directly in real applications, as it may either leave gaps in the second image or project the same point of the first image multiple times. For these reasons, when an image is processed, we always work with the inverse transformation f^{-1} which, for every point of the destination image $(u', v')^\top$, returns the point of the source image (u, v) from which to extract the color, that is:

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

This transformation will be referred to as *Inverse Warping*.

It is clear that the source image is also composed of pixels, but the knowledge of the point $(u, v)^\top$ allows for the straightforward use of techniques such as linear interpolation to derive the pixel value.

If the function f^{-1} is very complicated and the same transformation needs to be applied to multiple images, to save computational time, one can create a *Look Up Table* (LUT) of elements $(u, v)^\top$ as large as the destination image, where the result of the transformation (1.89) is stored for each element.

1.13 Sub-Pixel Accuracy

Every quantity, piece of information, characteristic that can be extracted from an image is limited by the pixel quantization of the image itself.

However, by examining a neighborhood around the point to be extracted, it is possible to provide an approximate estimate of the position of the feature with sub-pixel accuracy. These approaches are all based on the attempt to locally model the quantized image function, striving to reconstruct the information lost due to spatial quantization.

Every problem has a specific technique for extracting such information. In this section, some of them will be examined.

1.13.1 Minima and Maxima in 1D

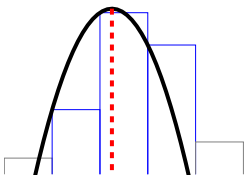


Figure 1.9: Construction of the parabolic model and detection of the maximum with sub-pixel accuracy.

If the point to be examined is the maximum or minimum of a one-dimensional sequence, one can approximate the first neighborhood of the point with a quadratic of equation $ax^2 + bx + c = y$. The quadratic is the least degree of the function that allows for the identification of local minima or maxima.

Let y_{-1}, y_0, and y_{+1} be the values of the function with deviations of -1, 0, and $+1$ with respect to the minimum/maximum identified with pixel precision. The equation of the quadratic passing through these 3 points takes the notable form

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

Such a curve has the notable maximum/minimum point at

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

$\hat{\delta}_x$ is to be understood as a deviation from the previously identified maximum/minimum, representing only its *sub-pixel* part.

This equation also provides another notable result: if y_0 is a local maximum/minimum point, it means that this value will, by definition, always be less/greater than both y_{+1} and y_{-1}. Thanks to this consideration, it is easily demonstrated that $\hat{\delta}_x$ is always between $-1/2$ and $1/2$.

There is an alternative formulation: denoting $\delta_+ = y_{+1} - y_0$ and $\delta_- = y_{-1} - y_0$, the equation of the parabola becomes

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

and the minimum is found at

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

where it is clear that the position of the minimum is obviously independent of y_0 but solely a function of the deltas.

1.13.2 Minima and Maxima in Higher Dimensions

In two dimensions, but the same reasoning applies to any dimension, the problem of maximizing has to be extended to increasingly complex functions.

The most immediate solution is to analyze the point along each spatial direction independently: in this way, the problem reduces entirely to the one-dimensional case.

If instead a broader neighborhood is to be exploited, the next simplest model to use is the paraboloid, a quadric expressed in the form

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

where the points (x, y) are always understood as deviations from the point to be modeled and z is the value that the function takes at that particular point. Compared to the solution with completely separated axes, with this equation, points not on the axes also actively contribute to the solution of the problem. Clearly, if only the 5 points along the axes are included in the system, the solution will be exactly the same as that seen in the previous section.

Each element of the problem therefore provides a constraint in the form

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

and all the constraints together generate a potentially overdetermined linear system. In this case, there are no remarkable results for obtaining a closed form solution, but the simplest approach is to precalculate a factorization of the system formed by the elements $\mathbf{x}_i$, representing a particular neighborhood of $(0, 0)$, in order to speed up the subsequent resolution when the values z_i are known.

The equation (1.94) has a zero gradient at the point

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

just as in the one-dimensional case, since the two components, that along x and that along y, remain separate during evaluation. This result extends to n-dimensional cases.

1.13.3 Minima, Maxima, and Saddle Points

As written, the model presented earlier applies to both minima/maxima as well as saddle points. However, this model does not account for any local rotations that the function may undergo. While such rotation is, in the first approximation, negligible for minima and maxima, it can be quite significant in the case of saddle points.

The version of the equation (1.94) that considers possible rotations of the axes is

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

The system is fully compatible with that shown in the previous section, with the only difference being that now there are 6 unknowns, thus requiring the processing of at least 6 points in the vicinity of the minimum/maximum/saddle point. Again, there are no remarkable solutions, but it is advisable to factor the matrix of known terms.

The gradient of the function (1.97) vanishes at the point corresponding to the solution of the linear system

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

which can be easily solved using Cramer's rule.

Saddle points can be useful, for example, for precisely locating *subpixel* checkerboard markers.

1.14 The Integral Image

Let I be a generic grayscale image. The value of pixel (x, y) of the integral image $\mathcal{I}$ represents the sum of the values of every pixel of the source image contained within the rectangle $(0, 0) - (x, y)$:

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

With this definition, it is worth noting that the corners of the rectangle are included in the summation (figure 1.10).

The computational trick of using the integral image allows optimization of various algorithms presented in this book, particularly SURF (section 5.4) and the extraction of Haar features (section 6.1).

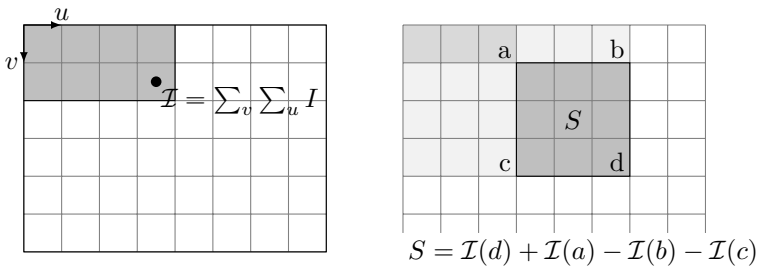


Figure 1.10: Construction of the integral image and use for area calculation.

Thanks to the integral image, it is possible, at a constant computational cost of 4 sums, to obtain the summation of any rectangular sub-region of the image I:

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

The value obtained in this way represents the sum of the elements of the original image within the rectangle (extremes included).

In addition to being able to quickly calculate the summation of any subset of the image, it is also possible to easily obtain convolutions with kernels of a particular shape in a very straightforward manner, while maintaining performance that is invariant with respect to the size of the filter. Examples of convolution masks can be seen in section 6.1.

Chapter 2

Elements of Statistics

Computer vision aims to enable the perception of the world through the *eyes* of a computer. However, whenever one seeks to examine an observable real quantity and relate it to a mathematical model, one must contend with statistics. For this reason, this second chapter will present some statistical techniques that are essential for those developing machine vision algorithms.

2.1 Mean and Variance

It is easy to assume that the notion of the average among numbers is a concept familiar to everyone, at least from a purely intuitive standpoint. In this section, a brief summary is provided, definitions are given, and some interesting aspects will be highlighted.

For n samples of an observed quantity x, the sample mean is denoted as $\bar{x}$ and is given by

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

The sample mean, by definition, is an *empirical* quantity.

If an infinite number of values of x and $\bar{x}$ could be sampled, they would converge to the theoretical expected value. This is known as the Law of Large Numbers.

The expected value (*expectation*, *mean*) of a random variable X is denoted by $\mathrm{E}[X]$ or μ and can be calculated for discrete random variables using the formula

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

and for continuous variables using

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

given the knowledge of the probability distribution $p_X(x)$.

Now we introduce the concept of the mean of a random variable function.

Definizione 4 *Let X be a random variable with probability function $p_X(x)$ and $g(x)$a generic measurable function in x. If the integral*

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

is absolutely convergent, it is referred to as the "mean value of the random variable $Y = g(X)$".

There are certain functions whose mean has a significant meaning. When $g(x) = x$ we refer to first-order statistics (*first statistical moment*), and generally when $g(x) = x^k$ we discuss statistics of k-order. The mean is therefore the first-order statistic, and another statistic of particular interest is the second-order moment:

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

This statistic is important because it allows us to estimate the variance of X.

The variance is defined as the expected value of the square of the random variable X after subtracting its mean, which corresponds to the second moment of the function $g(X) = X - \mathrm{E}[X]$:

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

Assuming X and $\mathrm{E}[X]$ are independent processes, one obtains the simplest and most widely used form of the variance:

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

The square root of the variance is known as the standard deviation and has the advantage of having the same unit of measurement as the observed quantity:

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

Let's extend the concepts discussed so far to the multivariable case. The multivariable case can be viewed as an extension to multiple dimensions, where each dimension is associated with a different variable.

The covariance matrix Σ is the extension to multiple dimensions (or multiple variables) of the concept of variance. It is constructed as

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

where each element of the matrix contains the covariance between the various components of the random vector X. Covariance indicates how the different random variables that make up the vector X are related to each other.

The possible ways to denote the covariance matrix are

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

The notation for cross-covariance is, instead, unambiguous

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

generalization of the concept of the covariance matrix. The cross-covariance matrix $\mathbf{\Sigma}$ has as elements in the position (i, j) the covariance between the random variable X_i and the variable Y_j: It seems that you've entered a placeholder or a command related to a LaTeX environment. If you have specific content or equations you'd like me to translate, please provide that text, and I'll be happy to assist you! The covariance matrix $\mathrm{cov}(X, X)$ is consequently symmetric.

The covariance matrix, which describes the relationships between variables and consequently how uncorrelated they are with each other, is also referred to as the scatter plot matrix. The inverse of the covariance matrix is known as the concentration matrix or precision matrix.

The correlation matrix $\mathbf{r}(X, Y)$ is the normalized cross-covariance matrix with respect to the covariance matrices:

$$\mathbf{r}(X, Y) = \frac{\mathrm{cov}(X, Y)}{\sqrt{\mathrm{var}(X)\mathrm{var}(Y)}} \tag{2.12}$$

This matrix has values that always lie within the range $[-1, 1]$ or $[-100\%, 100\%]$.

2.2 The Gaussian Distribution

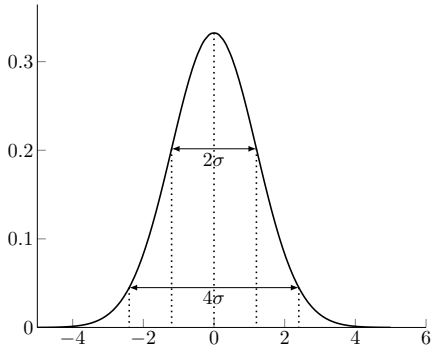


Figure 2.1: Gaussian distribution

The Gaussian distribution is one of the most widely used probability distributions in practical problems, as it models a significant portion of the probability distribution in real-world events. In this document, it is specifically utilized in filters (section 2.12) and Bayesian classifiers (section 4.2), as well as in LDA (section 4.3).

Definition 2 *The standard Gaussian distribution, denoted by the symbol $\mathcal{N}(0; 1)$, is characterized by the density*

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{\left(-\frac{1}{2}x^2\right)} \tag{2.13}$$

Definizione 5 *The general Gaussian distribution $\mathcal{N}(\mu;\sigma^2)$, with $\mu,\sigma \in \mathbb{R}, \sigma^2 \geqslant 0$, is the one obtained from the standard distribution through the transformation $x \mapsto \sigma x + \mu$.*

In the univariate case (univariate Gaussian), the Gaussian has the following distribution function:

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

where μ is the mean and σ^2 is the variance. Within $\pm\sigma$ from μ, 68

The multivariate Gaussian distribution (multidimensional Gaussian) is characterized by a vector $\boldsymbol{\mu}$ of dimension n, representing the mean value of the various components, and a covariance matrix $\boldsymbol{\Sigma}$ of dimensions $n \times n$:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{|\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} \tag{2.15}$$

normal distribution with mean $\boldsymbol{\mu} = [\mu_1, \mu_2, \dots \mu_n]^T$ and covariance $\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11} & \cdots & \sigma_{1n} \\ \vdots & \ddots & \vdots \\ \sigma_{n1} & \cdots & \sigma_{nn} \end{bmatrix}$.

It can be anticipated that the exponent quantity in equation (2.15) is the Mahalanobis distance (section 2.4) between $\mathbf{x}$ and $\boldsymbol{\mu}$.

When the random variables are independent and have equal variance, the matrix $\boldsymbol{\Sigma}$ is a diagonal matrix with all values equal to σ^2, and the multivariate normal distribution simplifies to

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

.

2.2.1 Sampled Gaussian

In practical applications of discrete signal processing, where the Gaussian is used as a convolutional filter, it must also be represented at discrete steps g_k. The Gaussian is typically sampled at a uniform step, but since it has infinite support, a sufficient number of samples are taken for only 3 or 4 times the standard deviation of the Gaussian:

$$g_k = \begin{cases} ce^{-\frac{k^2}{2\sigma^2}} & |k| < 3\sigma \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

with c being a normalization factor chosen such that $\sum_k g_k = 1$.

It is possible to extend the Gaussian to the multidimensional case in a very straightforward manner as follows:

$$g_{k_1,k_2,\dots,k_n} = g_{k_1} \cdot g_{k_2} \dots g_{k_n} \tag{2.18}$$

2.3 Mixture Models

Mixture models are a type of density model consisting of a certain number of density functions, typically Gaussian (*Gaussian Mixture Models*), which are combined to provide a multimodal density. Mixture models allow for the representation of probability distributions in the presence of subpopulations. They can, for example, be used to model the colors of an object and leverage this information for tracking or color-based segmentation.

The *mixture model* is a mathematical formalism sufficient to model a probability distribution as a sum of parametric distributions. In mathematical terms,

$$p_X(x) = \sum_{k=1}^{n} a_k h(x|\lambda_k) \tag{2.19}$$

where $p_X(x)$ is the modeled distribution function, n is the number of components in the model, and a_k is the proportion factor of component k. By definition, $0 < a_k < 1$, $\forall k = 1, \dots, n$, and $a_1 + \dots + a_n = 1$. $h(x|\lambda_k)$ is a probability distribution parameterized by a (generally) λ_k vector. In the case of Gaussian mixture models, the parameter vector consists of the mean and variance of the individual components.

Mixture models are often used when $h(x)$ is known, $p_X(x)$ can be sampled, and the goal is to determine only the parameters a_k and λ_k. An example of a practical situation where this formalism is employed is when analyzing a population composed of distinct subpopulations.

2.4 The Mahalanobis Distance

A common problem is to determine how much an element $\mathbf{x}$ may belong to a probability distribution, allowing for an approximate estimation of whether this element is an *inlier*, meaning it belongs to the distribution, or an *outlier*, indicating that it is external to it.

The Mahalanobis distance [Mah36] provides a measure of an observation normalized with respect to its variance, which is why it is also referred to as the "generalized distance."

Definizione 6 *The Mahalanobis distance of a vector* $\mathbf{x}$ *with respect to a distribution characterized by a mean* μ *and a covariance matrix* $\mathbf{\Sigma}$ *is defined as*

$$d(\mathbf{x}) = \sqrt{(\mathbf{x} - \mu)^\top \mathbf{\Sigma}^{-1} (\mathbf{x} - \mu)} \tag{2.20}$$

the generalized distance of the point from the mean.

Such a distance can be extended (the *generalized squared interpoint distance*) to the case of two vectors $\mathbf{x}$ and $\mathbf{y}$, which are realizations of the same random variable with covariance distribution $\mathbf{\Sigma}$:

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

In the particular case of a diagonal covariance matrix, one retrieves the normalized Euclidean distance. Conversely, when the covariance matrix is precisely the identity matrix (i.e., the components of the distribution are indeed uncorrelated), the formulation above reduces to the classical Euclidean distance.

The Mahalanobis distance allows for the measurement of distances in samples where the units of measurement are unknown, effectively assigning an automatic scaling factor to the data.

2.4.1 Standard Score

An alternative formulation to the Mahalanobis distance is the *Standard Score*. A random variable X is standardized using its empirical statistics by applying the transformation

$$Z = \frac{X - \mu}{\sigma} \tag{2.22}$$

, with μ as the mean and σ as the standard deviation of X. The new random variable Z has, by definition, a mean of zero and a unit variance.

It is possible to use this *Z-score* to filter potential *outliers* from the distribution.

2.5 Transformations of Random Variables

One of the fundamental problems in statistics is understanding how a random variable propagates within a complex system and to what extent it renders the output of that system random.

Let $f(\cdot)$ be a function that transforms the random variable X into the random variable Y, that is, $y = f(x)$, with x realizations of the random variable X, and suppose that f is invertible, meaning that there exists a function $x = g(y)$ such that $g(f(x)) = x$.

Let $\mathcal{I}_x$ be a generic interval of the domain of existence of the values x and $\mathcal{I}_y = \{y : y = f(x), x \in \mathcal{I}_x\}$ its corresponding image. It is evident that the probabilities of the events x in $\mathcal{I}_x$ and y in $\mathcal{I}_y$ must be equal, that is,

$$\int_{\mathcal{I}_y} p_Y(y) dy = \int_{\mathcal{I}_x} p_X(x) dx \tag{2.23}$$

Without loss of generality, it is possible to set the interval $\mathcal{I}_x$ to an infinitesimal value. Under this condition, the relationship (2.23) simplifies to

$$p_Y(y)|dy| = p_X(x)|dx| = p_X(g(y))|dx| \tag{2.24}$$

from which we obtain It seems that you have entered a placeholder for a mathematical block. Please provide the specific content or equations you would like me to translate, and I will be happy to assist you! This relationship can be easily extended to the case of non-injective functions by summing the different intervals, and to the multidimensional case by using the Jacobian instead of the derivative.

2.6 Propagation of Uncertainty

To understand how uncertainty propagates in a system, it is therefore necessary to undergo a process, which may be more or less complex, of both inversion and derivation of the system itself.

In many applications, it is therefore difficult, if not impossible, to obtain analytically the probability distribution at the output of a transformation of a generic input distribution. Fortunately, in practical applications, a lower precision is often required when addressing a problem of uncertainty propagation, typically focusing only on first and second-order statistics and limiting the cases to Gaussian-type probability distributions.

Sum or Difference of Quantities The random variable $Z = X \pm Y$, the sum/difference of independent random variables, has variance (covariance) equal to The variance of the resulting variable is therefore the sum of the individual variances.

Linear Transformations Let It seems that you've entered a placeholder for a mathematical block. Please provide the specific content or equations you'd like translated, and I'll be happy to assist you! a linear system where the random vector $\mathbf{x}$ is associated with the covariance matrix $\text{var}(X)$. The covariance matrix of the resulting random variable $\mathbf{y}$, which is the output of the system, is It seems that you've entered a placeholder or a command for a math block. Please provide the content or context you would like to translate, and I'll be happy to assist you!

This relationship also holds in the case of projections $y = \mathbf{b} \cdot \mathbf{x}$, and similarly to the linear system, the variance of the variable Y becomes

$$\text{var}(Y) = \text{var}(\mathbf{b}^\top X) = \mathbf{b}^\top \text{var}(X)\mathbf{b} \tag{2.25}$$

.

Generalizing the previous cases, the cross-covariance between $\mathbf{Ax}$ and $\mathbf{By}$ can be expressed as:

$$\text{cov}(\mathbf{A}X, \mathbf{B}Y) = \mathbf{A}\text{cov}(X, Y)\mathbf{B}^\top \tag{2.26}$$

As a special case, the cross-covariance between $\mathbf{x}$ and $\mathbf{Ax}$

$$\text{cov}(X, \mathbf{A}X) = \text{var}(X)\mathbf{A}^\top \tag{2.27}$$

It is worth noting that $\text{cov}(Y, X) = \text{cov}(X, Y)^\top = \mathbf{A}\text{var}(X)$.

The examples of uncertainty propagation discussed so far can be further generalized, anticipating important results for the nonlinear case, presented by the affine transformation $f(\mathbf{x})$ defined as

$$f(\mathbf{x}) = f_{\bar{\mathbf{x}}} + \mathbf{A}(\mathbf{x} - \bar{\mathbf{x}}) \tag{2.28}$$

that is, a transformation of random variables $Y = f(X)$ which effectively has a mean value $\bar{y} = f_{\bar{\mathbf{x}}}$ and a covariance matrix $\mathbf{\Sigma}_Y = \mathbf{A}\mathbf{\Sigma}_X\mathbf{A}^\top$.

Nonlinear Transformations The propagation of covariance in the nonlinear case is not easily obtainable in closed form and is generally achieved only in an approximate manner. Techniques such as Monte Carlo simulation can be employed to accurately simulate the probability distribution following a generic transformation at various orders of precision. The linear approximation is still widely used in practical problems; however, as will be discussed in the next section, modern techniques allow for the estimation of covariance at high orders of precision in a relatively straightforward manner.

Normally, for first-order statistics (*first-order error propagation*), the nonlinear transformation f is approximated, through a series expansion, by an affine transformation

$$f(\mathbf{x}) \approx f(\bar{\mathbf{x}}) + \mathbf{J}_f(\mathbf{x} - \bar{\mathbf{x}}) \tag{2.29}$$

with $\mathbf{J}_f$ being the matrix of partial derivatives (Jacobian) of the function f. With this approximation, the result from the previously shown linear affine case in equation (2.28) can be used to determine the covariance matrix of the variable $f(\mathbf{x})$, by replacing the matrix $\mathbf{A}$ with the Jacobian, yielding the covariance

$$\mathbf{\Sigma}_Y = \mathbf{J}_f\mathbf{\Sigma}_X\mathbf{J}_f^\top \tag{2.30}$$

and using the expected mean value $\bar{y} = f(\bar{\mathbf{x}})$.

2.6.1 Examples of Error Propagation

In the field of computer vision, the theory of error propagation is crucial, as basic operations for feature extraction are often affected by noise. For instance, measuring color intensity or determining the position of a particular feature in an image can be influenced by noise, making it essential to understand how this noise impacts subsequent calculations.

The measurement error due to additive noise is formalized as $x = \hat{x} + \varepsilon$, where x is the observed value, $\hat{x}$ is the true value, and ε is the additive noise, for instance, white Gaussian noise with variance σ_x^2.

In the case of vision, it might be interesting to estimate how the error generated by the imprecise observation of a point in the image propagates through the system. In this scenario, the observed variables will be (x, y), image coordinates both affected by localization error with variances σ_x^2 and σ_y^2 respectively, which are normally (at least in the first approximation) uncorrelated with each other.

Using the result from equation (2.29), the generic function $z(x, y)$, which is a function of two random variables, can be approximated to the first order through a Taylor series expansion as From which we can estimate the uncertainty on the value of z, by applying the variance propagation discussed in the previous section, yielding

$$\sigma_z^2 = \left(\frac{\partial z}{\partial x}\right)^2 \sigma_x^2 + \left(\frac{\partial z}{\partial y}\right)^2 \sigma_y^2 \tag{2.31}$$

having chosen to calculate the derivative at (x, y).

With this formulation, several examples can be presented:

Example 1 The propagation of the error of $z = x + y$ is

$$\sigma_z^2 = \sigma_x^2 + \sigma_y^2 \tag{2.32}$$

a result already seen previously.

Esempio 2 The propagation of the error of $z = xy$ is

$$\sigma_z^2 = y^2\sigma_x^2 + x^2\sigma_y^2 \tag{2.33}$$

Example 3 The propagation of the error of $z = \frac{1}{x \pm y}$ is

$$\sigma_z^2 = \frac{\sigma_x^2 + \sigma_y^2}{(x \pm y)^4} \tag{2.34}$$

Esempio 4 The propagation of the error of $z = \frac{x}{y}$ is

$$\sigma_z^2 = \frac{1}{y^2}\sigma_x^2 + \frac{x^2}{y^4}\sigma_y^2 \tag{2.35}$$

Esempio 5 The propagation of the error of $z = \sqrt{x^2 + y^2}$ is

$$\sigma_z^2 = \frac{x^2\sigma_x^2 + y^2\sigma_y^2}{x^2 + y^2} \tag{2.36}$$

It is interesting to note from these equations how the absolute values taken by the variables (x and y in the examples) directly influence the estimation of the error on the final variable z: some variables yield results with lower variance as their intensity increases, while others may exhibit the opposite behavior. For these reasons, depending on the transformation and consequently the model estimation one aims to achieve, certain points in the image may be more important to observe than others.

2.6.2 Error Propagation through Linearized Statistics

The Sigma-Point Approach (SPA) allows for the estimation of the mean and variance of a random variable at the output of a system modeled by a nonlinear function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$.

To estimate the mean and variance, the input random variable $\mathbf{x} \in \mathbb{R}^n$ is approximated by $2n+1$ points $\boldsymbol{\mathcal{X}}_i$, referred to as *sigma points*, each weighted by a weight w_i, in order to achieve a distribution with mean and variance $\bar{\mathbf{x}}$ and $\boldsymbol{\Sigma}_\mathbf{x}$ respectively, that is, parameters exactly equal to those of $\mathbf{x}$.

One way to obtain a set of points whose distribution has the same mean and variance as the original distribution is to take $2n + 1$ *sigma-points* and their respective weights as follows:

$$\begin{aligned} \boldsymbol{\mathcal{X}}_0 &= \bar{\mathbf{x}} \\ \boldsymbol{\mathcal{X}}_i &= \bar{\mathbf{x}} + \zeta \left(\sqrt{\boldsymbol{\Sigma}_\mathbf{x}}\right)_i \\ \boldsymbol{\mathcal{X}}_{i+n} &= \bar{\mathbf{x}} - \zeta \left(\sqrt{\boldsymbol{\Sigma}_\mathbf{x}}\right)_i \end{aligned} \tag{2.37}$$

where ζ is a scaling factor that accounts for how spread out the sigma points are relative to the mean $\bar{\mathbf{x}}$. Associated with each sigma point is a pair of weights w_i^m and w_i^c used in the calculation of the mean and covariance, respectively.

Unlike the *Monte Carlo* methods, the *sigma points* are chosen deterministically to best represent the statistics of the variable.

Once the *sigma-points* are obtained, they are transformed (*unscented transformation*) through the function f into transformed sigma points

$$\boldsymbol{\mathcal{Y}}_i = f(\boldsymbol{\mathcal{X}}_i) \quad {\scriptstyle i=0,\dots,2n} \tag{2.38}$$

.

From these points, it is possible to calculate the mean and variance of the output variable using

$$\begin{array}{l} \bar{\mathbf{y}} \approx \sum_{i=0}^{2n} w_i^m \boldsymbol{\mathcal{Y}}_i \\ \boldsymbol{\Sigma}_{\mathbf{y}} \approx \sum_{i=0}^{2n} w_i^c (\boldsymbol{\mathcal{Y}}_i - \bar{\mathbf{y}})(\boldsymbol{\mathcal{Y}}_i - \bar{\mathbf{y}})^\top \end{array} \tag{2.39}$$

for each point $i = 0, \dots, 2n$. The mean and variance obtained in this way provide a good approximation of the mean and variance of the input distribution transformed by the function f.

The problem addressed by the Sigma Point approach is inherently ill-defined because there exist infinite probability distributions that share the same mean and covariance. The *Unscented Transform* (UT) [JU97], one of the possible *Sigma-Point Approaches*, sets the values as $\zeta = \sqrt{n+\lambda}$, where n is the dimension of the space and λ is a number defined as $\lambda = \alpha^2(n+\kappa) - n$ with $\alpha \in]0.001, 1]$ being a small positive number and κ typically set to 0 or $3-n$. In some articles, $\alpha = 1$ and $\kappa = 3 - n$ are used for Gaussian distributions.

In the *unscented* transformation, the sigma points are weighted points, and the weights differ in the calculation of the mean and the covariance matrix. The *unscented* transformation therefore sets these weights to

$$\begin{array}{l} w_0^m = \frac{\lambda}{n+\lambda} \\ w_0^c = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \\ w_i = w_{i+n} = \frac{1}{2(n+\lambda)} \end{array} \tag{2.40}$$

The difference between the weights w_i^m and w_i^c lies solely in the central term. The value $\beta = 2$ is fixed for Gaussian distributions.

It is important to emphasize that the variants of the *sigma-point* approaches have their weights calculated differently.

2.7 Conditioning in Overparameterized Linear Systems

In section 2.6 and the following sections, we discussed how noise propagates through both linear and nonlinear transformations.

In this section, we examine the complementary case where the estimation of noise on the output variable of the linear system is known, while we seek to determine the estimation of noise on the input variables, specifically the quality with which the solution of a linear system has been obtained. For a significant portion of this section, we refer to the theory discussed in section **??**, which serves as a continuation of that discussion. This will then be integrated in section 3.5 into the broader discourse on regression with nonlinear models.

Let therefore

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.41}$$

be a linear system, ideal and free from noise, with $\mathbf{x}$ being the exact solution to the problem.

A perturbation $\delta\mathbf{b}$ on the column of known terms (observations, outputs), in

$$\mathbf{A}\mathbf{x} = \tilde{\mathbf{b}} \tag{2.42}$$

with $\tilde{\mathbf{b}} = \mathbf{b} + \delta\mathbf{b}$, causes a perturbation $\tilde{\mathbf{x}} = \mathbf{x} + \delta\mathbf{x}$ on the solution of magnitude equal to

$$\delta\mathbf{x} = \mathbf{A}^{-1}\delta\mathbf{b} \tag{2.43}$$

. In this way, we fall back into the previously discussed case of noise propagation in a linear system.

An interesting index consists of calculating the norm of the error in relation to the expected value. This relationship holds as

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}\|\|\mathbf{A}^{-1}\|\frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} = \kappa(\mathbf{A})\frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} \tag{2.44}$$

having defined $\kappa(\mathbf{A})$ as the condition number of the coefficient matrix (*sensitivity matrix*) $\mathbf{A}$. In the particular case where $\mathbf{A}$ is singular, the conditioning of the matrix is set to $\kappa(\mathbf{A}) = \infty$.

Let us now extend the analysis to the case where the system is an overdetermined linear system. For this purpose, it is possible to derive the conditioning of a matrix using an additional property of the SVD decomposition. Let

$$\mathbf{x} = \mathbf{V}\mathbf{S}^{-1}\mathbf{U}^*\mathbf{b} \tag{2.45}$$

be the solution to an overdetermined linear problem obtained through the SVD decomposition method. By expanding equation (2.45), it can be shown that the solution of a linear system, obtained through the SVD decomposition, takes the form

$$\mathbf{x} = \sum \frac{\mathbf{u}_i^\top \mathbf{b}}{\sigma_i} \mathbf{v}_i \tag{2.46}$$

From this formulation, it is evident that when the singular values σ_i are low, any small variation in the numerator is amplified: under the Euclidean norm, the condition number of a matrix is precisely the ratio of the largest singular value to the smallest. The condition number is always positive, and a condition number close to one indicates a well-conditioned matrix.

In summary, conditioning has the following important properties:

- $\kappa(\mathbf{A}) = \kappa(\mathbf{A}^{-1})$
- $\kappa(c\mathbf{A}) = \kappa(\mathbf{A})$ for every $c \neq 0$
- $\kappa(\mathbf{A}) \geq 1$
- $\kappa(\mathbf{A}) = \frac{\sigma_1}{\sigma_n}$ if the norm is Euclidean
- $\kappa(\mathbf{A}) = 1$ if $\mathbf{A}$ is orthogonal

As noted in section **??**, the solution to the perpendicular equations tends to amplify errors compared to alternative solutions. It is indeed straightforward to demonstrate that in this case

$$\kappa\left(\mathbf{A}^\top \mathbf{A}\right) = \left(\frac{\sigma_1}{\sigma^n}\right)^2 \tag{2.47}$$

2.8 The Maximum Likelihood Estimator

From a statistical perspective, the data vector $\mathbf{x} = \{x_1 \dots x_n\}$ consists of realizations of a random variable from an unknown population. The task of data analysis is to identify the population that most likely generated those samples. In statistics, each population is characterized by a corresponding probability distribution, and associated with each probability distribution is a unique parameterization $\boldsymbol{\vartheta}$: by varying these parameters, a different probability distribution should be generated.

Let $f(\mathbf{x}|\boldsymbol{\vartheta})$ be the probability density function (PDF) that indicates the probability of observing $\mathbf{x}$ given a parameterization $\boldsymbol{\vartheta}$. If the individual observations x_i are statistically independent of each other, the PDF of $\mathbf{x}$ can be expressed as the product of the individual PDFs:

$$f(\mathbf{x} = \{x_1 \dots x_n\} | \boldsymbol{\vartheta}) = f_1(x_1|\boldsymbol{\vartheta}) f_2(x_2|\boldsymbol{\vartheta}) \dots f_n(x_n|\boldsymbol{\vartheta}) \tag{2.48}$$

Given a parameterization $\boldsymbol{\vartheta}$, it is possible to define a specific PDF that illustrates the probability of the occurrence of certain data relative to others. In the real case, we face the exact reciprocal problem: the data have been observed, and we need to identify which $\boldsymbol{\vartheta}$ generated that specific PDF.

Definizione 7 *To solve the inverse problem, we define the function $\mathcal{L} : \boldsymbol{\vartheta} \mapsto [0, \infty)$, the likelihood function, defined as*

$$\mathcal{L}(\boldsymbol{\vartheta}|\mathbf{x}) = f(\mathbf{x}|\boldsymbol{\vartheta}) = \prod_{i=1}^{n} f_i(x_i|\boldsymbol{\vartheta}) \tag{2.49}$$

in the case of statistically independent observations.

$\mathcal{L}(\boldsymbol{\vartheta}|\mathbf{x})$ indicates the likelihood of the parameter $\boldsymbol{\vartheta}$ following the observation of the events $\mathbf{x}$.

The principle of the maximum likelihood estimator (*MLE*) $\hat{\boldsymbol{\vartheta}}_{MLE}$, originally developed by R.A. Fisher in the 1920s, selects the best parameterization that best fits the probability distribution generated by the observed data.

In the case of a Gaussian probability distribution, an additional definition is useful.

Definizione 8 *Let ℓ be the log-likelihood function defined as*

$$\ell = \log \mathcal{L}(\boldsymbol{\vartheta}|x_1 \dots x_n) = \sum_{i=1}^{n} \log f_i(x_i|\boldsymbol{\vartheta}) \tag{2.50}$$

having utilized the properties of the logarithm.

The best estimate of the model parameters is the one that maximizes the likelihood, specifically the log-likelihood

$$\hat{\boldsymbol{\vartheta}}_{ML} = \arg\max_{\boldsymbol{\vartheta}} \mathcal{L}(\boldsymbol{\vartheta}|x_1 \dots x_n) = \arg\max_{\boldsymbol{\vartheta}} \sum_{i=1}^{n} \log f_i(x_i|\boldsymbol{\vartheta}) \tag{2.51}$$

since the logarithm is a monotonically increasing function.

It is possible to find in the literature, as an optimal estimator, instead of maximizing the likelihood function, the minimization of its opposite

$$\hat{\boldsymbol{\vartheta}}_{ML} = \arg\min_{\boldsymbol{\vartheta}} \left(-\sum_{i=1}^{n} \log f_i(x_i|\boldsymbol{\vartheta}) \right) \tag{2.52}$$

, that is, the minimization of the *negative log likelihood*.

This formulation is particularly useful when the noise distribution is Gaussian. Let (x_i, y_i) be the realizations of the random variable. In the case of a generic function $y_i = g(x_i; \boldsymbol{\vartheta}) + \epsilon$ with normally distributed noise, constant time, and zero mean, the likelihood is given by

$$\mathcal{L}(\boldsymbol{\vartheta}|\mathbf{x}) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y_i - g(x_i; \boldsymbol{\vartheta}))^2}{2\sigma^2} \right) \tag{2.53}$$

. Therefore, the maximum likelihood estimate (MLE) obtained by minimizing the *negative log likelihood* can be expressed as

$$\hat{\boldsymbol{\vartheta}}_{ML} = \arg\min_{\boldsymbol{\vartheta}} \left(-\sum_{i=1}^{n} \log N(x_i, y_i|\boldsymbol{\vartheta}) \right) = \arg\min_{\boldsymbol{\vartheta}} \sum_{i=1}^{n} (y_i - g(x_i; \boldsymbol{\vartheta}))^2 \tag{2.54}$$

. In other words, the traditional least squares solution is the maximum likelihood estimator in the case of additive Gaussian noise with zero mean.

Now, the m partial derivatives of the log-likelihood form a vector $m \times 1$.

$$\mathbf{u}(\boldsymbol{\beta}) = \frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \begin{bmatrix} \dfrac{\partial \ell}{\partial \beta_1} \\ \vdots \\ \dfrac{\partial \ell}{\partial \beta_m} \end{bmatrix} \tag{2.55}$$

The vector $\mathbf{u}(\boldsymbol{\beta})$ is referred to as the *score vector* (or *Fisher's score function*) of the log-likelihood. If the log-likelihood is concave, the maximum likelihood estimator thus identifies the point for which

$$\mathbf{u}(\hat{\boldsymbol{\beta}}) = \mathbf{0} \tag{2.56}$$

The moments of $\mathbf{u}(\boldsymbol{\beta})$ therefore satisfy important properties: as we noted earlier, the mean of $\mathbf{u}(\boldsymbol{\beta})$ calculated at the maximum likelihood point is equal to zero, and the variance-covariance matrix is

$$\operatorname{var}(\mathbf{u}(\boldsymbol{\beta})) = \mathrm{E}\left[\mathbf{u}(\boldsymbol{\beta})\mathbf{u}(\boldsymbol{\beta})^\top\right] = -\mathrm{E}\left[\frac{\partial^2 \ell(\boldsymbol{\beta})}{\partial \beta_j \partial \beta_k}\right] = \mathcal{I}(\boldsymbol{\beta}) \tag{2.57}$$

The matrix $\mathcal{I}$, defined as the negative of the Hessian, is referred to as the *expected Fisher information matrix*, and its inverse is known as the *observed information matrix*.

2.8.1 Maximum a Posteriori Estimation

The *Maximum a Posteriori estimator*, or *maximum a posteriori probability* (MAP), provides an estimate (one of the) modes of the posterior distribution. Unlike the maximum likelihood estimation, the MAP derives a posterior density using Bayesian theory, combining prior knowledge $f(\boldsymbol{\vartheta})$ with the conditional density $\mathcal{L}(\boldsymbol{\vartheta}|\mathbf{x}) = f(\mathbf{x}|\boldsymbol{\vartheta})$ of likelihood, resulting in the new estimate

$$\hat{\boldsymbol{\vartheta}}_{MAP} = \arg\max_{\boldsymbol{\vartheta}} f(\boldsymbol{\vartheta}|\mathbf{x}) = \arg\max_{\boldsymbol{\vartheta}} \frac{f(\mathbf{x}|\boldsymbol{\vartheta}) f(\boldsymbol{\vartheta})}{f(\mathbf{x})} = \arg\max_{\boldsymbol{\vartheta}} f(\mathbf{x}|\boldsymbol{\vartheta}) f(\boldsymbol{\vartheta}) \tag{2.58}$$

and in the case of uncorrelated events, the formula transforms into

$$\hat{\boldsymbol{\vartheta}}_{MAP} = \arg\max_{\boldsymbol{\vartheta}} \prod_{i=1}^{n} f(x_i|\boldsymbol{\vartheta}) f(\boldsymbol{\vartheta}) = \arg\max_{\boldsymbol{\vartheta}} \left\{ \sum_{i=1}^{n} \log f(x_i|\boldsymbol{\vartheta}) \right\} + \log f(\boldsymbol{\vartheta}) \tag{2.59}$$

where, to simplify the calculations, the properties of logarithms have been utilized.

Clearly, if the prior probability $f(\boldsymbol{\vartheta})$ is uniform, MAP and MLE coincide.

2.9 Weighted Mean with Variance

Having multiple observations with different variances σ_i^2, the goal is to merge these various observations. This is the case, for example, when multiple measurements of the same observable are taken by different sensors at the same time, or with the same sensor for a quantity assumed to be constant but with variable observation noise over time. The objective is to obtain a weighted average of each individual observation of the form

$$\bar{x} = \sum_i w_i x_i \tag{2.60}$$

. The variance of the variable $\bar{x}$ will be defined as follows.

$$\sigma_{\bar{x}}^2 = \sum_i w_i^2 \sigma_i^2 \tag{2.61}$$

. The optimal solution (maximum likelihood estimator) is obtained by minimizing this quantity under the additional constraint $\sum_i w_i = 1$.

The weight that minimizes this quantity is

$$w_i = \frac{\frac{1}{\sigma_i^2}}{\sum_j \frac{1}{\sigma_j^2}} \tag{2.62}$$

In this way, the variance of the mean is lower than the variance of the individual measuring instruments and is given by

$$\sigma_{\bar{x}}^2 = \frac{1}{\sum 1/\sigma_i^2} \tag{2.63}$$

A direct consequence is the ability to combine n readings from the same sensor and the same observable (assuming the observation noise has constant variance) but at different time instances. The final variance is reduced to

$$\sigma_{\bar{x}}^2 = \frac{\sigma_0^2}{n} \tag{2.64}$$

It is possible to construct this result iteratively through the sequence:

$$\bar{x}_{i+1} = (1-k)\bar{x}_i + k x_{i+1} \quad k = \frac{\sigma_{\bar{x}}^2}{\sigma_{\bar{x}}^2 + \sigma_{i+1}^2} \tag{2.65}$$

with k blending factor. Written in this form, the estimate of the observable is in the same format as the one-dimensional Kalman filter (see this result in section 2.12.2): in the absence of process noise, the gain k approaches zero.

2.10 Eigenvalue Analysis

This section bridges analysis, statistics, and classification, addressing topics related to data analysis by leveraging the information provided by eigenvalues and eigenvectors.

2.10.1 PCA

Principal Component Analysis, or the discrete Karhunen-Loeve transform *KLT*, is a technique that has two significant applications in data analysis:

- it allows for the "ordering" of a vector distribution of data in such a way as to maximize its variance, and through this information, reduce the dimensionality of the problem: thus, it is a lossy data compression technique, or alternatively, a method to represent the same amount of information with fewer data;
- it transforms the input data such that the covariance matrix of the output data is diagonal, and consequently, the components of the data are uncorrelated with one another.

Similarly, there are two formulations of the PCA definition:

- it projects the data onto a lower-dimensional space such that the variance of the projected data is maximized;
- it projects the data onto a lower-dimensional space such that the distance between the point and its projection is minimized.

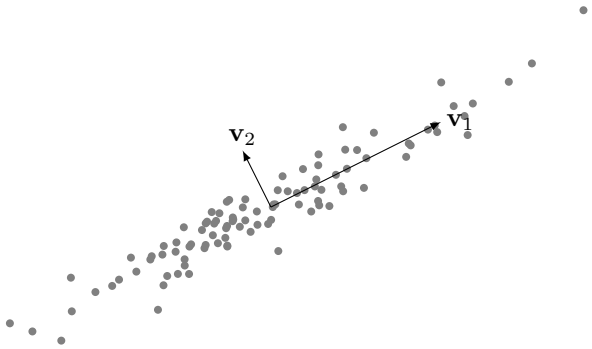


Figure 2.2: Principal Components.

A practical example of dimensionality reduction is the equation of a hyperplane in d dimensions: there exists a basis of the space that transforms the equation of the plane, reducing it to $d-1$ dimensions without losing information, thereby saving one dimension in the problem.

Let us consider $\mathbf{x}_i \in \mathbb{R}^d$ random vectors representing the outcomes of some experiment, realizations of a zero-mean random variable, which can be stored in the *rows*[1] of the matrix $\mathbf{X}$ of dimensions $d \times n$, which therefore stores n random vectors of dimensionality d and with $n > d$. Each line corresponds to a different result $\mathbf{x}$, and the distribution of these experiments must have a mean, at least the empirical one, equal to zero.

Assuming that the points have zero mean (which can always be achieved by simply subtracting the centroid), their covariance of occurrences $\mathbf{x}$ is given by

$$\mathbf{\Sigma} = E(\mathbf{x}\mathbf{x}^\top) \approx \frac{1}{n}\mathbf{X}^\top\mathbf{X} \tag{2.66}$$

. If the input data $\mathbf{x}$ are correlated, the covariance matrix $\mathbf{\Sigma}$ is not a diagonal matrix.

The objective of PCA is to find an optimal transformation $\mathbf{V}$ that transforms the correlated data into uncorrelated data

$$\mathbf{y} = \mathbf{V}^\top\mathbf{x} \tag{2.67}$$

and arranged according to their informational content in such a way that, by selecting a subset of the bases, this approach can reduce the dimensionality of the problem.

If there exists an orthonormal basis $\mathbf{V}$ such that the covariance matrix of $\mathbf{\Sigma}_X$ expressed in this basis is diagonal, then the axes of this new basis are referred to as the principal components of $\mathbf{\Sigma}$ (or of the distribution of X). When a covariance matrix is obtained where all elements are 0 except for those on the diagonal, it indicates that under this new basis of the space, the events are uncorrelated with each other.

This transformation can be found by solving an eigenvalue problem: it can indeed be demonstrated that the elements of the diagonal correlation matrix must be the eigenvalues of Σ_X, and for this reason, the variances of the projection of the vector $\mathbf{x}$ onto the principal components are the eigenvalues themselves:

$$\mathbf{\Sigma V} = \mathbf{V\Delta} \tag{2.68}$$

where $\mathbf{V}$ is the matrix of eigenvectors (orthogonal matrix $\mathbf{V}\mathbf{V}^\top = \mathbf{I}$) and $\mathbf{\Delta}$ is the diagonal matrix of eigenvalues $\lambda_1 \geq \ldots \geq \lambda_d$.

To achieve this result, there are two approaches. Since $\mathbf{\Sigma}$ is a symmetric, real, positive definite matrix, it can be decomposed into

$$\mathbf{\Sigma} = \mathbf{V\Delta V}^\top \tag{2.69}$$

, referred to as the spectral decomposition, with $\mathbf{V}$ being an orthonormal matrix, the right eigenvalues of $\mathbf{\Sigma}$, and $\mathbf{\Delta}$ is the diagonal matrix containing the eigenvalues. Since the matrix $\mathbf{\Sigma}$ is positive definite, all eigenvalues will be positive or zero. By multiplying the equation (2.69) on the right by $\mathbf{V}$, it is shown that it is indeed the solution to the problem (2.68).

This technique, however, requires the explicit computation of $\mathbf{\Sigma}$. Given a rectangular matrix $\mathbf{X}$, the SVD technique allows us to precisely find the eigenvalues and eigenvectors of the matrix $\mathbf{X}^\top\mathbf{X}$, that is, of $\mathbf{\Sigma}$, and therefore it is the most efficient and numerically stable method to achieve this result.

Through the SVD, it is possible to decompose the event matrix $\mathbf{X}$ such that

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

using the Economy/Compact SVD representation where $\mathbf{U}$ are the left singular vectors, $\mathbf{S}$ are the eigenvalues of $\mathbf{\Sigma}$, and $\mathbf{V}$ are the right singular vectors.

[1] In this document, the convention for rows has been chosen: in the literature, one can find both row and column representations of data, and consequently, the nomenclature might differ, referring to $\mathbf{U}$ instead of $\mathbf{V}$ and vice versa.

It is noteworthy that using the SVD, it is not necessary to explicitly compute the covariance matrix $\mathbf{\Sigma}$. However, this matrix can be derived later through the equation

$$\mathbf{\Sigma} = \mathbf{X}^\top \mathbf{X} = \mathbf{V}\mathbf{S}^2\mathbf{V}^\top \tag{2.70}$$

.

By comparing this relation with that of equation (2.69), it can also be concluded that $\mathbf{\Delta} = \mathbf{S}^2$.

It is important to remember the properties of eigenvalues:

- The eigenvalues of $\mathbf{X}\mathbf{X}^\top$ and $\mathbf{X}^\top\mathbf{X}$ are the same.
- The singular values are the eigenvalues of the matrix $\mathbf{X}^\top\mathbf{X}$, which is the covariance matrix;
- The largest eigenvalues are associated with the direction vectors of maximum variance;

and also an important property of the SVD

$$\mathbf{x}^{(l)} = \sum_{i=1}^{l} \mathbf{u}_i \sigma_i \mathbf{v}_i^\top \tag{2.71}$$

which is the rank l approximation closest to $\mathbf{X}$. This fact, combined with the inherent characteristic of SVD to return the singular values of $\mathbf{X}$ ordered from largest to smallest, allows for the approximation of a matrix to one of lower rank.

By selecting the number of eigenvectors with sufficiently large eigenvalues, it is possible to create an orthonormal basis $m \times n$ of the space $\tilde{\mathbf{V}}$ such that $\mathbf{y} \in \mathbb{R}^m$ obtained as a projection

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

represents a reduced-dimensional space that still contains most of the information of the system.

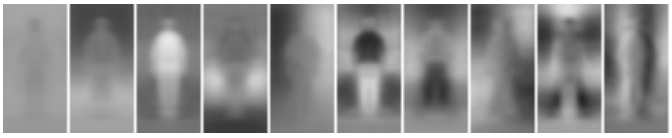


Figure 2.3: Example of the first 10 eigenvectors 24×48 extracted from the Daimler-DB pedestrian dataset

2.10.2 ZCA

PCA is a technique that allows for the decorrelation of components, but this does not prevent the eigenvalues from being different. If all eigenvalues are forced to be equal (see also 2.4.1), and in fact the unit of measurement is changed so that all principal components are equal (the variances are equal), the distribution is referred to as spherical, and the process is known as data whitening.

The matrix referred to as the whitening matrix is denoted as the solution to Zero Components Analysis (ZCA) of the equation

$$\mathbf{Y}^\top \mathbf{Y} = \mathbf{I} \tag{2.72}$$

. After the whitening transformation, the data will not only have zero mean and be decorrelated, but will also exhibit identity covariance.

The whitened matrix from PCA is obtained as

$$\mathbf{X}_{PCA} = \mathbf{V}^\top \mathbf{X}^\top = \mathbf{S}\mathbf{U}^\top \tag{2.73}$$

or equivalently $\mathbf{W}_{PCA} = \mathbf{V}^\top$, while the whitening matrix from ZCA can be derived from

$$\mathbf{X}_{ZCA} = \mathbf{\Delta}^{-1}\mathbf{X}_{PCA} = \mathbf{S}^{-1}\mathbf{X}_{PCA} = \mathbf{S}^{-1}\mathbf{V}^\top\mathbf{X}^\top = \mathbf{U}^\top \tag{2.74}$$

or, in other words, $\mathbf{W}_{ZCA} = \mathbf{S}^{-1}\mathbf{V}^\top$. Most importantly, the remarkable result is given by $\mathbf{X}_{ZCA} = \mathbf{U}^\top$.

It is noteworthy that the matrix after the PCA transformation may have a number of components less than the input data, whereas ZCA always retains the same number of components.

2.11 Elements of Probability

In this section, some useful probability relations are presented for the subsequent section.

Let's define the probability density function (PDF) as

$$p_X(x) = P(X = x) \tag{2.75}$$

to facilitate the transition from the discrete case to the continuous case.

Bayes' theorem (or Bayes' formula) is a relationship that is derived by combining the theorem of compound probability with the theorem of absolute probability.

Starting from the definition of conditional probability $P(A, B) = P(A|B)P(B)$ (*multiplication rule*), we obtain:

$$P(A|B) = \frac{P(A, B)}{P(B)} \tag{2.76}$$

and conversely

$$P(B|A) = \frac{P(B, A)}{P(A)} \tag{2.77}$$

with the consideration that $P(A, B) = P(B, A)$ is obtained

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.78}$$

The same reasoning can be applied in the case of three variables:

$$P(A, B, C) = P(A|B, C)P(B, C) = P(B|A, C)P(A, C) = P(B, A, C) \tag{2.79}$$

leading to Bayes' theorem

$$P(A|B, C) = \frac{P(B|A, C)P(A|C)}{P(B|C)} \tag{2.80}$$

where the dependence on a third variable C is evident.

Another important formula that will be used in the next section is the law of total probability:

$$P(B) = \sum P(A_i, B) = \sum P(A_i)P(B|A_i) \tag{2.81}$$

or in the continuous case

$$p_X(x) = \int p_{X,Y}(X = x, Y = y)dy = \int p(x|Y = y)p(y)dy \tag{2.82}$$

the marginal density of $\mathbf{X}$.

2.12 Bayesian Filters

In this section, the problem of statistical filtering is discussed, which refers to the class of problems where data from one or more sensors affected by noise is available. This data represents the observation of the dynamic state of a system that is not directly observable, but for which an estimate is required. The process through which one seeks to find the best estimate of the internal state of a system is called "filtering," as it is a method for filtering out the various components of noise. The evolution of a system (the evolution of its internal state) must adhere to known physical laws, influenced by a noise component (process noise). It is precisely through the understanding of the equations governing the evolution of the state that it becomes possible to provide a better estimate of the internal state.

A physical process can be viewed, in its state space representation (*State Space Model*), through a function that describes how the state $\mathbf{x}_t$ evolves over time:

$$\dot{\mathbf{x}}_t = f(t, \mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t) \tag{2.83}$$

with $\mathbf{u}_t$ any known inputs to the system, and $\mathbf{w}_t$ a parameter representing the *process noise*, which accounts for the randomness that governs its evolution. Similarly, the observation of the state is also a process influenced by noise, in this case referred to as *observation noise*. In this case, it is also possible to define a function that models the observation $\mathbf{z}_t$ as

$$\mathbf{z}_t = h(t, \mathbf{x}_t, \mathbf{v}_t) \tag{2.84}$$

with $\mathbf{v}_t$ representing the observation noise and being a function solely of the current state.

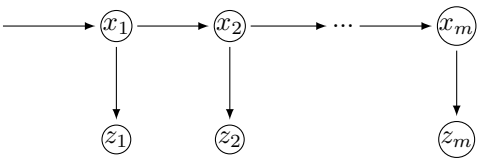


Figure 2.4: Example of evolution and observation of a Markovian system.

This formalism is described in the continuous time domain. In practical applications, signals are sampled at discrete time k and therefore a discrete-time version is typically used in the form

$$\begin{aligned} x_{k+1} &= f_k(x_k, u_k, w_k) \\ z_{k+1} &= h_k(x_k, v_k) \end{aligned} \tag{2.85}$$

where w_k and v_k can be viewed as sequences of white noise with known statistics.

In systems that satisfy the equations (2.85), the evolution of the state is solely a function of the previous state, while the observation is only a function of the current state (see figure 2.4). If a system meets these assumptions, it is said to be a Markov process: the evolution of the system and the observation must depend only on the current state and not on past states. Access to information about the state always occurs indirectly through observation (known as a *Hidden Markov Model*).

Many approaches to estimate the unknown state of a system from a set of measurements do not account for the noisy nature of such observations. It is indeed possible to construct an algorithm that performs nonlinear regression on the observations to obtain estimates of all the states of the problem, solving an optimization problem with a high number of unknowns.

Filters, unlike regressions, aim to provide the best estimate of the variables (state) as the observation data arrives. From a theoretical standpoint, regressions represent the optimal case, while filtering converges to the correct result only after a sufficiently large number of samples.

Bayesian filters aim to estimate, at the discrete time instant k, the state of the random variable $\mathbf{x}_k \in \mathbb{R}^n$ given an indirect observation of the system, $\mathbf{z}_k \in \mathbb{R}^m$.

Filtering techniques allow for both obtaining the best estimate of the unknown state $\mathbf{x}_k$ and the multivariate probability distribution $p(\mathbf{x}_k)$ that represents the knowledge of the state itself.

Given the observation of the system, it is possible to define a probability density of $\mathbf{x}_k$ *a posteriori* to the observation of the event $\mathbf{z}_k$ due to the additional knowledge gained from such observation:

$$p^+(\mathbf{x}_k) = p(\mathbf{x}_k|\mathbf{z}_k) \tag{2.86}$$

where the conditional probability $p(\mathbf{x}_k|\mathbf{z}_k)$ indicates the probability that the hidden state is $\mathbf{x}_k$ given the observation $\mathbf{z}_k$. The "function" $p(\mathbf{x}_k|\mathbf{z}_k)$ represents the measurement model. In the literature, the posterior distribution $p^+(\mathbf{x}_k)$ is also referred to as *belief*.

Applying Bayes' theorem to equation (2.86) yields

$$p(\mathbf{x}_k|\mathbf{z}_k) = c_k p(\mathbf{z}_k|\mathbf{x}_k) p(\mathbf{x}_k) \tag{2.87}$$

with c_k as the normalization factor such that $\int p(\mathbf{x}_k|\mathbf{z}_k) = 1$. The knowledge of $p(\mathbf{z}_k|\mathbf{x}_k)$ is essential, which represents the probability that the observation is precisely the quantity $\mathbf{z}_k$ observed given the possible state $\mathbf{x}_k$. The use of Bayes' theorem to estimate the state given the observation is the reason why this class of filtering is referred to as *Bayesian*.

In addition to the *a posteriori* knowledge of the probability distribution, it is possible to leverage further information to improve the estimation: the *a priori* knowledge regarding the observation, obtained from the constraint that the state does not evolve in a completely unpredictable manner but rather can only evolve in certain ways with specific probabilities. These ways in which the system can evolve are solely a function of the current state.

The Markovian process hypothesis implies that the only past state influencing the evolution of the system is the state at time $k-1$, that is, $p(x_k|x_{1:k-1}) = p(x_k|x_{k-1})$.

It is therefore possible to perform the prediction *a priori*, thanks to the Chapman-Kolmogorov equation:

$$p^-(\mathbf{x}_k) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) p(\mathbf{x}_{k-1}) d\mathbf{x}_{k-1} \tag{2.88}$$

where $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$ represents the dynamics of the system (*dynamic model*) and $\mathbf{u}_k$ are the potential inputs that influence the evolution of the system, of which, however, the knowledge is complete.

From the knowledge of the *a priori* state and the observation z_k, it is possible to rewrite equation (2.86) in the state update equation

$$p^+(\mathbf{x}_k) = c_k p(\mathbf{z}_k|\mathbf{x}_k) p^-(\mathbf{x}_k) \tag{2.89}$$

.

The state is estimated by alternating between a prediction phase (a priori estimation) and an observation phase (a posteriori estimation). This iterative process is known as Recursive Bayesian Estimation.

The techniques described in this section will refer only to the most recent observation available for state estimation, for reasons of performance and simplicity. Formally, it is possible to extend the discussion to the case where all observations are utilized to obtain a more accurate estimate of the state. In this case, the filtering and prediction equations become

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{z}_{1:k}) &= \int p(\mathbf{x}_{1:k}|\mathbf{z}_{1:k})d\mathbf{x}_{1:k-1} \\ p(\mathbf{x}_{k+1}|\mathbf{z}_{1:k}) &= \int p(\mathbf{x}_{k+1}|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k})d\mathbf{x}_k \end{aligned} \tag{2.90}$$

For simplicity and due to the reduced computational burden, typically only the latest observation is evaluated; however, in certain cases (for example, in particle filters), it is possible to incorporate the knowledge of the entire past history into the equations quite easily.

As an estimate of continuous variables, it is not possible to exploit Bayesian theory "directly"; however, several approaches have been proposed in the literature to enable efficient estimation both from a computational perspective and in terms of memory usage.

Depending on whether the problem is linear or non-linear, and whether the noise probability distribution is Gaussian or not, each of these filters performs in a more or less optimal manner.

The Kalman Filter (section 2.12.2) is the optimal filter when the problem is linear and the noise distribution is Gaussian. The Extended Kalman Filter and the Unscented Kalman Filter, discussed in sections 2.12.4 and 2.12.5 respectively, are sub-optimal filters for nonlinear problems with Gaussian noise distribution (or slightly deviating from it). Finally, particle filters provide a sub-optimal solution for nonlinear problems with non-Gaussian noise distribution.

The *grid-based* filters (section 2.12.1) and particle filters (section 2.12.8) operate on a discrete representation of the state, while the Kalman, Extended, and Sigma-Point filters work on a continuous representation of the state.

Kalman, Extended Kalman, and Sigma Point Kalman filters estimate the uncertainty distribution (of the state, process, and observation) as a single Gaussian. There are multimodal extensions such as *Multi-hypothesis tracking* (*MHT*) that allow the application of Kalman filters to distributions modeled as mixtures of Gaussians, while particle filters and *grid-based* methods are inherently multimodal.

An excellent *survey* on Bayesian filtering is [Che03].

2.12.1 Grid-based Methods

Grid-based approaches are particularly well-suited for problems where the state assumes only a limited number of discrete values (these are referred to as Discrete Filters), while they also allow for an approximate estimation in the case of continuous states (*histogram filters*) transformed into discrete representations through spatial quantization. Each element of the grid (or histogram) is associated with the probability that the state is actually located in that particular cell. The theory of Bayesian filters (hence multimodal distributions and strongly nonlinear systems) is directly utilized, albeit limited to the discrete points where the state can exist.

Let us assume that m points are used to represent the state $\mathbf{x} \in \mathbb{R}^n$. If the original state is continuous, this is clearly an approximation, and it is preferable that $m \gg n$. At each iteration k, there are therefore $\mathbf{x}_{i,k} \in \mathbb{R}^n$ with $i = 1, \ldots, m$ possible states associated with a probability distribution $p_{i,k}$ that evolves over time according to the dynamics of the problem.

The previously discussed equations hold, namely the *a priori* estimate:

$$p^-_{i,k} = \sum_{j=1}^{m} p(x_{i,k}|x_{j,k-1})p^+_{j,k-1} = \sum_{j=1}^{m} f_{i,j}p^+_{j,k-1} \quad \forall i \tag{2.91}$$

and the *a posteriori* state update equation for the observation z_k:

$$p^+_{i,k} = c_k p(z_k|x_{i,k})p^-_{i,k} \quad \forall i \tag{2.92}$$

with c_k always being the normalization factor such that $\sum p^+_i = 1$.

The *grid-based* methods thus allow for the direct application of Bayesian recursive theory.

2.12.2 Kalman Filter

The Kalman filter [WB95] aims to estimate the internal state $\mathbf{x} \in \mathbb{R}^n$, which is not accessible, of a discrete-time system, given that the model knowledge is complete. In fact, the Kalman filter is the optimal recursive estimator: if the noise in the problem is Gaussian, the Kalman filter provides the least squares estimate of the internal state of the system.

For historical reasons, the Kalman filter specifically refers to the filtering of a system where the state transition and the observation are linear functions of the current state.

According to linear systems theory, the dynamics of a continuous-time "linear" system is represented by a differential equation of the form

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{w}(t) \tag{2.93}$$

state update equation, to which an indirect observation of this state is associated through a linear system:

$$\mathbf{z}(t) = \mathbf{H}(t)\mathbf{x}(t) + \mathbf{v}(t) \tag{2.94}$$

with $\mathbf{z} \in \mathbf{R}^m$ the observable.

The discrete-time Kalman filter assists real systems where the world is sampled at discrete intervals, transforming the continuous-time linear system into a linear system of the form

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \mathbf{w}_k \\ \mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \end{cases} \tag{2.95}$$

If the system evolves according to this model, it is referred to as a *Linear-Gaussian State Space Model* or a *Linear Dynamic System*. If the values of the matrices are time-independent, the model is termed *stationary*.

The variables w_k and v_k represent the process noise and observation noise, respectively, with a mean value of zero $\bar{w_k} = \bar{v_k} = 0$ and known variances $\mathbf{Q}$ and $\mathbf{R}$ (assuming white Gaussian noise). $\mathbf{A}$ is a state transition matrix, $n \times n$ is a matrix $\mathbf{B}$ that connects the optional control input $n \times l$ with the state $\mathbf{u} \in \mathbb{R}^l$, and finally $\mathbf{x}$ is a matrix $\mathbf{H}$ that links the state with the measurement $m \times n$. All these matrices, representing the system model, must be known with absolute precision, otherwise systematic errors will be introduced.

The Kalman filter is a recursive estimation filter that requires at each iteration the knowledge of the estimated state from the previous step $\hat{\mathbf{x}}_{k-1}$ and the current observation $\mathbf{z}_k$, which serves as an indirect observation of the system's state.

Let $\hat{\mathbf{x}}_k^-$ be the *a priori* estimate of the system state, based on the estimate obtained at time $k-1$ and the dynamics of the problem, and let $\hat{\mathbf{x}}_k$ be the *a posteriori* estimate of the state based on the observation $\mathbf{z}_k$. From these definitions, it is possible to define the error of the *a priori* and *a posteriori* estimates as

$$\begin{aligned} \mathbf{e}_k^- &= \mathbf{x}_k - \hat{\mathbf{x}}_k^- \\ \mathbf{e}_k &= \mathbf{x}_k - \hat{\mathbf{x}}_k \end{aligned} \tag{2.96}$$

To these errors, it is possible to associate

$$\begin{aligned} \mathbf{P}_k^- &= \mathrm{E}[\mathbf{e}_k^-\mathbf{e}_k^{-\top}] \\ \mathbf{P}_k &= \mathrm{E}[\mathbf{e}_k\mathbf{e}_k^\top] \end{aligned} \tag{2.97}$$

the *a priori* and *a posteriori* covariance matrices, respectively.

The objective of the Kalman filter is to minimize the covariance of the *a posteriori* error $\mathbf{P}_k$ and to provide a method for obtaining the estimate of $\hat{\mathbf{x}}_k$ given the a priori estimate $\hat{\mathbf{x}}_k^-$ and the observation $\mathbf{z}_k$.

The Kalman filter provides a *posterior* state estimate through a linear combination of the previous state estimate and the observation error:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-) \tag{2.98}$$

shifting the estimation problem to that of deriving the gain factor $\mathbf{K}_k$ (*blending factor*). The difference $\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-$ is referred to as the *residual*, or *innovation*, and represents the discrepancy between the predicted observation and the actual observation. It is noteworthy that the metric used to calculate the residual may depend on the specific characteristics of the problem.

The Kalman filter is typically presented in two phases: the time update (prediction phase) and the measurement update (observation phase).

In the first phase, the *a priori* estimate of both $\hat{\mathbf{x}}_k$ and the covariance $\mathbf{P}_k$ is obtained. The *a priori* estimate $\hat{\mathbf{x}}_k^-$ derives from a good understanding of the system dynamics given by equation (2.95):

$$\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k \tag{2.99}$$

Similarly, the *a priori* estimate of the error covariance is updated:

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top + \mathbf{Q}_k \tag{2.100}$$

These are the best estimates of the state and the covariance at the instant k that can be obtained *a priori* from the observation of the system.

In the second phase, the gain

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^\top \left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^\top + \mathbf{R}_k\right)^{-1} \tag{2.101}$$

is calculated to minimize the *a posteriori* covariance, and with this factor, the *a posteriori* state is updated using equation (2.98).

Using this value for the gain $\mathbf{K}$, the *a posteriori* estimate of the covariance matrix becomes

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^- \tag{2.102}$$

To unify the various forms of the Kalman filters, these equations can be expressed using the variance-covariance matrices as follows:

$$\begin{aligned} \operatorname{cov}(x_k, z_k) &= \mathbf{P}_k^- \mathbf{H}_k^\top \\ \operatorname{cov}(z_k) &= \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\top \end{aligned} \tag{2.103}$$

so that we can express equation (2.101) as

$$\mathbf{K}_k = \operatorname{cov}(x_k, z_k) \left(\operatorname{cov}(z_k) + \mathbf{R}_k\right)^{-1} \tag{2.104}$$

and, by substituting the covariances (2.103) into (2.102), we obtain

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k \operatorname{cov}(x_k, z_k)^\top \tag{2.105}$$

It can be easily observed that the covariance matrix and the Kalman gain do not depend at all on the state, the observations, or the residual, and have an independent history.

However, the Kalman filter requires an initial value for the state variable and the covariance matrix: the initial state value should be as close as possible to the true value, and the degree of similarity to this value should be reflected in the initial covariance matrix.

One-Dimensional Kalman Filter

It is interesting to illustrate, as an example, the simplified case of a one-dimensional Kalman state filter that coincides with the observable. The transition and observation equations are

$$\begin{aligned} x_i &= x_{i-1} + u_i + w_i \\ z_i &= x_i + v_i \end{aligned} \tag{2.106}$$

where w_i is the process noise whose variance q_i represents the estimate of the probability of variation of the signal itself (low if the signal varies little over time, high if the signal varies significantly), while v_i is the observation noise with variance r_i, which is the noise associated with the observation of the state.

The prediction cycle is very simple and becomes:

$$\begin{aligned} x_i^- &= x_{i-1} + u_i \\ p_i^- &= p_{i-1} + q_i \end{aligned} \tag{2.107}$$

The Kalman gain k becomes

$$k_i = \frac{p_i^-}{p_i^- + r_i} \tag{2.108}$$

and finally the observation phase becomes

$$\begin{aligned} x_i &= x_i^- + k_i(z_i - x_i^-) = k_i z_i + (1 - k_i) x_i^- \\ p_i &= (1 - k_i) p_i^- \end{aligned} \tag{2.109}$$

It is usually possible to estimate the value of r a priori, while the value of q must be determined through experiments.

As seen in the first of the equations (2.109), the factor k is essentially a *blending factor* between the observation of the state and the previously estimated state.

In the one-dimensional case, it is easy to see how the gain k and the variance p are independent of the state and the observations, let alone the error. If r and q do not vary over time, k and p are numerical sequences that converge to a constant value determined solely by the characterization of the noise, regardless of the initial values. This result should be compared with what is obtained from equation (2.65).

2.12.3 Correlated Noise

In the case where the noise is not merely additive, but propagates through the system via some linear transformation, the Kalman filter is generalized to

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k + \mathbf{W}_k \mathbf{w}_k \\ \mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{V}_k \mathbf{v}_k \end{cases} \tag{2.110}$$

The process noise is correlated through a matrix $\mathbf{W}_k$ to the source, and the observation noise through a matrix $\mathbf{V}_k$.

In this case, it is possible to apply the same equations of the Kalman system by introducing the substitutions

$$\begin{aligned} \mathbf{Q}'_k &= \mathbf{W}_k \mathbf{Q}_k \mathbf{W}_k^\top \\ \mathbf{R}'_k &= \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k^\top \end{aligned} \tag{2.111}$$

This result will be useful in the following section on the extended Kalman filter.

Clearly, if the matrices $\mathbf{W}_k$ and $\mathbf{V}_k$ are identities, meaning that the noise is simply additive, the expression simplifies and reverts to the form seen previously.

2.12.4 Extended Kalman Filter

The Extended Kalman Filter (EKF) is a nonlinear version of the Kalman filter used when the evolution or observation of the system state is nonlinear.

A discrete-time nonlinear system, consisting of state evolution and observation, can be expressed in a generalized form as

$$\begin{cases} \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \\ \mathbf{z}_k &= h(\mathbf{x}_k, \mathbf{v}_k) \end{cases} \tag{2.112}$$

where, in addition to the state $\mathbf{x}_k$ and the inputs $\mathbf{u}_k$, the process errors $\mathbf{w}_k$ and the observation errors $\mathbf{v}_k$ can also non-linearly affect the evolution of the state f and the observation h, thereby generalizing the concept of additive noise used previously.

In order to be applied, EKF requires the computation of the Jacobians of both f and h. By applying the theory presented in section 2.6 regarding the propagation of uncertainty in nonlinear functions, it is possible to leverage the same mathematical formulations used for the linear Kalman case on nonlinear functions by utilizing the matrices

$$\begin{aligned} \mathbf{A}_k &= \left.\frac{\partial f(\mathbf{x}, \mathbf{u}_k, \bar{\mathbf{w}})}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^-} & \mathbf{W}_k &= \left.\frac{\partial f(\hat{\mathbf{x}}_k^-, \mathbf{u}_k, \mathbf{w})}{\partial \mathbf{w}}\right|_{\bar{\mathbf{w}}} \\ \mathbf{H}_k &= \left.\frac{\partial h(\mathbf{x}, \bar{\mathbf{v}})}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^-} & \mathbf{V}_k &= \left.\frac{\partial h(\mathbf{x}, \mathbf{v})}{\partial \mathbf{v}}\right|_{\bar{\mathbf{x}}} \end{aligned} \tag{2.113}$$

and employing the update equation

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-)) \tag{2.114}$$

. It is also worth noting that the computation of the residual $\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-)$ can be a nonlinear function (for example, when comparing angles, where there exists a periodicity of the error).

Compared to the linear Kalman filter, the Extended Kalman Filter (EKF) is considered a sub-optimal choice as an estimator, yet it remains widely accepted and utilized in practical applications. The Extended Kalman Filter, by its very design, achieves only first-order accuracy but still allows for results that are close to optimal in scenarios where the second derivatives are negligible.

2.12.5 Sigma-Point Kalman Filter

An alternative in the non-linear case to the Extended Kalman Filter is the Sigma Point Kalman Filter. In the results reported from various experiments, in the case of the non-linear functions f and h, the *Sigma Point Kalman Filter* (SPKF) tends to provide better performance compared to the EKF: the statistical linearized error propagation (SPKF) is generally superior to the propagation based on Taylor series expansion (EKF).

Not only the state, but also the various points around the mean (the *sigma points*) are propagated through the functions that constitute the state update and observation in the Kalman filter. The advantage of the Sigma Point Kalman Filter (SPKF) is that it does not require the computation of Jacobians and typically allows for a better estimation of the mean and variance of the process.

The Unscented Kalman filter is one of the various versions of the Kalman filter based on Sigma Points. In this case, the theory for uncertainty propagation discussed in section 2.6.2 is utilized to estimate the mean and covariance of the a priori state and the observation error.

Even with the Unscented filter, it is possible to handle the case where noise enters the system in a non-additive manner. To generalize the case of non-additive noise, we define, in order to maintain a syntax consistent with that discussed in section 2.6.2, a variable called *augmented state* $\mathbf{x}^a \in \mathbb{R}^{n^a}$ with $n^a = n + q$ formed by the state $\mathbf{x} \in \mathbb{R}^n$ and the process noise w, which has zero mean, so as to use the function

$$\boldsymbol{\mathcal{X}}^- = f(\mathbf{x}_{k-1}^a, \mathbf{u}_k) \tag{2.115}$$

for state update that allows for the nonlinear and non-additive consideration of the contribution from the process noise. Similarly, we define the augmented covariance matrix as:

$$\mathbf{P}_{\mathbf{x}}^a = \begin{bmatrix} \mathbf{P}_{\mathbf{x}} & 0 \\ 0 & \mathbf{Q} \end{bmatrix} \tag{2.116}$$

In the case where the process noise is additive, the system becomes similar to that of the linear Kalman filter in the form

$$\mathbf{P}_k^- = \sum_{i=0}^{2n} w_i^c (\boldsymbol{\mathcal{X}}_i^- - \bar{\boldsymbol{\mathcal{X}}}_i^-)(\boldsymbol{\mathcal{X}}_i^- - \bar{\boldsymbol{\mathcal{X}}}_i^-)^\top + \mathbf{Q}_k \tag{2.117}$$

From the *sigma points* $\boldsymbol{\mathcal{X}}_i^-$, projected through f and representing the *a priori* state distribution, it is possible to generate additional sigma points in order to obtain the *a priori* observation estimate:

$$\boldsymbol{\mathcal{Z}}_i = h(\boldsymbol{\mathcal{X}}_i^-) \tag{2.118}$$

with which to calculate the most probable value of the observation $\hat{\mathbf{z}}$ by weighting the results $\boldsymbol{\mathcal{Z}}_i$ with the weights of the associated *sigma points* as in equation (2.39). In this case as well, the observation noise can be incorporated as an augmented state or, if assumed to be additive and independent, it can be added to the covariance matrix.

By utilizing the knowledge of the sigma points $\boldsymbol{\mathcal{X}}_i^-$ and $\boldsymbol{\mathcal{Z}}_i$, it is possible to easily obtain the covariance $\text{cov}(\boldsymbol{\mathcal{Z}})$ and also the cross-covariance $\text{cov}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}})$ by generalizing the equation (2.39):

$$\text{cov}\left(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}}\right) \approx \sum_{i=0}^{2n} w_i^c (\boldsymbol{\mathcal{X}}_i - \bar{\mathbf{x}})(\boldsymbol{\mathcal{Z}}_i - \bar{\mathbf{z}})^\top \tag{2.119}$$

Given the knowledge of the covariance $\text{cov}(\boldsymbol{\mathcal{Z}})$ and the cross-covariance $\text{cov}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}})$, the Kalman *sigma-point* gain becomes exactly as expressed by equation (2.104), and the covariance update $\mathbf{P}_k$ follows the equation (2.105).

2.12.6 IEKF and ISPKF

The extended Kalman filter utilizes the Jacobian of the observation function h centered at $\hat{\mathbf{x}}^-$, the *a priori* state, and, through the knowledge of the observation, enables the estimation of the *a posteriori* state.

In fact, this procedure is precisely a single iteration of the Gauss-Newton method.

It is possible to increase the iterations in order to obtain the class of iterative Kalman filters, which typically demonstrate significantly better performance than their non-iterative counterparts.

The only difference compared to the respective non-iterative filters lies in the observation part (see equation (2.114)), which is replaced by iterations in the form of:

$$\mathbf{x}_{i+1} = \hat{\mathbf{x}} + \mathbf{K}(\mathbf{z} - h(\mathbf{x}_i) - \mathbf{H}_i(\hat{\mathbf{x}} - \mathbf{x}_i)) \tag{2.120}$$

with the gain $\mathbf{K}$ calculated iteratively as

$$\mathbf{K} = \mathbf{P}\mathbf{H}_i^\top (\mathbf{H}_i \mathbf{P} \mathbf{H}_i^\top + \mathbf{R})^{-1} \tag{2.121}$$

and using the value $\mathbf{x}_0 = \hat{\mathbf{x}}^-$ as the initial value for minimization.

The value of $\mathbf{K}$, associated with the last iteration, is finally used to update the process covariance matrix.

The same procedure can be applied to the SPKF filter to obtain the *Iterated Sigma Point Kalman Filter* [SSM06], where the iteration to compute the state takes the form

$$\mathbf{x}_{i+1} = \hat{\mathbf{x}} + \mathbf{K}\left(\mathbf{z} - h(\mathbf{x}_i) - \text{cov}(\mathcal{X}, \mathcal{Z})^\top \mathbf{P}^{-1}(\hat{\mathbf{x}} - \mathbf{x}_i)\right) \tag{2.122}$$

2.12.7 Gaussian Mixture Kalman Filter

The *Gaussian sum Kalman filters* (GS-KF) describes a Bayesian filter where the multimodal distribution is approximated as a mixture of Gaussians. It is handled in the same manner as the linear Kalman filter in the case of linear transformations, treating each state separately.

2.12.8 Particle Filter

The linear and quasi-linear approaches proposed by Kalman can be applied to problems where the state is Gaussian or nearly Gaussian with a unimodal distribution: the state estimate at time k is a direct function of the unique state estimate at time $k-1$ and the covariance of that estimate.

When it is required to derive the non-Gaussian probability distribution of the system state $p(x_k; u_{k-1}; z_k)$ at time k, as a function of the inputs and observations, Kalman-type approaches are no longer satisfactory.

Grid-based approaches are suitable for those problems, which are quite uncommon, where the state is discretizable and finite. *Histogram-based/occupancy grid* approaches, on the other hand, are applicable to a broader class of problems; however, due to the uniform sampling of the state, they scale poorly as the dimensions increase.

Consider again the result expressed by equation (2.4): to extract a generic statistic $h(\cdot)$ (for example, the mean or variance) from a probability distribution $p(x)$, one utilizes the expression

$$\bar{h} \overset{def}{=} \int_X h(x)p(x)dx \tag{2.123}$$

. In cases where such an estimate cannot be obtained analytically, it is still possible to derive it indirectly through the analysis of x_i independent samples, with $1 \leq i \leq N$, randomly drawn from a distribution that is exactly p.

Given the samples x_i generated in this manner, the Monte Carlo estimate of $h(\cdot)$ is given by

$$\bar{h} \approx \frac{1}{N} \sum_{i=1}^{N} h(x_i) \tag{2.124}$$

Monte Carlo does not solve all problems nor does it suggest how to efficiently obtain random samples. The issue becomes sensitive in multidimensional cases where the areas where the probability takes on significant values are extremely small. The objective of *Importance Sampling* (IS) is to sample the distribution $p(x)$ in "important" regions in order to maximize computational efficiency.

The idea of *Importance Sampling* is to take a simpler distribution $q(x)$ (*Importance density*) instead of the true $p(x)$, which is usually difficult to sample (or reproduce), by making the substitution

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

where we have introduced the weight system $w(x)$. Through the use of appropriate weights, it is thus possible to modify equation (2.124) to

$$\bar{h} \approx \frac{1}{N}\sum_{i=1}^{N} w_i h(x_i) \tag{2.125}$$

where $w_i \propto W_i = p(x_i)/q(x_i)$ represents a corrective weight, the importance factor (*important weights*), to convert the support distribution q to the true distribution p. The weights W_i must be normalized

$$w_i = \frac{W_i}{\sum W_i} \tag{2.126}$$

to be usable.

The distribution $q(x)$ is more similar to $p(x)$, the more accurate the estimate will be. On the other hand, the distribution $q(x)$ should be very simple to sample, for example by choosing a uniform or Gaussian distribution.

Given the knowledge of Bayesian filters and Monte Carlo techniques, it is possible to address the theory of particle filters. The state at time k is represented by a set of samples (*particles*), and each sample is a hypothesis of the state to be evaluated. One can refer to a series of particles obtained *a priori* of the observation by applying equation (2.125) to the state evolution function.

If Bayesian theory is applied directly to the samples of the estimated distribution, it is possible to modify the weights w_i associated with the samples while simultaneously using the system and perception model (*Sequential Importance Sampling*):

$$w_{k,i} \propto w_{k-1,i}\frac{p(z_k|x_{k,i})p(x_{k,i}|x_{k-1,i})}{q(x_{k,i}|x_{k-1,i}, z_k)} \tag{2.127}$$

In this way, the initial samples remain the same, but only the weights w_i associated with them change.

When possible, it is advantageous to use the *Important density* as the *a priori* distribution

$$q(x_{k,i}|x_{k-1,i}, z_k) = p(x_{k,i}|x_{k-1,i}) \tag{2.128}$$

so that, when introduced in (2.127), it yields

$$w_{k,i} \propto w_{k-1,i}p(z_k|x_{k,i}) \tag{2.129}$$

The issue with the SIS approach is that after a few iterations, only a few particles will have a non-negligible weight factor (*weight degeneracy*).

BootStrap/Sequential Importance Resampling

A simpler solution is the *Sequential Importance Resampling*, where the weights do not depend on previous iterations; instead, it is the samples that change, following a *resampling* phase.

The resampling phase involves generating a new set of particles x' by resampling N_s times a discretely approximated version of $p(\mathbf{x}_k|\mathbf{z}_k)$ given by

$$p(\mathbf{x}_k|\mathbf{z}_k) \approx \sum_{i=1}^{N_s} w_{k,i}\delta(\mathbf{x}_k - \mathbf{x}_{k,i}) \tag{2.130}$$

having defined

$$w_{k,i} \propto p(\mathbf{z}_k|\mathbf{x}_k) \tag{2.131}$$

SIR filters do not avoid the degenerate case (in fact, they effectively eliminate unlikely particles), however, they lead to significant computational savings and focus the search for the solution around the most probable states.

There are various algorithms for performing resampling. A non-exhaustive list of such algorithms includes: *Simple Random Resampling*, *Roulette Wheel / Fitness Proportionate Selection*, *Stochastic Universal Sampling*, *Multinomial Resampling*, *Residual Resampling*, *Stratified Resampling*, and *Systematic Resampling*.

2.12.9 Parameter Estimation

Kalman, in all its variants, is classically viewed as a filter or state estimator. However, it is widely used, primarily in *machine learning*, to apply these techniques for estimating the parameters of a model (the meta-model):

$$\mathbf{y}_k = f(\mathbf{x_k}, \boldsymbol{\beta}) \tag{2.132}$$

where $\mathbf{y}_k$ are the system outputs, $\mathbf{x_k}$ are the inputs, and f is a function based on the parameters $\boldsymbol{\beta}$ to be estimated. The concept of training, or *fitting*, the model consists of determining the parameters $\boldsymbol{\beta}$.

Kalman allows for the determination of parameters, which may be variable, of the model by using as the state to be estimated precisely $\boldsymbol{\beta}$, thereby obtaining an iterative system of the form

$$\begin{cases} \boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \mathbf{w}_k \\ \mathbf{y}_k = f(\mathbf{x}_k, \boldsymbol{\beta}_k) \end{cases} \tag{2.133}$$

, where the optional noise $\mathbf{w}_k$ is used to model any variations of the model over time: the choice of the variance of $\mathbf{w}$ determines the responsiveness to changes in the model parameters.

2.12.10 Alpha-Beta Filter

The *alpha-beta filter* can be viewed as a simplified version of the Kalman filter, where the state is represented by only two variables, one of which is the integral of the other. By drawing a simple analogy with physical systems, we can denote these variables as position $\mathbf{x}$ and velocity $\mathbf{v}$. Assuming that the velocity remains constant over a small time interval ΔT, the *a priori* estimate (prediction) of the position at time k is given by

$$\hat{\mathbf{x}}_k^- = \hat{\mathbf{x}}_{k-1} + \Delta T \mathbf{v}_{k-1} \tag{2.134}$$

while the velocity is always considered constant:

$$\hat{\mathbf{v}}_k^- = \hat{\mathbf{v}_{k-1}} \tag{2.135}$$

The output, however, is affected by noise, and the observed value $\mathbf{x}_k$ differs from the predicted value $\hat{\mathbf{x}}_k^-$. This prediction error $\mathbf{r}$ is referred to as the residual (a posteriori error estimate):

$$\mathbf{r}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k^- \tag{2.136}$$

Let us define two parameters α and β in order to obtain the posterior estimate as

$$\begin{cases} \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \alpha \mathbf{r}_k \\ \hat{\mathbf{v}}_k &= \hat{\mathbf{v}}_k^- + \beta \frac{\mathbf{r}_k}{\Delta T} \end{cases} \tag{2.137}$$

In this way, an asymptotic observer for the position and velocity variables is obtained. Unlike the Kalman filter, the alpha-beta filter is a suboptimal filter where the parameters α and β are tuned experimentally without any statistical validation. This approach is often supported by the fact that even in the Kalman filter, it is sometimes necessary to impose the noise matrices entirely empirically.

Chapter 3

Regression and Optimization Methods for Model Analysis

One of the most common problems within computer vision (and more broadly within information theory) is the adaptation of a set of noisy measurements (for example, the *pixels* of an image) to a predefined model.

In addition to the presence of noise, which could be white Gaussian but potentially of any statistical distribution, one must consider the issue of potential *outliers*, a term used in statistics to refer to data points that are too distant from the model to be considered part of it.

In this chapter, various regression techniques aimed at estimating the parameters $\boldsymbol{\beta}$ of a stationary model given a dataset affected by noise are presented, along with methods for identifying and removing outliers from the input data.

In the next chapter, techniques of "regression" more closely related to the theme of classification will be presented.

To estimate the parameters of a model, several techniques presented in the literature are as follows:

Least Squares Fitting If the data consists entirely of *inliers*, there are no *outliers*, and the only disturbance is additive white Gaussian noise, least squares regression is the optimal technique (section 3.2);

M-Estimator The presence of even a few *outliers* can significantly shift the model since the errors are squared [Hub96]: weighting the distant points of the estimated model in a non-quadratic manner leads to improvements in the estimation itself (section 3.8);

IRLS *Iteratively reweighted least squares* is used when the *outliers* are very distant from the model and present in low quantities: under this condition, an iterative regression can be performed (section 3.9), where in each cycle, points with excessively high errors are either removed (*ILS*) or weighted differently (*IRLS*);

Hough If the input data is affected by both errors and numerous *outliers*, and there is potentially a multimodal distribution, but the model is formed by a few parameters, the Hough transform [Hou59] allows for the extraction of the most statistically prevalent model (section 3.11);

RANSAC If the number of *outliers* is comparable to that of the *inliers* and the noise is very low (relative to the position of the *outliers*), *RANdom SAmpling and Consensus* [FB87] enables the identification of the best model present in the scene (section 3.12);

LMedS The *Least Median of Squares* is an algorithm, similar to RANSAC, that ranks the points based on the distance from the randomly generated model and selects the model with the smallest median error [Rou84] (section 3.12.2);

Kalman It is finally possible to use a Kalman filter to derive the parameters of a model (see 2.12.9) when such information is required at runtime.

Only RANSAC and the Hough Transform can effectively handle the case where two or more distributions are simultaneously approaching the model in the measurement.

Nothing ultimately prevents the use of mixed techniques; for example, a relatively coarse (and thus fast and low-memory) Hough transform can be employed to remove the *outliers*, followed by a least squares regression to obtain the model parameters more precisely.

3.1 The Cramer-Rao Bound

The Cramer-Rao Lower Bound (CRLB) establishes a lower limit for the variance of any unbiased estimator of the parameter θ (to maintain consistent notation with the literature, *beta* in our case).

Let X be a multidimensional random variable and θ an unknown deterministic parameter. Let $f_x^{\vartheta}(X)$ be the probability density of X given θ. We assume that such a probability density exists and is twice differentiable with respect to θ.

Theorem 1 (Cramer-Rao Inequality) *Let $T(\cdot)$ be an unbiased estimator of the scalar parameter ϑ, and suppose that the observation space X is independent of θ. Then (under certain regularity conditions...)*

$$E^{\theta}\left[(T(X)-\theta)^2\right] \geq \left[I_n(\theta)\right]^{-1} \tag{3.1}$$

where $I_n(\theta) = E^{\theta}\left[\left(\frac{\partial \ln f_x^{\theta}(X)}{\partial \theta}\right)^2\right]$ *(Fisher Information quantity).*

Since the parameter θ is not known, the Cramer-Rao theorem only allows us to determine whether the estimator is optimal or not.

3.2 Least Squares Regression

Let's start by looking at the most common situation in real-world applications: when the measurements we collect are affected by random noise that follows a Gaussian (bell-shaped) distribution and is added to the data.

We can describe this with the equation:

$$y = f(\mathbf{x}, \boldsymbol{\beta}) + \varepsilon \tag{3.2}$$

Here, $f(\mathbf{x}, \boldsymbol{\beta})$ is a function-usually nonlinear-that depends on some parameters $\boldsymbol{\beta}$and input values $\mathbf{x}$. The term εrepresents the added noise, which is assumed to be Gaussian, with zero average and a certain variance σ.

To estimate the parameters reliably, we need a large number of input samples, written as $\mathbf{x} = \{\mathbf{x}_1 \dots \mathbf{x}_n\}$. Ideally, the number of samples should be much greater than the number of parameters we are trying to estimate.

One can consider that the function of the parameters may not be the same for all samples, but there could be different functions, in fact observing different quantities, always as a function of the same parameters $\boldsymbol{\beta}$. In this case, the equation (3.2) can be generalized as follows:

$$y_i = f_i(\boldsymbol{\beta}) + \varepsilon_i \tag{3.3}$$

where the subscript i implicitly denotes both the type of function and the i-th input sample (essentially a constant parameter of the function).

The vector $\mathbf{r}$ is introduced, defined as

$$r_i = y_i - f_i(\boldsymbol{\beta}) \tag{3.4}$$

contains the residual associated with the i-th observation (or the i-th function). r_i is a function of $\boldsymbol{\beta}$ just as f_i is, and it shares the derivatives (up to a sign with this formalism).

To obtain a maximum likelihood estimator, the quantity to minimize is the *negative log likelihood* (section 2.8) of the function (3.2). In the case of Gaussian noise, the likelihood function can be expressed as

$$\mathcal{L}(r_i|\boldsymbol{\beta}, \sigma) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{r_i^2}{2\sigma_i^2}} \tag{3.5}$$

for independent observations. By applying the definition of *negative log likelihood* to the likelihood function, it follows that in the case of Gaussian noise, the maximum likelihood estimator is equivalent to the least squares method.

Least squares regression is a standard optimization technique for overdetermined systems that identifies the parameters $\boldsymbol{\beta} = (\beta_1, \dots, \beta_m)$ of a function $f(\mathbf{x}, \boldsymbol{\beta}) : \mathbb{R}^m \mapsto \mathbb{R}^n$ that minimize an error S calculated as the sum of squares (*Sum Of Squared Error*) of the residuals r_i over a set of n observations $y_1 \dots y_n$:

$$S(\boldsymbol{\beta}) = SSE(\boldsymbol{\beta}) = \mathbf{r} \cdot \mathbf{r} = \sum_{i=1}^{n} \|r_i\|^2 = \sum_{i=1}^{n} \|y_i - f_i(\boldsymbol{\beta})\|^2 \tag{3.6}$$

$S(\boldsymbol{\beta})$ is defined as the *residual sum of squares* or alternatively as the *expected squared error*.

$S : \mathbb{R}^m \mapsto \mathbb{R}$ is a function that is analyzed by varying the parameters $\boldsymbol{\beta} \in \mathbb{R}^m$ to find its minimum value.

$$\beta^+ = \arg\min_{\beta} S(\beta) \tag{3.7}$$

For this reason, it is referred to as the *objective function* or *cost function*. A minimum obtained through a procedure such as that described by equation (3.7) is defined as a *global minimum*.

A global minimum is challenging to identify from a purely computational standpoint, and typically, techniques can only be employed to locate local minima.

Let us therefore consider $S(\boldsymbol{\beta})$[1] differentiable, that is, f differentiable. The necessary condition for $\boldsymbol{\beta}$ to be a minimum is that, at that point in the parameter space, the gradient of $S(\boldsymbol{\beta})$ vanishes, that is,

$$\frac{\partial S(\boldsymbol{\beta})}{\partial \beta_j} = 2\mathbf{J}^{\top}\mathbf{r} = -2\sum_{i=1}^{n} r_i \frac{\partial f_i(\boldsymbol{\beta})}{\partial \beta_j} = 0 \qquad j = 1, \dots, m \tag{3.8}$$

[1] In the literature, the function S is often encoded with a scale factor $1/2$ to ensure that the gradient of S is not biased by the factor 2 and that the sign aligns with f to simplify notation.

A sufficient condition for a stationary point ($S'(\boldsymbol{\beta}) = 0$) to be a minimum is that $S''(\boldsymbol{\beta})$ (the Hessian) is positive definite. Clearly, the existence of a local minimum guarantees only that there exists a neighborhood δ of $\boldsymbol{\beta}$ such that the function $S(\boldsymbol{\beta} + \delta) \geq S(\boldsymbol{\beta})$.

All the discussions addressed so far assume that the noise is additive ε with constant variance across all samples (*homoscedasticity*). In the case where the measurement noise is still Gaussian and additive with a mean of zero but with non-constant variance, each individual observation y_i is an independent random variable associated with the variance σ_i^2. Intuitively, it is understood that the optimal regression in this case will need to give more weight to samples with low variance while giving less weight to samples with high variance.

To achieve this result, a normalization is employed, similar to that shown in section 2.4.1 and a direct consequence of the *likelihood* in equation (3.5). Therefore, one should no longer minimize the simple sum of squared residuals, but rather the *weighted* sum of the residuals:

$$\chi^2 = \sum_{i=1}^{n} \frac{\|r_i\|^2}{\sigma_i} \tag{3.9}$$

The cost function, now the sum of a random variable with unit variance squared, becomes a chi-squared distribution and is thus referred to as χ^2. The minimum of this cost function coincides with that obtained previously from the least squares when the variance is constant instead. The condition (3.8) for obtaining the minimum is also modified accordingly:

$$\sum_{i=1}^{n} \frac{r_i}{\sigma_i} \frac{\partial f_i(\boldsymbol{\beta})}{\partial \beta_j} = 0 \qquad j = 1, \ldots, m \tag{3.10}$$

Further generalizing this concept, when the observation is subject to Gaussian noise with a known covariance matrix $\boldsymbol{\Sigma}$, the *Weighted Sum of Squared Error* (*WSSE*) can ultimately be expressed as It seems that you have entered a placeholder or a command related to a LaTeX environment. If you have specific content or text that you would like me to translate from Italian to English, please provide that text, and I will be happy to assist you! It is noteworthy that this formulation of the cost function is equivalent to that of equation (3.6), where, however, the Mahalanobis distance is used instead of the Euclidean distance (section 2.4).

Any *Weighted Least Squares* can be reduced to an unweighted problem $\boldsymbol{\Sigma} = I$ by premultiplying the residuals $\mathbf{r}$ (and consequently the derivatives) by a matrix $\mathbf{L}^\top$ such that $\boldsymbol{\Sigma}^{-1} = \mathbf{L}\mathbf{L}^\top$, using, for instance, a Cholesky decomposition in the case where this matrix is not diagonal.

All these estimators, which take into account the variance of the observation, coincide with the *negative log likelihood* for the variable $\mathbf{y}$ perturbed by Gaussian noise with zero mean and covariance $\boldsymbol{\Sigma}$.

3.2.1 Least Squares Linear Regression

When f is a linear function with respect to the parameters $\boldsymbol{\beta}$, it is referred to as linear least squares regression (or Ordinary Least Squares, OLS). This function can be represented in the form of a linear system.

$$y_i = \mathbf{x}_i \boldsymbol{\beta} + \varepsilon_i \tag{3.11}$$

where $\boldsymbol{\beta}$ are the unknown parameters to be estimated and ε_i is zero-mean white Gaussian additive noise. The parameters $\boldsymbol{\beta}$ are the regression coefficients: they measure the association between the variable $\mathbf{x}$ and the variable y.

Each observation is a constraint, and all individual constraints can be collected in matrix form.

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{3.12}$$

$\mathbf{y} \in \mathbb{R}^n$ is the vector of *responses* (dependent variables), and the matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ The collection of independent variables (*explanatory variables*) is referred to as the *design matrix*, and finally, $\boldsymbol{\varepsilon}$ is the vector of additive noise with a mean of zero $\mathrm{E}[\boldsymbol{\varepsilon}] = 0$ and variance $\boldsymbol{\Sigma}$. The parameter vector $\boldsymbol{\beta}$ is called the *Linear Projection Coefficient* or *Linear Predictor*. The random variable $\mathbf{y}$ is therefore composed of a *deterministic* part and a *stochastic* part.

The objective is to find the hyperplane $\boldsymbol{\beta}$ in m dimensions that best fits the data $(\mathbf{y}, \mathbf{X})$.

The value $\boldsymbol{\beta}$ that minimizes the cost function defined in equation (3.6), specifically in the case of observation noise with a mean value of zero and constant variance across all samples, is indeed the best linear estimator that minimizes variance (the *Best Linear Unbiased Estimator BLUE*).

Definizione 9 *The Best Linear Unbiased Estimate (BLUE) of a parameter $\boldsymbol{\beta}$ based on a dataset Y is*

1. *a linear function of Y, such that the estimator can be expressed as $\hat{\boldsymbol{\beta}} = \mathbf{A}Y$;*
2. *it must be* unbiased *($E[\mathbf{A}Y] = 0$),*
3. *among all possible linear estimators, it is the one that produces the minimum variance.*

The Gauss-Markov theorem demonstrates that a least squares estimator is the best choice among all minimum variance *BLUE* estimators when the variance of the observations is constant (*homoscedastic*).

The best least squares estimate $\hat{\boldsymbol{\beta}}$ that minimizes the sum of the residuals is the solution to the linear problem.

$$\hat{\boldsymbol{\beta}} = \arg\min_{\mathbf{b}} \|\boldsymbol{\varepsilon}\|^2 = \arg\min_{\mathbf{b}} \sum \|\mathbf{y}_i - \mathbf{x}_i\mathbf{b}\|^2 = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} \tag{3.13}$$

The same result was already obtained in section **??** concerning the pseudoinverse of a matrix: an SVD decomposition of the matrix $\mathbf{X}$ also yields the best solution in terms of minimizing the propagation of computational errors.

The matrix $\mathbf{P}$, defined as

$$\mathbf{P} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top \tag{3.14}$$

is a projection matrix that transforms the outputs (*response vector*) $\mathbf{y}$ into their estimate $\hat{\mathbf{y}}$ (the estimate of the observation without noise):

$$\mathbf{P}\mathbf{y}_i = \mathbf{x}_i\hat{\boldsymbol{\beta}} = \hat{\mathbf{y}}_i \tag{3.15}$$

Due to this property, $\mathbf{P}$ is referred to as the hat matrix.

In the case of noise with non-constant variance among the observed samples (*heteroscedastic*), weighted least squares regression is the *BLUE* choice.

$$w_i = \frac{1}{\sigma_i} \tag{3.16}$$

with $w_i > 0$ that take into account the various uncertainties associated with each observation y_i such that $1/w_i$ is the standard deviation of the i-th measurement. By inserting the weights w_i into a diagonal matrix $\mathbf{W}$, a new linear system is obtained where each row effectively has the same observation variance. The solution that minimizes $\boldsymbol{\varepsilon}$ can always be expressed as

$$\hat{\boldsymbol{\beta}} = (\mathbf{W}\mathbf{X})^+\mathbf{W}\mathbf{y} \tag{3.17}$$

with $\mathbf{W} = \boldsymbol{\Sigma}^{-1}$.

Further generalizing, in the case of noise with non-constant variance among the observed samples and correlated with each other, the best *BLUE* estimate in the linear case must take into account the covariance of the noise $\boldsymbol{\Sigma}$:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\boldsymbol{\Sigma}^{-1}\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{\Sigma}^{-1}\mathbf{y} \tag{3.18}$$

This estimator is referred to as *Generalized Least Squares* (*GLS*).

Such a system minimizes the variance

$$Var[\hat{\boldsymbol{\beta}}_{GLS}] = (\mathbf{X}^\top\boldsymbol{\Sigma}^{-1}\mathbf{X})^{-1} \tag{3.19}$$

3.2.2 Total Least Squares

We now extend the linear problem $\mathbf{A}\mathbf{x} = \mathbf{b} + \delta$ to the more general case where the coefficient matrix $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{E}$ is also perturbed (Errors-In-Variables model, EIV [VHV91]). This type of least squares regression problem is referred to as *Total Least Squares* (TLS).

The solution of the perturbed system

$$(\mathbf{A} + \mathbf{E})\mathbf{x} = \mathbf{b} + \boldsymbol{\delta} \tag{3.20}$$

corresponds to finding the solution $\mathbf{x}$ that minimizes the Frobenius norm $\|(\mathbf{E}\ \boldsymbol{\delta})\|_F$, subject to the constraint (3.20). In classical TLS, all columns of the data matrix contain noise. If some columns are error-free, then the solution is referred to as *mixed TLS-LS*.

The system (3.20) can be rewritten as

$$([\mathbf{A}|\mathbf{b}] + [\mathbf{E}|\boldsymbol{\delta}]) \begin{bmatrix} \mathbf{x} \\ -1 \end{bmatrix} = \mathbf{0} \tag{3.21}$$

Utilizing the SVD decomposition and the Eckart-Young-Mirsky theorem (the matrix formed by the first n terms of the SVD decomposition is the matrix that best approximates the matrix $\mathbf{Z}$ under the Frobenius norm), it is possible to find the solution to the problem (3.20). Let us denote

$$\mathbf{C} := [\mathbf{A}|\mathbf{b}] = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top \tag{3.22}$$

the Singular Value Decomposition of the matrix $\mathbf{C}$, where $\boldsymbol{\Sigma} = \text{diag}\,(\sigma_1 \dots \sigma_{n+d})$. The *Total Least Squares* solution, if it exists, can be expressed as

$$\hat{\mathbf{X}}_{tls} = -\mathbf{V}_{12}\mathbf{V}_{22}^{-1} \tag{3.23}$$

having partitioned

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}_{11} & \mathbf{V}_{12} \\ \mathbf{V}_{21} & \mathbf{V}_{22} \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_2 \end{bmatrix} \tag{3.24}$$

and it is possible to obtain the best estimate of $\hat{\mathbf{C}}$ as

$$\hat{\mathbf{C}}_{tls} = \mathbf{C} + \Delta\mathbf{C}_{tls} = \mathbf{U}\,\text{diag}\,(\boldsymbol{\Sigma}_1, 0)\,\mathbf{V}^\top \tag{3.25}$$

3.3 Optimization Methods

Now, let us consider a generic problem of modeling (optimization) of an unconstrained function, applicable, for example, to classification problems in the field of computer vision. The considerations expressed in this section apply to the case of least squares but can be extended to a generic *loss function*.

Let $\mathbf{z}$ be the dataset involved in the modeling operation, consisting of a pair $(\mathbf{x}_i, y_i)$ composed of an arbitrary input $\mathbf{x}_i$ and the output y_i. Let $\ell(\hat{y}, y)$ be the cost function (*loss function*) that returns the quality of the estimate on y. The objective is to find the weights $\boldsymbol{\beta}$ that parameterize the function $f(\mathbf{x}; \boldsymbol{\beta})$ that minimize a cost function $S(\boldsymbol{\beta})$.

$$S(\boldsymbol{\beta}) = \int \ell(\mathbf{z}; \boldsymbol{\beta}) dP(\mathbf{z}) \qquad S(\boldsymbol{\beta}) = \sum_{i=1}^{n} \ell_i(\boldsymbol{\beta}) \tag{3.26}$$

both in the continuous case and in the discrete case, having defined $\ell_i(\boldsymbol{\beta}) = \ell(f_i(\mathbf{x}_i; \boldsymbol{\beta}), y_i)$. For simplicity, we will always refer to the second case, the discrete one, to describe the cost function.

In the case of normal additive Gaussian error, the maximum likelihood estimator is the quadratic *loss function* given by equation (3.6):

$$\ell_i(\boldsymbol{\beta}) = r_i^2(\boldsymbol{\beta}) = (y_i - f_i(\mathbf{x}_i; \boldsymbol{\beta}))^2 \tag{3.27}$$

In practical applications, it is almost never possible to obtain the minimum of the function in closed form; therefore, it is necessary to resort to appropriate iterative methods, which, starting from an initial state and moving in suitable directions $\boldsymbol{\delta}$, gradually approach the minimum of the objective function.

3.3.1 Newton-Raphson Method

The problem of finding the minima of a function can be reduced to the problem of finding the zeros of a function, specifically the first derivative of the cost function S.

Let $\mathbf{g} : \mathbb{R}^m \mapsto \mathbb{R}^n$ be a differentiable multivariable function for which we need to find

$$\mathbf{g}(\mathbf{x}) = \mathbf{0} \tag{3.28}$$

. By expanding the function $\mathbf{g}$ in a Taylor series locally around an appropriate point $\mathbf{x}$, we obtain

$$\mathbf{g}(\mathbf{x} + \boldsymbol{\delta}) = \mathbf{g}(\mathbf{x}) + \mathbf{J}_g \boldsymbol{\delta} + O(\boldsymbol{\delta}^2) \tag{3.29}$$

where $\mathbf{J}_g$ is the matrix $n \times m$ Jacobian of the function $\mathbf{g}$ evaluated at $\mathbf{x}$.

The objective is to modify the value of $\mathbf{x}$ by an amount $\boldsymbol{\delta}$ such that the cost function calculated at $\mathbf{x}_t + \boldsymbol{\delta}$ is exactly zero. Ignoring contributions of higher order than $\boldsymbol{\delta}^2$, the estimate of $\boldsymbol{\delta}$ that, in first approximation, brings the function $\mathbf{g}$ close to zero is the solution of the linear system (3.29) with the condition (3.28), namely

$$\mathbf{J}_g \boldsymbol{\delta} = -\mathbf{g}(\mathbf{x}) \tag{3.30}$$

. This system, if $\mathbf{J}_g$ has no rank deficiencies, is a simple linear system, which may also be overdetermined, and can be solved using one of the techniques shown in section **??**. The idea behind iterative methods is to modify the point $\mathbf{x}_t$ of the quantity $\boldsymbol{\delta}_t$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \boldsymbol{\delta}_t \tag{3.31}$$

for the iterations $t = 1, 2, \ldots$, calculated in such a way as to progressively approach zero for the function.

In the case of a single variable $n = m = 1$, the Newton's method reduces to

$$x_{t+1} = x_t - \frac{g(x)}{g'(x)} \tag{3.32}$$

In numerical analysis, this is known as the Newton method (or Newton-Raphson method) for finding the zeros of a function.

The points of maximum and minimum of a function are the points at which the gradient can be set to zero. Therefore, this technique can be applied to find the maxima and minima of a function $f(\mathbf{x}) : \mathbb{R}^m \mapsto \mathbb{R}$ by defining

$$\begin{aligned} \mathbf{g}(\mathbf{x}) &= \nabla f(\mathbf{x}) \\ \mathbf{J}_g(\mathbf{x}) &= \mathbf{H}_f(\mathbf{x}) \end{aligned} \tag{3.33}$$

where $\nabla f(\mathbf{x})$ is the gradient function $\mathbb{R}^m \mapsto \mathbb{R}^m$ and $\mathbf{H}_f(\mathbf{x})$ is the Hessian matrix $m \times m$, with the gradient and Hessian functions of f evaluated at $\mathbf{x}$. The modification of the Newton point $\mathbf{x}$ thus becomes

$$\mathbf{H}_f(\mathbf{x}) \boldsymbol{\delta}_t = -\nabla f(\mathbf{x}) \tag{3.34}$$

. When used for optimization, the Newton method effectively approximates the function $f(\mathbf{x})$ in the vicinity of $\mathbf{x}$ with a quadratic. If $f(\mathbf{x})$ is a quadratic function, convergence is guaranteed in a single iteration.

Now, in the specific case of optimization methods, the function $f(\mathbf{x})$ is the cost function $S(\boldsymbol{\beta})$. Therefore, when the Hessian matrix of $S(\boldsymbol{\beta})$ is non-singular, the parameter variation equation is obtained as follows:

$$\boldsymbol{\delta}_t = -\mathbf{H}_S^{-1}(\boldsymbol{\beta}_t)\nabla S(\boldsymbol{\beta}_t) \tag{3.35}$$

through the Newton optimization method.

3.3.2 Gradient Descent

The gradient descent algorithm (*gradient descent* GD or *steepest descent*) updates the weights $\boldsymbol{\beta}$ at each iteration using the gradient (specifically, the negative gradient) of $S(\boldsymbol{\beta})$:

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \gamma\nabla S(\boldsymbol{\beta}_t) = \boldsymbol{\beta}_t - \gamma\sum_{i=1}^{n}\nabla\ell_i(\boldsymbol{\beta}_t) \tag{3.36}$$

or

$$\boldsymbol{\delta}_t = -\gamma\sum_{i=1}^{n}\nabla\ell_i(\boldsymbol{\beta}_t) \tag{3.37}$$

where γ is an appropriately chosen optimization factor (in *machine learning* it is referred to as the *learning rate*). Under suitable assumptions, if the starting point is close to the solution and the factor γ is sufficiently low, the rate of convergence that can be achieved is practically linear.

By introducing a user-defined parameter γ into the expression (3.36), this approach is empirical and problem-dependent, if not reliant on the experience of the human user. Following this observation and comparing the gradient descent equation with equation (3.35), it becomes evident that Newton's method is, in fact, a special case of gradient descent. Therefore, one can achieve better optimization by replacing the scalar parameter γ with the positive definite matrix $\boldsymbol{\Gamma}_t$ obtained from the inverse of the Hessian at the point:

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \boldsymbol{\Gamma}_t\nabla S(\boldsymbol{\beta}_t) \tag{3.38}$$

A second-order gradient descent is thus the Newton algorithm, which, under appropriate assumptions, yields quadratic convergence instead of linear convergence.

3.3.3 Stochastic Gradient Descent

The stochastic gradient descent (SGD) algorithm is a simplification of the gradient descent algorithm. Instead of calculating the exact gradient of $S(\boldsymbol{\beta})$ for each iteration, the gradient of one randomly chosen sample ℓ_i is used.

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \gamma_t\nabla\ell_i(\boldsymbol{\beta}_t) \tag{3.39}$$

SGD can be employed to optimize any convex function over a convex domain.

The potential second-order stochastic gradient descent does not typically lead to improvements.

From a practical standpoint, it is often performed the update on a small number of samples (*batch size*) greater than 1, in order to reduce the noise from individual samples while avoiding overly averaged contributions as seen in the case of full gradient descent.

Finally, to simulate inertia to change, a term α called *momentum* is added: The values of α are typically small (for example, 0.05).

The *momentum* is the simplest modification to SGD that addresses some common issues in optimization within *machine learning*. In addition to SGD with momentum, there are numerous variants designed to accelerate convergence in gradient descent-type algorithms. A non-exhaustive list includes:

- Nesterov Accelerated Gradient (NAG)
- AdaDelta
- Adaptive Gradient (AdaGrad)
- RMSProp (Rprop + SGD)
- Adam
- AdaMax
- Momentum
- Resilient Propagation (RProp)

A survey of the various algorithms can be found in [Rud16].

Adam

Adam [KB14] (*Adaptive Momentum estimation*) aims to leverage the strengths of both AdaGrad (Adaptive Gradient Algorithm) and RMSProp (Root Mean Square Propagation).

AdaGrad assigns a learning rate to each parameter, which is advantageous for problems with sparse gradients. RMSProp also utilizes a learning rate for each parameter, but this rate is adjusted based on the magnitude of the gradient: this algorithm performs well in online settings and on non-stationary problems.

Instead of updating the learning rate by analyzing the first-order moment (the average) as in RMSProp, Adam leverages the second-order moment of the gradients.

3.3.4 Gauss-Newton

The methods discussed so far allow considerable freedom in the choice of a particular *loss function* over another. In practical cases where the cost function ℓ is quadratic, further optimizations can be made to the Newton method, avoiding the cumbersome computation of the Hessian. In this case, the loss function takes the form already seen previously,

$$S(\boldsymbol{\beta}) = \frac{1}{2}\mathbf{r}^\top \mathbf{r} = \frac{1}{2}\sum_{i=1}^{n} r_i^2(\boldsymbol{\beta}) \tag{3.40}$$

. The term $1/2$ in the cost function serves to provide a more compact expression of the Jacobian.

With this cost function, the gradient and Hessian are expressed as

$$\begin{aligned} \nabla S(\boldsymbol{\beta}) &= \textstyle\sum_{i=1}^{n} r_i(\boldsymbol{\beta})\nabla r_i(\boldsymbol{\beta}) = \mathbf{J}_r^\top \mathbf{r} \\ \mathbf{H}_S(\boldsymbol{\beta}) &= \textstyle\sum_{i=1}^{n} \nabla r_i \nabla r_i^\top + \sum_{i=1}^{n} r_i \mathbf{H}_{r_i} = \mathbf{J}_r^\top \mathbf{J}_r + \sum_{i=1}^{n} r_i \mathbf{H}_{r_i} \end{aligned} \tag{3.41}$$

When the parameters are close to the exact solution, the residual is small, and the Hessian can be approximated by only the first term of the expression, namely

$$\mathbf{H}_S(\boldsymbol{\beta}) \approx \mathbf{J}_r^\top \mathbf{J}_r \tag{3.42}$$

. Under these conditions, the gradient and the Hessian of the cost function S can be expressed solely in terms of the Jacobian of the functions $r_i(\boldsymbol{\beta})$. The approximated expression for the Hessian can be incorporated into equation (3.34):

$$-\mathbf{J}_r^\top \mathbf{r} = \mathbf{H}_S \boldsymbol{\delta}_\beta \approx \mathbf{J}_r^\top \mathbf{J}_r \boldsymbol{\delta}_\beta \tag{3.43}$$

. This, similar to the case of Newton, is a linear minimization problem that can be solved using the *normal equations*:

$$\boldsymbol{\delta}_\beta = -\left(\mathbf{J}_r^\top \mathbf{J}_r\right)^{-1} \mathbf{J}_r^\top \mathbf{r} \tag{3.44}$$

. The significance of the *normal equations* is geometric: the minimum is achieved when $\mathbf{J}\boldsymbol{\delta}_\beta - \mathbf{r}$ becomes orthogonal to the column space of $\mathbf{J}$.

In the particular case of the residue function written as

$$r_i = y_i - f_i(\mathbf{x}_i; \boldsymbol{\beta}) \tag{3.45}$$

, that is similar to those in equation (3.6), it is possible to use $\mathbf{J}_f$, the Jacobian of f, instead of $\mathbf{J}_r$.

$$\boldsymbol{\delta}_\beta = \left(\mathbf{J}_f^\top \mathbf{J}_f\right)^{-1} \mathbf{J}_f^\top \mathbf{r} \tag{3.46}$$

Having observed that the derivatives of r_i and $f_i(\mathbf{x}_i)$ are equal up to a sign[2].

3.3.5 Levenberg-Marquardt

In the previous sections, the algorithms for solving nonlinear systems have been divided into gradient descent algorithms and Gauss-Newton algorithms. For a more in-depth reading, I recommend [MBT04].

In Gauss-Newton methods, when $\mathbf{J}^\top\mathbf{J}$ is positive definite, the method can always indicate a direction where the cost decreases. When $\mathbf{J}^\top\mathbf{J}$ becomes singular, the Gauss-Newton method becomes numerically unstable.

The technique proposed by Levenberg-Marquardt aims to leverage the strengths of both Gauss-Newton and gradient descent methods in order to benefit from both.

The Levenberg-Marquardt (LM) algorithm is an iterative regression technique that is now considered standard for solving multivariable nonlinear problems. An excellent description of the algorithm can be found in [Lou05]. The algorithm can be viewed as consisting of a slow yet convergent gradient descent phase, followed by a faster Gauss-Newton type solver.

[2] Clearly, the derivatives coincide when a residue of the type $r_i = \hat{y}_i - y_i$ is chosen.

The Levenberg-Marquardt algorithm solves a slightly different version of the equation (3.43), a special case of the equation (3.34), known as the *augmented normal equations*:

$$\mathbf{N}\delta_\beta = -\mathbf{J}_r^\top \mathbf{r} \tag{3.47}$$

where $\mathbf{N} = \mathbf{H}_S + \mu\mathbf{I}$ with $\mu > 0$ a damping factor: when the factor μ is high, the matrix $\mathbf{N}$ is nearly diagonal, and the algorithm approaches a steepest descent gradient method, while when the term μ is close to zero, the algorithm approximates the Newton method. Instead of a line search, Levenberg-Marquardt is a technique that implements the concept of a trust region.

In the Levenberg-Marquardt method, the approximation of the Hessian $\mathbf{H}_S(\boldsymbol{\beta}) \approx \mathbf{J}_r^\top \mathbf{J}_r$ is also typically utilized, as seen in the Gauss-Newton method, but is limited to the case where the *loss function* is quadratic.

How to set and modify between iterations μ is, however, a problem left to the solver, and various techniques are proposed in the literature.

One of the most widely used implementations [Nie99] selects as μ_0 a value of the type

$$\mu_0 = \tau \max \operatorname{trace} \mathbf{H} \tag{3.48}$$

with τ chosen freely by the user based on their confidence in the value of $\boldsymbol{\beta}$. The modification of μ across the various iterations is managed by the gain factor ρ (*gain ratio*):

$$\rho = \frac{S(\boldsymbol{\beta}) - S(\boldsymbol{\beta} + \boldsymbol{\delta}_\beta)}{\frac{1}{2}\boldsymbol{\delta}_\beta^\top (\mu\boldsymbol{\delta}_\beta + \mathbf{J}^\top \mathbf{r})} \tag{3.49}$$

A high value of ρ indicates that the linearized version of f is very good, and μ can be decreased. Conversely, if ρ is high, then the value of μ should be increased. Finally, when $\rho \approx 1$ is present, there is a good correspondence between the predicted model and the data. In the limiting case, when ρ is negative, it indicates a deteriorating solution that should be discarded, and μ should be increased in order to approach a gradient descent method. Through ρ, it is possible to modify μ according to the rule

if $\rho > 0$ **then**
  $\mu \leftarrow \mu \max\left(\frac{1}{3}, 1 - (2\rho - 1)^3\right)$
  $\nu \leftarrow 2$
else
  $\mu \leftarrow \mu\nu$
  $\nu \leftarrow 2\nu$
end if

3.3.6 DogLeg

Similar to Levenberg-Marquardt, the Dog-Leg algorithm attempts to combine the Gauss-Newton method with the gradient descent method. Unlike Levenberg-Marquardt, where this behavior is managed by the Dumped Hessian, in the case of Dog Leg, the choice remains explicit through the analysis of a *Trust Region*.

3.3.7 Sampson Error

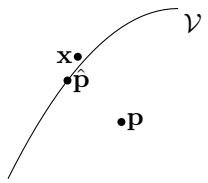


Figure 3.1: Between a variety $\mathcal{V}$ and a point $\mathbf{p}$, one can identify the point at minimum geometric distance $\hat{\mathbf{p}}$ and the point determined by the Sampson distance $\mathbf{x}$.

In many regression problems, it is necessary to have some metric to understand how far a $\mathbf{p}$ is from the true model. To achieve this, it would be useful to have an estimate $\hat{\mathbf{p}}$ of the observation without the noise component, that is, a data point that exactly belongs to the model. Both of these quantities are typically not directly obtainable without introducing unknown auxiliary variables. However, it is possible to obtain an estimate of these values by linearizing the model function around the observation.

Let $\mathbf{p}$ be an observation affected by noise, and let $f(\mathbf{x}) = \mathbf{0}$ be a multidimensional variety *manifold* representing a specific model to which the observation must belong, that is, $\mathbf{p} = \hat{\mathbf{p}} + \epsilon$.

The residual $f(\mathbf{p})$ is an *algebraic* measure of the proximity between the point and the model and does not provide any useful information in absolute terms: if the function is replaced by a non-zero multiple of itself, it will obviously represent the same locus of points, but the function's output will change accordingly. The correct metric from the perspective of the maximum likelihood estimator in the case of additive white Gaussian noise on the observations is the geometric distance between the point $\mathbf{p}$ and the point $\hat{\mathbf{p}}$ belonging to the model, that is, to estimate ϵ.

We therefore examine the problem of calculating an approximate distance between the point $\mathbf{p} \in \mathbb{R}^m$ and a geometric variety $f(\mathbf{x}) = \mathbf{0}$ where $f : \mathbb{R}^m \mapsto \mathbb{R}^n$ is a differentiable function in a neighborhood of $\mathbf{p}$.

The point $\hat{\mathbf{p}}$ that lies on the manifold closest to the point $\mathbf{p}$ is, by definition, the point that minimizes the geometric error

$$\hat{\mathbf{p}} = \arg\min_{\mathbf{x}} \|\mathbf{p} - \mathbf{x}\| \tag{3.50}$$

under the constraint $f(\mathbf{x}) = 0$ (or $\min ||\epsilon||^2$ under the constraint $f(\mathbf{p} + \epsilon) = \mathbf{0}$).

The difference between minimizing an algebraic quantity in a linear manner and a geometric quantity in a non-linear manner has prompted the search for a potential compromise. The Sampson method, initially developed for varieties such as conics, requires a hypothesis that can instead be applied to various problems: the derivatives of the cost function in the vicinity of the minimum $\hat{\mathbf{p}}$ must be nearly linear and thus approximable through a series expansion. The variety $f(\mathbf{p}) = 0$ can be approximated using Taylor expansion such that

$$\tilde{f}(\mathbf{x}) \approx f(\mathbf{p}) + \mathbf{J}_f \boldsymbol{\delta}_{\mathbf{x}} = \mathbf{0} \tag{3.51}$$

with $\mathbf{J}_f$ being the $n \times m$ Jacobian matrix of the function f evaluated at $\mathbf{p}$ and $\boldsymbol{\delta}_{\mathbf{x}} = \mathbf{x} - \mathbf{p}$.

This is the equation of a hyperplane in $\mathbf{x}$ and the distance between the point $\mathbf{p}$ and the plane $\tilde{f}(\mathbf{x}) = 0$ is known as the Sampson distance or the *approximate maximum likelihood* (AML). The Sampson error represents the geometric distance between the point and the approximated version of the function (*geometric distance to first order approximation function*).

At this point, the problem becomes finding the point $\mathbf{x}$ that is closest to $\mathbf{p}$, which means minimizing $\|\boldsymbol{\delta}_{\mathbf{x}}\|$, while satisfying the linear constraint

$$\mathbf{J}_f \boldsymbol{\delta}_{\mathbf{x}} = -f(\mathbf{p}) \tag{3.52}$$

.

As it is a minimization problem with constraints, it is solved using Lagrange multipliers, from which a remarkable result is obtained. An interesting result when compared to the Gauss-Newton method, for instance, equation (3.44).

The value $\boldsymbol{\delta}_{\mathbf{x}}$ represents an estimate of the distance from the point $\mathbf{p}$ to the variety and can be used both to determine whether the point belongs to the variety (for example, within algorithms like RANSAC to discern outliers) and potentially as an alternative cost function to the Euclidean norm. $\boldsymbol{\delta}_{\mathbf{x}}$ is the Sampson error, and its norm, given by

$$\|\boldsymbol{\delta}_x\|^2 = \boldsymbol{\delta}_x^\top \boldsymbol{\delta}_x = f(\mathbf{p})^\top \left(\mathbf{J}\mathbf{J}^\top\right)^{-1} f(\mathbf{p}) \tag{3.53}$$

, indicates the squared distance between the point and (the first-order approximation of) a point on the variety.

In the notable case $n = 1$, the Sampson distance reduces to

$$\|\boldsymbol{\delta}_x\|^2 = \frac{(f(\mathbf{p}))^2}{\|\nabla f(\mathbf{p})\|^2} \tag{3.54}$$

Practical applications of the use of the Sampson error include, for example, the distance from a point to a conic (see section 3.6.7), the distance of a pair of points from a homography, or the distance of a pair of corresponding points with respect to the Fundamental matrix (section 9.4.2).

The Sampson distance can be generalized in the case of multiple constraints using the Mahalanobis distance, specifically by minimizing

$$\min_{\epsilon} \sum ||\epsilon||_\Sigma^2 = \min_{\epsilon} \sum \epsilon^\top \boldsymbol{\Sigma}^{-1} \epsilon \tag{3.55}$$

subject to the constraint $f(\mathbf{p} + \epsilon) = 0$. Thus, the above equation generalizes to

$$\|\boldsymbol{\delta}_x\|^2 = \boldsymbol{\delta}_x^\top \boldsymbol{\delta}_x = f(\mathbf{p})^\top \left(\mathbf{J}\boldsymbol{\Sigma}\mathbf{J}^\top\right)^{-1} f(\mathbf{p}) \tag{3.56}$$

.

3.3.8 Regression with Anisotropic Noise

In the case where the observation noise is not isotropic, it is no longer possible to use the Euclidean distance to measure the error, and it is necessary to switch to the Mahalanobis distance. Under this different metric, the cost function (3.40) can be expressed as

$$S(\boldsymbol{\beta}) = (\mathbf{r}(\boldsymbol{\beta}))^\top \boldsymbol{\Omega} (\mathbf{r}(\boldsymbol{\beta})) \tag{3.57}$$

where $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$ is the information matrix, also known as the concentration matrix or precision matrix. The Mahalanobis distance is the optimal estimator in the sense of Maximum Likelihood when the noise is Gaussian and anisotropic with zero mean. In the particular case where the covariance matrix is diagonal, this approach can be completely reduced to the weighted least squares approach.

The Taylor series expansion of equation (3.57) is expressed as

$$\begin{aligned} S(\boldsymbol{\beta}+\boldsymbol{\delta}) &= (\mathbf{r}(\boldsymbol{\beta}+\boldsymbol{\delta}))^{\top}\,\mathbf{\Omega}\,(\mathbf{r}(\boldsymbol{\beta}+\boldsymbol{\delta})) \\ &\approx (\mathbf{r}+\mathbf{J}\boldsymbol{\delta})^{\top}\,\mathbf{\Omega}\,(\mathbf{r}+\mathbf{J}\boldsymbol{\delta}) \\ &= \mathbf{r}\mathbf{\Omega}\mathbf{r}^{\top} + 2\mathbf{r}^{\top}\mathbf{\Omega}\mathbf{J}\boldsymbol{\delta} + \boldsymbol{\delta}^{\top}\mathbf{J}^{\top}\mathbf{\Omega}\mathbf{J}\boldsymbol{\delta} \\ &= \mathbf{r}\mathbf{\Omega}\mathbf{r}^{\top} + 2\mathbf{b}\boldsymbol{\delta} + \boldsymbol{\delta}^{\top}\mathbf{H}\boldsymbol{\delta} \end{aligned} \tag{3.58}$$

with $\mathbf{r}$ and $\mathbf{J}$ calculated in $\boldsymbol{\beta}$. The matrix $\mathbf{H} = \mathbf{J}^{\top}\mathbf{\Omega}\mathbf{J}$ is the information matrix of the entire system, as it is obtained from the projection of the measurement error in the parameter space through the Jacobian $\mathbf{J}$, while $\mathbf{b} = \mathbf{r}^{\top}\mathbf{\Omega}\mathbf{J}$ has been introduced for compactness.

The derivatives of the function S consequently become

$$\frac{\partial S(\boldsymbol{\beta}+\boldsymbol{\delta})}{\partial \boldsymbol{\delta}} \approx 2\mathbf{b} + 2\mathbf{H}\boldsymbol{\delta} \tag{3.59}$$

From this result, if one wishes to find the minimum of the cost function S using Gauss-Newton, a result similar to the one observed previously is obtained

$$\mathbf{H}\boldsymbol{\delta} = -\mathbf{b} \tag{3.60}$$

which is very similar to that of equation (3.43) obtained from Gauss-Newton with isotropic noise.

3.3.9 Optimization on a Manifold

All the optimization methods discussed so far have been designed to operate in a "flat" Euclidean space. When optimizing a state vector that contains one or more variables where the Euclidean space loses its meaning (for example, rotations or matrices), every parametrization results in sub-optimal solutions and is affected by singularities. In recent years, techniques that utilize the overparameterized version of the state vector [Her08] have gained significant traction, allowing for the optimization of the problem directly on the manifold, which is locally homeomorphic to a linear space.

The idea is to transform the classical minimization of $S \in \mathcal{M}$, with $\mathcal{M}$ being an n-dimensional manifold,

$$\boldsymbol{\delta} \Leftarrow \left.\frac{\partial S(\mathbf{x}+\boldsymbol{\delta})}{\partial \boldsymbol{\delta}}\right|_{\boldsymbol{\delta}=0} = 0 \qquad \mathbf{x} \Leftarrow \mathbf{x}+\boldsymbol{\delta} \tag{3.61}$$

into

$$\boldsymbol{\epsilon} \Leftarrow \left.\frac{\partial S(\mathbf{x} \boxplus \boldsymbol{\epsilon})}{\partial \boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=0} = 0 \qquad \mathbf{x} \Leftarrow \mathbf{x} \boxplus \boldsymbol{\epsilon} \tag{3.62}$$

with $\boldsymbol{\epsilon} \in \mathbb{R}^n$, assuming that in the neighborhood $\boldsymbol{\epsilon} = 0$ the function operates in a Euclidean space. The operator $\boxplus$ allows for the addition between elements of the manifold space and elements of the Euclidean space $\mathbb{R}^n$.

A very classic example is to consider the optimization of an orientation expressed in 3 dimensions through the use of a quaternion in 4 dimensions.

3.4 Convex Functions and Convex Optimization Problems

Definizione 10 *A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if for $\forall \mathbf{x}, \mathbf{y}$ belonging to the domain of f and for $\lambda \in [0,1]$, it holds that*

$$f\left(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}\right) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}) \tag{3.63}$$

A function f is defined as concave if $-f$ is convex.

3.4.1 Linear Programming (LP)

3.4.2 Quadratic Programming (QP)

3.5 Model Parameter Evaluation

Neglecting the presence of *outliers* in the input data on which regression is performed, two important open questions remain: one is to assess the quality of the obtained model, and at the same time, to provide an index of how far this estimate may be from the true model, due to errors in the input data.

This section extensively addresses the non-linear case: the linear case is equivalent by using the parameter matrix $\mathbf{X}$ instead of the Jacobian $\mathbf{J}$, which has already been partially discussed in section 2.7.

Let $\mathbf{y} = (y_1, \dots, y_n)^\top$ be a vector of realizations of statistically independent random variables $y \in \mathbb{R}$ and $\boldsymbol{\beta} \in \mathbb{R}^m$ model parameters. An intuitive estimator of the goodness of fit of the model is the *root-mean-squared residual error* (*RMSE*), also referred to as the *standard error of the regression*:

$$s = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}} \tag{3.64}$$

with $\hat{y}_i = f(\mathbf{x}_i, \hat{\boldsymbol{\beta}})$ estimated value thanks to the model f from which the parameters $\hat{\boldsymbol{\beta}}$ and $S = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$ have been derived. Typically, this function has already been seen expressed in terms of the residual $r_i = \mathbf{y}_i - \hat{\mathbf{y}}_i$. If the estimator is not affected by *bias* (as occurs, for example, in least squares regression) $\mathrm{E}[r_i] = 0$. Therefore, in the case where the noise on the observations is Gaussian with zero mean, the value of $s \geq \sigma$ and the two values are equal when the model is optimal.

However, this is not a direct indicator of the quality of the identified solution, but rather how well the found model matches the input data: consider, for example, the limiting case of underdetermined systems where the residual will always be zero, regardless of the amount of noise affecting the individual observations.

The most suitable index for estimating the model is the variance-covariance matrix of the parameters (*Parameter Variances and Covariances matrix*).

The forward propagation of covariance has already been demonstrated in section 2.6, and with a quick reference, there are three methods to perform this operation: The first is based on the linear approximation of the model and involves the use of the Jacobian, the second is based on the more generic technique of Monte Carlo simulation, and finally, a modern alternative that averages between the first two is the *Unscented Transformation* (section 2.12.5), which empirically allows for estimates up to the third order in the case of Gaussian noise.

The desire to assess the quality of the identified parameters $\hat{\boldsymbol{\beta}}$ given the estimated noise covariance (*Covariance Matrix Estimation*) is precisely the opposite case, as it requires calculating the backward propagation of the variance (*backward propagation*). In fact, once this covariance matrix is obtained, it is possible to define a confidence interval around $\hat{\boldsymbol{\beta}}$.

The goodness of fit of the parameter estimates $\hat{\boldsymbol{\beta}}$, in the nonlinear case, can be assessed in a first approximation by inverting the linearized version of the model (although in this case, techniques such as Monte Carlo or Unscented Transform can also be employed for more rigorous estimates).

It is possible to identify the covariance matrix associated with the proposed solution $\hat{\boldsymbol{\beta}}$ in the case where the function f is one-to-one and differentiable in the vicinity of that solution. Let $f : \mathbb{R}^m \to \mathbb{R}^n$ be a multivariate multidimensional function. It is possible to estimate the mean value $\bar{\mathbf{r}} = \mathrm{E}\left[\mathbf{y} - f(\hat{\boldsymbol{\beta}})\right] \approx \mathbf{0}$ and the cross-covariance matrix $\boldsymbol{\Sigma}_r$ of the residuals; thus, the inverse transformation f^{-1} will have a mean value $\hat{\boldsymbol{\beta}}$ and a covariance matrix

$$\Sigma_{\boldsymbol{\beta}} = (\mathbf{J}^\top \Sigma_r^{-1} \mathbf{J})^{-1} \tag{3.65}$$

with $\mathbf{J}$ being the Jacobian of the model f evaluated at the point $\hat{\boldsymbol{\beta}}$:

$$J_{i,j} = \frac{\partial r_i}{\partial \beta_j}(\hat{\boldsymbol{\beta}}) = -\frac{\partial f_i}{\partial \beta_j}(\hat{\boldsymbol{\beta}}) \tag{3.66}$$

The equation (3.65) is derived by manipulating equation (2.30), which calculates the forward propagation of uncertainty.

Note that this (the inverse of the information matrix) is the lower bound of the Cramer-Rao inequality on the covariance that a consistent estimator of the parameter $\boldsymbol{\beta}$ can achieve.

In cases where the transformation f is underdetermined, the rank of the Jacobian d, with $d < m$, is referred to as the number of essential parameters. In the event of an underdetermined transformation f, the formula (3.65) is not invertible; however, it can be shown that the best approximation of the covariance matrix can be obtained using the pseudo-inverse:

$$\Sigma_{\boldsymbol{\beta}} = (\mathbf{J}^\top \Sigma_r^{-1} \mathbf{J})^{+}$$

Alternatively, it is possible to perform a QR decomposition with pivoting of the Jacobian, identify the linearly dependent columns (through the analysis of the diagonal of the matrix R), and remove them during the inversion of the matrix itself.

In the very common case where f is a scalar function and the observation noise is independent with constant variance, the asymptotically estimated covariance matrix (*Asymptotic Covariance Matrix*) can be expressed more simply as

$$\Sigma_{\boldsymbol{\beta}} = (\mathbf{J}^\top \mathbf{J})^{-1} \sigma^2 \tag{3.67}$$

with σ^2 representing the variance of the observation noise, having applied the assumption $\boldsymbol{\Sigma}_r = \sigma^2 \mathbf{I}$ which holds in the case of independent realizations. Since $\mathbf{J}$ is a function solely of the geometry of the problem, the matrix $(\mathbf{J}^\top \mathbf{J})^{-1}$ is also a function only of the problem and not of the observations. Asymptotically, the estimate tends to $\boldsymbol{\beta} = \mathcal{N}\left(\hat{\boldsymbol{\beta}}, \Sigma_{\boldsymbol{\beta}}\right)$. The Jacobian matrix, as it indicates how the outputs are sensitive to the parameters, is also referred to as the sensitivity matrix.

The estimation of observation noise can be empirical, assuming the law of large numbers $\sigma = s$, calculated through

$$\sigma^2 \approx \frac{\sum_{i=1}^{n} r_i^2}{n-m} \tag{3.68}$$

using the posterior statistics of the error on the data r_i. The denominator $n-m$ represents the statistical degrees of freedom of the problem: in this way, the estimated variance is infinite when the number of unknowns in the model equals the number of collected data points.

The Eicker-White covariance estimator is slightly different and its study is left to the reader.

The variance-covariance matrix of the parameters represents the error ellipsoid.

A useful metric for evaluating the problem is the D-optimal configuration (D-optimal design):

$$\det\left(\mathbf{J}^\top \mathbf{J}\right)^{-1} \tag{3.69}$$

which minimizes the determinant of the variance-covariance matrix, or conversely, maximizes the Fisher information matrix:

$$\det \mathbf{F}\left(\boldsymbol{\beta}\right) \tag{3.70}$$

Geometrically, this approach minimizes the volume of the error ellipsoid.

Other metrics include the E-optimal design, which consists of maximizing the smallest eigenvalue of the Fisher matrix, or equivalently, minimizing the largest eigenvalue of the variance-covariance matrix. Geometrically, this minimizes the maximum diameter of the ellipsoid.

3.6 Notable Regressions

In this section, we will examine some notable regressions related to very simple models, such as planes and circles.

3.6.1 Linear Regression

Let

$$y = mx + q + \varepsilon \tag{3.71}$$

be the equation of the line expressed in explicit form with the measurement error fully incorporated along the y axis. With the error along the y axis, the cost function to be minimized is

$$S = \frac{1}{2n}\sum_{i=1}^{n}\left(mx_i + q - y_i\right)^2 \tag{3.72}$$

.

The solution to the problem is the point where the gradient of S in m and q is zero, that is:

$$\begin{aligned} m &= \frac{\overline{(xy)} - \bar{x}\bar{y}}{\overline{x^2} - \bar{x}^2} = \frac{\operatorname{cov}(x,y)}{\operatorname{var}(x)} \\ q &= -m\bar{x} + \bar{y} \end{aligned} \tag{3.73}$$

With $\bar{x}$, the mean value of the samples x_i (using the same formalism, other quantities are also indicated). The line passes through the point $(\bar{x}, \bar{y})$, the centroid of the distribution.

It is easy to modify this result if one wishes to minimize the deviation along the x instead of along the y, or to represent the equation of the line in implicit form.

3.6.2 Orthogonal Distance Fit

In the case where the error is present on both axes (noise as a function of distance), the formulation of the cost function S that maximizes the likelihood is referred to as the *Orthogonal least-squares line fit*. The error can indeed be expressed using the distance between the point and the line, according to equation (1.31). The regression that utilizes this metric, therefore known as *Perpendicular Regression* or *Total least squares* (see section 3.2.2), is meaningful when both coordinates are affected by error, meaning they are both random variables. The amount of noise in the two components is assumed to be equal (for the more general case, see the discussion in section 2.4). The error function S to be minimized is the distance between the point and the line:

$$S = \frac{1}{2n}\sum_{i=1}^{n}\frac{\left(ax_i + by_i + c\right)^2}{a^2 + b^2} \tag{3.74}$$

. The minimum is found at $\nabla S = 0$. It is noteworthy that in the case of perpendicular distance, there exists both a minimum and a maximum as solutions, and therefore there will be two values of lines (orthogonal to each other), both of which are solutions to the system.

From the partial derivative $\frac{\partial S}{\partial c} = 0$, it follows that the regression line passes through the centroid $(\bar{x}, \bar{y})$ of the distribution, that is,

$$c = -a\bar{x} - b\bar{y} \tag{3.75}$$

with $\bar{x}$ and $\bar{y}$ being the means of samples x_i and y_i respectively.

The error function (3.74), using the relationship (3.75), can be expressed as: It seems that you've entered a placeholder or a command for a math block. Please provide the specific content or context you would like to translate, and I'll be happy to assist you! that is, by making appropriate substitutions $S_{xx} = \text{var}(x)$, $S_{yy} = \text{var}(y)$ and $S_{xy} = \text{cov}(x, y)$:

$$S = \frac{a^2 S_{xx} + 2abS_{xy} + b^2 S_{yy}}{a^2 + b^2} \tag{3.76}$$

more easily derivable. The expression (3.76) for the error is not of a general nature, but is valid only for all lines that pass through the centroid of the distribution. Being a homogeneous form, it is known up to a multiplicative factor: therefore, there is not a single solution but a relationship that links the parameters. Excluding the cases $a = 0$, $b = 0$ (to be treated separately), the constraint for deriving the minimum/maximum has the form of type

$$(a^2 - b^2)S_{xy} + ab(S_{yy} - S_{xx}) = 0 \tag{3.77}$$

, which is the solution to the problem.

It is worth noting that the same result can be obtained in a much simpler manner by applying the SVD decomposition to the equation of the lines. In the case of linear regression, the SVD decomposition minimizes both the algebraic and geometric errors (the algebraic and geometric errors coincide when all noise-affected terms remain confined to the known term).

3.6.3 Orthogonal Regression to a Plane

The considerations made for the line can also be extended to the plane. It should be emphasized that the orthogonal regressions of a line, a plane, or a hyperplane are to be regarded as an eigenvalue problem and can be solved through Singular Value Decomposition (SVD), which is precisely the main application of Principal Component Analysis (PCA).

Let $\mathbf{p_0} = \mathrm{E}[\mathbf{p}]$ be the centroid of the points involved in the regression. Given the equation of the plane (1.49) and using the sum of distances as the error function (1.52), we immediately obtain the constraint:

$$k = -\mathbf{p_0} \cdot \hat{n} \tag{3.78}$$

that is, as already noted in the linear case, the centroid of the distribution lies on the plane. Starting from this initial constraint, it is possible to describe the plane as

$$(\mathbf{p} - \mathbf{p_0}) \cdot \hat{n} = 0 \tag{3.79}$$

an overdetermined homogeneous system, whose solution can be obtained using the pseudoinverse (for example, through QR factorization or SVD). The value of $\hat{n}$ thus derived will be known up to a multiplicative factor, and for this reason, it can always be normalized, forcing it to unit length (the solutions obtained through factorizations are usually already normalized).

3.6.4 Polynomial Function Linear Regression

The method applied to obtain linear regression to a line expressed in explicit form can be generalized to any polynomial function of the type:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_m x^m + \varepsilon \tag{3.80}$$

where $\beta_0 \ldots \beta_m$ are the parameters of the curve to be determined, parameters that are obtained by seeking the minimum of the error function described in (3.6). The derivatives of a polynomial function are noteworthy:

$$\begin{aligned} \tfrac{\partial S}{\partial \beta_j} &= \textstyle\sum_{i=0}^{n} (\beta_0 + \ldots + \beta_m x_i^m - y_i) x_i^j \\ &= \beta_0 \textstyle\sum x_i^j + \ldots + \beta_m \sum x_i^{j+m} - \sum y_i x_i^j \end{aligned} \tag{3.81}$$

Setting the gradient to zero means solving the associated system:

$$\begin{bmatrix} \sum 1 & \ldots & \sum x_i^m \\ \sum x_i & \ldots & \sum x_i^{m+1} \\ \vdots & \ddots & \vdots \\ \sum x_i^m & \ldots & \sum x_i^{2m} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_m \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum y_i x_i \\ \vdots \\ \sum y_i x_i^m \end{bmatrix} \tag{3.82}$$

which is a symmetric matrix.

Alternatively, it is possible to leverage the theory of the pseudoinverse (see section **??**) and directly use equation (3.80) to construct an overdetermined linear system:

$$\begin{bmatrix} 1 & x_1 & \dots & x_1^m \\ 1 & x_2 & \dots & x_2^m \\ \vdots & & & \vdots \\ 1 & x_n & \dots & x_n^m \end{bmatrix} \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{3.83}$$

Vandermonde matrix. The solution to this system allows us to obtain the coefficients of the polynomial that minimizes the square of the residuals. When considering the pseudoinverse solved with the method of *normal equations*, it becomes evident that the resulting system is exactly the same as that of equation (3.82).

As will be seen in other parts of this book, matrices such as the Vandermonde matrix, where the different columns have varying orders of magnitude, are ill-conditioned and require normalization to enhance their numerical stability.

3.6.5 Regression to a Circle

The regression of a set of points to the equation of a circle (*circular regression*) can be achieved by minimizing both an algebraic distance and a geometric distance.

To calculate the linear regression of a dataset towards the equation of a circle centered at (x_0, y_0) with radius r, the function to minimize is

$$S = \sum \left((x_i - x_0)^2 + (y_i - y_0)^2 - r^2 \right)^2 \tag{3.84}$$

where the orthogonal distance between the points and the model is minimized.

To solve the problem, it is advisable to perform a change of variables and minimize the algebraic form:

$$S = \sum \left(z_i + Bx_i + Cy_i + D \right)^2 \tag{3.85}$$

where $z_i = x_i^2 + y_i^2$ has been introduced for simplicity.

The problem reduces to solving a linear system 3×3 of equations

$$\begin{array}{llll} \sum z_i x_i & +B \sum x_i^2 & +C \sum y_i x_i & +D \sum x_i = 0 \\ \sum z_i y_i & +B \sum x_i y_i & +C \sum y_i^2 & +D \sum y_i = 0 \\ \sum z_i & +B \sum x_i & +C \sum y_i & +D \sum 1 = 0 \end{array} \tag{3.86}$$

which is symmetric and easily solvable.

Once the parameters B, C, and D are obtained, it is possible to derive the original parameters of the circle:

$$x_0 = -\frac{B}{2} \quad y_0 = -\frac{C}{2} \quad r^2 = x_0^2 + y_0^2 - D \tag{3.87}$$

The same result can be achieved using the linear solvers discussed earlier. Consider, for example, an algebraic representation of a circle

$$f(\mathbf{x}) = a\mathbf{x}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c = 0 \tag{3.88}$$

where $\mathbf{x}$ is the locus of points on the circumference.

Given a list of points that belong to a noisy circumference, the parameters (a, b_x, b_y, c) that describe the circle are obtained by solving the homogeneous constraint system (3.88). As will be detailed in subsequent problems, for purely computational reasons, it is advantageous to normalize the input data, as the different unknowns are associated with data of significantly varying magnitudes.

The algebraic solution is often used as an initial solution for iterative techniques that minimize a different metric. To perform geometric regression, it is necessary to minimize the distances $d_i^2 = \left(\|\mathbf{x}_i - (x_0, y_0)^\top\| - r \right)^2$. To minimize this quantity, a nonlinear least squares solver is required, such as Levenberg-Marquardt, along with the computation of the derivatives of the cost function.

An alternative is to parameterize the problem in a space different from the Cartesian one. By using the parametric form of the equation of the circle

$$\begin{aligned} x &= x_0 + r \cos \varphi \\ y &= y_0 + r \sin \varphi \end{aligned} \tag{3.89}$$

the quantities to be minimized become

$$\begin{aligned} x_i - x_0 + r \cos \varphi_i &\approx 0 \\ y_i - y_0 + r \sin \varphi_i &\approx 0 \end{aligned} \tag{3.90}$$

which can be easily differentiated. To each input data (x_i, y_i) an additional unknown φ_i, a subsidiary variable, is associated. In this way, a nonlinear system is created with $3 + n$ unknowns and $2n$ equations.

3.6.6 Ellipse Regression

As with the circle, it is possible to perform both algebraic and geometric minimization.

The quadratic equation of an ellipse is

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c = 0 \tag{3.91}$$

where $\mathbf{A}$ is a symmetric, positive definite matrix. In this case as well, the solution to the homogeneous problem (3.91) allows us to derive the 6 unknowns (known up to a multiplicative factor) of the system.

The nonlinear solution that minimizes the geometric quantity can be obtained using the parametric representation of the ellipse

$$\mathbf{x} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} a\cos\varphi \\ b\sin\varphi \end{bmatrix} \tag{3.92}$$

where (x_0, y_0) represents the center of the ellipse, (a, b) the lengths of the two semi-axes, and α the rotation of the ellipse with respect to the center. As with the circle, the φ_i will be auxiliary variables, and the nonlinear problem becomes one of $5 + n$ unknowns with $2n$ equations.

3.6.7 Regression on a Conic

It is clearly possible to generalize the regression of the parabola, the circle, and the ellipse to any conic (section 1.6) that is arbitrarily oriented.

Let $(x_i, y_i)^\top$, with $i = 1, \ldots, n$, be points affected by noise belonging to the set of points to be estimated.

The equation (1.56) can be rewritten in the form

$$\mathbf{a}_i^\top \boldsymbol{\beta} = 0 \tag{3.93}$$

where $\mathbf{a}_i = \left\{x_i^2, x_i y_i, y_i^2, x_i, y_i, 1\right\}$ and $\boldsymbol{\beta} = \{a, b, c, d, e, f\}$ from which it is evident that to obtain the parameters $\boldsymbol{\beta}$ of any conic, one can proceed by solving a homogeneous problem of type $\mathbf{A}\boldsymbol{\beta} = 0$ in 6 unknowns, minimizing a quantity of the form

$$S = \sum_{i=1}^{n} \mathbf{a}_i^\top \boldsymbol{\beta} \tag{3.94}$$

Such a solution clearly minimizes an algebraic error rather than a geometric one; therefore, this is not the optimal estimator.

An alternative formulation for deriving the parameters of conics can be found in [FPF99].

Finally, to determine whether a point is close to the equation of a conic, or to obtain a geometric approximation of the point-conic distance, one can calculate the Sampson error (section 3.3.7) by leveraging the fact that, for a conic defined by equation (1.56), the gradient of the variety takes on a very simple form to compute:

$$\nabla f(x, y) = (2ax + by + d, bx + 2cy + e) \tag{3.95}$$

3.7 Logistic Regression

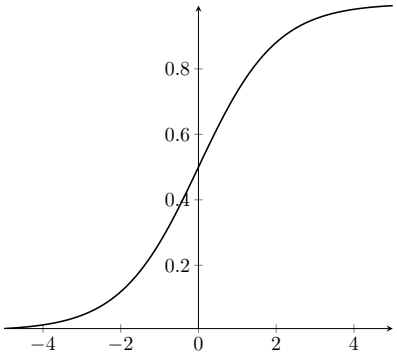


Figure 3.2: Logistic Function

There exists a family of linear models that relate the dependent variable to the explanatory variables through a nonlinear function, known as generalized linear models. Logistic regression falls within this class of models, specifically in the case where the variable y is dichotomous, meaning it can only take on values 0 or 1. By its nature, this type of problem holds significant importance in classification tasks.

In the case of binary problems, it is possible to define the probability of success and failure.

$$\begin{aligned} P[Y=1|\mathbf{x}] &= p(\mathbf{x}) \\ P[Y=0|\mathbf{x}] &= 1-p(\mathbf{x}) \end{aligned} \tag{3.96}$$

The response of a linear predictor of the form

$$y' = \boldsymbol{\beta}\cdot\mathbf{x}+\varepsilon \tag{3.97}$$

is not constrained between 0 and 1, thus it is unsuitable for this purpose. It is necessary to associate the response of the linear predictor with the response of a certain function g, a probability function $p(\mathbf{x})$

$$g(p(\mathbf{x})) = \boldsymbol{\beta}\cdot\mathbf{x}+b \tag{3.98}$$

where $g(p)$, the *mean function*, is a nonlinear function defined over $[0,1]$. $g(p)$ must be invertible, and the inverse $g^{-1}(y')$ is the *link function*.

A widely used model for the function $g(p)$ is the *logit* function defined as:

$$logit(p) = \log\frac{p}{1-p} = \boldsymbol{\beta}\cdot\mathbf{x} \tag{3.99}$$

The function $\frac{p}{1-p}$, since it represents how many times success is greater than failure, is referred to as the *odds-ratio*. Consequently, the function (3.99) represents the logarithm of the probability of an event occurring relative to the probability of the same event not occurring (*log-odds*).

Its inverse function exists and is given by

$$\mathrm{E}[Y|\mathbf{x}] = p(\mathbf{x}) = \frac{e^{\boldsymbol{\beta}\cdot\mathbf{x}}}{1+e^{\boldsymbol{\beta}\cdot\mathbf{x}}} \tag{3.100}$$

'and it is the logistic function.

The maximum likelihood method in this case does not coincide with the least squares method but with

$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{i=1}^{n} f(y_i|\mathbf{x}_i) = \prod_{i=1}^{n} p^{y_i}(\mathbf{x}_i)\left(1-p^{y_i}(\mathbf{x}_i)\right) \tag{3.101}$$

from which the log-likelihood function

$$\log\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{n} y_i(\boldsymbol{\beta}\cdot\mathbf{x}_i) - \log\left(1+e^{\boldsymbol{\beta}\cdot\mathbf{x}_i}\right) \tag{3.102}$$

is derived. The maximization of this function, through iterative techniques, allows for the estimation of the parameters $\boldsymbol{\beta}$.

3.8 M-Estimator

The use of least squares regression is chosen both due to the maximum likelihood function and, more importantly, because of the simplicity of the derivatives obtained in the Jacobian.

In the problems discussed so far, constant variance and normally distributed observation errors have almost always been assumed. If the noise were solely Gaussian, this approach would be theoretically correct; however, real-world applications typically exhibit distributions that consist of Gaussian noise belonging to the model and noise associated with elements that do not belong to the model itself (outliers). In this condition, least squares regression results in treating all points as if the error were Gaussian, meaning that points close to the model are given little weight, while points far from the model are heavily weighted. These latter points are usually outliers, based purely on probability considerations.

The approach to uniquely address these problems was introduced by John Nelder, who coined the term generalized linear models (*GLM*, *Generalized Linear Models*) for these techniques.

To address this problem, it is necessary to change the metric through which errors are evaluated: a first example of an alternative metric that could resolve the issue is the regression to the absolute value. However, calculating the minimum of the error function expressed as the absolute distance (*Least absolute deviations regression*) is not straightforward, as the derivative is not continuous and requires the use of iterative optimization techniques: differentiable metrics are preferable in this case.

Peter Huber proposed in 1964 a generalization of the minimization concept to maximum likelihood by introducing *M-estimators*.

Some examples of regression functions are shown in Figure 3.3.

An *M-Estimator* replaces the metric based on the sum of squares with a metric based on a generic function ρ (*loss function*) that has a single minimum at zero and exhibits sub-quadratic growth. M-Estimators generalize least squares regression: by setting $\rho(\mathbf{r}) = \|\mathbf{r}\|^2$, one obtains the classical form of regression.

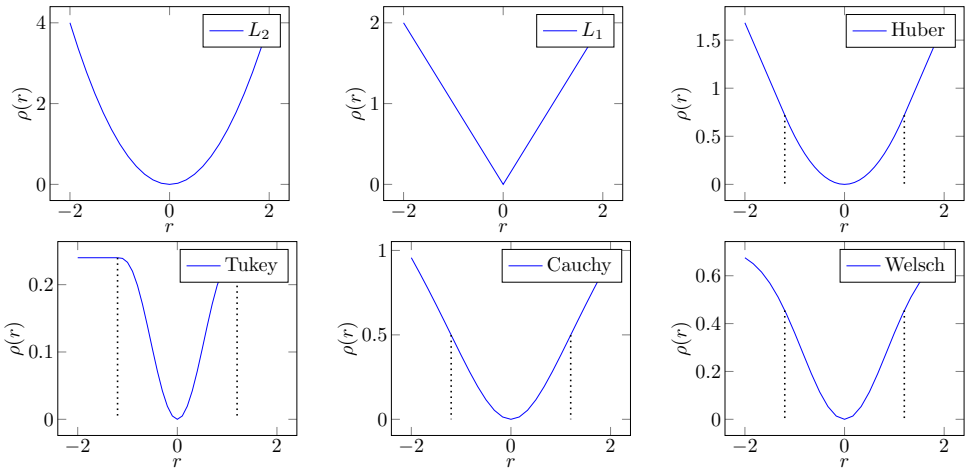


Figure 3.3: Some examples of weight functions for regressions: least squares regression (L2 metric), linear regression (L1), Huber estimators, Tukey's biweight, the Lorentzian function (Cauchy), and the Welsch function (Leclerc).

Finally, if the loss function is monotonically increasing, it is referred to as an *M-estimator*, whereas if the loss function is increasing near zero but decreasing away from zero, it is referred to as a *Redescending M-estimator*.

The estimation of parameters is achieved through the minimization of a summation of weighted generic quantities:

$$\min_{\beta} \sum \rho\left(\frac{\mathbf{r}_i}{\sigma_i}\right) \tag{3.103}$$

whose solution, whether in closed form or iterative, differs from that of the least squares due to the different derivative of the function ρ:

$$\sum_{i=1}^{n} \frac{1}{\sigma_i} \rho'\left(\frac{\mathbf{r}_i}{\sigma_i}\right) \frac{\partial \mathbf{r}_i}{\partial \beta_j} = 0 \qquad j = 1, \ldots, m \tag{3.104}$$

3.9 Iteratively Reweighted Least Squares

An orthogonal technique to M-Estimators is the Iteratively Reweighted Least Squares (IRLS) method [gre84]. This technique involves estimating new weights at each iteration, which are then used to derive a new solution. This method can be applied to both linear and non-linear problems.

In the linear case, the objective is to minimize a cost function of the form

$$\|\mathbf{W}\mathbf{r}\|^2 = \sum_i w_i^2 r_i^2 = \mathbf{r}^\top \mathbf{W}^\top \mathbf{W}\mathbf{r} \tag{3.105}$$

where the matrix $\mathbf{W}$ is a diagonal matrix with the weights w_i placed along the diagonal.

In the case of an over-dimensioned problem, this has a solution

$$\mathbf{x} = \left[\mathbf{A}^\top \mathbf{W}^\top \mathbf{W}\mathbf{A}\right]^{-1} \mathbf{A}^\top \mathbf{W}^\top \mathbf{W}\mathbf{b} \tag{3.106}$$

The same approach is applied to non-linear systems.

3.10 Black-Rangarajan duality

The iterative least squares technique becomes optimal when the weights are appropriately selected, aiming to combine the aspects of robust estimation with the rejection of outliers.

The duality described by Black-Rangarajan [BR96] connects these two aspects by showing that robust estimation can be viewed as a process of outlier removal. This dualism enables the use of Graduated Non-Convexity (GNC), a technique that gradually transforms a non-convex problem into a convex one, making it easier to find a global solution without the need for an initial hypothesis.

In practice, this dualism combined with GNC is used to develop algorithms that are robust to a high percentage of outliers, surpassing traditional methods such as RANSAC in terms of accuracy and speed.

The Black-Rangarajan duality provides a framework for establishing the relationship between M-estimators and linear processes. Typical robust loss functions include Welsch (Leclerc), Cauchy (Lorentzian), Charbonnier (pseudo-Huber, ℓ_1-ℓ_2), Huber, Geman-McClure, smooth truncated quadratic, truncated quadratic, Tukey's biweight functions, and so on. The Black-Rangarajan duality of these functions can be found in [ZB17], from which I present an excerpt in the table below:

Name	$\rho(x)$	$\omega(x)$
Quadratic	$\frac{x^2}{2}$	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}
Truncated Quadratic	$\min\left\{\tau, x\right\}^2/2$	$\begin{cases} 1 & \lvert x\rvert \le \tau \\ 0 & \lvert x\rvert > \tau \end{cases}$

where there is the *loss function* $\rho(x)$ and its corresponding weight update function $\omega(x)$, and let $\tau \stackrel{\text{def}}{=\!=} \max\{x : \omega(x) = 1\}$ be the radius of the unconditional *inliers*.

3.11 Hough Transform

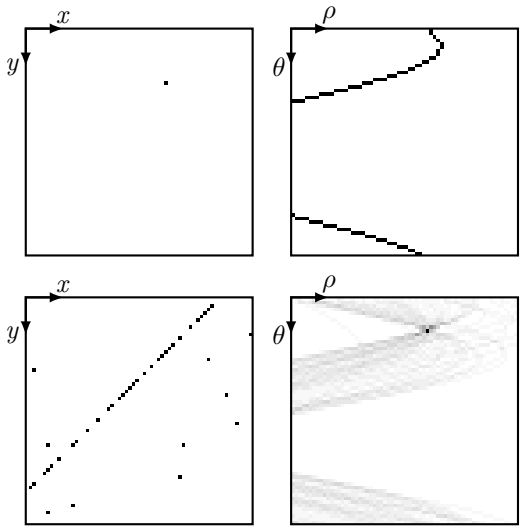


Figure 3.4: Example of the Hough Transform for detecting lines in polar coordinates: accumulator map (top right) of a single point (top left), and accumulator map (bottom right) of a series of collinear points along with *outliers* (bottom left).

Let $g(\mathbf{x}, \boldsymbol{\beta}) = 0$ be a continuous variety in $\mathbf{x}$ for which it is required to estimate the parameters $\boldsymbol{\beta} \in \mathbb{R}^m$. To derive these parameters and fully define the function, a set of coordinates $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is available, which belong to the locus of points of the function, potentially affected by noise but, above all, potentially *outliers*.

The Hough Transform is a technique that allows for the grouping of a "highly probable" set of points that satisfy certain parametric constraints [PIK92].

For every possible point $\boldsymbol{\beta}^*$ in the parameter space, it is possible to associate a score $H(\boldsymbol{\beta})$ of the form

$$H(\boldsymbol{\beta}^*) = \{\mathbf{x} : g(\mathbf{x}, \boldsymbol{\beta}^*) = 0, \mathbf{x} \in S\} \tag{3.107}$$

which represents the number of elements in S that satisfy the constraint expressed by g. The parameter $\boldsymbol{\beta}^*$ that maximizes this score is the statistically most probable solution to the problem.

Let the function $p(\mathbf{x}, \boldsymbol{\beta})$ be a likelihood index between the pair $(\mathbf{x}, \boldsymbol{\beta})$ and the constraint expressed by $g(\mathbf{x}, \boldsymbol{\beta}) = 0$. The function p is typically a binary function, but by generalizing, it can also comfortably represent a probability. Through the function p, it is possible to incrementally construct the Hough transform $H(\boldsymbol{\beta})$ via

$$H(\boldsymbol{\beta}) = \sum_{i=1}^{n} p(\mathbf{x}_i, \boldsymbol{\beta}) \tag{3.108}$$

. The Hough transform is the sum of all these functions.

For specific constraints, it is possible to further simplify this approach in order to reduce computational load and memory usage.

Let there be $\beta_1 \ldots \beta_m$ parameters to estimate, which are quantifiable and bounded, and let f and β_1 be a function and a parameter such that the function $g(\mathbf{x}, \boldsymbol{\beta}) = 0$ can be expressed as

$$\beta_1 = f(\mathbf{x}, \beta_2, \ldots, \beta_m) \tag{3.109}$$

If the function g can be expressed as in equation (3.109), it is possible to estimate the parameters $\boldsymbol{\beta}$ that represent the most "likely" model among all the provided points $\mathbf{x}$ using the method of the discrete Hough transform. For each element $\mathbf{x}$, it is possible to vary the parameters $\beta_2 \dots \beta_m$ within their range and insert the values of β_1 returned by the function (3.109) into the accumulator image $H(\boldsymbol{\beta})$.

In this way, it is possible to generate an n-dimensional probability map using observations $\mathbf{x}$ affected by noise and potentially *outliers*. Similarly, the Hough method allows for the estimation of a model in the presence of a mixture of models with different parameters.

The Hough method allows for progressively improved performance as the number of constraints increases, dynamically limiting, for example, the range of parameters associated with the sample $\mathbf{x}$. The Hough algorithm can be viewed as a degenerate form of *template matching*.

The use of Hough is typically interesting when the model has only 2 parameters, as it can be easily visualized on a two-dimensional map.

A very common example of the Hough transform is the case where g (the model) is a line, expressed in polar form as in equation (1.46), where the parameters to be derived are θ and ρ: it is evident that for every pair of points (x, y) and for all possible quantized and limited angles of θ (since the angle is a bounded parameter), there exists one and only one ρ that satisfies equation (1.46).

It is therefore possible to create a map $H(\theta, \rho)$ where, for each point $(x, y) \in S$ and for each $\theta \in [\theta_{min}, \theta_{max}]$, the element associated with $(\theta, \cos\theta x + \sin\theta y)$ is incremented on the accumulator map, a relationship that satisfies the equation (1.46) of the line expressed in polar coordinates.

3.12 RANSAC

The *RANdom Sample And Consensus* algorithm is an iterative method for estimating the parameters of a model where the dataset is heavily influenced by the presence of *outliers*. This algorithm [FB81] is a non-deterministic approach based on the random selection of the elements that generate the model.

RANSAC and all its variants can be viewed as algorithms that iteratively alternate between two phases: the hypothesis generation phase and the hypothesis evaluation phase.

The algorithm, in brief, consists of randomly selecting s samples from all n input samples $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, with s large enough to derive a model (the hypothesis). Once a hypothesis is obtained, the number of n elements from X that are close enough to it to belong to it is counted. A sample $\mathbf{x} \in S$ belongs or does not belong to the hypothetical model (i.e., it is a hypothetical *inlier* or *outlier*) if its distance from the model $d_{\boldsymbol{\beta}}(\mathbf{x})$ is less than or greater than a given threshold τ, a threshold that is typically dependent on the problem. The threshold τ encounters practical issues where the additive error is Gaussian, meaning that the support is infinite. In this case, it is still necessary to define a probability p of detecting the *inliers* to establish a threshold τ.

All the samples that satisfy the hypothesis are called consents (*consensus*).

The set of consents S associated with the hypothesis $\boldsymbol{\beta}$ is the *consensus set* of $\boldsymbol{\beta}$:

$$S(\boldsymbol{\beta}) = \{\mathbf{x} \in X : d_{\boldsymbol{\beta}}(\mathbf{x}) < \tau\} \tag{3.110}$$

Among all the randomly generated models, the model that meets a specific metric is selected; for instance, in the original RANSAC, this is the model that has the maximum consensus cardinality.

One of the challenges is determining how many hypotheses to generate in order to have a good probability of obtaining the correct model.

There exists a statistical relationship between the number of iterations N and the probability p of identifying a solution composed solely of *inliers*. The number of attempts N must satisfy $(1 - P)^N \leq 1 - p$, which can be expressed as

$$N \geq \frac{\log(1-p)}{\log(1-P)} \tag{3.111}$$

, where P is the probability of having selected a solution consisting only of *inliers*.

Typically, as an approximation of P, one can use $P = (1 - \epsilon)^s$[3] and therefore

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)} \tag{3.112}$$

with ϵ being the a priori probability of selecting an *outlier* and s the number of elements needed to define the model. The size of a minimum *consensus set* can be statistically deduced simply as $T = (1 - \epsilon)n$.

Typically, s is chosen to be equal to the number of elements required to create the model; however, if it exceeds this number, the generated model must be constructed through numerical regression based on the provided constraints. This

[3] The correct estimate is $P = \frac{\binom{pn}{s}}{\binom{n}{s}}$ with pn being the total number of *inliers*, n the total number of elements, and s the number of elements required.

condition is necessary when the noise variance is high, although it increases the risk of incorporating *outliers* into the constraints.

3.12.1 M-SAC

The RANSAC policy is to return, among all the generated hypotheses, the one that has the smallest number of outliers beyond a fixed threshold. This policy can be viewed as an *M-estimator* that minimizes a *loss function* of the type

$$\rho = \begin{cases} 0 & |e| < \tau \\ 1 & |e| > \tau \end{cases} \tag{3.113}$$

which assigns a score of 1 to all elements that are farther than the threshold from the evaluated model and a score of 0 to the elements within the threshold τ.

The concept can therefore be generalized in M-SAC techniques (*M-Estimator Sample and Consensus*), where the *loss function* of RANSAC is modified.

As noted in the previous section, the noise in the data can be partially viewed as Gaussian noise on the *inliers* associated with a uniform distribution of *outliers*. The *negative Maximum Likelihood* is, in fact, the theoretically correct *loss function*, which underlies the MLESAC methods, but it is quite computationally intensive.

A good approximation, characteristic of M-SAC techniques, is to use the following *loss function*:

$$\rho = \begin{cases} e^2 & |e| < \tau \\ \tau^2 & |e| > \tau \end{cases} \tag{3.114}$$

This *loss function* models quite well the case of *inliers* affected by zero-mean Gaussian noise and *outliers* distributed uniformly.

3.12.2 LMedS

The outlier rejection algorithm known as the *Least Median of Squares* (*LMedS*) is conceptually very similar to RANSAC: Similar to RANSAC, a model is generated from random samples of the input data; however, instead of selecting the model that gathers the highest number of inliers (or that minimizes a *loss function*), LMdeS selects the model that has the lowest median error among all candidates. Consequently, all input data are compared with the model, sorted by error, and the median value is examined.

The relationship between the probability of identifying *inliers* and the number of iterations is the same as in RANSAC. However, RANSAC requires two parameters (the number of iterations and the threshold to determine whether an element belongs to the dataset), while LMedS only requires the first. By design, LMedS, however, tolerates a maximum presence of 50% *outliers*.

A good overview of the RANSAC, M-SAC, and LMedS techniques can be found in [CKY09].

Chapter 4

Classification

A predominant role in Computer Vision is played by Classification techniques and *machine learning*. The vast amount of information that can be extracted from a video sensor far exceeds the quantity that can be obtained from other sensors; however, it requires complex techniques that enable the exploitation of this wealth of information.

As previously mentioned, statistics, classification, and model fitting can be viewed as different facets of a single topic. Statistics seeks the most accurate Bayesian approach to extract the hidden parameters (of the state or model) of a system, potentially affected by noise, aiming to return the most probable output given the inputs. Meanwhile, classification offers techniques and methods for efficiently modeling the system. Finally, if the exact model underlying a physical system were known, any classification problem would reduce to an optimization problem. For these reasons, it is not straightforward to delineate where one topic ends and another begins.

The classification problem can be reduced to the task of deriving the parameters of a generic model that allows for generalization of the problem, given a limited number of examples.

A classifier can be viewed in two ways, depending on the type of information that the system aims to provide:

1. as a "likelihood" function towards a model, as in equation (4.1);
2. as a partitioning of the input space, as in equation (4.2).

In the first case, a classifier is represented by a generic function

$$f : \mathbb{R}^n \to \mathbb{R}^m \tag{4.1}$$

that allows associating with the input element $\mathbf{x}$, composed of n characteristics representing the *example* to be classified, confidence values regarding the possible $\{y_1, \ldots, y_m\}$ output classes (*categories*):

$$f(\mathbf{x}) = (p(y_1|\mathbf{x}), \ldots, p(y_m|\mathbf{x}))$$

that is, the probability that the observed object is precisely y_i given the observed quantity $\mathbf{x}$.

Due to the infinite nature of possible functions and the lack of additional specific information regarding the problem's structure, the function f cannot be a well-defined function but will be represented by a parameterized model in the form

$$\mathbf{y} = f(\mathbf{x}, \boldsymbol{\beta})$$

where $\mathbf{y} \in \mathbb{R}^m$ is the output space, $\mathbf{x} \in \mathbb{R}^n$ is the input space, and $\boldsymbol{\beta}$ are the model parameters f to be determined during the training phase.

The training phase is based on a set of examples (*training set*) consisting of pairs $(\mathbf{x}_i, \mathbf{y}_i)$, and through these examples, the training phase must determine the parameters $\boldsymbol{\beta}$ of the function $f(\mathbf{x}, \boldsymbol{\beta})$ that minimize, under a specific metric (cost function), the error on the *training set* itself.

To train the classifier, it is therefore necessary to identify the optimal parameters $\boldsymbol{\beta}$ that minimize the error in the output space: classification is also an optimization problem. For this reason, *machine learning*, model *fitting*, and statistics are closely related research areas. The same considerations applied in Kalman or for Hough, as well as everything discussed in the chapter on *fitting* models using least squares, can be utilized for classification, and specific classification algorithms can be employed, for instance, to fit a series of noisy observations to a curve.

It is generally not feasible to produce a complete training set: it is not always possible to obtain every type of input-output association in order to systematically map the entire input space to the output space. Even if this were possible, it would still be costly to have the memory required to represent such associations in the form of a Look Up Table. These are the main reasons for the use of models in classification.

The fact that the *training set* cannot cover all possible input-output combinations, combined with the generation of a model optimized for such incomplete data, can lead to a lack of generalization in the training: elements not present in the

training set may be misclassified due to excessive adaptation to the *training set* (the *overfitting* problem). This phenomenon is typically caused by an optimization phase that focuses more on reducing the error on the outputs rather than on generalizing the problem.

Returning to the ways to visualize a classifier, it often proves simpler and more generalizable to directly derive from the input data a surface in $\mathbb{R}^n$ that separates the categories in the n-dimensional input space. A new function g can be defined that associates a unique output label to each group of inputs, in the form of

$$g : \mathbb{R}^n \to \mathbb{Y} = \{y_1, \ldots, y_m\} \tag{4.2}$$

. This represents the second way to view a classifier.

The expression (4.1) can always be converted into the form (4.2) through a majority voting process:

$$g(\mathbf{x}) = \arg\max_{y_i} p(y_i|\mathbf{x}) \tag{4.3}$$

The classifier, from this perspective, is a function that directly returns the symbol most similar to the provided input. The *training set* must now associate each input (each element of the space) with exactly one output class $y \in \mathbb{Y}$. Typically, this way of viewing a classifier allows for a reduction in computational complexity and resource utilization.

If the function (4.1) indeed represents a transfer function, a response, the function (4.2) can be viewed as a partitioning of the space $\mathbb{R}^n$ where regions, generally very complex and non-contiguous in the input space, are associated with a unique class.

For the reasons stated previously, it is physically impossible to achieve an optimal classifier (except for very small dimensional problems or for simple models that are perfectly known). However, there are several *general purpose* classifiers that can be considered sub-optimal depending on the problem and the required performance.

In the case of classifiers (4.2), the challenge is to obtain an optimal partitioning of the space; thus, a set of fast primitives is required that do not consume too much memory in the case of high values of n. Meanwhile, in case (4.1), a function that models the problem very well is explicitly required, while avoiding specializations.

The information (*features*) that can be extracted from an image for classification purposes is manifold. Generally, using the grayscale/color values of the image directly is rarely employed in practical applications because such values are typically influenced by the scene's brightness and, most importantly, because they would represent a very large input space that is difficult to manage.

Therefore, it is essential to extract from the portion of the image to be classified the critical information (*features*) that best describes its appearance. For this reason, the entire theory presented in section 6 is widely used in *machine learning*.

Indeed, both *Haar features* are extensively utilized due to their extraction speed, and Histograms of Oriented Gradients (*HOG*, section 6.2) are favored for their accuracy. As a compromise and generalization between the two classes of *features*, Integral Channel Features (*ICF*, section 6.3) have recently been proposed.

To reduce the complexity of the classification problem, it can be divided into multiple layers to be addressed independently: the first layer transforms the input space into the feature space, while a second layer performs the actual classification starting from the feature space.

Under this consideration, classification techniques can be divided into three main categories:

Rule-based learning In this case, both the feature space and the parameters of the classification function are determined by a human user, without utilizing any dataset or training examples;

Machine Learning The transformation from input space to feature space is chosen by the user from a finite set of functions, while the extraction of model parameters is left to the processor by analyzing the provided examples;

Representation learning Both the transformation to feature space and the extraction of model parameters are performed by the computer.

Recently, techniques of *Representation Learning* constructed with multiple cascading layers (*Deep Learning*) have been very successful in solving complex classification problems.

Among the techniques for transforming the input space into the feature space, it is important to mention PCA, a linear unsupervised technique. The *Principal Component Analysis* (section 2.10.1) is a method that allows for the reduction of the number of inputs to the classifier by removing linearly dependent or irrelevant components, thereby reducing the dimensionality of the problem while striving to preserve as much information as possible.

Regarding the models and modeling techniques, widely used are:

Regression a regression of data to a model is essentially a classifier. Consequently, all the theory presented in chapter 3 can and should be applied to classify data;

Neural Networks Neural networks allow the generation of functions of the type (4.1) by concatenating sums, multiplications, and strongly nonlinear functions such as sigmoids. Regression techniques enable the estimation of the parameters of this generic model;

Bayesian Classifiers It is possible to use Bayes' theorem directly as a classifier or to combine multiple classifiers in order to maximize the *a posteriori* probability of identifying the correct class (section 4.2);

Decision Tree where classifiers are cascaded with other classifiers (and each node represents some attribute extracted from the input data);

Decision Stump a degenerate decision tree (1 node), allows partitioning the feature space using a simple threshold, thus becoming the simplest classifier of type (4.2) and an example of a weak classifier;

Ensemble Learning Multiple weak classifiers can be related to each other (Ensemble Learning, see section 4.6) in order to maximize some global metric (for example, the margin of separation between classes). In fact, they are not true classifiers but rather techniques for combining several simple classifiers to generate a complex classifier (ensemble).

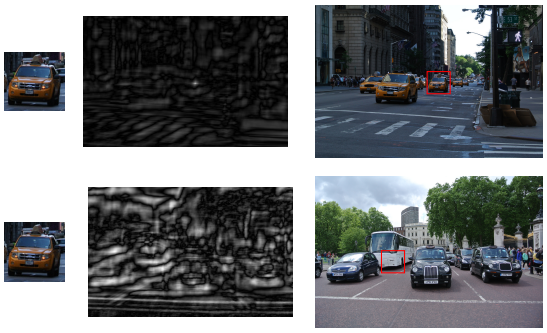


Figure 4.1: Example of *Template Matching*. The approach works well on the source image, but it cannot be extended to other images, especially with significant variations in brightness and scale.

4.1 Binary Classifiers

A particularly common case of classifier is that of a binary classifier. In this scenario, the problem consists of finding a relationship that links the *training-set* $S = \{(\mathbf{x}_1, y_1) \dots (\mathbf{x}_l, y_l)\} \in (\mathbb{X} \times \mathbb{Y})$ where $\mathbb{X} \subseteq \mathbb{R}^n$ is the vector that gathers the information to be used for training and $\mathbb{Y} = \{+1, -1\}$ is the space of the associated classes.

Examples of intrinsically binary classifiers include:

LDA *Linear Discriminant Analysis* (section 4.3) is a technique that finds the separating plane between classes that maximizes the distance between the distributions;

Decision Stump One-level decision trees have only two possible outputs;

SVM *Support Vector Machines* (section 4.4) partition the feature space by maximizing the margin using hyperplanes or simple surfaces.

Particular interest is given to linear classifiers (LDA and Linear SVM), which, in order to solve the binary classification problem, identify a separating hyperplane $(\mathbf{w}, b)$ between the two classes.

The equation of a hyperplane, slightly modifying the formula (1.49), is

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{4.4}$$

where the normal vector $\mathbf{w}$ may not necessarily be of unit norm. A hyperplane divides the space into two subspaces where the equation (4.4) has opposite signs. The separating surface is a hyperplane that divides the space into two subregions representing the two categories of binary classification.

A linear classifier is based on a discriminant function

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{4.5}$$

. The vector $\mathbf{w}$ is referred to as the *weight vector*, and the term b is called the *bias*. Linear classifiers hold significant importance as they transform the problem from multidimensional to scalar by projecting along the axis $\mathbf{w}$.

The sign of the function $f(\mathbf{x})$ represents the outcome of the classification. A separating hyperplane corresponds to identifying a linear combination of the elements $\mathbf{x} \in \mathbf{X}$ in such a way as to obtain

$$\hat{y} = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b) \tag{4.6}$$

.

4.2 Bayesian Classifiers

The *Bayes* theorem, associated with Computer Vision, represents a fundamental technique for pattern classification, based on experience (*training set*).

To understand Bayes' theorem, it is essential to consider a simple example. Suppose we want to classify fruit that is presented to an observer (or a processor in the extreme case). For simplicity, let us assume that there are only two types of fruit (the categories of the classifier), for instance, oranges and apples. For humans, as well as for machines, determining the type of fruit being observed involves examining specific characteristics (*features*) extracted from the observation of the fruit, using appropriate techniques.

If the fruits are selected completely at random and no additional information can be extracted from them, the optimal approach for classifying them would be to provide a completely random response.

The Bayesian decision theory plays an important role only when some *a priori* information about the objects is known.

As a first step, let us assume that there is no prior knowledge about the characteristics of the fruits, but it is known that 80% of the fruits are apples and the remaining are oranges. If this is the only information available to make a decision, one would instinctively tend to classify the fruit as an apple (the optimal classifier): every fruit will be classified as an apple since, in the absence of other information, this is the only way to minimize the error. The *a priori* information in this case consists of the probabilities that the chosen fruit is an apple or an orange.

Let us now examine the case where it is possible to extract additional information from the observed scene. The concept of Bayes applied to classification is very intuitive from this perspective: if I observe a particular measurable characteristic of the image x (*features*), I can estimate the probability that this image represents a certain class y_i *a posteriori* of the observation. From this standpoint, Bayesian classifiers provide exactly the probability that the input data vector represents the specified output class.

4.2.1 Bayes' Theorem

The definition of conditional probability allows us to immediately obtain the following fundamental result:

Teorema 1 (Bayes' theorem) *Let $\{\Omega, \mathcal{Y}, p\}$ be a probability space. Consider the events $y = y_i$ (abbreviated as y_i) with $i = 1..n$ being a complete system of events of Ω and $p(y_i) > 0\ \forall i = 1..n$.*

In this case, $\forall y_i \in \mathcal{Y}$ with $p(y_i) > 0$ will yield:

$$p(y_i|x) = \frac{p(y_i)p(x|y_i)}{\sum_{j=1}^{n} p(y_j)p(x|y_j)} \tag{4.7}$$

and this $\forall i = 1..n$.

The Bayes' theorem is one of the fundamental elements of the *subjectivist* or *personal* approach to probability and statistical inference. The system of alternatives y_i with $i = 1..n$ is often interpreted as a set of *causes*, and Bayes' theorem, given the prior probabilities of the different causes, allows for the assignment of probabilities to the causes given an effect x. The probabilities $p(y_i)$ with $i = 1..n$ can be interpreted as the *a priori* knowledge (usually denoted by π_i), which is the knowledge available before conducting a *statistical experiment*. The probabilities $p(x|y_i)$ with $i = 1..n$ are interpreted as the *likelihood* or information regarding x that can be obtained by performing an appropriate statistical experiment. Thus, Bayes' formula suggests a mechanism for *learning from experience*: by combining some a priori knowledge about the event y_i provided by $p(y_i)$ with the knowledge that can be acquired from a statistical experiment given by $p(x|y_i)$, one arrives at a better understanding represented by $p(x_i|y)$ of the event x_i, also referred to as *posterior probability* after conducting the experiment.

We can have, for example, the probability distribution for the color of apples, as well as that for oranges. To use the notation introduced earlier in the theorem, let y_1 denote the state in which the fruit is an apple, y_2 the condition in which the fruit is an orange, and let x be a random variable representing the color of the fruit. With this notation, $p(x|y_1)$ represents the density function for the event color x conditioned on the fact that the state is apple, and $p(x|y_2)$ that it is orange.

During the training phase, it is possible to construct the probability distribution of $p(x|y_i)$ for i apple or orange. In addition to this knowledge, the prior probabilities $p(y_1)$ and $p(y_2)$ are always known, which simply represent the total number of apples compared to the number of oranges.

What we are looking for is a formula that indicates the probability of a fruit being an apple or an orange, given that a certain color x has been observed.

The Bayes' formula (4.7) allows precisely this:

$$p(y_i|x) = \frac{p(x|y_i)p(y_i)}{p(x)} \tag{4.8}$$

given the prior knowledge, it enables the calculation of the posterior probability that the state of the fruit is y_i given the measured *feature* x. Therefore, upon observing a certain x on the conveyor belt, having calculated $p(y_1|x)$ and $p(y_2|x)$, one

would be inclined to decide that the fruit is an apple if the first value is greater than the second (or vice versa):

$$p(y_1|x) > p(y_2|x)$$

that is:

$$p(x|y_1)p(y_1) > p(x|y_2)p(y_2)$$

In general, for n classes, the Bayesian estimator can be defined as a *discriminant function*:

$$f(x) = \hat{y}(x) = \arg\max_i p(y_i|x) = \arg\max_i p(x|y_i)\pi_i \tag{4.9}$$

It is also possible to calculate an index, given the prior knowledge of the problem, indicating how much this reasoning will be subject to errors. The probability of making an error given an observed *feature* x will depend on the maximum value of the n curves of the distribution 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 The Bayesian Classifier

Through the Bayesian approach, it would be possible to construct an *optimal* classifier if both the prior probabilities $p(y_i)$ and the class-conditional densities $p(x|y_i)$ were known perfectly. Typically, such information is rarely available, and the adopted approach is to build a classifier from a set of examples (the *training set*).

To model $p(x|y_i)$, a *parametric approach* is typically employed, and whenever possible, this distribution is aligned with that of a Gaussian or with spline functions.

The most commonly used estimation techniques are *Maximum-Likelihood (ML)* and *Bayesian Estimation*, which, although differing in their underlying logic, yield nearly identical results. The Gaussian distribution is typically an appropriate model for most *pattern recognition* problems.

Let us examine the quite common case in which the probabilities of the various classes follow a multivariate Gaussian distribution with mean $\boldsymbol{\mu}_i$ and covariance matrix $\boldsymbol{\Sigma}_i$. The optimal Bayesian classifier is given by

$$\begin{aligned} \hat{y}(\mathbf{x}) = &\ \arg\max_i p(\mathbf{x}|y_i)\pi_i \\ = &\ \arg\max_i \log\left(p(\mathbf{x}|y_i)\pi_i\right) \\ = &\ \arg\min_i \left((\mathbf{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) - \log\det\boldsymbol{\Sigma}_i - 2\log\pi_i\right) \end{aligned} \tag{4.11}$$

It seems that you've provided a snippet of LaTeX code that includes the end of an array and an equation environment. If you need assistance with translating specific content or text that accompanies this code, please provide that text, and I'll be happy to help! using the *negative log-likelihood* (section 2.8). In the case of equal prior probabilities π_i, equation (4.11) corresponds to the problem of finding the minimum of the Mahalanobis distance (section 2.4) between the classes of the problem.

4.2.3 Naive Bayes

Typically, using a single feature extracted from the object to be classified does not yield a high classification accuracy. Fortunately, the features that can be extracted from an image are numerous.

Let the characteristics be indicated by x_j and $j = 1, \ldots, m$. It is very important to note that the observed events x_j used to construct the Bayesian classifier must be independent events (conditional independence); otherwise, Bayes' theorem is no longer valid (one of the limitations of Bayesian classifiers): for example, classifiers analyzing overlapping parts of the image cannot be combined, nor can the estimator "is orange" be combined with "is not red."

The *Naive Bayes* (or *idiot Bayes*) assumption leverages the simplifying hypothesis of independence among the observed attributes (*features*): in this case, given m observed variables $x_1 \ldots x_m$, the probability that the event y_i occurs will be:

$$p(x_1 \ldots x_m|y_i) = \prod_{j=1}^{m} p(x_j|y_i) \tag{4.12}$$

4.3 LDA

An example of dimensionality reduction for classification purposes is Linear Discriminant Analysis (Fisher, 1936).

When analyzing the functioning of PCA (section 2.10.1), this technique is limited to maximizing information without distinguishing between the potential classes that make up the problem: PCA does not take into account that the data may represent different categories. PCA is not a true classifier; rather, it is a technique useful for simplifying the problem by reducing its dimensionality. In contrast, LDA aims to maximize both the discriminatory information between classes and the information represented by variance.

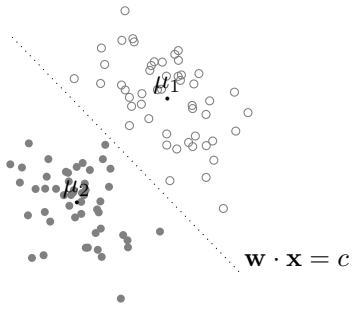


Figure 4.2: Linear Discriminant Analysis.

In the case of a two-class problem, the optimal Bayesian classifier is the one that identifies the decision boundary formed by the hypersurface along which the conditional probability of the two classes is equal.

If we impose the hypothesis that the two classes of the binary problem have a multivariate Gaussian distribution and an equal covariance matrix, it is straightforward to demonstrate that the Bayesian decision margin, as shown in equation (4.11), becomes linear.

In LDA, the assumption of homoscedasticity is made, and under this assumption, the goal is to obtain a vector $\mathbf{w}$ that allows projecting the n-dimensional space of events into a scalar space, which maximizes the separation between classes and enables them to be linearly separated through a separation margin of the form

$$\mathbf{w}^\top \mathbf{x} = c \tag{4.13}$$

.

To determine this separating surface, various metrics can be employed. The term LDA currently encompasses several techniques, among which Fisher's Linear Discriminant Analysis is the most widely referenced in the literature.

It can be demonstrated that the projection that maximizes the separation between the two classes from a "statistical" perspective, namely the decision hyperplane, is obtained with

$$\mathbf{w} = \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{4.14}$$

and the optimal separation value is located halfway between the projections of the two means

$$c = \mathbf{w}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)/2 \tag{4.15}$$

in the case where the prior probabilities of the two sets are identical.

This decision margin is the solution to the maximum likelihood in the case of two classes with uniform distribution and the same covariance.

4.4 SVM

LDA aims to maximize the statistical distance between classes but does not assess the actual margin of separation between them.

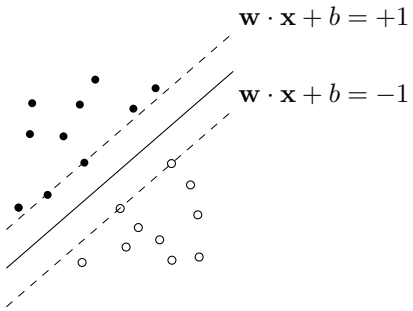


Figure 4.3: Separation hyperplane between two classes obtained through SVM. The points on the margin (dashed line) are the *Support Vectors*.

SVM [CV95], like LDA, allows for the creation of a linear classifier based on a discriminant function in the same form as shown in equation (4.5). However, SVM goes further: the optimal hyperplane in $\mathbb{R}^n$ is generated in such a way as to "physically" separate (the *decision boundary*) the elements of the binary classification problem, aiming to maximize the separation margin between the classes. This reasoning greatly benefits the generalization of the classifier.

Let us therefore define the classification classes typical of a binary problem in the form $y_i = \{+1, -1\}$ and refer to the hyperplane given by formula (4.4).

Suppose there exist optimal parameters $(\mathbf{w}_0, b_0)$ that satisfy the constraint

$$\begin{array}{ll} \mathbf{x}_i \cdot \mathbf{w}_0 + b_0 \geq +1 & \text{per } y_i = +1 \\ \mathbf{x}_i \cdot \mathbf{w}_0 + b_0 \leq -1 & \text{per } y_i = -1 \end{array} \tag{4.16}$$

or, in a more compact form:

$$y_i(\mathbf{x}_i \cdot \mathbf{w}_0 + b_0) - 1 \geq 0 \tag{4.17}$$

for every $(y_i, \mathbf{x}_i)$ samples provided during the training phase.

It can be assumed that for each of the categories, there exists one or more vectors $\mathbf{x}_i$ where the inequalities (4.17) become equalities. These elements, referred to as *Support Vectors*, are the most extreme points of the distribution, and their distance represents the measure of the separation margin between the two categories.

The distance ρ point-plane (see eq.(1.52)) is given by

$$\rho = \frac{\|\mathbf{w} \cdot \mathbf{x} + b\|}{\|\mathbf{w}\|} \tag{4.18}$$

Given two points of opposite classes that satisfy the equality (4.17), the margin can be derived from the equation (4.18), and it is equal to

$$\rho = \frac{2}{\|\mathbf{w}_0\|} \tag{4.19}$$

To maximize the margin ρ of equation (4.19), one must minimize its inverse, that is,

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \tag{4.20}$$

under the series of constraints expressed by the inequality (4.17). This is what is referred to as the standard *primal* optimization problem of the SVM.

This class of problems (minimization with constraints such as inequalities or *primal optimization problem*) is solved using the Karush-Kuhn-Tucker approach, which is the method of Lagrange multipliers generalized to inequalities. Through the KKT conditions, the Lagrangian function is obtained:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i \left(y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1\right) \tag{4.21}$$

to be minimized in $\mathbf{w}$ and b and maximized in $\boldsymbol{\alpha}$. The weights $\alpha_i \geq 0$ are the Lagrange multipliers.

From the nullification of the partial derivatives, we obtain:

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum y_i \alpha_i = 0 \tag{4.22}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \rightarrow \mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \tag{4.23}$$

By substituting these results (the primal variables) into the Lagrangian (4.21), it becomes a function of the multipliers only, the *dual*, from which the Wolfe dual form is derived:

$$\Psi(\boldsymbol{\alpha}) = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{4.24}$$

under the constraints $\alpha_i \geq 0$ and $\sum \alpha_i y_i = 0$. The maximum of the function Ψ evaluated over $\boldsymbol{\alpha}$ corresponds to the α_i associated with each training vector $\mathbf{x}_i$. This maximum allows us to find the solution to the original problem.

In this relationship, the KKT conditions are valid, among which the constraint known as *Complementary slackness* is of considerable importance.

$$\alpha_i \left(y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1\right) = 0 \tag{4.25}$$

This constraint states that the maximum of the Lagrangian is either on the boundary of the constraint ($\alpha_i \neq 0$) or is a local minimum ($\alpha_i = 0$). As a consequence, only the α_i on the boundary are non-zero and contribute to the solution: all other training samples are, in fact, irrelevant. These vectors, associated with the $\alpha_i > 0$, are the *Support Vectors*.

By solving the quadratic problem (4.24), under the constraint (4.22) and $\alpha_i \geq 0$, the weights that exhibit $\alpha_i \neq 0$ will be the *Support Vectors*. These weights, when substituted into equations (4.23) and (4.25), will lead to the derivation of the maximum margin hyperplane.

The most commonly used method to solve this QP problem is the *Sequential Minimal Optimization* (*SMO*). For a comprehensive discussion of the topics related to SVM, one can refer to [SS02].

4.4.1 Soft Margin SVM

In real-world applications, a margin does not always exist, meaning that classes are not always linearly separable in the feature space through a hyperplane. The concept underlying the Soft Margin allows us to overcome this limitation by introducing an additional variable ξ for each sample, thereby relaxing the constraint on the margin

$$\begin{aligned} &y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \\ &\xi_i \geq 0, \forall i \end{aligned} \tag{4.26}$$

. The parameter ξ represents the slackness associated with the sample. When $0 < \xi \leq 1$, the sample is correctly classified but lies within the margin area. When $\xi > 1$, the sample enters the decision space of the opposing class and is therefore classified incorrectly.

To search for a more optimal separating hyperplane, the cost function to minimize must also take into account the distance between the sample and the margin:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum \xi_i \tag{4.27}$$

subject to the constraints (4.26). The parameter C represents a degree of freedom in the problem, indicating how much a sample must "pay" for violating the margin constraint. When C is small, the margin is wide, whereas when C approaches infinity, it reverts to the *Hard Margin* formulation of SVM discussed earlier.

Each sample $\mathbf{x}_i$ can fall into one of three possible states:

- it may lie beyond the margin $y_i(\mathbf{w}^\top \mathbf{x}_i + b) > 1$ and therefore not contribute to the function;
- it may lie on the margin $y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1$ not participating directly in the minimization but only as a *support vector*;
- finally, it may fall within the margin and be penalized according to how much it deviates from the hard constraints.

The Lagrangian of the system (4.27), with the constraints introduced by the variables ξ, is

$$\mathcal{L}(\mathbf{w}, b, \xi, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i) - \sum_i \gamma_i \xi_i \tag{4.28}$$

With the increase in the number of constraints, the dual variables are both $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$.

The remarkable result is that, upon applying the derivatives, the dual formulation of (4.28) becomes exactly the same as the dual of the *Hard Margin* case: the variables ξ_i do not appear in the dual formulation, and the only difference between the *Hard Margin* case and the *Soft Margin* case lies in the constraint on the parameters α_i, which in this case are limited to

$$0 \leq \alpha_i \leq C \tag{4.29}$$

instead of being subject to the simple inequality $\alpha_i \geq 0$. The significant advantage of this formulation is precisely in the high simplicity of the constraints and in the fact that it allows us to reduce the *Hard Margin* case to a particular case ($C = \infty$) of the *Soft Margin*. The constant C serves as an upper limit on the values that the α_i can assume.

4.4.2 SVM and Kernel Functions

Despite the Soft Margin, some problems are intrinsically non-separable in the feature space. However, from the understanding of the problem, it is possible to infer that a nonlinear transformation $\phi : X \to F$ transforms the input feature space $\mathcal{X}$ into the feature space $\mathcal{F}$ where the separating hyperplane allows for better discrimination of the categories. The discriminant function in the space $\mathcal{F}$ is given by

$$f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b \tag{4.30}$$

To enable separation, the space $\mathcal{F}$ is typically of higher dimensions than the space $\mathcal{X}$. This increase in dimensions leads to a rise in the computational complexity of the problem and an increased demand for resources. Kernel methods address this issue.

The vector $\mathbf{w}$ is a linear combination of the training samples (the *support vectors* in the case of *hard margin*):

$$\mathbf{w} = \sum_i \alpha_i \phi(\mathbf{x}_i) \tag{4.31}$$

The discriminant function thus takes the form

$$\begin{aligned} f(\mathbf{x}) &= \textstyle\sum_i \alpha_i \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}) + b \\ &= \textstyle\sum_i \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b \end{aligned} \tag{4.32}$$

with the evaluation of the kernel function $k(\mathbf{x}, \mathbf{x}')$.

At the time of evaluating the discriminant function, it is therefore necessary to utilize the support vectors (at least those with a non-negligible associated parameter α_i). In fact, SVM with a kernel identifies certain samples from the training set as useful information to determine how close the evaluation sample in question is to them.

The *bias* is calculated instantaneously from equation (4.32), averaging

$$b = \mathrm{E}[y_j - \sum_i \alpha_i k(\mathbf{x}_j, \mathbf{x}_i)] \tag{4.33}$$

The most commonly used kernels, due to their ease of evaluation, are Gaussian kernels in the form

$$k(\mathbf{x}, \mathbf{x}') = e^{-\gamma \|\mathbf{x}-\mathbf{x}'\|^2} \tag{4.34}$$

with γ as the parameter to be set, and polynomial kernels of degree d in the form

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^d \tag{4.35}$$

where in the case of $d = 1$, the formulation reduces to the linear case.

The use of kernel functions, combined with the ability to precalculate all combinations $k(\mathbf{x}_i, \mathbf{x}_j)$, allows for the establishment of a common interface between linear and nonlinear training, effectively maintaining the same level of performance.

It is noteworthy that the prediction $f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$ takes the form of

$$\phi(\mathbf{x}) = [k_1(\mathbf{x}, \mathbf{x}_1), \ldots, k_n(\mathbf{x}, \mathbf{x}_n)] \tag{4.36}$$

where $\mathbf{x}_i$ represents a subset of the training data. The models written in this form effectively perform a *template matching* between the sample $\mathbf{x}$ to be evaluated and the prototypes $\mathbf{x}_i$.

4.4.3 Empirical Risk Minimization

The *Soft Margin* constraint (4.26) can be rewritten as

$$y_i f(\mathbf{x}_i) \geq 1 - \xi_i \tag{4.37}$$

where $f(\mathbf{x}_i)$ can also be a generic kernel function. This inequality is equivalent to

$$\xi_i \geq \max(0, 1 - y_i f(\mathbf{x}_i)) \tag{4.38}$$

Since $\xi_i \geq 0$. The loss function (4.38) is referred to as the hinge loss function (*Hinge Loss*)

$$\ell(y, \hat{y}) = \max(0, 1 - y\hat{y}) \tag{4.39}$$

and has the advantage of being convex and non-differentiable only at 1. The *hinge loss* is always greater than the 0/1 loss function.

The training problem of SVM in the case of non-linearly separable data is equivalent to an unconstrained optimization problem on $\mathbf{w}$ of the form

$$\min_{\mathbf{w} \in \mathbb{R}^d} \|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \ell(y_i, f(\mathbf{x}_i)) \tag{4.40}$$

The objective function continues to be described in two clearly distinct parts: the first is Tikhonov regularization, and the second is the empirical risk minimization with the hinge loss function. SVM can therefore be viewed as a linear classifier that optimizes the hinge loss function with L2 regularization.

The input data $\mathbf{x}_i$ can fall into 3 different categories:

- $y_i f(\mathbf{x}_i) > 1$ are the points outside the margin and do not contribute to the cost function;
- $y_i f(\mathbf{x}_i) = 1$ are the points on the margin and do not contribute to the cost as in the case of a "hard margin";
- $y_i f(\mathbf{x}_i) < 1$ are the points that violate the constraint and contribute to the cost.

4.5 Multiclass Classification

SVM returns an objective function $f(\mathbf{x}_i, \mathbf{W}, b) = \mathbf{W}\mathbf{x}_i + b$ whose absolute value lacks a true meaning as it is an uncalibrated output. The extension to the multiclass case is challenging because the different objective functions for each class are not directly comparable to one another.

The concept of *hinge loss* can, however, be extended to the multiclass case. In this scenario, a *SVM Loss* of the form

$$\ell_i = \sum_{j \neq y_i} \begin{cases} 0, & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1, & \text{otherwise} \end{cases} = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \tag{4.41}$$

is defined, where $s_j = f_j(\mathbf{x}_i)$ denotes, for simplicity, the objective function associated with class j for the i-th sample.

Another similar metric is the *squared hinge loss*:

$$\ell_i = \sum_{j \neq y_i} \max\left(0, s_j - s_{y_i} + 1\right)^2 \tag{4.42}$$

Finally, a loss function is defined over the entire dataset as the average

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \ell_i + \lambda R((W)) \tag{4.43}$$

with the optional regularization term on the weights.

A different metric, extended to the multiclass case, is the normalized exponential function known as *Softmax*:

$$\ell_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} = -s_{y_i} + \log \sum_j e^{s_j} \tag{4.44}$$

The objective function s_j can be interpreted as an unnormalized logarithmic probability for each class, and therefore, the cardinal loss function can be replaced with the cross-entropy loss function. A *Softmax* classifier minimizes the cross-entropy among the classes, and since it minimizes the negative log likelihood of the correct class, it can be viewed as a maximum likelihood estimator. In the case of *Softmax*, the regularization term $R((W))$ can be seen, from a statistical perspective, as a *prior* on the weights: in this case, it represents a Maximum a posteriori (*MAP*) estimate.

4.6 Ensemble Learning

The concept of *Ensemble* training refers to the use of various distinct classifiers, combined in a certain manner to maximize performance by leveraging the strengths of each while mitigating the weaknesses of the individual classifiers.

At the core of the concept of *Ensemble Learning* are weak classifiers: a weak classifier is capable of classifying at least the $50\% + 1$ of the samples in a binary problem. When combined in a certain way, weak classifiers allow for the construction of a strong classifier, simultaneously addressing typical issues associated with traditional classifiers, with *overfitting* being the primary concern.

The origin of *Ensemble Learning*, the concept of a weak classifier, and in particular the notion of *probably approximately correct learning* (*PAC*) were first introduced by Valiant [Val84].

In fact, *Ensemble Learning* techniques do not provide *general purpose* classifiers; rather, they indicate the optimal way to combine multiple classifiers together.

Examples of *Ensemble Learning* techniques include

Decision Tree Decision Trees, being constructed from multiple *Decision Stumps* in a cascade, serve as a primary example of *Ensemble Learning*;

Bagging *BootStrap AGGregatING* aims to mitigate *overfitting* issues by training various classifiers on subsets of the *training set* and ultimately performing a majority vote;

Boosting Instead of using purely random subsets of the *training set*, samples that remain incorrectly classified are partially utilized;

AdaBoost *ADAptive BOOSTing* (section 4.6.2) is the most well-known *Ensemble Learning* algorithm and the progenitor of the highly successful family of *AnyBoost* classifiers;

Random Forest™ is a *BootStrap Aggregating* (*bagging*) method applied to *Decision Trees*, forming an *Ensemble Classifier* composed of multiple decision trees, each created from a subset of the training data and features to be analyzed, which vote for the majority;

and many others.

Examples of weak classifiers widely used in the literature include *Decision Stumps* [AL92] associated with Haar *features* (section 6.1). The *Decision Stump* is a binary classifier in the form

$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } pf(\mathbf{x}) > p\theta \\ -1 & \text{otherwise} \end{cases} \tag{4.45}$$

where $f(\mathbf{x})$ is a function that extracts a scalar from the sample to be classified, $p = \{+1, -1\}$ is a parity that indicates the direction of the inequality, and θ is the decision threshold (figure 4.4).

4.6.1 Decision Trees

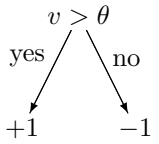


Figure 4.4: Example of a *Decision Stump*. v is a feature extracted from the image and θ is a threshold.

A Decision Tree is a very simple and effective method for creating a classifier, and the training of decision trees is one of the most successful current techniques. A decision tree is a tree of classifiers (*Decision Stump*) where each internal node is associated with a specific "question" regarding a feature. From this node, as many branches emanate as there are possible values that the feature can take, leading to the leaves that indicate the category associated with the decision. Special attention is typically given to binary decision nodes.

A good "question" divides samples of heterogeneous classes into subsets with fairly homogeneous labels, stratifying the data in such a way as to minimize variance within each stratum.

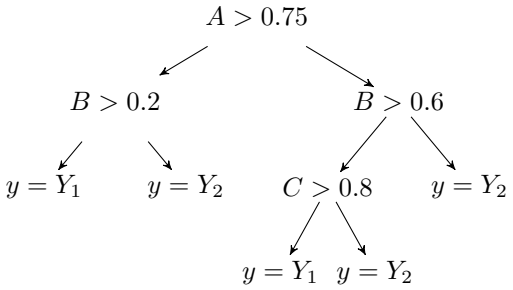


Figure 4.5: Example of a *Decision Tree*.

To enable this, it is necessary to define a metric that measures this impurity. We define X as a subset of samples from a particular training set consisting of m possible classes. X is, in fact, a random variable that takes only discrete values (the continuous case is analogous). It is possible to associate with each discrete value x_i, which can take X, the probability distribution $p(x_i) = p_i$. X is a *data set* composed of m classes, and p_i is the relative frequency of class i within the set X.

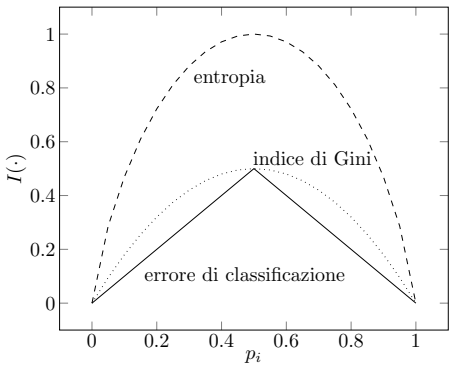


Figure 4.6: Comparison of impurity measurement metrics in the case of a binary classification problem.

Given the definition of X, the following metrics are widely used in decision trees:

Entropy From information theory, the entropy I_H of X is given by:

$$I_H(X) = -\sum_{i=1}^{m} p_i \log_2 p_i \tag{4.46}$$

Gini Index The Gini impurity index is defined as

$$I_G(X) = 1 - \sum_{i=1}^{m} p_i^2 \tag{4.47}$$

Classification Error From Bayesian theory:

$$I_E(X) = 1 - \max_i p_i \tag{4.48}$$

Intuitively, a node with class distribution $(0, 1)$ has minimal impurity, while a node with uniform distribution $(0.5, 0.5)$ has maximal impurity.

A "question" $h_j(x)$, which has k possible answers, divides the set $\mathcal{E}$ into the subsets $\mathcal{E}_1, \ldots, \mathcal{E}_k$.

To evaluate how well the condition is executed, one must compare the impurity level of the child nodes with the impurity of the parent node: the greater their difference, the better the chosen condition.

Given a metric $I(\cdot)$ that measures impurity, the gain Δ is a criterion that can be used to determine the quality of the split:

$$\Delta = I(\mathcal{E}) - \sum_{i=1}^{k} \frac{N(\mathcal{E}_i)}{N(\mathcal{E})} I(\mathcal{E}_i) \tag{4.49}$$

where $N(\mathcal{E})$ is the number of samples in the parent node and $N(\mathcal{E}_i)$ is the number of samples in the i-th child node.

If entropy is used as a metric, the gain Δ is known as *Information Gain* [TSK06].

Decision trees induce algorithms that select a test condition that maximizes the gain Δ. Since $I(\mathcal{E})$ is the same for all possible classifiers and $N(\mathcal{E})$ is constant, maximizing the gain is equivalent to minimizing the weighted sum of the impurities of the child nodes:

$$\hat{h} = \arg\min_{h_j} \sum_{i=1}^{k} N(\mathcal{E}_i) I(\mathcal{E}_i) \tag{4.50}$$

The best question $h_j(x)$ is the one that minimizes this quantity.

In the case of binary classifiers, the Gini metric is widely used, as the gain to be minimized reduces to

$$\frac{p_1 n_1}{p_1 + n_1} + \frac{p_2 n_2}{p_2 + n_2} \tag{4.51}$$

with p_1, n_1 being the number of positive and negative samples that the classifier moves to the left branch and p_2, n_2 being the number of samples in the right branch.

Decision trees adapt both very well and quickly to training data and consequently, if not constrained, they systematically suffer from the problem of *overfitting*. Typically, a pruning algorithm is applied to trees to reduce, where possible, the issue of *overfitting*.

Pruning approaches are generally of two types: *pre-pruning* and *post-pruning*. *Pre-pruning* involves stopping the creation of the tree under certain conditions to avoid excessive specialization (for example, maximum tree size). In contrast, *post-pruning* refines an already created tree by removing branches that do not meet specific conditions on a previously selected *validation set*.

This technique for creating a decision tree is commonly referred to as *Classification and Regression Trees* (CART) [B+84]. Indeed, in real cases where the analyzed features are statistical quantities, we do not refer to creating a classification tree, but more appropriately to constructing a regression tree. Finding the optimal partition of the data is an NP-complete problem; therefore, greedy algorithms, such as the one presented in the section, are typically employed.

4.6.2 ADAptive BOOSTing

One of the *Ensemble* classifiers that has garnered significant interest from researchers in recent years is undoubtedly *AdaBoost* [FS95]. The fundamental idea behind *AdaBoost* is to construct a list of classifiers by iteratively assigning a weight to each new classifier based on its ability to recognize samples that were not correctly identified by the other classifiers already involved in the training process. All these classifiers will cast their votes with the weights assigned to them, and the final decision will be made by majority vote.

The *Boosting* techniques allow for the generation of a classifier in the form of an additive model:

$$F_T(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x}) + \ldots + f_T(\mathbf{x}) = \sum_{t=1}^{T} f_t(\mathbf{x}) \tag{4.52}$$

with $f_1, \ldots, f_T$ individual classifiers.

Let us examine the case of binary classification, and let $S = (\mathbf{x}_1, y_1) \ldots (\mathbf{x}_m, y_m) \in (\mathbb{X} \times \{-1, 1\})$ be the set of m samples available for training.

A common choice in the context of model fitting is to use least squares regression (an optimal metric in the presence of Gaussian noise, for example) to obtain the additive model $F_T(\mathbf{x})$, by minimizing the quantity $\sum (y_i - F_T(\mathbf{x}_i))^2$. However, following numerous experiments, it has been observed that the quadratic cost function is not the optimal choice in classification problems.

The *AdaBoost* approach suggests that the combination of all these classifiers minimizes a different, more favorable cost function, namely the exponential loss function (*exponential loss*):

$$\min_F \sum_{i=1}^{m} e^{-y_i F(\mathbf{x}_i)} \tag{4.53}$$

Since the global minimization of the function (4.53) is usually impossible, there are two possible approaches to proceed:

- optimizing one classifier at a time in a cyclic manner until reaching a stable situation (*generalized backfitting algorithm*);
- adding one new classifier at a time to the additive model (*"greedy" forward stepwise approach*).

AdaBoost addresses the classification problem using the second approach.

Under these considerations, the objective of the training process is reduced to identifying an additional classifier $f(\mathbf{x})$ that minimizes the quantity

$$f_{T+1} = \arg\min_f \sum_{i=1}^{m} e^{-y_i(F_T(\mathbf{x}_i)+f(\mathbf{x}_i))} = \arg\min_f \sum_{i=1}^{m} w_i e^{-y_i f(\mathbf{x}_i)} \tag{4.54}$$

each time, having defined $w_i = e^{-y_i F_T(\mathbf{x}_i)}$ and leveraging the properties of the exponential function.

AdaBoost is a technique that addresses all these requirements.

Let us assume that we have at our disposal $\mathcal{H} = \{h_1, \ldots, h_T\}$ binary classifiers, each of which, by evaluating the feature $\mathbf{x}_i$, with $1 \leq i \leq m$, returns an opinion $y_i = \{-1, +1\}$.

Let the function $F_T(\mathbf{x}; \boldsymbol{\alpha})$, defined as

$$F_T(\mathbf{x}; \alpha_1, \ldots, \alpha_T) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \tag{4.55}$$

be a function whose sign represents the classification hypothesis and whose magnitude reflects the quality of the prediction. The model described by equation (4.55) is referred to as the *Extended Additive Model* or *Adaptive Basis-Function Model*.

The objective is to obtain a strong classifier $H(\mathbf{x}_i)$ as a weighted linear sum of the classifiers h_t, whose sign determines the global hypothesis:

$$H(\mathbf{x}_i) = \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_i)\right) = \operatorname{sgn} F_T(\mathbf{x}_i; \boldsymbol{\alpha}) \tag{4.56}$$

This is a majority voting scheme: the hypothesis that receives the most votes from the classifiers, each with a different weight α_t, is chosen as the winner. It is precisely the constants α_t, the weights assigned to each classifier, that are the result provided by this training technique.

To enable the assignment of a score to the classifier, it is necessary that each input sample x_i is assigned a certain weight w_i: the higher the weight, the more the sample has been misclassified up to this point in the training, while the lower the weight, the more it has been classified correctly. In the first iteration, all weights are set equal to $w_i^{(0)} = 1/m$, ensuring an accurate statistical distribution. Variants such as *Asymmetric AdaBoost* assign different weights to the various categories involved.

Let $u_i = y_i h_t(\mathbf{x}_i)$ be the function that expresses the success $(+1)$ or failure (-1) of the classifier h_t in evaluating the sample x_i. Given the weights associated with each sample, it is possible for each classifier to calculate W_{-1}, the sum of the weights associated with failures, and W_{+1}, the sum of the weights associated with correct classifications, through the definition of u_i, in compact form as follows:

$$W_b = \sum_{u_i = b} w_i \tag{4.57}$$

with $b = +1$, representing success, and $b = -1$, representing failure.

Let ϵ_t be the measure of the classifier's error h_t calculated as

$$\epsilon_t = \sum_{y_i \neq h_t(i)} w_i^{(t)} = \sum_{u_i = -1} w_i^{(t)} = W_- \tag{4.58}$$

, the sum of the weights associated only with the samples classified incorrectly, and let

$$r_t = W_+ - W_- = \sum_{i=1}^{m} w_i^{(t)} u_i \tag{4.59}$$

be the weighted average, using the weights w_i, of the classification performances u_i.

The iterations of the *AdaBoost* algorithm are as follows:

1. An *Oracle* provides a classifier h_t (the choice is essentially left to the user, aiming to select the classifier that minimizes the error ϵ_t, although it is not mandatory for it to be the best);
2. The error ϵ_t produced by the classifier h_t on the input samples is calculated. When it is not possible to find a classifier for which $\epsilon_t > 1/2$, the training cannot proceed and must therefore be terminated;
3. Given the error, the classifier h_t is assigned a weight α_t, calculated as described subsequently;
4. For each sample x_i, the associated distribution $w_i^{(t+1)}$ is updated through the function

$$w_i^{(t+1)} = \frac{1}{Z_t} w_i^{(t)} e^{-\alpha_t u_i} = \frac{1}{Z_t} w_i^{(t)} e^{-y_i f_t(\mathbf{x}_i)} \tag{4.60}$$

.

The weight associated with samples that were successfully classified is decreased by an amount proportional to $e^{-\alpha_t}$, while the weight of samples that were misclassified is increased by e^{α_t}. Z_t is a normalization factor chosen such that $\sum w_i^{(t)} = 1$, but it also holds significant meaning as explained immediately below.

The normalization parameter Z_t is given by

$$Z_t = \sum_{i=1}^{m} w_i^{(t)} e^{-\alpha_t u_i} = e^{-\alpha_t} W_+ + e^{\alpha_t} W_- \tag{4.61}$$

and, an important result of *AdaBoost*, it can be demonstrated that the classification error is upper-bounded by

$$\frac{1}{m} \{i : H(x_i) \neq y_i\} \leq \prod_{t=1}^{T} Z_t \tag{4.62}$$

For this reason, Z_t is precisely the quantity to minimize in order to obtain the optimal classifier. A direct consequence of this result is that *AdaBoost* can be viewed as a framework that minimizes $\prod_t Z_t$.

The optimal choice of α_t (and consequently that of h_t) is the one where the function (4.61) reaches its minimum, specifically at

$$\alpha_t = \frac{1}{2} \log \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) = \frac{1}{2} \log \frac{W_+}{W_-} = \frac{1}{2} \log \left(\frac{1 + r_t}{1 - r_t} \right) \tag{4.63}$$

. With this particular choice of α_t, Z_t also reaches its minimum and is equal to

$$Z_t = 2\sqrt{\epsilon_t (1 - \epsilon_t)} = 2\sqrt{W_- W_+} \tag{4.64}$$

. From equation (4.64), it follows that Z_t is minimized by selecting the classifier h_t that has the lowest value of ϵ_t, which corresponds to the maximum of W_+.

Choosing the weight from equation (4.63) that minimizes Z_t, after each iteration of *AdaBoost*, the weights associated with correctly identified samples are decreased by a factor of $\exp(-\alpha_t)$, that is $\sqrt{W_-/W_+}$, while the weights associated with samples incorrectly classified by the hypothesis h_t are increased by a factor of $\exp(\alpha_t)$, that is $\sqrt{W_+/W_-}$.

This algorithm is what is referred to in the literature as *AdaBoost.M1* or *Discrete AdaBoost* [FHT00]. The hypotheses $h_t(\mathbf{x})$ used by AdaBoost are *features* that can take on only the values $\{+1, -1\}$.

The intuitive functioning of *AdaBoost* is quite straightforward: *AdaBoost* focuses on the input *patterns* that have been misclassified the most as each new classifier is added to the ensemble.

AdaBoost has several interpretations: as a classifier that maximizes the margin, logistic regression for an additive model, as a discrete step gradient descent minimizer, and also as regression using Newton's method.

AdaBoost, like SVM, aims to maximize the separation margin between classes, albeit using different metrics. In this way, both methods are able to be less sensitive to issues such as *overfitting*.

4.6.3 AdaBoost and Its Variants

The problem of *Boosting* can be generalized and viewed as a problem where it is necessary to search for predictors $f_t(\mathbf{x})$ that minimize the global cost function:

$$\sum_{i=1}^{m} \phi \left(y_i \left(f_1(\mathbf{x}_i) + \ldots + f_n(\mathbf{x}_i) \right) \right) \tag{4.65}$$

where $\phi \in \mathcal{C}^1$ is a convex function that is non-increasing with respect to $\lim_{z \to \infty} \phi(z) = 0$.

From an analytical perspective, *AdaBoost* is an example of a coordinate-wise gradient descent optimizer that minimizes the *potential function* $\phi(z) = e^{-z}$, optimizing one coefficient α_t at a time [LS10], as can be seen from equation (4.54).

A non-exhaustive list that sheds light on some peculiarities of this technique, the variants of *AdaBoost* are:

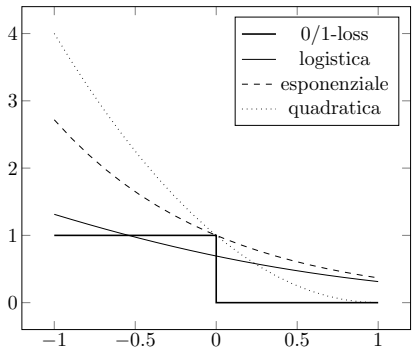


Figure 4.7: Comparison of *loss functions*: 0/1-loss, logistic, exponential, and quadratic

AdaBoost with Abstention

AdaBoost can also be extended to cases involving classifiers with abstention, where the possible outputs are $h_j(x_i) \in \{-1, 0, +1\}$. By expanding the definition (4.57), for simplicity let W_- denote the failures, W_0 the abstentions, and W_+ the successes of the classifier h_t.

In this case, Z_t also reaches the minimum with the same value as α_t in the case without abstention, see (4.63). and with such a choice Z_t would hold true

$$Z_t = W_0 + 2\sqrt{W_- W_+} \tag{4.66}$$

However, there exists a more conservative choice of α_t proposed by Freund and Shapire

$$\alpha_t = \frac{1}{2} \log \left(\frac{W_+ + 1/2W_0}{W_- + 1/2W_0} \right) \tag{4.67}$$

that allows for setting an upper limit on Z_t.

Real AdaBoost

Real AdaBoost generalizes the previous case but, above all, generalizes the same extended additive model [FHT00]. Instead of using dichotomous hypotheses $h_t(x)$ and associating a weight α_t with them, it directly seeks the *feature* $f_t(x)$ that minimizes the equation (4.54).

Real AdaBoost allows the use of weak classifiers that provide the probability distribution $p_t(x) = P[y = 1|x, w^{(t)}] \in [0, 1]$, the probability that class y is indeed $+1$ given the observation of feature x.

Given a probability distribution $p_t(x)$, the *feature* $f_t(x)$, which minimizes equation (4.54), is

$$f_t(x) = \frac{1}{2} \log \frac{P[y = +1|x, w^{(t)}]}{P[y = -1|x, w^{(t)}]} = \frac{1}{2} \log \frac{p_t(x)}{1 - p_t(x)} \tag{4.68}$$

. This result is equal to half of the logistic transformation. Since the objective remains to minimize the exponential cost function, the weight update still follows equation (4.60).

Both *Discrete* and *Real AdaBoost*, by selecting a weak classifier that satisfies equation 4.68, ensure that *AdaBoost* asymptotically converges to

$$\lim_{T \to \infty} F_T(x) = \frac{1}{2} \log \frac{P[y = +1|x]}{P[y = -1|x]} \tag{4.69}$$

demonstrating how the *AdaBoost* algorithm is an iterative procedure that combines multiple weak classifiers to approximate a Bayesian classifier.

Real AdaBoost can also be used with a discrete classifier such as the *Decision Stump*. By directly applying the equation (4.68) to the two possible output states of the *Decision Stump* (it is still straightforward to obtain the minimum of Z_t algebraically), the classifier's responses must take the values

$$f(x) = \begin{cases} \frac{1}{2} \log \frac{W_{TP}}{W_{FP}} & x > \theta \\ \frac{1}{2} \log \frac{W_{FN}}{W_{TN}} & x \leq \theta \end{cases} \tag{4.70}$$

with the values W_*, which represent the sum of the weights associated with False Positives (FP), False Negatives (FN), True Positives (TP), and True Negatives (TN). With this choice of values, Z_t takes on a significant value of

$$Z_t = 2 \left(\sqrt{W_{TP} W_{FP}} + \sqrt{W_{FN} W_{TN}} \right) \tag{4.71}$$

, which serves as a metric to use for selecting the best feature x and threshold θ.

Gentle AdaBoost

The weights associated with the *outliers* in *Real AdaBoost* can be very high due to the presence of the logarithm in the equation. In this case, it becomes necessary to make the regression more "gentle."

Gentle AdaBoost further generalizes the concept of Ensemble Learning to additive models [FHT00] by employing regression with steps typical of Newton's methods: It seems you have provided a placeholder for a mathematical block, but there is no content to translate. Please provide the text or equations you would like me to translate, and I'll be happy to assist!

The hypothesis $f_t(x)$, to be added to the additive model at iteration t, is selected from all possible hypotheses f_k as the one that optimizes a weighted least squares regression

$$f_t = \arg\min_{f_k} \sum_i w_i (y_i - f_k(x_i))^2 \tag{4.72}$$

but for each iteration, the weight update from *AdaBoost* (4.60) is used, specifically the exponential loss function.

Also, *Gentle AdaBoost* can be used with the *Decision Stump*. In this case, the minimum of (4.72) of the decision algorithm takes on a remarkable form in

$$f(x) = \begin{cases} \frac{W_{TP} - W_{FP}}{W_{TP} + W_{FP}} & x > \theta \\ \frac{W_{FN} - W_{TN}}{W_{TN} + W_{FN}} & x \leq \theta \end{cases} \tag{4.73}$$

LogitBoost

For historical reasons, *AdaBoost* does not explicitly exhibit a statistical formalism. The first observation is that the output of the *AdaBoost* classifier is not a probability, as it is not constrained between $[0, 1]$. In addition to this issue, which is partially addressed by *Real AdaBoost*, minimizing the *loss-function* (4.53) does not appear to be a statistical approach, unlike maximizing the likelihood. However, it is possible to demonstrate that the cost function of *AdaBoost* maximizes a function very similar to the *Bernoulli log-likelihood*.

For these reasons, it is possible to extend *AdaBoost* to the theory of logistic regression, as described in section 3.7.

The additive logistic regression takes the form

$$\log \frac{P[y = +1|x]}{P[y = -1|x]} = F_T(x) = \sum_{t=1}^{T} f_t(x) \tag{4.74}$$

an interesting expression when compared to that of AdaBoost in equation (4.69). By inverting equation (4.74), we obtain the logistic relationship

$$p(x) = P[y = +1|x] = \frac{e^{F_T(x)}}{1 + e^{F_T(x)}} = \frac{1}{1 + e^{-F_T(x)}} \tag{4.75}$$

which associates a probability estimate with the additive model $F(x)$.

The problem then becomes one of finding an appropriate *loss function* for this representation, specifically identifying a variant of *AdaBoost* that precisely maximizes the *Bernoulli log-likelihood* [FHT00].

Maximizing the likelihood of (4.75) is equivalent to minimizing the *log-loss*

$$\sum_{t=1}^{T} \log\left(1 + \exp(-y_i F_T(x_i))\right) \tag{4.76}$$

LogitBoost first extends *AdaBoost* to the problem of logistic optimization of a function $F_T(x)$ under the cost function $\phi(z) = \log(1 + e^{-z})$, maximizing the *Bernoulli log-likelihood* using Newton-type iterations.

The weights associated with each sample are derived directly from the probability distribution

$$\begin{aligned} z_i &= \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))} \\ w_i &= \frac{p(x_i)}{1 - p(x_i)} \end{aligned} \tag{4.77}$$

with $y^* = \{0, 1\}$ and choosing the hypothesis $f_t(x)$ as the least squares regression of z_i on x_i using the weights w_i. The future estimate of $p(x_i)$ is directly derived from equation (4.75).

Asymmetric-AdaBoost

Asymmetric-AdaBoost introduces a variant in the weight update rule [VJ01]. The issue with AdaBoost is that it does not allow for direct control over the weight assigned to classification errors across different classes and does not enable explicit minimization of the number of false positives, focusing only on the classification error. The *Asymmetric-AdaBoost* variants, on the other hand, modify at each iteration t the weights associated with positive and negative samples by a cost factor $c_+^{(t)}$ and $c_-^{(t)}$, respectively.

Cascade

Regardless of the use of *Cascade* classifiers [VJ02], the weights are modified by a factor $\beta_t = \epsilon_t/(1-\epsilon_t) = W_-/W_+$ only in the case of correct classification; otherwise, the weights remain unchanged. The weight associated with a classifier is assigned as $\alpha_t = -\log \beta_t$, which is double the weight assigned by *AdaBoost.M1*.

MAdaBoost

The *MAdaBoost* algorithm features a distinct weight update mechanism aimed at reducing the contribution of *outliers* (or overly complex examples) during training. The maximum weight $w_i^{(t)}$ that a sample can assume is capped at the upper limit of $w_i^{(0)}$, which is the weight assigned at the beginning of the algorithm.

This behavior can be represented by a cost function of the form

$$\phi(z) = \begin{cases} 1-z & z \leq 0 \\ e^{-z} & z > 0 \end{cases} \tag{4.78}$$

4.7 Neural Networks

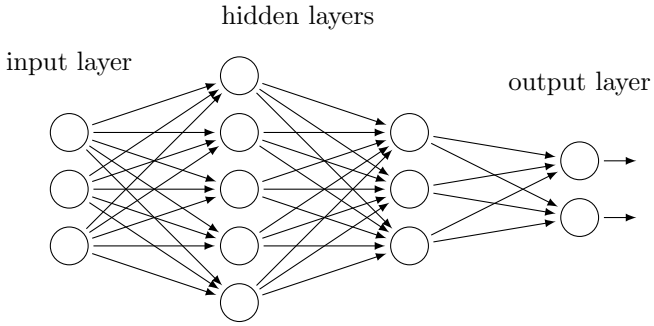


Figure 4.8: Example of a neural network topology.

Research in Machine Learning (and, more broadly, in Computer Vision) has consistently sought inspiration from the human brain for the development of algorithms. Artificial Neural Networks (ANNs) are based on the concept of the "artificial neuron," which is a structure that, similar to the neurons of living beings, applies a nonlinear transformation (known as the activation function) to the weighted contributions of the various inputs to the neuron:

$$\text{output} = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{4.79}$$

where x_i are the various inputs related to the j-th neuron, associated with the weights w_i, output is the response of the neuron, and the activation function f, which is highly nonlinear, is typically a step function, a sigmoid, or a logistic function. The bias b is sometimes simulated with a constant input x_0.

The simplest neural network, consisting of an input layer and an output layer, is analogous to the perceptron model introduced by Rosenblatt in 1957. Similar to the brains of living beings, an artificial neural network consists of the interconnection of various artificial neurons.

The geometry of a *feedforward* neural network, the topology commonly used in practical applications, is that of the *MultiLayer Perceptron* (MLP) and consists of the combination of multiple hidden layers of neurons that connect the input stage to the output stage, which will serve as the input for the subsequent layer. A multilayer perceptron can be likened to a function

The training phase involves estimating the weights w_i^k that minimize the error between the training labels and the values predicted by the network $f_w(\mathbf{x})$:

$$S(\mathbf{w}) = \sum_i \|\mathbf{y}_i - f_{\mathbf{w}}(\mathbf{x}_i)\|^2 \tag{4.80}$$

The estimation of the weights w_i^k can be achieved using well-known optimization techniques: typically, the *back propagation* method is utilized, which is essentially a gradient descent approach employing the *chain rule* for derivative calculations, given that MLPs are layered structures.

4.7.1 Activation Functions

The activation function of the artificial neuron transforms the input values into outputs. Common activation functions include

- step (Heaviside) $h(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$
- sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$
- softsign $s(x) = \frac{1}{1+|x|}$
- hyperbolic tangent $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$
- ramp $r(x) = \max(0, x)$

Originally, only sigmoid activation functions (sigmoid and hyperbolic tangent) were commonly used; however, recently, the class of ramp functions has also been employed, which, in contrast to sigmoid functions, are unbounded. Artificial Neurons with the ramp activation function are defined as *Rectified Linear Units* (ReLUs) [KSH12a].

4.8 Deep Learning

Neural networks, and in particular their training through *backpropagation*, present several practical issues:

- they require labeled training data, while in contexts based on large amounts of data (*big data*) most of the data may not be categorized;
- the training time scales poorly both with the increase in network size and with the increase in the amount of data;
- the optimization can get stuck in local minima, making the network sub-optimal;
- the supervised training of deep models (networks with many hidden layers) is a particularly complex optimization problem.

To address these challenges, a branch of *machine learning* has been developed, known as *deep learning*, which leverages deep architectures and advanced techniques to improve learning effectiveness.

Humans tackle complex problems by breaking them down into sub-problems and multiple levels of abstract representation. Similarly, *deep learning* allows a system to learn hierarchical representations of data, directly mapping complex functions between input and output without relying on manually engineered features.

This approach makes it possible to generate high-level abstractions, often not expressible by humans, but more manageable by the computer.

With the growing availability of data and applications of *machine learning*, automatic learning techniques are evolving rapidly. The goal of *deep learning* is to build high-level representations of data through the use of multiple layers of nonlinear operations, as in *Deep Neural Networks* (DNN). ands, techniques that enable automatic learning are continually growing. The goal of *deep learning* is to create high-level representations of data through the use of multiple layers of nonlinear operations (*DNN Deep Neural Network*).

4.8.1 Representation learning

Feature Learning or *Representation Learning* serves as a bridge between traditional *machine learning* techniques and the underlying theory of *Deep Learning*. It involves leveraging unsupervised learning methods to reduce the dimensionality of the problem while preserving as much information as possible. Subsequently, these extracted data (where the input space is transformed into a lower-dimensional feature space) can be utilized for supervised classification, employing classical *machine learning* techniques.

The performance of a *machine learning* algorithm is highly dependent on the choice of input data representation (*features*). The greatest efforts to achieve a better *machine learning* algorithm are focused on designing a sequence of transformations of the input data into a format that allows the classification algorithm to maximize its performance.

Various unsupervised learning techniques, some already discussed previously and others that will be covered in the upcoming sections, include

- K-means segmentation
- Hopfield network
- Sparse Coding

- Principal Component Analysis (PCA in section 2.10.1)
- Restricted Boltzmann Machines (RBM in section 4.8.2)
- AutoEncoders (in section 4.8.3)

These techniques allow for the reduction of the problem's dimensionality while striving to retain maximum information, specifically preserving those aspects of the training data that most effectively describe the samples.

4.8.2 Restricted Boltzmann Machines

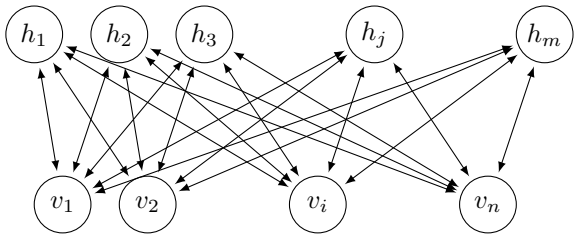


Figure 4.9: Restricted Boltzmann Machines.

The breaking point between shallow and deep training techniques is considered to be 2006, when Hinton and others at the University of Toronto introduced *Deep Belief Networks* (DBNs) [HOT06], an algorithm that "greedily" trains a layered structure by training one layer at a time using an unsupervised training algorithm. The distinctive feature of DBNs lies in the fact that the layers are composed of *Restricted Boltzmann Machines* (RBMs) [Smo86, FH94].

Let $\mathbf{v} \in \{0,1\}^n$ be a binary stochastic variable associated with the visible state and $\mathbf{h} \in \{0,1\}^m$ a binary stochastic variable associated with the hidden state. Given a state $(\mathbf{v}, \mathbf{h})$, the energy of the configuration of the visible and hidden layers is given by [Hop82]

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{n} a_i v_i - \sum_{j=1}^{m} b_j h_j - \sum_{i=1}^{n}\sum_{j=1}^{m} w_{i,j} v_i h_j \tag{4.81}$$

where v_i and h_j are the binary states of the visible layer and the hidden layer, respectively, while a_i, b_j are the weights, and $w_{i,j}$ are the weights associated between them. A *Boltzmann Machine* is similar to a Hopfield network, with the distinction that all outputs are stochastic. Therefore, a *Boltzmann Machine* can be defined as a special case of the Ising model, which in turn is a particular case of a *Markov Random Field*. Similarly, RBMs can be interpreted as stochastic neural networks where the nodes and connections correspond to neurons and synapses, respectively.

The probability of the joint configuration $(\mathbf{a}, \mathbf{b}, \mathbf{W})$ is given by the Boltzmann distribution:

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z(\cdot)} e^{-E(\mathbf{v},\mathbf{h})} \tag{4.82}$$

where the partition function Z is defined as

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{4.83}$$

the sum of the energies of all possible pairs of visible and hidden states.

The term *restricted* refers to the fact that direct interactions between units belonging to the same layer are not permitted, but only interactions between adjacent layers are allowed.

Given an input $\mathbf{v}$, the hidden binary state h_j is activated with probability:

$$p(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_i v_i w_{i,j}\right) \tag{4.84}$$

where $\sigma(x)$ is the logistic function $1/(1+exp(-x))$. Similarly, it is straightforward to obtain the visible state from the hidden state:

$$p(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_j h_i w_{i,j}\right) \tag{4.85}$$

Obtaining the model $(\mathbf{a}, \mathbf{b}, \mathbf{W})$ that allows for the representation of all training input values is, however, a very complex task. A much faster procedure was proposed by Hinton in 2002: it is only since then that RBMs can be trained using the *contrastive divergence* (CD) algorithm [Hin12].

4.8.3 Auto-Encoders

Auto-Encoders are a specific type of unsupervised neural networks that aim to approximate the RBMs trained with the *Contrastive Divergence* algorithm. An Auto-Encoder (and in this context, RBMs serve as a perfect representation) allows for encoding the input $\mathbf{x}$ into a representation $\mathbf{c}(\mathbf{x})$ such that the input can still be reconstructed in some manner, minimizing the *negative log-likelihood*

$$-\log P\left(\mathbf{x}|\mathbf{c}(\mathbf{x})\right) \tag{4.86}$$

. It is noteworthy that in the case where the distribution is Gaussian, one would recover the classical form of least squares regression (see section 2.8).

When the inputs $\mathbf{x}_i$ are binary (or the distribution is of a binomial type), the cost function becomes

$$-\log(P\left(\mathbf{x}|\mathbf{c}(\mathbf{x})\right) = \sum_i \mathbf{x}_i \log \mathbf{f}_i\left(\mathbf{c}(\mathbf{x})\right) + (1-\mathbf{x}_i)\log\left(1-\mathbf{f}_i(\mathbf{c}(\mathbf{x}))\right) \tag{4.87}$$

where $\mathbf{f}(\cdot)$ is the *decoder* associated with the *encoder* $\mathbf{c}(\cdot)$.

The function $\mathbf{c}(\mathbf{x})$ is a lossy compression function. This function is effectively a good compression method only for the data observed during the unsupervised training phase, but it will not perform well for all data in general that were not involved in the learning phase.

4.8.4 Deep Neural Networks

In traditional neural networks, the optimization of weights starts from randomly chosen initial values. Although simple to implement, this choice implies that as the network depth increases, performance tends to degrade, while shallower architectures (with one or two hidden layers) are generally more stable and easier to train.

Historically, the training of multilayer neural networks (MLP) through gradient descent faced two main obstacles:

- the presence of numerous local minima that hinder convergence;
- the appearance of large plateaus in the error surface, which drastically slow down optimization.

As a consequence, the idea of using very deep networks to model complex problems was long considered impractical.

Starting in 2012, thanks to the availability of large amounts of data (*Big Data*), the increasing computational power provided by GPUs, and the development of more effective optimization techniques (such as *Adam*) 3.3.3, deep neural networks experienced a true renaissance.

A key turning point was the victory of *AlexNet* [KSH12b] at the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC): for the first time, a deep convolutional neural network clearly outperformed traditional approaches, marking the beginning of the modern era of *deep learning*.

Since then, deep networks have become the de facto standard in many areas of *machine learning*. In particular, the processing of structured data such as images has greatly benefited from *convolutional neural networks* (CNN), which exploit the spatial structure of the visual signal to more efficiently learn hierarchical and invariant representations.

4.8.5 Convolutional Neural Networks

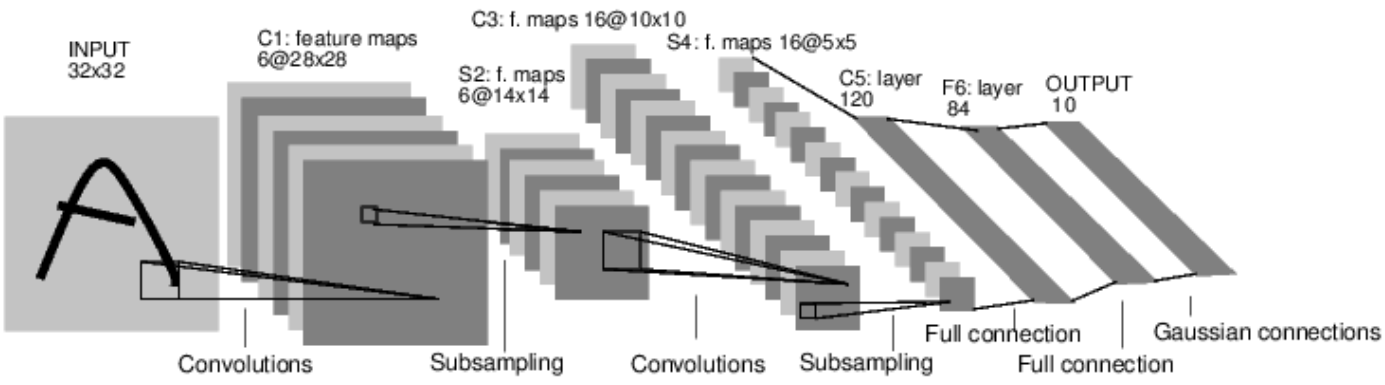


Figure 4.10: Example of the architecture of a Convolutional Neural Network, LeNet5.

Convolutional Neural Networks (CNNs) represent a natural evolution of deep neural networks for the processing of spatially structured data, such as images. In contrast to classical *MultiLayer Perceptrons* (MLPs), CNNs exploit local pixel correlations and spatial redundancy through convolutional layers in which weights are shared. This architectural

principle enables the automatic learning of hierarchical and translation-invariant features, significantly reducing the number of parameters to be optimized and thereby improving training efficiency. The following paragraphs describe the main components of a CNN, including convolutional layers, activation functions, pooling, and the typical architectures used in deep training.

CNNs are multi-layer neural networks similar to *MultiLayer Perceptrons*, but with a distinctive structure: at least one of the layers consists of sets of neurons with shared weights, referred to as *convolutional layers*.

In a convolutional layer, the activation of a neuron depends on the dot product between a kernel (or filter) and a local region of the input:

$$a_{i,j} = \sum_k \sum_l \sum_m w_{k,l,m}\, x_{i+k,j+l,m} + b = \mathbf{w}^\top \mathbf{x}_{i,j} + b \tag{4.88}$$

where $\mathbf{w}$ denotes the filter weights, $\mathbf{x}_{i,j}$ the local portion of the image centered at (i, j), and b a possible bias term.

Convolution can be applied with a *stride* greater than 1, thereby producing a downsampled activation map relative to the input. Since filters progressively reduce the spatial resolution of activation maps, it is common to introduce *padding* in order to preserve the output dimensions.

Activation maps are then transformed by a nonlinear activation function, typically a ReLU. Subsequently, a *pooling* layer is often employed to reduce dimensionality and introduce local invariance: *max pooling* is the most widely used variant, whereas *average pooling* and *L2-norm pooling* were more common in earlier architectures.

CNNs are specifically designed to leverage multi-channel two-dimensional inputs. Typically, a CNN accepts as input a third- or fourth-order tensor; for example, an image of size $w \times h$ with three channels (R,G,B) corresponds to a third-order tensor. Each convolutional layer may contain k distinct kernels, producing output activation maps of size $w \times h \times k$.

The architecture of a deep neural network (DNN) may include:

- convolutional layers (**C**);
- pooling layers (**S**);
- normalization layers;
- fully connected layers (MLP);
- loss functions.

At the final stage (the *loss layer*), a traditional classification method (MLP, AdaBoost, or SVM) is applied to the reduced representation of the input, preserving most of the useful information.

The training of DNNs is typically performed through variants of stochastic gradient descent (see section 3.3.3), iteratively optimizing the weights of the filters and fully connected layers on the basis of the classification error.

4.8.6 Recurrent Neural Networks and Transformers (for sequences and images)

Convolutional Neural Networks (*CNNs*) represented a breakthrough in the processing of static images, due to their ability to capture hierarchies of local features through convolutional filters and pooling. However, traditional CNNs are not well suited for handling sequential data such as text, time signals, or video, where order and temporal dependencies among elements are crucial. To address these challenges, *Recurrent Neural Networks* (*RNNs*) were introduced, in which neurons are equipped with recurrent connections that enable the maintenance of a memory of past states. RNNs have been applied to tasks such as speech recognition, machine translation, time-series analysis, and automatic image captioning. Nevertheless, standard RNNs struggle to learn long-term dependencies due to the vanishing and exploding gradient problems. To mitigate these limitations, more sophisticated architectures such as *Long Short-Term Memory* (LSTM) and *Gated Recurrent Units* (GRU) were developed, capable of regulating which information to retain or discard through *gating* mechanisms.

Despite these advances, the first generation of deep sequential networks relied on an *encoder?decoder* paradigm, in which the information from the input sequence was compressed into a lower-dimensional tensor, expected to preserve as much relevant information as possible for the downstream task. This approach, however, suffers from an intrinsic limitation: important input elements may be overlooked or attenuated during compression.

A fundamental shift came with the introduction of the *attention mechanism*. The underlying idea of attention is simple yet powerful: instead of compressing all information into a single vector, the model can dynamically ?focus? on the most relevant parts of the input sequence. Attention weights are computed according to the relevance of each element with respect to the others, overcoming the limitations of compressed recurrent representations.

Let $X \in \mathbb{R}^{n \times d}$ be the input sequence matrix, where n is the number of *tokens* (the individual units into which the algorithm divides the input sequence, varying from sequence to sequence) and d is the embedding dimension (a numerical vector representing each token, fixed for the model and encoding semantic and syntactic information).

The *query* (Q), *key* (K), and *value* (V) matrices are obtained through linear projections:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V \tag{4.89}$$

where:

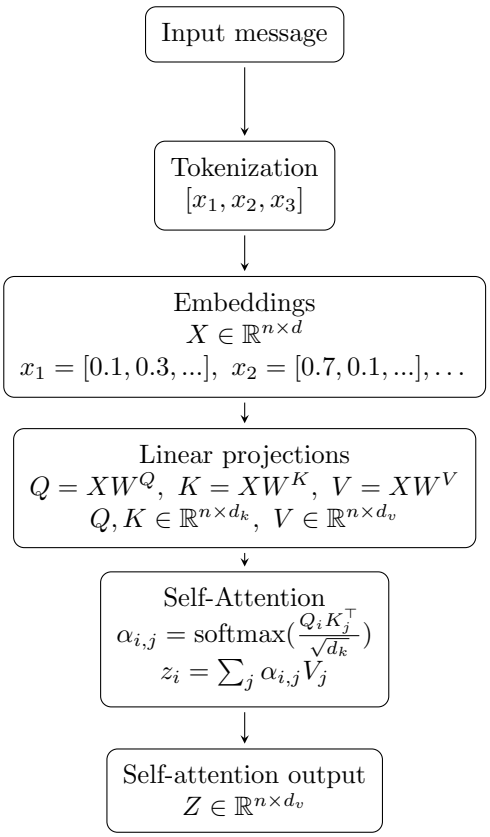


Figure 4.11: Conceptual diagram of the *self-attention* mechanism in a Transformer.

- $W^Q, W^K \in \mathbb{R}^{d \times d_k}$ and $W^V \in \mathbb{R}^{d \times d_v}$ are weight matrices learned during training;
- $Q, K \in \mathbb{R}^{n \times d_k}$ and $V \in \mathbb{R}^{n \times d_v}$.

The *self-attention* mechanism can then be expressed in scalar form (for a single token) as:

$$\alpha_{i,j} = \text{softmax}\left(\frac{Q_i K_j^\top}{\sqrt{d_k}}\right), \quad z_i = \sum_{j=1}^{n} \alpha_{i,j} V_j \tag{4.90}$$

where:

- $Q_i, K_j \in \mathbb{R}^{d_k}$ are, respectively, the *query* of token i and the *key* of token j;
- $V_j \in \mathbb{R}^{d_v}$ is the *value* of token j;
- $\alpha_{i,j}$ is the scalar weight indicating the degree to which token i attends to token j;
- $z_i \in \mathbb{R}^{d_v}$ is the new representation of token i resulting from the weighted combination of values.
- The *softmax* function, which normalizes the weights $\alpha_{i,j}$ between 0 and 1, is defined as:

$$\text{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}, \quad i = 1, \ldots, n \tag{4.91}$$

  where $\mathbf{x}$ is the input vector (in our case, $\mathbf{x} = \frac{Q_i K^\top}{\sqrt{d_k}}$ for row i).

Intuitively, attention can be seen as a dynamic generalization of weighting methods such as *Bag of Words* or *TF-IDF*. Unlike TF-IDF, which assigns static weights to terms, attention assigns context- and task-dependent weights, enabling the model to focus selectively on the most relevant parts. Semantically, the resulting matrix Z has the same length as the input (n tokens in, n tokens out), but each token is enriched with contextual information from the entire sequence.

The attention mechanism directly led to the development of *Transformers* [VSP+17], which are now the *de facto* standard for sequence modeling in natural language processing, computer vision, and multimodal learning. In Transformers, the central operator is *self-attention*, which models relationships among all sequence elements in parallel and directly. Compared

to RNNs, Transformers provide substantial advantages in terms of parallelization, numerical stability, and the ability to capture long-range dependencies.

In computer vision, the application of Transformers gave rise to the *Vision Transformers* (ViT) [DBK+20], where an image is partitioned into small patches treated as a sequence, analogous to words in a text. These models have demonstrated performance that is competitive with or superior to CNNs on various classification, recognition, and segmentation tasks, particularly in the presence of large-scale datasets.

In practical applications, a single *self-attention* module is insufficient to extract adequate information from the input tokens. The *multi-head self-attention* mechanism extends the idea of self-attention by allowing the model to examine the sequence from multiple perspectives simultaneously. Specifically:

- h distinct heads are employed. For each head $p = 1, \ldots, h$, separate projection matrices are defined:

$$Q^{(p)} = XW^{Q,(p)}, \quad K^{(p)} = XW^{K,(p)}, \quad V^{(p)} = XW^{V,(p)}, \tag{4.92}$$

  where each matrix $W^{Q,(p)}, W^{K,(p)}, W^{V,(p)} \in \mathbb{R}^{d \times d_k}$.

- Each head computes self-attention (scaling, softmax, combination) within its subspace. In compact matrix form, where Attention denotes the *scaled dot-product attention* operator:

$$Z^{(p)} = \text{Attention}\left(Q^{(p)}, K^{(p)}, V^{(p)}\right) = \text{softmax}\left(\frac{Q^{(p)}(K^{(p)})^\top}{\sqrt{d_k}}\right) V^{(p)} \tag{4.93}$$

- The outputs of the different heads are concatenated:

$$Z_{\text{concat}} = \left[Z^{(1)}\,;\, Z^{(2)}\,;\, \ldots\,;\, Z^{(h)}\right] \in \mathbb{R}^{n \times (h \cdot d_v)} \tag{4.94}$$

  and then projected back into a d-dimensional space (the model dimension) through an output matrix $W^O \in \mathbb{R}^{(h \cdot d_v) \times d}$:

$$Z = Z_{\text{concat}}\, W^O \tag{4.95}$$

- Often one chooses $d_v = d_k$ and $d = h \cdot d_k$, so that concatenation followed by projection yields a dimension consistent with the input to the next layer.

- This enables the modeling of different ?types? of attention (e.g., syntactic relations, semantic dependencies, local vs. global interactions) within the same layer.

Table 4.1: Examples of geometry (*layers*, *embedding*, *heads*) for well-known Transformer models.

Model	**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)	many decoders	12288	96	128

Today, RNNs and Transformers are best seen as complementary tools: the former remain useful in scenarios with relatively short sequences or limited resources, while the latter constitute the foundation of modern state-of-the-art *deep learning* architectures. The evolution from recurrent to attention-based mechanisms has marked a paradigm shift: from the notion of compressed memory to dynamic, context-dependent representations, where the model autonomously decides ?what to attend to? for each element of the sequence.

4.9 Generalizing the Training

It is important to note that *Machine Learning* techniques encompass much more than mere optimization. One of the objectives set during training is to ensure that the system is capable of classifying new samples that it has not yet encountered. One approach to combat *overfitting* is "regularization." There are various techniques in the literature for regularization: the main ones are L1/L2 regularization and early stopping (*early-stopping*).

4.9.1 L1 and L2 Regularization

L1 and L2 regularization involves adding an additional term to the cost function that penalizes certain configurations. Regularizing, for example, the cost function

$$S(\boldsymbol{\beta}, \mathbf{X}) = -\sum_i \log P(Y = y_i|\mathbf{x}_i; \boldsymbol{\beta}) \tag{4.96}$$

means adding a term, which is a function solely of $\boldsymbol{\beta}$, in order to obtain the new cost function of the form

$$E(\boldsymbol{\beta}, \mathbf{X}) = S(\boldsymbol{\beta}, \mathbf{X}) + \lambda R(\boldsymbol{\beta}) \tag{4.97}$$

with $R(\boldsymbol{\beta})$ being a regularizing function.

A widely used regularization function is

$$R(\boldsymbol{\beta}) = \left(\sum_j |\beta_j|^p \right)^{1/p} \tag{4.98}$$

Common values for p are 1 or 2 (hence it is referred to as L1 or L2 regularization). When $p = 2$ can also be defined in the literature as *weight decay*. This type of regularization function penalizes parameters with excessively high values.

4.9.2 Early Exit

Early stopping directly combats *overfitting* by monitoring the model's performance on an additional set of examples known as the validation set. When the objective function on the validation set stops decreasing for a certain number of iterations and begins to worsen, training is halted due to the suspicion that the model is starting to overfit the problem.

4.10 Performance Evaluation

Given a classifier trained on a specific training set (*Training Set*), it is necessary to evaluate it on another set (*Validation Set* or *Certification Set*). From this comparison, it is possible to extract metrics that allow for the assessment of the classifier and enable the comparison of different classifiers with one another. It is absolutely essential that the performance metrics are calculated on a set of samples not used during the training phase (the *validation set*) in order to detect issues such as data *overfitting* or lack of generalization.

Once the parameters of the classifier are set, the contingency table (*Confusion Matrix*) can be created:

It seems that you've provided a LaTeX command that indicates the end of an HTML-only section. If you have specific text or content that you would like translated from Italian to English, please provide that text, and I will be happy to assist you with the translation while maintaining the LaTeX commands and structure.

		True Value	
		p	n
Classification	p'	VP	FP
	n'	FN	VN

False Positives (FP) are also referred to as False Alarms. False Negatives (FN) are known as *misses*.

From the table, several performance metrics are typically extracted, such as:

- *Accuracy* is the ratio of the Number of Correct Predictions to the Total Number of Predictions = (VP+VN)/(VP+VN+FN+F
- *Error Rate* is the Total Number of Incorrect Predictions divided by the Total Number of Predictions = (FP+FN) / (VP+VN+FN+FP);
- *Precision* (or *specificity* or *PPV*) is the probability that a positive returned by the classifier is correct = VP / (VP+FP);
- *Recall* (or *hit-rate* or *TPR*) is the percentage of positives correctly identified = VP / (VP+FN);
- *Miss-Rate* or FNR is the complement of Recall = 1-Recall = FN/(VP+FN).

Each classifier has one or more parameters that, when modified, change the ratio between correct recognitions and the number of false positives. Consequently, it becomes challenging to objectively compare two classifiers, as one may exhibit a higher number of correct detections at the same threshold, but also a higher number of false positives. Therefore, to compare the performance of different binary classifiers obtained from various training sessions, it is common to use curves that vary with this internal threshold of the classifier.

The performance curves that can be found are:

- The ROC curve (*Receiver Operating Characteristic*) is a Cartesian graph where the abscissa indicates the number of false positives (percentage *FPR*, per frame or absolute) and the ordinate displays the percentage of correct recognitions (*True Positive Rate TPR*), generated by the classifier as the threshold varies. Every classifier must have a ROC curve that outperforms the random classifier, represented by the line connecting points $(0, 0)$ and $(1, 1)$ on the ROC graph.

- *Precision-Recall* (*PRC*) focuses the analysis primarily on the positives. The area under the PRC is referred to as *Average Precision* (*AP*). In real-world detection problems, the number of True Negatives is extremely high, making it essential to concentrate on false positives. The PRC has the significant advantage of obscuring the scale of False Positives: typically, these are indicated per frame, per minute, or in another unit of measurement.

- *Detection Error Tradeoff* (*DET*) allows for the representation of negative errors (misses) and positive errors (false alarms) on the axes. It is a curve that focuses the analysis purely on errors.

It is important to note that these indices pertain to any class of problems that involves the concept of correct or incorrect results. Therefore, they are applicable not only to classifiers but also, for example, to associations of characteristic points and more.

Recently, to enable a more streamlined comparison of classifier performance, functions have been proposed that, when applied to ROC curves, yield a single scalar representing a score of classification quality. These functions are typically averages of samples taken from the ROC curve in the regions of practical interest.

Chapter 5

Characteristic Points

The identification (extraction) of keypoints (*keypoint detection*), their characterization (*feature description*), and finally comparison (*matching*) are closely related topics within the field of computer vision. Applications that utilize keypoints range from panoramic image creation to three-dimensional reconstruction, from visual odometry to object tracking, and in many other use cases.

The concept of a keypoint emphasizes that not all points in an image have a high probability of being identified unambiguously during a comparison; rather, only certain points possess this property. These are notable, stable, and easily identifiable points.

In the last decade, as in nearly all fields of Computer Vision, significant advancements have been made in the development of *local invariant features*, characteristic points that enable applications to define a local geometry of the image and encode it in such a way that it remains invariant to image transformations, such as translation, rotation, scaling, and affine deformations.

In this chapter, we will address topics that are closely related to keypoint extraction algorithms. The discussion regarding point description will be covered in the following chapter, as it is an orthogonal topic to both the description of points and the concept of classification.

A non-exhaustive list of algorithms for keypoint detection includes:

Harris Corner Harris mathematically formalizes the concept of an edge and, through the study of the eigenvalues of the covariance matrix in the vicinity of a point, allows for the determination of the presence or absence of a corner. It is invariant to changes in brightness, geometric transformations such as translations and rotations, and minimally to scale variations (see section 5.2);

KLT The Kanade-Lucas-Tomasi method utilizes a variant of Harris (Shi-Tomasi) as a *corner detector* and performs matching using pyramid representations of the scene (details in 7.2);

AST The class of *Advance Segment Test* (see section 5.5) identifies a characteristic point by observing the difference in brightness of points on a circumference;

SIFT Analyzes the image in a multi-resolution manner and is invariant to similarity transformations (see section 5.3);

SURF A more efficient variant of SIFT based on the integral image (see section 5.4).

5.1 Hessian Identifier

The problem of identifying notable points that can be easily recognized between two images has initially been addressed by shifting the focus towards detecting corner points in the image, thereby discarding those portions of the image that lack texture or contain only edges.

The Hessian operator (*Hessian detector*) [Bea78], based on the Hessian matrix derived from the Taylor series expansion around the point to be described, seeks those regions of the image that exhibit strong derivatives in orthogonal directions. This algorithm is based on the analysis of the matrix of second derivatives, known as the Hessian.

$$\mathbf{H}(\mathbf{x}, \sigma) = \begin{bmatrix} I_{xx}(\mathbf{x}, \sigma) & I_{xy}(\mathbf{x}, \sigma) \\ I_{xy}(\mathbf{x}, \sigma) & I_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{5.1}$$

The algorithm computes the second derivatives of the image I_{xx}, I_{xy}, I_{yy} for each point in the image and identifies the points where the determinant of the Hessian

$$\det\left(\mathbf{H}(\mathbf{x}, \sigma)\right) = I_{xx}(\mathbf{x}, \sigma) I_{yy}(\mathbf{x}, \sigma) - I_{xy}^2(\mathbf{x}, \sigma) \tag{5.2}$$

reaches its maximum. This search is typically performed on the image of the Hessian determinant, to which a Non-Maxima Suppression is applied over a window 3×3. The maxima of the Hessian determinant response are usually located at corners and in areas of the image with strong texture. The use of the Hessian determinant makes this algorithm invariant to rotation.

In practical applications, the original image is never used; instead, a low-pass filtered version is obtained through a Gaussian filter.

5.2 Förstner-Harris

The Förstner-Harris algorithm [FG87, HS88] has been explicitly designed to achieve high geometric stability. It defines characteristic points as those points that exhibit a local maximum when compared to the least squares of their translated version. This algorithm has been highly successful because it enables the identification of variations in image intensity in the vicinity of a point by utilizing the autocorrelation matrix of the first derivatives of the image.

Let the gradient images be defined (these can be generated by a differential operator, such as Sobel, Prewitt, or Roberts) as $I_x(x, y)$ and $I_y(x, y)$, representing the horizontal gradient and the vertical gradient of the image to be analyzed, respectively.

From these two images, it is possible to calculate a function $\mathbf{C}(x, y)$, representing the covariance matrix (autocorrelation) of the gradient images in the vicinity of (x, y), defined as It seems that you have included a placeholder for a mathematical block, but there is no content provided for translation. Please provide the specific text or equations that you would like to have translated from Italian to English, and I will be happy to assist you. with $\boldsymbol{\delta} \in \Omega$ around (x, y) and $w(\boldsymbol{\delta})$ an optional kernel, typically a Gaussian centered at (x, y), to allow for differential weighting of the points in the vicinity, or a constant window on Ω. Originally, $w(\boldsymbol{\delta})$ were very small filters, but as computational power has increased, larger Gaussian kernels have been adopted.

In fact, Harris employs two convolution filters: a derivative filter to compute the derived images and an integral filter to calculate the elements of the matrix. The dimensions of these filters and the use of a Gaussian filter to weight the points refer to the discussion in the following section regarding the scale of feature detection.

The matrix $\mathbf{C}$ is the second-order moment matrix. To identify characteristic points, one can analyze the eigenvalues λ_0 and λ_1 of the matrix $\mathbf{C}$ (for a more in-depth discussion, refer to section 2.10.1). The eigenvalues of the autocorrelation matrix $\mathbf{C}$ allow for the characterization of the type of image contained within the window around the given point.

If there are two very high eigenvalues, the point is a *corner*; if there is only one high eigenvalue, it is an *edge*; otherwise, it is a reasonably flat area, represented as a function like

$$C = \min(\lambda_0, \lambda_1) \tag{5.3}$$

where the local maxima correspond to the *corners* identified by the Shi-Tomasi algorithm [ST94].

For a matrix 2×2, the eigenvalues are obtained as solutions of the quadratic characteristic polynomial

$$p(x) = x^2 - \text{trace}(\mathbf{C})x + \det(\mathbf{C}) \tag{5.4}$$

.

Harris, to avoid explicitly calculating the eigenvalues of $\mathbf{C}$, introduces an operator $H(x, y)$ defined as

$$H(x, y) = \det(\mathbf{C}) - \alpha \,\text{trace}(\mathbf{C})^2 \tag{5.5}$$

where α is a parameter that lies between 0 and 0.25 and is typically set to 0.04.

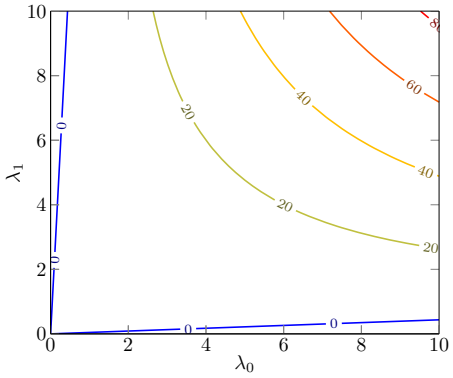


Figure 5.1: Response in the eigenvalue plane provided by the Harris equation at different threshold values, with $\alpha = 0.04$ set. The area of interest is very similar to that provided by the Shi-Tomasi method but without the need to explicitly compute the eigenvalues.

According to Harris, the point (x, y) is a characteristic point (*corner*) if $H(x, y) > H_{thr}$, with H_{thr} being a threshold to be defined. The parameter α regulates the sensitivity of the feature detector. Qualitatively, increasing α removes edges, while increasing H_{thr} removes flat areas (figure 5.1).

5.3 Scale and Rotation Invariance

Harris is a feature point detector that is not invariant to scale variations. To overcome this series of limitations, Lindeberg [Lin94, Lin14] introduces the concept of automatic scale selection, allowing for the identification of characteristic points at a specific level of resolution. The pyramid representation of the scene, a computationally efficient algorithm widely used previously, effectively becomes a special case of this scale-space representation.

Let $G(x, y; t)$ be the two-dimensional Gaussian with variance $t > 0$, described by the equation

$$G(x, y; t) = \frac{1}{2\pi t} e^{-\frac{x^2+y^2}{2t}} \tag{5.6}$$

(see section 2.2).

The convolution $L(x, y; t)$ between the image $I(x, y)$ and the Gaussian $G(x, y; t)$

$$L(x, y; t) = G(x, y; t) * I(x, y) \tag{5.7}$$

generates the scale-space representation of the image itself. The variance $t = \sigma^2$ of the Gaussian kernel is referred to as the scale parameter. The representation of the image at the degenerate scale $t = 0$ is the original image itself.

It is noteworthy that applying a Gaussian filter to an image does not create new structures: all the information generated by the filter was already contained in the original image.

Figure 5.2: Scale-space representation of an image 512×512: from the original image $t = 0$ to scales 1, 4, 16, 64, and 256.

The scale factor t is a continuous number; however, for computational reasons, discrete steps of this value are used, typically exponential sequences, such as $t = 2^i$ or $t = \frac{1}{2}e^i$.

Applying a scale-space image operator, using the commutative property between convolution and differentiation, is equivalent to convolving the original image with the derivative of the Gaussian:

$$L_{x^\alpha}(\cdot; t) = \partial_{x^\alpha} L(\cdot; t) = (\partial_{x^\alpha} g(\cdot; t)) * f(\cdot) \tag{5.8}$$

with α multi-index notation for the derivative. Similarly, it is possible to extend the definition of all edge or feature point filters to any scale factor. Through the work of Lindeberg, it has been possible to extend the concept of Harris *Corners* to scale-invariant cases (Harris-Laplace and Hessian-Laplace methods [MS02]).

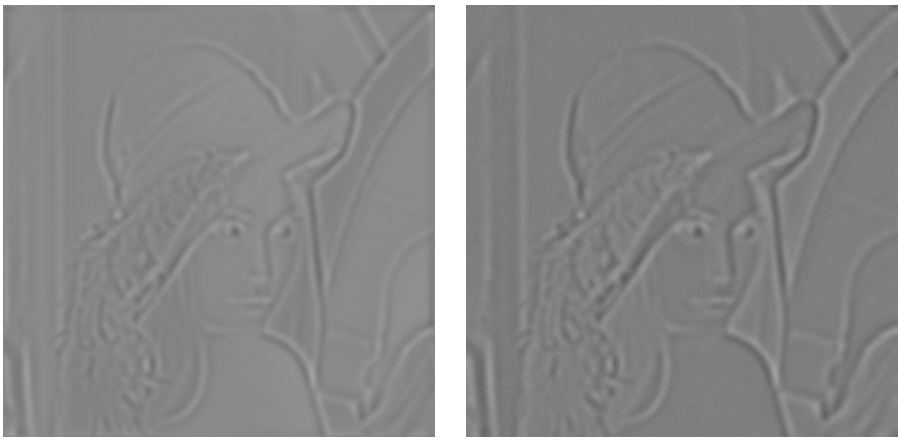
Some interesting operators for identifying characteristic points include the gradient magnitude $|\nabla L|$, the Laplacian $\nabla^2 L$, and the determinant of the Hessian $\det \mathcal{H}(L)$. All of these operators are invariant to rotations, meaning that the location of the minimum/maximum point exists independently of the rotation of the image.

Among these operators, a widely used one for identifying characteristic points is the normalized Laplacian of Gaussian (LoG) operator:

$$\nabla^2_n L(x,y,t) = t(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})G = -\frac{1}{t\pi}\left(1 - \frac{x^2+y^2}{2t}\right) e^{-\frac{x^2+y^2}{2t}} \tag{5.9}$$

Through the LoG operator, it is possible to identify characteristic points such as local maxima or minima in spatial coordinates and scale.

For example, a circle with radius r has the maximum response to the Laplacian at the scale factor $\sigma = r/\sqrt{2}$.



(a) (b)

Figure 5.3: Comparison between the normalized LoG image *(a)* and DoG *(b)*

Lowe [Low04], in the *Scale-invariant feature transform* (*SIFT*) algorithm, enhances performance by approximating the Laplacian of the Gaussian (LoG) with a Difference of Gaussians (DoG):

$$\begin{aligned} D(x,y;\sigma) &= (G(x,y;k\sigma) - G(x,y;\sigma)) * I(x,y) \\ &\approx L(x,y;k\sigma) - L(x,y;\sigma) \\ &\approx (k-1)\sigma^2 LoG(x,y;\sigma) \end{aligned} \tag{5.10}$$

This procedure is more efficient because the Gaussian image at scale $k\sigma$ can be computed from the Gaussian image σ by applying a smaller filter $(k-1)\sigma$, and therefore is overall much faster compared to performing the convolution $k\sigma$ with the original image.

If in LoG the characteristic points were the local minima/maxima, both in space and scale, of the Laplacian image, in this case, the characteristic points are the minimum and maximum points in the difference image between the scale images $\sigma, k\sigma, \ldots, k^n\sigma$ through which the image is processed (figure 5.4).

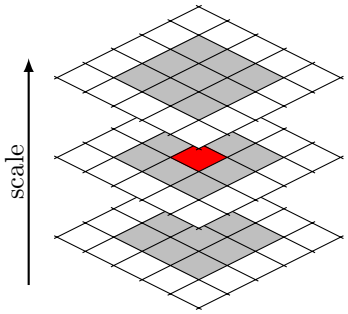


Figure 5.4: Identification of local minima and maxima: for each pixel and for each scale, a neighborhood $3\times3\times3$ is compared.

With the introduction of step k, the domain of the variable σ is effectively divided into discrete logarithmic steps, grouped into octaves, and each octave is subdivided into S sub-levels. In this way, σ takes on the discrete values

$$\sigma(o,s) = \sigma_0 2^{o+\frac{s}{S}} \leftrightarrow k = 2^{\frac{1}{S}} \tag{5.11}$$

with σ_0 as the base scaling factor.

The characteristic points, identified as maxima/minima in both discrete scale and space, are interpolated using a regression on a three-dimensional quadratic to determine the characteristic point with subpixel and subscale precision.

Between one octave and the next, the image is downsampled by a factor of 2: in addition to the multi-scale analysis within each octave, the image is processed again in the subsequent octave by halving both the horizontal and vertical dimensions, and this procedure is repeated multiple times.

The second phase of a feature detection and matching algorithm involves extracting a descriptor to perform comparisons, with the descriptor centered on the identified feature point. In fact, to be scale-invariant, the descriptor must be extracted at the same scale factor associated with the feature point.

To be invariant to rotation, the descriptor must be extracted from an image that has undergone some form of normalization with respect to the dominant direction identified in the vicinity of the evaluated point.

From this rotated image at the scale of the characteristic point, it is possible to extract a descriptor that emphasizes the edges in the surrounding area, ultimately making it invariant to brightness.

Among the numerous variants, PCA-SIFT should be noted, which utilizes PCA to reduce the dimensionality of the problem to a descriptor consisting of only 36 elements. PCA is employed in a prior training phase.

5.4 SURF

The *Speeded Up Robust Features* algorithm [BETVG08] is inspired by the SIFT algorithm and the scale-space representation theory, proposing an optimized version that utilizes approximate Hessians by employing the integral image, both for detecting keypoints and for extracting their descriptors.

SURF is invariant to translation, scale, and rotation, but there exists a simplified variant, referred to as "U-SURF," which is only invariant to variations in translation and scale. In this case, the area around the identified point is not normalized with respect to rotation when the descriptor is extracted.

In SURF, the characteristic points are detected by calculating local maxima on the determinant of the Hessian image defined as:

$$\mathcal{H}(x,y;t) = \begin{bmatrix} \frac{\partial}{\partial x^2}G(t) * I & \frac{\partial}{\partial xy}G(t) * I \\ \frac{\partial}{\partial yx}G(t) * I & \frac{\partial}{\partial y^2}G(t) * I \end{bmatrix} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{5.12}$$

an image formed by the convolutions of the second-order derivatives of the Gaussian with variance $t = \sigma^2$ and the image at point (x, y). For performance reasons, the derivatives of the Gaussians are quantized to integer values and approximated using rectangular regions (box filters), meaning that certain rectangular areas around the point are positively weighted, while others are negatively weighted, and their sum forms the element of the matrix $\mathcal{H}$.

The bandwidth of these approximate filters can be estimated as

$$\sigma = \frac{1.2}{9} l \tag{5.13}$$

with l being the size of the filter. The filter 9×9, the smallest possible, for instance, approximates the derivatives of the Gaussian with variance $\sigma = 1.2$.

The determinant image is calculated as

$$\det(\mathcal{H}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{5.14}$$

where w is a factor that accounts for quantization, attempting to compensate for various rounding errors, and is typically set to $w = 0.912$ constant. Finally, the determinant is normalized with respect to the scale size involved, allowing for comparisons across different scales.

The image is analyzed across multiple octaves (each octave has a scale factor that is double that of the previous octave). Each octave is divided into an equal number of scale levels. The number of scales per octave is constrained by the inherently quantized nature of the filter, and the approximated Gaussians are not as evenly spaced as in the case of SIFT. In fact, 4 intervals per octave is the only feasible number of subdivisions.

Within each octave, as the scale s and position vary, a *Non-Maxima Suppression* $3 \times 3 \times 3$ is performed on the determinant image of $\mathcal{H}$. The local minima/maxima, interpolated through a three-dimensional quadratic as in SIFT, are the interest points identified by SURF. The scale is set equal to the variance of the associated filter $s = \sigma$.

From the identified maxima, using the integral image, the dominant orientation is extracted in the vicinity of the point (within a radius of $6s$ and sampled at a step of s). In this case, Haar features of size $4s$ are also utilized and weighted with a Gaussian distribution of $\sigma = 2s$.

Through the orientation information, a descriptor is generated based on the directions of the gradients by sampling the area around $20s$, divided into 4×4 regions and weighting the points with a Gaussian $\sigma = 3.3s$. Within each region, d_x, d_y, $|d_x|$, and $|d_y|$ are calculated. Both the orientation and the gradient histogram are extracted at the detection scale of the *feature*.

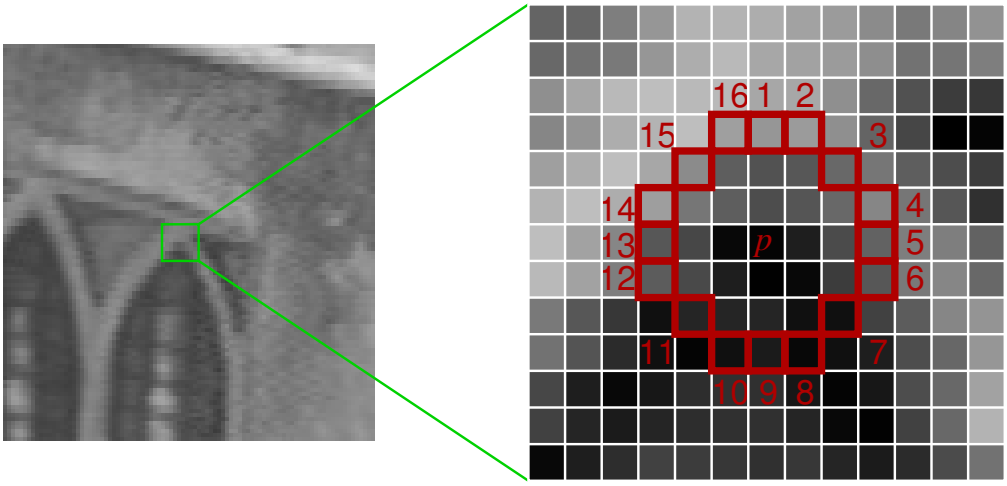


Figure 5.5: FAST: the 16 pixels on the circumference of radius 3 on which to perform the continuity test.

5.5 AST

The latest class of feature point extractors falls under the name of *Accelerated Segment Test* developed by Rosten. Currently, there are three slightly different versions of this algorithm.

The first version of *Features from Accelerated Segment Test FAST* [RD05] is probably the most intuitive: in this case, points are considered characteristic if they have a continuous sequence of n pixels along a circumference of a given radius, all being either (more or less) luminous than the central pixel used as a reference for the gray tone. In the case of FAST-9, for example, the 16 pixels on the circumference of radius 3 are analyzed to check if there are 9 contiguous pixels that are all above or all below a certain threshold relative to the central pixel. In subsequent versions [RD06], the extraction is optimized through the use of decision trees trained to identify feature points that maximize the local amount of information. These trees consistently process the pixels on the circumference.

This approach has become typical in recent years, as the abundance of public datasets has led to widespread use of classifiers to construct stable feature point detectors. In fact, given primitives that describe the neighborhood of a point, the application of an optimization technique allows for the identification of those that exhibit greater stability for the specific task. The article by Rosten, among other things, provides an excellent survey of previous feature point extraction techniques.

In the latest variant (FAST-ER), the area to be analyzed is extended not only to the points on a circumference but to all the pixels in the vicinity of the central point.

Chapter 6

Descriptors

Another concept that has a cross-cutting relevance among the themes of computer vision is that of *descriptor* (*Visual Descriptor*). The descriptor is indeed utilized in various contexts: it is employed to perform comparisons between characteristic points or to generate the disparity map in stereoscopic vision, to provide a compact representation of a portion of the image to expedite its identification or retrieval, and due to this compact solution that preserves a significant amount of information, it is used to generate the feature space in classification algorithms.

Depending on the transformation that the image undergoes from which the points are to be characterized, the descriptor must satisfy certain invariance principles:

translation This is the simplest case and is automatically resolved by the feature point extractor;

scale This is another transformation that is typically handled by the feature point extractor;

brightness Images may experience variations in brightness;

rotation Images may represent the same scene rotated;

perspective Changes in perspective complexly distort the observed portion of the world.

Before the introduction of the compact descriptor concept, the universally adopted method for comparing two feature points was the correlation between the areas surrounding the point:

$$d(\mathbf{p}_1, \mathbf{p}_2) = \sum_{\boldsymbol{\delta} \in \Omega} w_\delta (I_1(\mathbf{p}_1 + \boldsymbol{\delta}) - \bar{I}_1)(I_2(\mathbf{p}_2 + \boldsymbol{\delta}) - \bar{I}_2) \tag{6.1}$$

with Ω a fixed-size window centered on the point in the two images and $\bar{I}_n$ the average value of the image within the window Ω. w_δ is an optional weight (for example, a Gaussian) to assign different contributions to pixels that are near or far from the point. The correlation is invariant to changes in brightness but requires a high computational cost. In this case, the descriptor is precisely the portion of the image surrounding the identified point [Mor80].

A similar approach to correlation, which is not invariant to brightness but is more computationally efficient, is the SAD (*Sum of Absolute Differences*):

$$d(\mathbf{p}_1, \mathbf{p}_2) = \sum_{\boldsymbol{\delta} \in \Omega} |I_1(\mathbf{p}_1 + \boldsymbol{\delta}) - I_2(\mathbf{p}_2 + \boldsymbol{\delta})| \tag{6.2}$$

To make the SAD invariant to brightness, comparisons are typically performed not on the original image, but on the horizontal and vertical derivative images. This reasoning appears quite straightforward but can be further generalized to the concept of performing comparisons not on the original image, but between one or more images extracted using different *kernels*, which provide the descriptor with certain levels of invariance.

It is also worth noting that the comparison of pixels between images is nonetheless an algorithm of type $O(n^2)$: performing these comparisons point by point still requires significant computational weight and multiple memory accesses. Modern solutions aim to overcome this limitation by proposing the extraction of a descriptor from the neighborhood of the point, which is smaller in size than the number of represented pixels, yet maximizes the information contained within it.

Both SIFT (section 5.3) and SURF (section 5.4) extract their descriptors by leveraging scale and rotation information derived from the image (it is possible to extract this information independently, and therefore it can be applied to any class of descriptors to make them invariant to scale and rotation). The descriptors obtained from SIFT and SURF are different versions of the same concept, namely the histogram of oriented gradients (section 6.2), which serves as an example of how to compress the variability around a point into a reduced-dimensional space.

All currently used descriptors do not directly utilize the image points as descriptors, but it is easy to see that a sufficiently well-distributed subset of points is enough to achieve an accurate description of the point. In [RD05], a descriptor is created using the 16 pixels located along the discrete circumference of radius 3. This description can be made even more compact by

transitioning to the binary form of the *Local Binary Patterns* described later, or by not being constrained to the circumference, as in *Census* or *BRIEF*.

Another approach is to appropriately sample the kernel space [GZS11], extracting from m coordinates around the keypoint the values obtained from convolutions of the original image (horizontal and vertical Sobel), in order to create a descriptor consisting of just $2m$ values.

It is noteworthy that, for purely computational reasons related to resource reuse, a specific descriptor extractor is often associated with each particular feature point extractor.

From this introduction, it is clear that describing a key point with a smaller yet sufficiently descriptive dataset is a useful approach, especially in the context of classification. The concept of a descriptor arises from the attempt to extract local information from the image that allows for the preservation of a significant portion of the information. In this way, it becomes possible to perform (relatively) fast comparisons between points across images or to use such descriptors as features for training classifiers.

6.1 *Haar* Features

The Haar *features* (the name derives from their similarity to Haar *wavelets*) refer to a series of image filters constructed as sums and differences of purely rectangular subregions of the image itself [PP99]. Examples of Haar *features* are shown in Figure 6.1. The resulting value of the filter is the sum of the grayscale values of the pixels underlying the areas in white, minus the value of the pixels underlying the areas indicated in black. By their nature, these filters can be efficiently implemented using the integral image (Section 1.14).

Haar *Features* are used as an approximation of convolutions for calculating key points in the SURF algorithm, or as input features for decision trees to obtain weak classifiers.

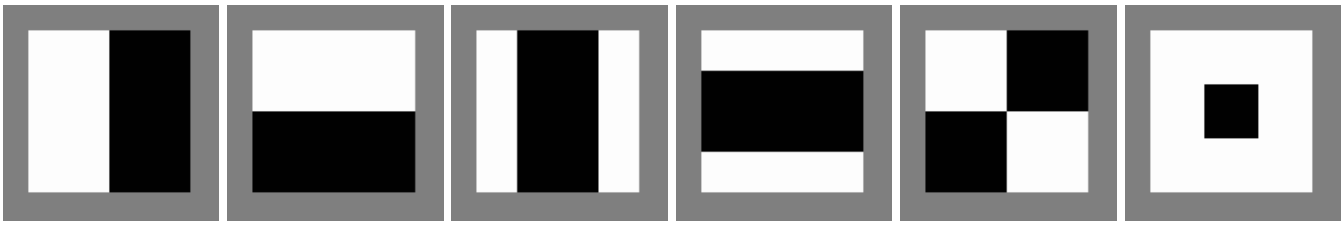


Figure 6.1: Examples of Haar *Features*. In the light areas, the underlying region is summed, while in the dark areas, it is subtracted, respectively.

Even though the form could potentially be anything, the number of bases for the *features* is typically limited (efforts are made, if possible, to avoid *features* that are too complex and computationally intensive).

In addition to the type of *feature*, it is necessary to select the sub-area of application: from each sub-window of the area to be analyzed, it is indeed possible to extract a value following the application of one of these many *features*. Identifying the most discriminative features is part of the training activity (*Decision Stump* ordered with *AdaBoost*) or through techniques such as PCA.

6.2 Histogram of Oriented Gradients

The Histogram of Oriented Gradient (HOG) is one of the techniques that has recently achieved significant success in effectively describing an area. This method was successfully used for the first time in SIFT to describe feature points and, in conjunction with SVM, to obtain highly performant classifiers [DT05].

Given the window within which to extract the descriptor, the magnitude and phase of a gradient operator (a derivative filter, Sobel, or any other) are calculated for each point. The extracted phase is then quantized: typically, between 6 to 9 *bins* are computed, and optionally, the phase is calculated with periodicity π, thereby ignoring the sign of the gradient.

The ideas underlying HOG are both to utilize the gradient phase to obtain a compact yet strongly illumination-invariant descriptor, and to decompose the window under examination into subparts, called cells, which may be overlapping and potentially of any shape and size. While HOG cells are typically square, it is possible to find rectangular cells in R-HOG or circular cells in C-HOG.

From each subpart into which the image is divided, a descriptor piece is extracted, formed by the histogram of the gradient magnitude. The most commonly used versions of HOG aim to locally normalize brightness and contrast. To achieve this, spatially adjacent cells are grouped into blocks. For each block, a normalization factor is extracted, which is used to adjust the weight of each sub-cell.

The histogram *bin* for each cell into which the area is divided represents the descriptor, a descriptor to be used for point comparisons or in training for object recognition.

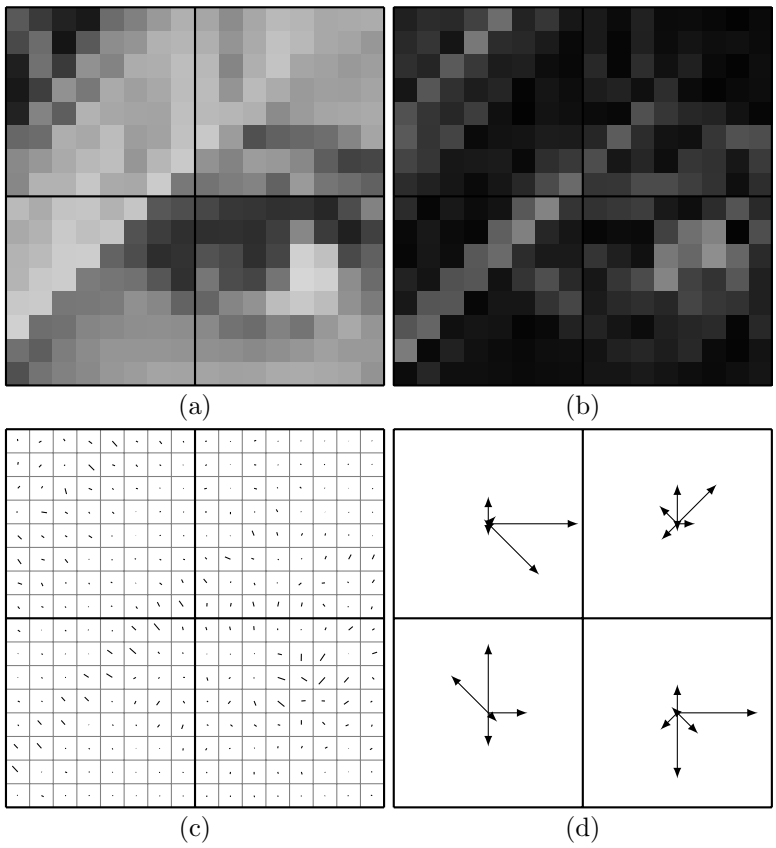


Figure 6.2: Calculation of the gradient histogram: from the cells, which may overlap, into which the image *(a)* is divided, the magnitude *(b)* and phase *(c)* of the gradients are computed, and for each cell, a histogram is constructed *(d)*.

6.3 Integral Channel Descriptor

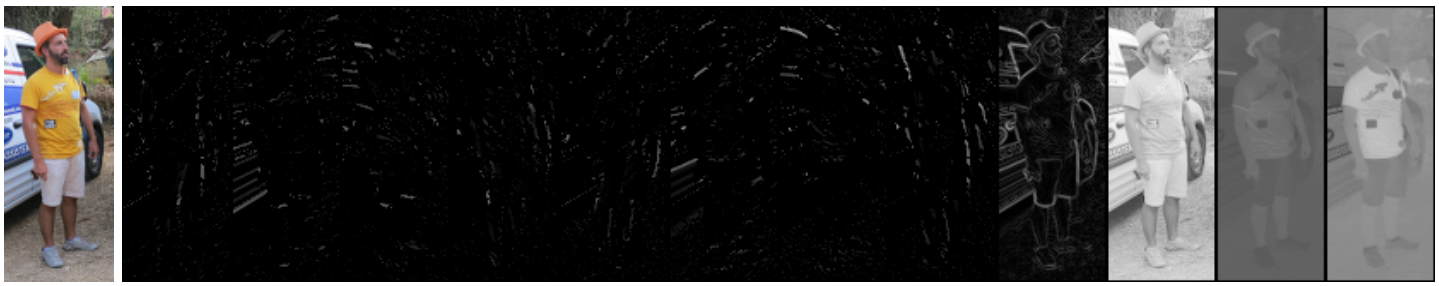


Figure 6.3: Image of the channels used by ICF. From left, the original image is followed by images of the different channels: 8 channels for the quantized gradient phase, 1 channel for the gradient magnitude, and 3 channels for the LUV components, respectively.

The variants of HOG exhibit variable cell shapes, and it has been observed that one way to accelerate the computation of the histogram of phase magnitudes is to leverage the integral image once again. Therefore, straddling the line between HOG and Haar features, the Integral Channel Features have recently demonstrated interesting performance, effectively serving as a generalization of HOG that utilizes the integral image.

The characteristic values that can be extracted derive from the summation of areas calculated not directly on the original image but from different secondary images, obtained through non-linear processing of the area to be characterized. Among the possible processing techniques, the most common are the gradient phase channels already seen in HOG, the gradient magnitude image, the grayscale image itself, and, if available, two additional channels representing chrominance.

Regarding the gradient, it is often calculated using the Sobel operator, but various experiments show that a simple derivative filter produces satisfactory results nonetheless. In this case, the Sobel phase can be used with or without sign, depending on the specific applications.

The scalar representing the characteristic to be extracted is simply the sum of a rectangular area within one of the calculated channels.

6.4 Binary Descriptors

One of the issues with traditional descriptors is that they are composed of a vector containing a certain number of values, which are typically floating-point numbers due to possible normalizations. As a result, both the extraction of this vector and the subsequent comparison phase require significant computational time.

One of the most promising alternatives is to extract a binary vector as a descriptor. The binary vector occupies less memory space, and to perform the comparison, it is sufficient to calculate the Hamming distance between the respective binary strings. The Hamming distance is computed very efficiently by performing the XOR operation on the binary strings and counting the active bits (POPCOUNT).

6.4.1 Census Transformation

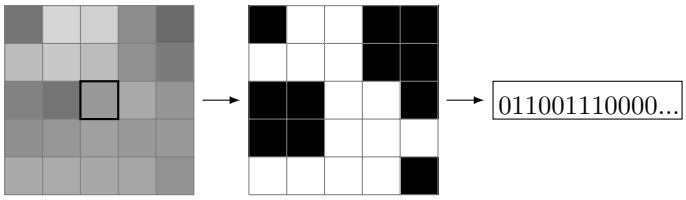


Figure 6.4: Calculation of the Census transformation in the vicinity of a point: the neighborhood of a point is binarized with respect to the value of the point itself, and from this thresholding, a binary string is constructed.

The Census Transformation [ZW94] involves describing the portion of an image around a point using a bit string. For each pixel in the image, the surrounding area, of fixed size and shape, is systematically analyzed, and each pixel in this area is compared to the generating pixel. If the pixel has a greater gray intensity, it is assigned a bit 1, while if it has a lower intensity, it is assigned a bit 0:

$$\tau(\mathbf{x}) = \begin{cases} 1 & I(\mathbf{x}) < I(0) \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

By systematically scanning the area, it is possible to generate a binary string. An example of the Census transformation of an area 5×5 is shown in Figure 6.4: from this area, a binary string of 25-1 bits is generated (the central pixel is effectively negligible).

The Census transformation demonstrates its potential in the case of comparisons: two generic points, instead of being compared through a SAD of the surrounding area, are compared using the Hamming distance between their respective binary strings of the Census transform.

Through the construction of the binary string, leveraging the difference in gray tone, the Census transform is quite invariant to brightness.

6.4.2 Local Binary Pattern (LBP)

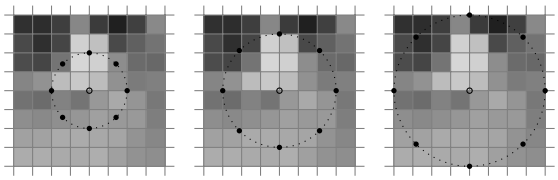


Figure 6.5: Pixels considered during the extraction of an 8-bit LBP descriptor, varying the radius of the circle.

In the first version of LBP [OPM02], the descriptor was found to be indistinguishable from the Census transform over a window 3×3: for each image point, the 8 neighboring pixels are examined, thresholded against the central pixel, thereby generating an 8-bit string, which corresponds to an integer descriptor ranging from 0 to 255.

This original concept has subsequently been extended to n points along a circle of radius ρ centered at the pixel for which the characteristic is to be calculated (figure 6.5). Since the point on the radius typically does not fall exactly on a pixel, bilinear interpolation can be performed to estimate the value to be thresholded in order to construct the binary string.

The LBP operator produces 2^n possible values for each point in the image. In the case where the image is rotated, the pixel values move along the circumference, and consequently, the bits within the binary string also rotate. It is possible to obtain a rotation-invariant LBP operator by normalizing the string through some transformation. One such transformation is, for example, performing n rotations on the binary string and taking, among all the results, the one with the minimum value:

$$\text{LBP}_{r.i.} = \min_i \text{ROR}_i(\text{LBP}) \quad i \in [0, n-1] \tag{6.4}$$

6.4.3 BRIEF

The Census transform does not prescribe a specific shape for the area over which comparisons are performed to generate the binary string. This limitation is addressed in [CLSF10], one of the first works to formally pose the problem of computing a discriminative binary descriptor.

Most binary descriptors are inspired by Census, which is generalized by no longer comparing each pixel solely with the center of the area, but instead performing comparisons between arbitrary pairs of points selected according to specific criteria. The comparison function is defined as:

$$\tau(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \tilde{I}(\mathbf{x}) < \tilde{I}(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the coordinates of two arbitrary pixels within the area surrounding the point to be described, and $\tilde{I}(\cdot)$ denotes the pixel intensity in a filtered (typically low-pass) version of the original image.

To obtain this coordinate mask, a training process is performed on sample images to identify the combination that maximizes correct detections.

Numerous approaches have been proposed in the literature to address the problem of how to filter the image and how to select the points used to construct the descriptor.

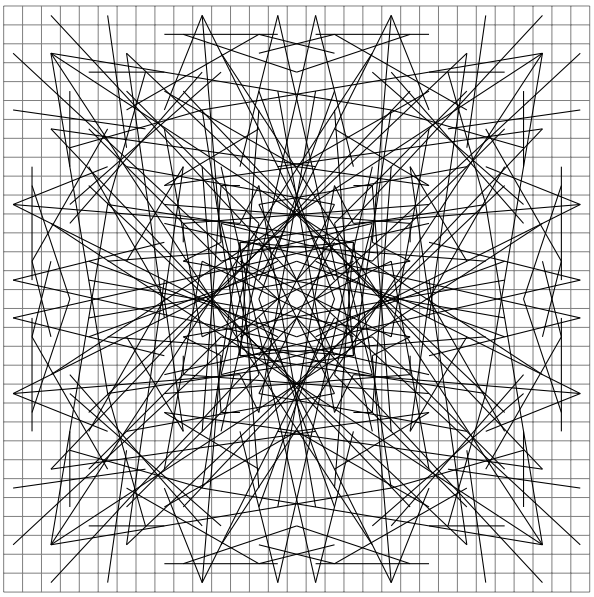


Figure 6.6: Example of a 256-bit BRIEF descriptor.

6.4.4 ORB

ORB (Oriented FAST and Rotated BRIEF) [RRKB11] is derived from the combination of the FAST keypoint detector and the BRIEF descriptor, modified to be rotation invariant. Since FAST does not provide orientation information, ORB computes it by determining the intensity centroid within the patch surrounding the keypoint: the vector connecting the keypoint to the centroid is invariant to rotation and sufficiently robust to illumination changes.

Using this angle, ORB rotates the BRIEF pattern (i.e., it employs precomputed rotated patterns, for example by quantizing the full rotation into 30 discrete parts) to generate a binary descriptor that is robust to rotation, known as rBRIEF. Additionally, by applying FAST and BRIEF across multiple levels of an image pyramid, scale invariance is also achieved.

ORB delivers performance comparable to SIFT and SURF in terms of accuracy, while offering significantly higher speed and, most importantly, being free of patent restrictions.

6.5 Machine Learning

With the evolution of Machine Learning techniques, the concept of descriptor has gradually faded, giving way to a much more harmonious theory regarding the concept of information. Observing certain algorithms such as RBNs (section 4.8.2) and CNNs (section 4.8.5) as *auto-encoders* when connected to a *decoder* brings the concept of descriptor to its limits, allowing for the extraction of the minimum number of features necessary for the reconstruction of a portion of the image (under a given metric).

Chapter 7

Tracking

Downstream of the extraction of characteristic points and their descriptors, there is a discussion on the association of descriptors and the concept of dense optical flow.

7.1 Descriptor Comparison and Association

As a conclusion to this chapter, it is essential to say a few words about the topic of comparison.

Let I_1 and I_2 be two images to be analyzed, and let $\mathbf{p}_1$ and $\mathbf{p}_2$ be two characteristic points identified in the first and second images, respectively. To determine whether these two image points represent the same point, typically observed from different viewpoints and thus affected by affine transformations (translations, scaling changes, rotations), homographies, and possibly changes in brightness, it is necessary to define some form of metric $d(\mathbf{p}_1, \mathbf{p}_2)$ to perform this comparison.

Associated with each descriptor, a specific metric can be defined. In general, the most commonly used metrics are L1 (Manhattan, *SAD*) and L2 (Euclidean, *SSD*).

Since the points extracted from the two images will certainly be more than one, a scanning process must be performed, and each point from the first image will be associated only with that point from the second image which has the minimum distance according to the selected metric:

$$\hat{\mathbf{p}_2} = \arg\min_i d(\mathbf{p}_1, \mathbf{p}_{2,i}) \tag{7.1}$$

Typically, to reduce the number of incorrect matches, the association is confirmed only if the metric is below a specified threshold and the ratio between the best match and the second-best match is below a second uniqueness threshold.

Finally, after finding $\mathbf{p}_2$, the best association of point $\mathbf{p}_1$ on the second image, it can be verified that $\mathbf{p}_2$ does not have better associations on the first image.

7.2 Lucas-Kanade

The Lucas-Kanade optical flow estimation method [LK81] is a technique for estimating the motion of interesting features across successive frames of a video. The goal is to associate a motion vector (u, v) with each "interesting" pixel in the scene by comparing two consecutive images.

The algorithm makes the following assumptions:

- The brightness between consecutive images remains constant;
- The two images must be temporally close enough so that the objects do not experience significant displacement (the algorithm performs well with slowly moving objects);
- The image contains objects with sufficient texture in grayscale (the original algorithm does not explicitly utilize color) whose gradient changes gradually.

Starting from the optical flow equation for each point (x, y):

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t) \tag{7.2}$$

where $I(t)$ is one image and $I(t + \delta t)$ is the subsequent one. With the first-order Taylor series expansion:

$$\begin{aligned} &I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t) \\ &I(x, y, t) + \tfrac{\partial I}{\partial x}(x, y, t)u\delta t + \tfrac{\partial I}{\partial y}(x, y, t)v\delta t + \tfrac{\partial I}{\partial t}(x, y, t)\delta t = I(x, y, t) \\ &\tfrac{\partial I}{\partial x}(x, y, t)u + \tfrac{\partial I}{\partial y}(x, y, t)v + \tfrac{\partial I}{\partial t}(x, y, t) = 0 \end{aligned} \tag{7.3}$$

The Lucas-Kanade algorithm assumes that the change in brightness of a pixel in the scene is fully compensated by the gradient of the scene itself, that is:

$$I_x u + I_y v + I_t = 0 \tag{7.4}$$

given the temporal gradient I_t and the spatial gradient (I_x, I_y).

Obviously, a single pixel does not contain enough information to solve this problem. To gather more observations, it is assumed that a neighborhood of the pixel exhibits the same motion, that is,

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_n) & I_y(\mathbf{p}_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_n) \end{bmatrix} \tag{7.5}$$

where $\mathbf{p}_1 \dots \mathbf{p}_n$ are the points in the neighborhood of the point to be estimated. The solution can be obtained through the method of *normal equations*

$$\begin{bmatrix} \sum I_x I_y & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} \tag{7.6}$$

It is noteworthy that this is also the matrix of characteristic points utilized by Shi-Tomasi or Harris (see 5.2): the characteristic points of this matrix are points that can be easily tracked using the Lucas-Kanade algorithm.

When the motion is greater than one pixel, an iterative algorithm is required to solve the problem, along with a *coarse-to-fine* approach to avoid local minima: there will exist a scale at which the motion of the pixel is less than one pixel.

Chapter 8

Pin-Hole Camera

In this chapter, we address the problem of describing the process through which incident light on objects is captured by a digital sensor. This concept is fundamental in image processing as it provides the relationship that connects the points of an image with their position in the world, allowing us to determine the area of the world associated with a *pixel* of the image or, conversely, to identify the portion of the image that corresponds to a specific region in world coordinates.

The universally accepted projective model, known as the *Pin-Hole Camera*, is based on simple geometric relationships[1].

In figure 8.1, a highly simplified diagram illustrates how the image is formed on the sensor. The observed point $(x_i, y_i, z_i)^\top$, expressed in camera coordinates, is projected onto a cell of the sensor $(\tilde{u}_i, \tilde{v}_i)^\top$. All these rays converge at a single point: the focal point (the *pin-hole*).

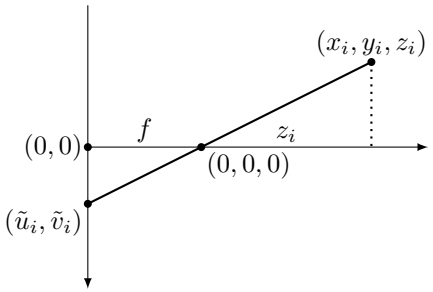


Figure 8.1: The *pin-hole* camera model. A world point in camera coordinates is projected onto the image plane.

Analyzing figure 8.1, it can be observed that the ratios between similar triangles generated by optical rays describe the equation that allows projecting a generic point $(x_i, y_i, z_i)^\top$, expressed in camera coordinates (one of the reference systems in which one can operate), into sensor coordinates $(\tilde{u}_i, \tilde{v}_i)^\top$:

$$\begin{bmatrix} \tilde{u}_i \\ \tilde{v}_i \end{bmatrix} = \frac{f}{z_i} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{8.1}$$

where f is the focal length (the distance between the pin-hole and the sensor). It should be noted that the coordinates $(x_i, y_i, z_i)^\top$, expressed in camera coordinates, in this book follow the *left-hand rule* (commonly used in *computer graphics*), as opposed to the *right-hand rule* (more commonly used in robotic applications), which is chosen to express world coordinates. The use of coordinate z to express distance is a purely mathematical requirement due to the transformations that will be presented shortly.

The sensor coordinates $(\tilde{u}_i, \tilde{v}_i)^\top$ are not the image coordinates but are still considered "intermediate" coordinates. Therefore, it is necessary to apply an additional transformation to obtain the image coordinates:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} D_u \tilde{u}_i \\ D_v \tilde{v}_i \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \tag{8.2}$$

where the coordinates (u_0, v_0) (*principal point*) account for the offset of the origin of the coordinates in the stored image relative to the projection of the focal point onto the sensor.

D_u and D_v are conversion factors between the units of the sensor's reference system (meters) and those of the image (pixels), accounting for the various conversion factors involved. With the advent of digital sensors, it is typically $D_u = D_v$.

In the absence of information available from the various *datasheets* regarding f, D_u, and D_v, there is a tendency to consolidate these variables into two new variables referred to as k_u and k_v, which represent the effective focal lengths

[1] Most models fall under the pinhole model, understood as a model where all light rays pass through a single point; however, there are models that generalize the pinhole model to catadioptric cameras, such as the Mei model, or other different models like the *double sphere*.

measured in pixels. These can be empirically obtained from the images, as will be discussed in the calibration section. These variables, involved in the conversion between sensor coordinates and image coordinates, are related to each other as

$$\begin{aligned} k_u &= D_u f = \frac{u_0}{\tan \alpha_u} \\ k_v &= D_v f = \frac{v_0}{\tan \alpha_v} \end{aligned} \tag{8.3}$$

with α_u and α_v angles that can be approximated to the half-angle of the camera aperture (horizontal and vertical, respectively). When the optics are undistorted and the sensor has square pixels, k_u and k_v tend to take on the same value.

Due to the presence of the ratio, the equation (8.1) cannot be clearly represented in a linear system. However, it is possible to modify this representation by adding an unknown λ and an additional constraint, allowing us to express this equation in the form of a linear system. To achieve this, we will leverage the theory presented in section 1.4 regarding homogeneous coordinates.

Using homogeneous coordinates, it can be easily shown that the system (8.1) can be written as

$$\begin{bmatrix} \lambda\, u_i \\ \lambda\, v_i \\ \lambda \end{bmatrix} = \lambda \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{8.4}$$

solved for $\lambda = z_i$. For this reason, λ is implied, and instead, homogeneous coordinates are used: to obtain the point in non-homogeneous coordinates, one must divide the first two coordinates by the third, resulting in equation (8.1). The use of homogeneous coordinates allows for the implicit representation of division by the coordinate z.

The matrix $\mathbf{K}$, combining the transformations (8.2) and (8.3), can be expressed as:

$$\mathbf{K} = \begin{bmatrix} \frac{u_0}{\tan \alpha_u} & k_\gamma & u_0 \\ 0 & \frac{v_0}{\tan \alpha_v} & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} k_u & k_\gamma & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8.5}$$

Such a matrix, not depending, as we will see later, on factors other than those of the chamber itself, is referred to as the matrix of intrinsic factors. The matrix $\mathbf{K}$ is an upper triangular matrix, defined by 5 parameters.

With modern digital sensors and the construction of cameras not manually but with precise numerical control machines, it is possible to set the *skew factor* k_γ, a factor that accounts for the fact that the angle between the axes in the sensor is not exactly 90 degrees, to zero.

Setting $k_\gamma = 0$, the inverse of the matrix (8.5) can be expressed as:

$$\mathbf{K}^{-1} = \begin{bmatrix} \frac{1}{k_u} & 0 & -\frac{u_0}{k_u} \\ 0 & \frac{1}{k_v} & -\frac{v_0}{k_v} \\ 0 & 0 & 1 \end{bmatrix} \tag{8.6}$$

The knowledge of these parameters (see section 8.5 regarding calibration) determines the ability to transform a point from camera coordinates to image coordinates or, conversely, to generate the line in camera coordinates corresponding to a point in the image.

With this modeling, in any case, the contributions due to lens distortion have not been taken into account. The pin-hole camera model is indeed valid only if the image coordinates used refer to undistorted images.

8.1 Lens Distortion

The majority of commercial cameras deviate from the *pin-hole* camera model, and this deviation is generally greater the larger the camera's field of view. Since each optical system consists of a certain number of lenses, the distortion arises from imperfections during the production and assembly phases of the optics. Achieving a distortion-free lens is an extremely costly process, and particularly in low-cost applications where reliance on economical optics is necessary, this issue becomes very evident.

These non-idealities generate a nonlinear distortion that is challenging to model, and due to the fact that this distortion depends on the interaction between the lens and the sensor, lens manufacturers typically do not provide, or are unable to provide, geometric information on how to represent such distortion.

It is important to note that the pin-hole camera model is only valid if the image being processed is not distorted; therefore, calibration, or correcting for geometric distortion, is a prerequisite for accurately reconstructing the three-dimensionality of the observed scene.

From the perspective of the optical ray, the distortion introduced by the lens is situated between the world and the *pin-hole*. The equation of the pin-hole camera modified with the optical distortion transforms into

$$\mathbf{p} = \mathbf{K} f_d([\mathbf{Rt}]\mathbf{x}) \tag{8.7}$$

With this formalism, the distortion f_d transforms a point from undistorted coordinates to distorted coordinates. This choice, as opposed to the inverse formulation, arises from purely practical considerations: Since the objective is to obtain a dense and undistorted output image (see the discussion in section 1.12), it is necessary to compute the function that transforms a non-distorted point into a distorted point.

In general, the distorting contributions of the lens are divided into radial (directly along the line connecting the point to the center of distortion) or tangential (which are perpendicular to the line). Tangential contributions (and other contributions not mentioned here) are typically small, while radial distortion is always detectable and generally increases in intensity as the focal length becomes shorter.

This section focuses on deriving a general relationship between the ideal point (x, y) and the actual distorted image point observed $(\breve{x}, \breve{y})$.

In the entire image, there exists a single point (x_d, y_d), defined as the distortion center, where distortion does not produce any effects. For this point $(x, y) = (\breve{x}, \breve{y})$.

To define the distortion, it is necessary to operate in a new set of coordinates, relative to the center of distortion:

$$\begin{aligned} \bar{x} &= x - x_d \\ \bar{y} &= y - y_d \end{aligned} \tag{8.8}$$

The distortion center is typically located near $(0, 0)$, but there is no guarantee that it coincides with the *principal point*. In several articles, it is proposed as an approximation to ignore the distortion center and align it with the *principal point* or to consider only the term of *decentering distortion*.

The classical formulation of *Brown-Conrady* [Bro66] models lens distortion in the form of a deviation:

$$\begin{aligned} \breve{x} &= x + \delta_x(\bar{x}, \bar{y}) \\ \breve{y} &= y + \delta_y(\bar{x}, \bar{y}) \end{aligned} \tag{8.9}$$

Such deviations can be divided into contributions:

radial distortion The deviation due to radial distortion is described by the equation

$$\begin{aligned} \delta_x^r &= \bar{x} f_r(r) \\ \delta_y^r &= \bar{y} f_r(r) \end{aligned} \tag{8.10}$$

where $f_r(r)$ is a function solely of the radius $r = \sqrt{\bar{x}^2 + \bar{y}^2}$, the Euclidean distance between the point and the center of distortion, subject to the constraint $f_r(0) = 1$.

The function $f_r(r)$ of radial distortion is not a well-known model but can be approximated using the first terms of the series expansion: The presence of only the powers of 2 is due to the symmetry of the function f_r.

thin prism distortion imperfections in the manufacturing process and misalignment between the sensor and the lens introduce additional asymmetric distortions. This is typically modeled as It seems that you have entered a placeholder or a command related to a mathematical block. Please provide the content or text you would like to have translated, and I will be happy to assist you!

Such contributions are often inadequate, however, to describe the effects of optical decentralization.

decentering distortion It is typically caused by improper assembly of the lens and the various components that make up the optics. The *Brown-Conrady* model represents the contribution of decentering in the form of

$$\begin{aligned} \delta_x^{(t)} &= (p_1(r^2 + 2\bar{x}^2) + 2p_2\bar{x}\bar{y})(1 + p_3 r^2 + \ldots) \\ \delta_y^{(t)} &= (p_2(r^2 + 2\bar{y}^2) + 2p_1\bar{x}\bar{y})(1 + p_3 r^2 + \ldots) \end{aligned} \tag{8.11}$$

This contribution consists of both a radial component and a tangential component.

Incorporating all these contributions into equation (8.9), the overall *Brown-Conrady* model can be expressed as In fact, the radial distortion is dominant, and in most applications, the first terms are more than sufficient.

For example, OpenCV models distortion using the *R3P1* model: 3 radial terms (k_1, k_2, k_3) and the first-degree decentering term (p_1, p_2).

The distortion coefficients are derived using various techniques available in the literature, applied to images acquired in a structured environment (calibration grids). Typically, a nonlinear minimizer is employed, and one either works with lines and iterates until all the curves in the image become straight *plumb-line method* [DF01], or enforces that points on a plane with known coordinates represent a homography. Such techniques are applicable only when operating in image coordinates (approach 1).

To calibrate the distortion in normalized camera coordinates (approach 2), both distortion and intrinsic camera parameters must be computed simultaneously [Zha99]. An initial estimate of the intrinsic parameters can be obtained from a linear minimizer, but the final estimate is achieved only through a nonlinear minimizer.

The widely used technique for estimating distortion parameters involves optimizing the observation of feature points in the image, whose positions in world coordinates are known, in order to enforce a complete perspective projection (section 8.5.6).

8.2 World Coordinates and Camera Coordinates

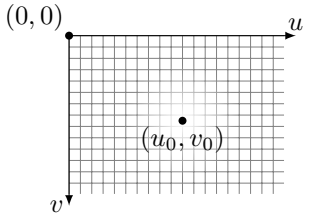


Figure 8.2: Image Coordinates (*Coordinate Immagine*)

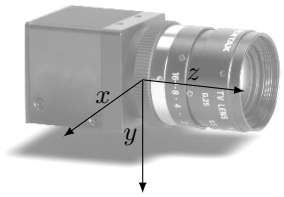


Figure 8.3: Coordinate Camera (*Camera coordinates*)

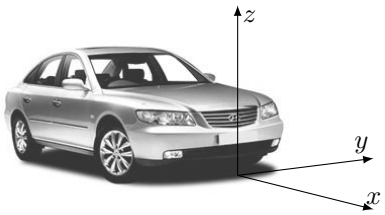


Figure 8.4: Example of "Vehicle" or "World" coordinates: Front-Left-Up or ISO 8855 (*World coordinates*)

When dealing with practical problems, it becomes necessary to transition from a reference system fixed to the camera, where point $(0,0,0)^\top$ coincides with the focal point (*pin-hole*), to a more generic reference system that better suits the user's needs. In this system, the camera is positioned at an arbitrary point in the "world" and oriented with respect to it in an arbitrary manner. This discussion applies to any generic sensor, including non-video sensors, by defining relationships that allow for the conversion of points from world coordinates to sensor coordinates and vice versa.

At this point, it is necessary to clarify the terminology related to reference systems in this book: the reference system termed "world" is defined as the system that is considered absolute and fixed at any given time, with respect to which the sensor is positioned. In Figure 8.4, for example, the origin of the "world" system is associated with a point on the vehicle (such as the front point). In this case, the "vehicle" (*body*) and "world" (*world*) systems are synonymous.

However, this distinction becomes less clear when there is a moving vehicle with respect to a "world" that can again be defined as the fixed reference system. In this case, we will have the sensor coordinates, the local coordinates of the vehicle/body, and finally those of the world. Typically, however, the coordinate system that distinguishes the sensor, vehicle, and world is kept consistent.

In camera coordinates, the special role that the coordinate z assumes is due to purely mathematical reasons, specifically the use of homogeneous coordinates, which during projection necessitates the division of the first two components by the third. In "sensor" coordinates, this limitation is no longer applicable.

Although not binding in any way, this book adopts the "sensor," "body," and "world" systems presented in Figure 8.4 (ISO 8855), which assigns the axis z the height of the point above the ground.

Therefore, to arrive at the definitive equation of the *pin-hole* camera, we start from equation (8.4) and apply the following considerations:

- the axes are exchanged through a permutation $\mathbf{\Pi}$ (which is still a rotation) to obtain the final reference system;
- the sensor must be rotated through a transformation ${}^{w}\mathbf{R}_{b}$ and consequently does not align with the axes of the "world" reference system;
- the pin-hole no longer coincides with point $(0,0,0)^\top$ but lies at a generic point $\mathbf{t}_0 = (x_0, y_0, z_0)^\top$ expressed in world coordinates.

The conversion from "world" coordinates to "camera" coordinates, being a composition of rotations, is also a rotation described by the equation $\mathbf{R} = {}^{c}\mathbf{R}_{w} = \mathbf{\Pi}\,{}^{w}\mathbf{R}_{b}^{-1}$.

Let $(x_i, y_i, z_i)^\top$ be a point in "world" coordinates and $(\tilde{x}_i, \tilde{y}_i, \tilde{z}_i)^\top$ the same point in "camera" coordinates. The relationship that connects these two points can be expressed as

$$\begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \\ \tilde{z}_i \end{bmatrix} = \mathbf{R} \left(\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} - \mathbf{t}_0 \right) = \mathbf{R} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \tilde{\mathbf{t}}_0 \tag{8.12}$$

where $\mathbf{R}$ is a matrix 3×3 that converts from world coordinates to camera coordinates, accounting for the rotations and the sign changes of the axes between world coordinates and camera coordinates (see appendix A), while the vector

$$\tilde{\mathbf{t}}_0 = -\mathbf{R}\mathbf{t}_0 \tag{8.13}$$

represents the position of the pin-hole $\mathbf{t}_0$ with respect to the origin of the world system, but expressed in the camera coordinate system.

It should be noted that rotation matrices are orthonormal matrices: they have a determinant of 1, thus preserving distances and areas, and the inverse of a rotation matrix is its transpose.

The matrix $\mathbf{R}$ and the vector $\mathbf{t}_0$ can be combined into a matrix form 3×4 by utilizing homogeneous coordinates. With this representation, it is possible to express the projection of a point, represented in world coordinates, homogeneous to $(x_i, y_i, z_i)^\top$, into a point with image coordinates, homogeneous to $(u_i, v_i)^\top$:

$$\lambda \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{R}|\tilde{\mathbf{t}}_0] \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{8.14}$$

From this equation, it is quite explicit that at each point of the image (u_i, v_i), there are infinitely many points in the world $(x_i, y_i, z_i)^\top$ that lie on a line as the parameter λ varies.

By implying λ and collecting the matrices, we obtain the final equation of the *pin-hole* camera (which does not account for, nor should it account for, distortion):

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{R}|\tilde{\mathbf{t}}_0] \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{8.15}$$

having defined $\mathbf{P} = \mathbf{K}[\mathbf{R}|\tilde{\mathbf{t}}_0]$ as the projection matrix (*camera matrix*) that will be used subsequently [Str87]. The matrix $\mathbf{P}$ is a 3×4 matrix and, being rectangular, it is not invertible.

It is important to note that by imposing an additional constraint on the points, for example $z_i = 0$, the matrix $\mathbf{P}$ is reduced to a matrix 3×3, which is invertible and is exactly the homographic matrix (see section 8.3.1) of the perspective transformation of the ground points. The matrix $\mathbf{P}_{z=0}$ is an example of an IPM (*Inverse Perspective Mapping*) transformation used to obtain a bird's eye view of the captured scene [MBLB91].

The inverse relationship of the equation (8.14), which transforms image points into world coordinates, can be expressed as:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \lambda \mathbf{R}^{-1}\mathbf{K}^{-1} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} + \mathbf{t}_0 = \lambda \mathbf{v}(u_i, v_i) + \mathbf{t}_0 \tag{8.16}$$

where it is clear that each image point corresponds to a line (as λ varies) in the world that passes through the pin-hole ($\mathbf{t}_0$) and is directed in the direction of

$$\mathbf{v}(u_i, v_i) = \mathbf{R}^{-1}\mathbf{K}^{-1} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \tag{8.17}$$

with $\mathbf{v} : \mathbb{R}^2 \to \mathbb{R}^3$ being a function that associates each image point with the vector that connects the pin-hole to the corresponding sensor point.

By directly using the *Camera Matrix* $\mathbf{P} = [\mathbf{P}_{3\times3}|\mathbf{p}_4]$, it is possible to achieve a result equivalent to equation (8.16) in the form of

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \lambda \mathbf{P}_{3\times3}^{-1} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} - \mathbf{P}_{3\times3}^{-1}\mathbf{p}_4 \tag{8.18}$$

, thereby avoiding the explicit use of the intrinsic and extrinsic parameter matrices. The two formulations are, of course, equivalent.

8.2.1 Properties of the Rotation Matrix

The rotation matrix will often be referred to in the text, for the sake of brevity, as an array in the `C` programming language:

$$\mathbf{R} = \begin{bmatrix} r_0 & r_1 & r_2 \\ r_3 & r_4 & r_5 \\ r_6 & r_7 & r_8 \end{bmatrix}$$

The rotation matrix is a highly overparameterized matrix: its 9 linearly independent parameters are, in fact, generated by 3 variables in a nonlinear manner (see appendix).

Without explicitly stating the angles from which the matrix is generated, it is possible to provide some additional constraints. The rotation matrix has the property of not altering distances, being orthonormal and $\det(\mathbf{R}) = 1$. Each row and each column must have unit length, and every row and every column are orthonormal to each other, as they form orthonormal bases of the space. Therefore, knowing two row or column vectors of the matrix $\mathbf{r}_1$ and $\mathbf{r}_2$, it is possible to determine the third basis vector as the cross product of the previous two:

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \tag{8.19}$$

Similarly, the dot product between two row vectors or two column vectors must yield a zero value, as they are orthogonal to each other. Under these constraints, there exist two exact solutions, one of which is:

$$\mathbf{R} = \begin{bmatrix} r_0 & r_1 & (1 - r_0^2 - r_1^2)^{\frac{1}{2}} \\ r_3 & r_4 & s(1 - r_3^2 - r_4^2)^{\frac{1}{2}} \\ (1 - r_0^2 - r_3^2)^{\frac{1}{2}} & s(1 - r_1^2 - r_4^2)^{\frac{1}{2}} & (r_0^2 + r_1^2 + r_3^2 + r_4^2 - 1)^{\frac{1}{2}} \end{bmatrix} \tag{8.20}$$

where $s = sgn(r_1 r_4 + r_2 r_5)$, while the other solution has exactly the opposite signs. Knowing a submatrix 2×2, it is possible to derive the other elements of the matrix itself up to a sign, always based on the fact that each row and column have unit norm.

8.2.2 Notable Results

We can use the rotation matrix and the pin-hole equation (8.15) to demonstrate some notable results. We define, from the system, the function f_{pm} of $\mathbb{R}^3$ in $\mathbb{R}^2$ called *perspective mapping*, defined as:

$$f_{pm}(x, y, z) = \left(k_u \frac{r_0 x + r_1 y + r_2 z}{r_6 x + r_7 y + r_8 z} + u_0, k_v \frac{r_3 x + r_4 y + r_5 z}{r_6 x + r_7 y + r_8 z} + v_0 \right) \tag{8.21}$$

the function of the *pin-hole camera* model written explicitly. For simplicity, it is assumed that the *pin-hole* coincides with the origin of the reference system.

Vanishing Points and Calibration

For each image, there are 3 vanishing points, closely related to the choice of reference axes.

Let us consider the first axis as an example. In our reference system, the coordinate x represents the distance (the discussion is similar for the other two coordinates). We take this coordinate to infinity while keeping the other two constant. What we obtain is the point

$$\lim_{x \to \infty} f_{pm}(x, y, z) = \left(k_u \frac{r_0}{r_6} + u_0, k_v \frac{r_3}{r_6} + v_0 \right) \tag{8.22}$$

.

Using homogeneous matrices, it is possible to achieve the same result with a more compact formalism.

Taking the perspective transformation (8.14) and sequentially sending $x \to \infty$, $y \to \infty$, and $z \to \infty$, the image points (in homogeneous coordinates) obtained, representing the vanishing points in the three directions, are exactly the columns of the matrix $[\mathbf{v}_x \mathbf{v}_y \mathbf{v}_z] = \mathbf{K} \cdot \mathbf{R}$, namely:

$$\begin{aligned} \mathbf{v}_x &= \mathbf{K}\mathbf{r}_1 \\ \mathbf{v}_y &= \mathbf{K}\mathbf{r}_2 \\ \mathbf{v}_z &= \mathbf{K}\mathbf{r}_3 \end{aligned} \tag{8.23}$$

where we have indicated with the syntax $\mathbf{r}_i$ the columns of the matrix $\mathbf{R}$. This is a first example of camera calibration that leverages knowledge of the image, specifically the position of the vanishing points.

In particular, considering the simplified case $u_0 = 0$, $v_0 = 0$, and $k_\gamma = 0$, the vanishing points are located at

It seems that you've provided a placeholder or a command for a math block (

$$\begin{aligned} \mathbf{v}_x &= \left(k_u \frac{r_0}{r_6}, k_v \frac{r_3}{r_6} \right) \\ \mathbf{v}_y &= \left(k_u \frac{r_1}{r_7}, k_v \frac{r_4}{r_7} \right) \\ \mathbf{v}_z &= \left(k_u \frac{r_2}{r_8}, k_v \frac{r_5}{r_8} \right) \end{aligned} \tag{8.24}$$

). If you have specific content or equations that you would like to translate from Italian to English, please provide that text, and I'll be happy to assist you!

It is noteworthy that since the 3 columns of $\mathbf{R}$ are orthonormal, it is sufficient to know 2 vanishing points to always obtain the third (see previous section).

Horizon Line

If we send to infinity not one variable but multiple ones, we obtain more than one point. For $x \to \infty$ but with $y = mx$, the vanishing point degenerates into a line described by the equation

$$k_v(r_3r_7 - r_4r_6)u + k_u(r_6r_1 - r_7r_0)v + k_uk_v(r_4r_0 - r_3r_1) = 0 \tag{8.25}$$

, known as the horizon line.

Degenerate Points and Lines

As a point in the projected image degenerates into a line, a line described by the equation $au + bv + c = 0$ becomes in the projected image

$$ak_u(r_0x + r_1y + r_2z) + bk_v(r_3x + r_4y + r_5z) + c(r_6x + r_7y + r_8z) = 0$$

that is

$$(ak_u\ r_0 + bk_vr_3 + cr_6)x + (ak_ur_1 + bk_vr_4 + cr_7)y + (ak_ur_2 + bk_vr_5 + cr_8)z = 0 \tag{8.26}$$

which represents the degenerate plane (with a normal as given by the equation) in three dimensions that passes through the origin (the *pin-hole*).

8.3 Notable Homographic Transformations

It is possible to provide a brief list of the transformations in computer vision that can be represented through a homography:

1. **Perspective Transformation**: This includes the transformation of images taken from different viewpoints, allowing for the correction of perspective distortions.
2. **Image Rectification**: Homographies can be used to align images taken from different angles to a common plane, making it easier to analyze or stitch images together.
3. **Panorama Stitching**: By applying homographies, multiple images can be combined to create a seamless panoramic view.
4. **Camera Calibration**: Homographies play a crucial role in the calibration process, helping to relate the 2D image plane to the 3D world coordinates.
5. **Object Tracking**: Homographies can assist in tracking objects across frames by mapping their positions in different views.
6. **Augmented Reality**: In augmented reality applications, homographies are used to overlay virtual objects onto real-world scenes accurately.
7. **Image Warping**: Homographies can be employed to warp images for various applications, including artistic effects or correcting lens distortion.
8. **Scene Reconstruction**: By using homographies, it is possible to reconstruct 3D scenes from multiple 2D images taken from different perspectives.

The transformations described in this section allow, given the knowledge of the camera orientation and intrinsic parameters, to derive the matrix $\mathbf{H}$ that determines the transformation. Conversely, by obtaining the homographic matrix through the association of points between the two images, it is possible to derive certain parameters that relate the views to each other.

It is indeed important to note that for all transformations involving a homography (change of viewpoint, perspective projection, IPM, and rectification), when the knowledge of the parameters necessary to generate the transformation is required, the representative matrices can still be implicitly derived by knowing how (at least) 4 points of the image are transformed (see section 8.5.1 for details).

The parameters obtained from the decomposition of the homographic matrix are derived from algebraic minimization. The maximum likelihood solution requires a nonlinear minimization but uses the result obtained from this phase as a starting point.

8.3.1 Perspective Mapping and Inverse Perspective Mapping

Using homography, it is possible to perform the transformation of *inverse perspective mapping* (or *bird eye view*) by simply inverting the matrix of the *perspective mapping*.

The homographic matrix $\mathbf{H} = \mathbf{P}_Z$ of the perspective mapping of a plane, *perspective mapping*, related to a constant plane z, where typically $z = 0$ is the ground, the most significant plane, can be derived quite simply as follows:

$$\mathbf{P}_Z = \mathbf{K} \cdot \mathbf{R}_Z \tag{8.27}$$

where $\mathbf{R}_Z$ is the rotation-translation matrix of a plane that can be expressed as

$$\mathbf{R}_Z = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & z\mathbf{r}_3 + \tilde{\mathbf{t}} \end{bmatrix} = \begin{bmatrix} r_0 & r_1 & r_2 z + \tilde{t}_x \\ r_3 & r_4 & r_5 z + \tilde{t}_y \\ r_6 & r_7 & r_8 z + \tilde{t}_z \end{bmatrix} \tag{8.28}$$

having defined the vector $\tilde{\mathbf{t}}$ as the translation expressed in camera coordinates, as shown in equation (8.13).

This matrix is very important and will be discussed extensively in section 8.5 on calibration.

The transformation (8.27), being a homography, is invertible. When it densely transforms all image points into world points, it is referred to as *Inverse Perspective Mapping*, whereas when it transforms all world points into image points, it is denoted as *Perspective Mapping*. In both cases, only the plane z is correctly projected.

It is always interesting to note how even the simplest model of the pin-hole camera with 9 parameters (6 extrinsic and 3 intrinsic) cannot be derived from the 8 parameters constraints provided by the homography matrix. However, knowing the intrinsic parameters allows for an estimation of the camera's rotation and position (section 8.5), as the equation 8.27 becomes invertible:

$$\mathbf{R}_Z = \mathbf{K}^{-1}\mathbf{H} \tag{8.29}$$

8.3.2 Vanishing Point and Horizon Line

Due to the limitation to plane transformations, it is possible to easily calculate the limits of the coordinates x and y through the transformation (8.27) as follows:

$$\begin{aligned} \lim_{x\to\infty} \mathbf{H}(x, y, 1)^\top &= \left(\tfrac{h_0}{h_6}, \tfrac{h_3}{h_6}\right) \\ \lim_{y\to\infty} \mathbf{H}(x, y, 1)^\top &= \left(\tfrac{h_1}{h_7}, \tfrac{h_4}{h_7}\right) \end{aligned} \tag{8.30}$$

These limits are the *vanishing points* (see section 8.2.2) of the image.

8.3.3 Change of Perspective

The generic equation that relates the image points between two generic viewpoints can be written as

$$\begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} \equiv \mathbf{K}_2 \left(\mathbf{R}\mathbf{K}_1^{-1} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} + \mathbf{t} \right) \tag{8.31}$$

where $\mathbf{t} = \mathbf{t}_1 - \mathbf{t}_2$ is the vector connecting the two *pin-holes* and $\mathbf{R}$ is the relative orientation between the two views as indicated in section 1.9. A more detailed discussion is provided in chapter 9 on stereoscopy.

In general, it is not possible to transform a view generated by one camera into the view generated by another. This is only feasible if one aims to correctly remap points on a specific plane or when the cameras share the same *pin-hole*.

The second case will be discussed in the next section. In the first case, it is possible to remap points from one view to another by utilizing a combination of a *Perspective Mapping* followed by an *Inverse Perspective Mapping*, under the assumption that the observed scene consists solely of a plane (for example, the ground). The image points are projected into world coordinates on a camera *1* and then reprojected back into image coordinates on a second camera *2* with different intrinsic and extrinsic parameters. Since a plane is always being reprojected, the composition of this transformation remains a homography:

$$\mathbf{H} = \mathbf{H}_2 \cdot \mathbf{H}_1^{-1} \tag{8.32}$$

Homographic transformations indeed combine through simple matrix multiplication. Expanding equation (8.32) with (8.27) yields:

$$\mathbf{H} = \mathbf{K}_2 \cdot \mathbf{R}_{Z2} \cdot \mathbf{R}_{Z1}^{-1} \cdot \mathbf{K}_1^{-1} \tag{8.33}$$

From a theoretical standpoint, the necessity to enforce a constant plane z only affects the situation if the translation vector changes. In cases where the translation vector is modified between the two views and there are points not belonging to the indicated plane, an incorrect remapping occurs between the two views (the homographic transformation is no longer respected). The transformation (8.32) can also be utilized to identify vertical obstacles within techniques such as *Ground Plane Stereo* and *Motion Stereo*.

This homographic matrix can be generalized by knowing the elements of the transformation between the two views $(\mathbf{R}, \mathbf{t})$ and the equation of the plane $(\mathbf{n}, d)$, where $\hat{\mathbf{n}}$ is the normal to the plane and d is the distance from the first camera to the plane itself.

In this case, a point $\mathbf{x}_1$ from the first view that lies on the plane satisfies the equation

$$\hat{\mathbf{n}}^\top \mathbf{x}_1 = d \tag{8.34}$$

, and this point is related to the same point, but viewed from the second camera in accordance with the equation

$$\mathbf{x}_2 = \mathbf{R}\mathbf{x}_1 + \mathbf{t} \tag{8.35}$$

.

By combining these two equations, the homographic constraint is obtained:

$$\mathbf{H} = \mathbf{K}_2 \left(\mathbf{R} + \frac{1}{d} \mathbf{t} \hat{\mathbf{n}}^\top \right) \mathbf{K}_1^{-1} \tag{8.36}$$

.

A homography can always be decomposed into $\left[\mathbf{R}, \frac{1}{d}\mathbf{t}, \hat{\mathbf{n}}\right]$ (there are 4 possible decompositions, and the one that satisfies the input points must be chosen).

8.3.4 Rectification

The case of pure rotation is a special scenario: a camera rotating around its optical center captures images of a 3D scene as if the scene were projected onto a plane infinitely far from the pin-hole.

In the case where $\mathbf{t} = 0$, that is, the coordinates of the two pinholes of the two views are coincident $\mathbf{t}_1 = \mathbf{t}_2$, the transformation (8.31) reduces in dimension and results in an equation compatible with a homography, and consequently valid for any point in the image, regardless of the presence of a dominant plane. Therefore, when the pinhole is common between the two views (thus representing a pure rotation or modification of intrinsic parameters), it is possible to achieve a perfect transformation for all points in the image. This process of projecting points from one camera to another by modifying intrinsic parameters and rotation is referred to as rectification.

To rectify an image, that is, to generate a dense image 1 from the points of image 2, it is necessary to use the homography matrix

$$\mathbf{H}_{1,2} = \mathbf{K}_2 \mathbf{R}_2 \mathbf{R}_1^{-1} \mathbf{K}_1^{-1} \tag{8.37}$$

which allows us to derive all the points of image 1 from the points of image 2. Specifically, for each pixel (u_1, v_1) of the image to be generated, the homographic transformation $\mathbf{H}$ is applied, and the point (u_2, v_2) of the source image is obtained from which to copy the pixel value.

Through the transformation (8.37), it is possible to convert an image captured by a camera with parameters $(\mathbf{K}_2, \mathbf{R}_2)$ into an image of a virtual camera with parameters $(\mathbf{K}_1, \mathbf{R}_1)$.

The discussion applies to all homographies, a method for obtaining the matrix $\mathbf{H}$ without knowledge of the intrinsic and extrinsic parameters but solely through correspondences between the views of the two cameras is presented in section 8.5.1. The homography can then be factored to retrieve the parameters that generated it.

8.4 Inverse Perspective Mapping

In the previous sections, we have seen examples of inverse perspective: the ability to derive the 3D point given a 2D image point and the knowledge of a constraint in the world on whose surface the point lies. It is always possible to create a system between the optical ray (8.16) and a manifold in $\mathbb{R}^3$:

$$\begin{cases} \mathbf{x} = \lambda \mathbf{V}\mathbf{p} + \mathbf{t} \\ f(\mathbf{x}) = 0 \end{cases} \tag{8.38}$$

denoting it as $\mathbf{V} = \mathbf{R}^{-1}\mathbf{K}$. This same formulation is used in computer graphics to refer to *RayTracing* techniques. In this section, we will generalize various cases.

Intersection of Optical Ray and Plane A generic plane in $\mathbb{R}^3$ expressed in the form

$$\hat{\mathbf{n}} \cdot \mathbf{x} + q = 0 \tag{8.39}$$

serves as a constraint to allow the intersection between the optical ray (8.16) and the plane (8.39). The system (8.38) is linear and can be solved for λ and, by substituting λ into the first equation, determine the 3D point. It is also possible to create a linear application associated with the intersection of a plane and a line in the form $\mathbf{x} \equiv \mathbf{A}_{4\times3}\mathbf{p}$, having defined

$$\mathbf{A}_{4\times3} = \begin{bmatrix} \left(\hat{\mathbf{n}}^\top \mathbf{t} - q\right)\mathbf{I} + \mathbf{t}\hat{\mathbf{n}}^\top \\ \hat{\mathbf{n}}^\top \end{bmatrix} \mathbf{V} \tag{8.40}$$

.

Intersection of an Optical Ray and a Sphere The manifold has the equation

$$\|\mathbf{x} - \mathbf{x}_0\|^2 = r^2 \tag{8.41}$$

which, when combined with the system (8.38), allows us to obtain

$$\lambda^2 \|\mathbf{Vp}\|^2 + 2\lambda (\mathbf{Vp}) \cdot (\mathbf{t} - \mathbf{x}_0) + \|\mathbf{t} - \mathbf{x}_0\|^2 = r^2 \tag{8.42}$$

The solution of the quadratic equation can therefore have 0 (no intersection), 1 (the optical ray is tangent to the sphere), or 2 roots (the optical ray intersects the sphere).

8.5 Calibration

The camera calibration phase allows for the extraction of (some or all) parameters that enable the pin-hole model to project points from world coordinates to camera coordinates. In English, camera calibration, which involves deriving intrinsic and/or extrinsic parameters, is referred to as *Camera resectioning*, as the concept of *Camera Calibration* can also refer to the problem of the photometric calibration of the system.

Calibration techniques can be divided into two categories depending on which pin-hole camera model is to be derived:

implicit where the elements of the projection matrix **P** or the homography matrix **H** are extracted in order to project points from one coordinate system to another, disregarding the internal structure of the sensor;

explicit where the physical parameters of the system involved in the perspective projection are extracted.

Implicit calibration is usually a faster process and, with a sufficient number of points, represents reality quite well. However, as we will see shortly, the linear version that minimizes an algebraic quantity is not the maximum likelihood estimator. Implicit calibration also overlooks some non-linearities that are always present in physical systems.

Explicit calibration, on the other hand, not only accurately represents the model with the minimum number of parameters but also provides greater flexibility in the use of the obtained parameters. This allows for operations on images or for dynamically varying certain parameters of the system, and it enables the implementation of the maximum likelihood estimator.

To enable the application of the calibration techniques presented in this section, it is essential to establish constraints between the involved projective spaces, such as image coordinates and their corresponding world coordinates. Through these relationships, it is possible to derive the parameters that represent the projective model being utilized.

The boundary that separates implicit calibration from explicit calibration sometimes tends to blur: under certain conditions, it is possible to transition from one mode to the other.

- With *Direct Linear Transformation*, as discussed in section 8.5.1, it is possible to implicitly calibrate the system by knowing the positions of points in world coordinates and image coordinates, deriving the projection matrix **P**, or the projection matrix of a single plane **H**, without explicitly stating any camera parameters. Subsequently, using equation (8.29), one can derive the matrix [**Rt**] of the extrinsic parameters, although information about the intrinsic parameters is still required.

- It has been previously mentioned (see section 8.2.2) how it is possible to derive the rotation matrix given the knowledge of the intrinsic parameter matrix and the positions of the vanishing points.

- If the rotation matrix **R** is known, it is possible to explicitly obtain the values of the angles that generated it (there may be multiple solutions in this case).

- If one can obtain the intrinsic parameter matrix **K**, it is straightforward to explicitly derive the intrinsic parameters of the camera.

- Zhang, in section 8.5.4, proposes a method to derive the intrinsic parameters of the camera if the relative positions of points belonging to the same plane are known, observed from multiple viewpoints.

8.5.1 Implicit Calibration

The fundamental idea of the *Direct Linear Transformation* proposed by Abdel-Aziz and Karara [AAK71] allows for the direct calculation of the coefficients of the matrices (8.47), (8.50), or the matrix (8.15), completely disregarding the parameters and the structure of the perspective transformation model. This article also presents an approach to solve overdetermined problems using the Pseudoinverse technique.

Given the system (8.15), it is necessary to derive the 12 parameters of the projection matrix **P** to achieve an *implicit* calibration of the system, where the internal parameters (ranging from 9 to 11 depending on the model) that generated

the elements of the matrix itself are unknown. This representation of the *pin-hole* camera is, of course, ideal (without non-linearities from the model).

The perspective function written in implicit form is

$$\begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \mathbf{P} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} = \begin{bmatrix} p_0 & p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 & p_7 \\ p_8 & p_9 & p_{10} & p_{11} \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \tag{8.43}$$

where the elements $p_0 \dots p_{11}$ are arranged in *row-major* order. It is possible to rearrange the system (8.43) to obtain 2 pairs of linear constraints for each point for which both image and world coordinates are known:

$$\begin{bmatrix} x_i & y_i & z_i & 1 & 0 & 0 & 0 & 0 & -u_i x_i & -u_i y_i & -u_i z_i & -u_i \\ 0 & 0 & 0 & 0 & x_i & y_i & z_i & 1 & -v_i x_i & -v_i y_i & -v_i z_i & -v_i \end{bmatrix} \begin{pmatrix} p_0 \\ \vdots \\ p_{11} \end{pmatrix} = 0 \tag{8.44}$$

This technique is called DLT (*direct linear transformation*). Since each point provides 2 constraints, at least 6 points that are not linearly dependent, meaning they do not lie on the same plane or the same line, are required to obtain these 12 parameters.

Being a homogeneous system, its solution will be the null subspace of $\mathbb{R}^{12}$, the kernel of the matrix of known terms. For this reason, the matrix $\mathbf{P}$ is known up to a multiplicative factor, resulting in only 11 degrees of freedom (even fewer when considering that a modern camera typically has only 3-4 intrinsic parameters and 6 extrinsic parameters).

Having rearranged the system, the propagation of noise across the points is no longer linear, and this solution does not satisfy the maximum likelihood criterion. The matrix $\mathbf{P}$ obtained through this procedure, although it conceals the internal structure of the sensor, allows for the projection of a point from world coordinates to image coordinates and enables the derivation of the line that underlies such a point in the world from a point in image coordinates.

The result is generally unstable when using only 6 points; therefore, the estimation is typically performed by processing more points than the minimum required. Techniques such as the pseudoinverse are employed to determine a solution that minimizes measurement errors.

Generalization in the Case of Homogeneous Coordinates

The equation 8.43 can be generalized to the case of an "image" point in homogeneous coordinates (u_i, v_i, w_i):

$$\begin{pmatrix} u_i \\ v_i \\ w_i \end{pmatrix} \equiv \mathbf{P} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \tag{8.45}$$

The problem is the same as previously encountered; the homogeneous solution exists, and the homogeneous resolutive equation (8.44) generalizes to

$$\begin{bmatrix} w_i x_i & w_i y_i & w_i z_i & w_i & 0 & 0 & 0 & 0 & -u_i x_i & -u_i y_i & -u_i z_i & -u_i \\ 0 & 0 & 0 & 0 & w_i x_i & w_i y_i & w_i z_i & w_i & -v_i x_i & -v_i y_i & -v_i z_i & -v_i \end{bmatrix} \begin{pmatrix} p_0 \\ \vdots \\ p_{11} \end{pmatrix} = 0 \tag{8.46}$$

for every i.

This formulation is useful when the projective model does not adhere to the pinhole model, but it is still possible to derive the "camera" coordinates of the optical rays corresponding to the pixel, which are therefore available in homogeneous format.

DLT Calculation of the Homography

Typically, to reduce the number of elements in the matrix $\mathbf{P}$, one can impose the constraint that all points involved in the calibration process lie on a specific plane (for example, the ground). This means setting the condition $z_i = 0 \; \forall i$, which implies the elimination of a column (related to the axis z) from the matrix. which reduces to the size 3×3, becomes invertible, and can be defined as homographic (see section 1.10).

We therefore define the matrix $\mathbf{H} = \mathbf{P}_Z$ (see (8.27)) as

$$\lambda \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \mathbf{H} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \tag{8.47}$$

. As discussed in section 8.3, this matrix is very useful because it allows, among other things, to remove the perspective from the image, synthesizing a fronto-parallel view of the plane, through a transformation known as *orthogonal rectification*,

bird eye view, or *inverse perspective mapping*. This transformation is applicable whether one aims to eliminate perspective (*perspective mapping* or *inverse perspective mapping*), to reproject a plane between two images (*ground plane stereo*), or to generate an image with different parameters (rectification, panoramic images) by utilizing a virtual plane.

As in the previous case, it is possible to transform the nonlinear relationship (8.47) in order to obtain linear constraints:

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -u_i x_i & -u_i y_i & -u_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -v_i x_i & -v_i y_i & -v_i \end{bmatrix} \begin{pmatrix} h_0 \\ \vdots \\ h_8 \end{pmatrix} = 0 \tag{8.48}$$

Since this matrix is also defined up to a multiplicative factor, it has only 8 degrees of freedom, and therefore an additional constraint can be imposed.

If you have a sufficiently modern linear systems solver, the additional constraint $|\mathbf{H}| = 1$ is automatically satisfied during the computation of the kernel of the matrix of known terms (QR factorization or SVD decomposition).

Another simpler and more intuitive method consists of imposing an additional constraint $h_8 = 1$: in this way, instead of solving a homogeneous system, one can solve a traditional linear problem. The system (8.47) can also be rearranged to obtain linear constraints in the form:

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i u_i & -y_i u_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_i v_i & -y_i v_i \end{bmatrix} \begin{pmatrix} h_0 \\ \vdots \\ h_7 \end{pmatrix} = \begin{pmatrix} u_i \\ v_i \end{pmatrix} \tag{8.49}$$

. This is a (non-homogeneous) system of two equations in 8 unknowns $h_0 \dots h_7$, and each point, for which both the world position on a plane and the position in the image are known, provides 2 constraints.

However, imposing $h_8 = 1$ implies that the point $(0, 0)$ cannot be a singularity of the image (e.g., the horizon line), and in general, it is not an optimal choice in terms of solution accuracy, as previously discussed.

It is important to note that the solution is heavily dependent on the chosen normalization. The choice $|H| = c$ can be referred to as *standard least-squares*.

In both cases, at least 4 points are required to obtain a homography $\mathbf{H}$, and each additional point allows for a solution with a lower error. These systems, when overdetermined, can be solved using the pseudoinverse method **??**.

The matrix $\mathbf{H}$ is defined by 4 intrinsic parameters and 6 extrinsic parameters. The separation of intrinsic parameters from extrinsic parameters suggests that these parameters should be extracted independently to strengthen the calibration process. After all, intrinsic parameters can be determined with a certain degree of accuracy offline and are applicable to all possible camera placements (see also 8.5.4).

Let us define the matrix $\mathbf{R}_Z$ (see (8.28)) as

$$\lambda \begin{pmatrix} \tilde{u_i} \\ \tilde{v_i} \\ 1 \end{pmatrix} = \begin{bmatrix} r_0 & r_1 & p_x \\ r_3 & r_4 & p_y \\ r_6 & r_7 & p_z \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \mathbf{R}_Z \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \tag{8.50}$$

, where $(\tilde{u_i}, \tilde{v_i})$ denotes the so-called normalized image coordinates (homogeneous coordinates of the point $(\tilde{x}_i, \tilde{y}_i, \tilde{z}_i)^\top$ in camera coordinates).

The matrix $\mathbf{H}$ is defined up to a scaling factor, while $\mathbf{R}_Z$ allows for the definition of the scale since it still has two orthonormal columns. The knowledge of the two columns of the rotation matrix enables the derivation of the third column, and therefore this calibration becomes valid for points even outside the plane $z = 0$.

As done previously, a non-linear system consisting of 3 homogeneous equations, when appropriately rearranged, yields two linear constraints: (Abdel-Aziz and Karara [AAK71]). It is therefore possible to construct a system of $2 \times N$ equations for all N control points, in order to solve for the 9 unknowns. The matrix is defined up to a multiplicative factor, but in this case, the internal structure of the matrix $\mathbf{R}_Z$ can be helpful in deriving the extrinsic parameters (see section 8.5.3). In fact, the two columns of the matrix must be orthonormal:

$$\begin{aligned} r_0^2 + r_3^2 + r_6^2 &= 1 \\ r_1^2 + r_4^2 + r_7^2 &= 1 \\ r_0 r_1 + r_3 r_4 + r_6 r_7 &= 0 \end{aligned} \tag{8.51}$$

These additional nonlinear constraints arise from the fact that such a matrix is explicitly defined by only 6 parameters (3 rotations and the translation).

Geometric Representation

The equations (8.44) and (8.48) can also be derived from purely geometric considerations since the image and camera vectors must be parallel (the factor $lamba_i$ is purely multiplicative and at most affects the vector through an affine transformation):

$$\mathbf{p} \times \mathbf{Px} = \mathbf{0} \qquad \mathbf{m}' \times \mathbf{Hm} = \mathbf{0} \tag{8.52}$$

This compact formulation is what is commonly referred to as *DLT* [HZ04] and applies to all those linear transformations known up to a multiplicative factor to transform this problem into a homogeneous problem.

8.5.2 MLE Calculation of the Homography

From a computational perspective, the equation (8.48) is ill-conditioned since each column represents a quantity with a different order of magnitude. To obtain a correct linear solution, a prior normalization phase is required. Hartley and Zisserman [HZ04] emphasize that normalization in the DLT is an essential step and cannot be considered purely optional.

The calculation of the homography in equation (8.49) has the drawback of not accounting for measurement errors on the points. In fact, the SVD decomposition minimizes something that, by pure coincidence, resembles the error on the known term (which is not the case in equation (8.48)), and in any case, it is not possible to assess the error on the parameter matrix. In this specific case, where a purely mathematical error is minimized using least squares without a corresponding geometric interpretation, it is referred to as *algebraic least squares* (*ALS*).

Since the DLT minimizes an algebraic error rather than a geometric one, even though from a computational standpoint the normalized DLT is superior, it may yield poorer results from the perspective of geometrically fitting the data. The version of the system (8.48) normalized by least squares is referred to as *normalized algebraic least squares* (*NALS*).

To overcome the limitations of algebraic error calculation, it is necessary to return to the original problem and not attempt to transform it into a linear problem, but rather to solve it, for instance, iteratively, using a nonlinear minimizer.

If the noise is present only in one of the two images, an appropriate cost function, with geometric significance, is the Euclidean distance between the measured points and the transformed points. This is commonly referred to as the transfer error and minimizes a nonlinear cost function of the form

$$\arg\min_{\mathbf{H}} \sum \|\mathbf{m}'_i - \mathbf{H}\mathbf{m}_i\|^2 \tag{8.53}$$

where $\mathbf{m}'_i$ is the image point affected by white Gaussian noise, while $\mathbf{m}_i$ is a perfectly known point. In this case, the function that minimizes the geometric error is also the one that represents the best estimate of the result from a Bayesian perspective (Maximum Likelihood Estimator or MLE).

However, when both data are affected by noise, the cost function (8.53) is not optimal. The simplest way to extend the previous solution is to attempt to minimize both the direct transfer error and the inverse transfer error (*symmetric transfer error*):

$$\arg\min_{\mathbf{H}} \sum \|\mathbf{m}'_i - \mathbf{H}\mathbf{m}_i\|^2 + \|\mathbf{m}_i - \mathbf{H}^{-1}\mathbf{m}'_i\|^2 \tag{8.54}$$

In this way, both contributions are taken into account in the solution of the problem.

This, however, is not yet the optimal solution, at least from a statistical standpoint. A maximum likelihood estimator must indeed properly account for the noise in both datasets when present (what Hartley and Zisserman refer to as the *Gold Standard*). The alternative solution, which is in fact the more accurate one, consists of minimizing the Reprojection error.

This solution significantly increases the size of the problem, as it aims to identify the optimal points that are not affected by noise $\hat{\mathbf{m}}_i$ and $\hat{\mathbf{m}}'_i$:

$$\arg\min_{\mathbf{H}} \sum \|\mathbf{m}'_i - \hat{\mathbf{m}}'_i\|^2 + \|\mathbf{m}_i - \hat{\mathbf{m}}_i\|^2 \tag{8.55}$$

under the constraint $\hat{\mathbf{m}'}_i = \mathbf{H}\hat{\mathbf{m}}_i$.

In the even more general case with covariance noise measured for each individual point, the correct metric is the Mahalanobis distance (see section 2.4):

$$\|\mathbf{m} - \hat{\mathbf{m}}\|^2_\Gamma = (\mathbf{m} - \hat{\mathbf{m}})^\top \Gamma^{-1} (\mathbf{m} - \hat{\mathbf{m}}) \tag{8.56}$$

In the case where the noise per point is constant, the previous expression simplifies to the more intuitive Euclidean distance.

Since it is a nonlinear minimization problem, an initial solution is required to start the search for the minimum that satisfies the cost equation: the linear solution remains useful and is employed as an initial reference to identify a minimum under a different metric.

The MLE estimator requires the use of an additional auxiliary variable $\hat{\mathbf{m}}_i$ for each point and iterative techniques to solve the problem. It is possible to use the Sampson error as an approximation of the geometric distance, as discussed in section 3.3.7. The homographic constraint (1.75) that relates the points of the two images can be expressed in the form of a two-dimensional manifold $\mathcal{V}_H$

$$\begin{aligned} h_0u_1 + h_1v_1 + h_2 - h_6u_1u_2 - h_7v_1u_2 - h_8u_2 = 0 \\ h_3u_1 + h_4v_1 + h_5 - h_6u_1v_2 - h_7v_1v_2 - h_8v_2 = 0 \end{aligned} \tag{8.57}$$

from which the Jacobian

$$\mathbf{J}_\mathcal{V} = \begin{bmatrix} h_0 - h_6u_2 & h_1 - h_7u_2 & -h_6u_1 - h_7v_1 - h_8 & 0 \\ h_3 - h_6v_2 & h_4 - h_7v_2 & 0 & -h_6u_1 - h_7v_1 - h_8 \end{bmatrix} \tag{8.58}$$

can be derived for use in the calculation of the Sampson distance [CPS05].

Error Propagation in Homography Calculation

In the case of an error on a single image, to calculate how the error propagates through the matrix $\mathbf{H}$, it is necessary to compute the Jacobian of the cost function (8.53). By specifying the homographic transformation, one obtains [HZ04]

$$\mathbf{J}_i = \frac{\partial r}{\partial \mathbf{h}} = \frac{1}{\hat{w}'} \begin{bmatrix} \mathbf{m}_i^\top & 0 & -\hat{u}_i' \mathbf{m}_i^\top / \hat{w}' \\ 0 & \mathbf{m}_i^\top & -\hat{v}_i' \mathbf{m}_i^\top / \hat{w}' \end{bmatrix} \tag{8.59}$$

with $\mathbf{m}_i = (u_i, v_i, 1)^\top$ and $\hat{\mathbf{m}}_i' = (\hat{u}_i', \hat{v}_i', \hat{w}_i')^\top = \mathbf{H}\mathbf{m}_i$.

Using the theory presented in section 3.5, it is possible to compute the covariance matrix of the homography parameters given the covariance on the points $\mathbf{m}_i'$. Since the total covariance matrix $\mathbf{\Sigma}$ of the noise on the individual points will be very sparse, as different points are assumed to have independent noise, the covariance $\mathbf{\Sigma}_h$ on the obtained parameters is given by [HZ04]

$$\mathbf{\Sigma}_h = \left(\sum \mathbf{J}_i^\top \mathbf{\Sigma}_i^{-1} \mathbf{J}_i \right)^+ \tag{8.60}$$

with $\mathbf{\Sigma}_i$ being the covariance matrix of the noise on the individual point.

8.5.3 Calibration according to Tsai

Camera calibration for various applications requires a comprehensive understanding of both intrinsic and extrinsic parameters. One of the most widely used methods is undoubtedly that of Tsai [Tsa87] from 1985. The merit of Tsai's work was to bring order to the previously discussed state of the art and to provide a unique and accepted nomenclature for the camera parameters as presented here.

The Tsai camera model is based on the perspective projection of the pin-hole camera and consists (in its classical form) of 11 parameters:

f Focal length of the camera

k First-order radial distortion coefficient

Cx,Cy Coordinates of the optical center of the lens

Sx A horizontal scale factor

Rx, Ry, Rz Rotation angles for the transformation between world coordinates and camera coordinates

Tx, Ty, Tz Translation vector for the transformation between world coordinates and camera coordinates

Tsai conducts a comprehensive analysis of all the techniques developed thus far for calibration, and ultimately proposes a modular system, where each module allows for the extraction of a series of these parameters.

It mainly points out that if the camera is distorted but the *principal point* is aligned with the *center of distortion*, the following relationships hold:

$$\frac{u_d}{v_d} = \frac{u_u}{v_u} \tag{8.61}$$

Consequently, it is possible to establish constraints under this condition using the distorted coordinates rather than the undistorted ones. This method is therefore also referred to as the *radial alignment constraint* (RAC).

Initially, using the camera parameters provided by the manufacturer, calculate the translation and rotation vector from a grid with coplanar points $z_i = 0$ of known coordinates, leveraging the constraint

$$(r_0 x_i + r_1 y_i + \tilde{t}_x) u_i' = (r_3 x_i + r_4 y_i + \tilde{t}_y) v_i' \tag{8.62}$$

with (u_i', v_i') normalized camera coordinates using the parameters of the camera and lens supplied by the manufacturer. From this constraint, a large overdetermined linear system of the type can be created.

$$\begin{bmatrix} x_i u_i' & y_i u_i' & u_i' & -v_i' x_i & -v_i' y_i \end{bmatrix} \begin{bmatrix} \frac{r_0}{\tilde{t}_y} \\ \frac{r_1}{\tilde{t}_y} \\ \frac{\tilde{t}_x}{\tilde{t}_y} \\ \frac{r_3}{\tilde{t}_y} \\ \frac{r_4}{\tilde{t}_y} \end{bmatrix} = v_i' \tag{8.63}$$

having set $\tilde{t}_y \neq 0$ (that is, the grid should not intersect the optical axis). The remaining parameters of the matrix $\mathbf{R}$ are obtained using equation (8.20).

Subsequently, it proceeds to derive the correct intrinsic parameters using these values for the rotation and translation matrix.

8.5.4 Calibration using the Sturm-Maybank-Zhang method

Zhang [Zha99] and simultaneously Sturm and Maybank [SM99] identify a method to derive a linear equation for obtaining the camera parameters, while also updating the calibration techniques (which remain valid but are now somewhat outdated, dating back to the 1980s) primarily developed by Tsai [Tsa87] and others [WM94].

This technique leverages the computation of various homographic matrices $\mathbf{H}$ obtained from the observation of a plane (for example, a calibration grid with equidistant *markers*) and seeks to explicitly derive the intrinsic parameters of the camera from these. As previously discussed, the matrix $\mathbf{H}$, the homographic transformation of a plane, possesses 8 degrees of freedom, but it is not possible to directly derive the 10 explicit parameters that generated it. Methods for obtaining the homographic matrix given correspondences between image points and points on the plane are discussed in section 8.5.1.

The matrix $\mathbf{H}$ and in particular the equation (8.27) can be expressed as

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = \lambda \mathbf{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \tag{8.64}$$

where λ is indicated to highlight the presence of an unknown multiplicative factor in the calculation of the homographic matrix. Let us focus on the part of the rotation matrix formed by the column vectors $\mathbf{r}_1$ and $\mathbf{r}_2$, which are orthonormal to each other.

Despite the presence of the factor λ, it is indeed possible to express relationships based on the orthogonality between the vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ in order to enforce the following two constraints:

$$\begin{aligned} \mathbf{h}_1^\top \mathbf{W} \mathbf{h}_2 &= 0 \\ \mathbf{h}_1^\top \mathbf{W} \mathbf{h}_1 &= \mathbf{h}_2^\top \mathbf{W} \mathbf{h}_2 \end{aligned} \tag{8.65}$$

having defined $\mathbf{W}$, neglecting the *skew* for simplicity, as

$$\mathbf{W} = (\mathbf{K}^{-1})^\top \mathbf{K}^{-1} = \begin{bmatrix} \frac{1}{k_u^2} & 0 & -\frac{u_0}{k_u^2} \\ 0 & \frac{1}{k_v^2} & -\frac{v_0}{k_v^2} \\ -\frac{u_0}{k_u^2} & -\frac{v_0}{k_v^2} & \frac{u_0^2}{k_u^2} + \frac{v_0^2}{k_v^2} + 1 \end{bmatrix} \tag{8.66}$$

a symmetric matrix. This equation represents the equation of a conic and is indeed the equation of the "absolute conic" [LF97].

The 4 (or 5 unknowns, not neglecting the *skew*) of the matrix $\mathbf{W}$ under the 2 constraints (8.65) can be solved using at least 2 (or 3) different planes, that is, matrices $\mathbf{H}$ whose columns are not linearly dependent on each other.

Once the matrix $\mathbf{W}$ is obtained, the original matrix can be determined using the Cholesky decomposition. Alternatively, Zhang provides the equations to directly obtain the intrinsic parameters of the camera from the matrix $\mathbf{W}$. It is indeed possible to transform $\mathbf{h}_i^\top \mathbf{W} \mathbf{h}_j = \mathbf{v}_{ij}^\top \mathbf{w}$ using appropriate values of the vector $\mathbf{v}_{ij}$ and with $\mathbf{w}$, a vector to be determined, using the non-zero values of the upper triangular matrix of $\mathbf{W}$. In this way, the system of equations (8.65) is transformed into the solution of a homogeneous linear system in $\mathbf{w}$.

Once the intrinsic parameters and the matrix $\mathbf{K}$ are determined, for each homographic matrix $\mathbf{H}$ used in the optimization phase, it is possible to estimate the rotation and translation:

$$\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} = \lambda \mathbf{K}^{-1} \mathbf{H} \tag{8.67}$$

The columns $\mathbf{r}_1$ and $\mathbf{r}_2$ are typically sufficient to derive the rotation angles. From each grid, it is possible to derive all the extrinsic parameters and measure the reprojection error in this way.

The system as a whole is still ill-conditioned, and it is challenging to arrive at a stable solution after repeated trials. However, the values obtained through this linear technique serve as a starting point in a phase of *Maximum Likelihood Estimation* to minimize reprojection errors (section 8.5.6).

One note: Zhang in his article equates the *Principal Point* with the distortion center, which is generally inaccurate.

8.5.5 The P3P and PnP Problems

The DLT solution is numerically unstable, while non-linear least squares minimization techniques are iterative methods that require a starting point for minimization that is close to the optimum (for example, the solution proposed by DLT). Currently, there are no definitive and elegant solutions to these problems, but several possible approaches exist, all of which are superior to the DLT approach. These techniques fall under the names of PnP (*perspective-n-point*) and P3P (*perspective-3-point*) in the theoretical optimal case, as only three points are needed to determine the pose of the video sensor.

Among the interesting approaches to mention are those described in [LFNP09] and in [HR11]. These are relatively lengthy and complex methods, but they provide pose estimates that are very close to those obtained through iterative techniques, which still suffer from local minimum issues.

8.5.6 Maximum Likelihood Estimation

When performing the calibration phase to relate image points to world points, it is easy to assume that the point in world coordinates has high precision, while the value of the point in image coordinates is known only up to Gaussian noise with a mean of zero.

The techniques discussed previously (in particular, the DLT) are mere approximations of the optimal solution and should be used as a starting point for nonlinear minimization. To obtain the optimal solution, it is necessary to minimize the sum of the squared errors between the measured position affected by noise and the position predicted by the model. The maximum likelihood estimator minimizes an objective function of the form

$$\min_{\beta} \|\mathbf{p}_i - f(\mathbf{x}_i, \beta)\|^2 \tag{8.68}$$

where $\mathbf{x}_i$ is a point in world coordinates and $\mathbf{p}_i$ is the corresponding point in image coordinates, affected by observation noise due to the point detection algorithm and the spatial quantization in pixels that any sensor applies to optical rays. β are the parameters of the perspective projection to be estimated, preferably the explicit ones considering the distortion of the optics.

8.6 Fish-Eye Camera Model

To generalize the concept of a pinhole camera, multiple classes of camera models can be introduced. It is important to note that real cameras are never ideal cameras that perfectly conform to a specific optical model. Therefore, there are various basic models that attempt to approximate the lens equation, to which several distortion terms must still be added.

In particular, *fish-eye* lenses are characterized by a dominant barrel distortion, which allows for achieving very high field of view angles, up to 180 degrees or more, while maintaining the same focal length.

Let ϑ be the incident angle of the optical ray at the camera point (x, y, z) with respect to the optical axis $(0, 0, 1)$. This angle is given by $\vartheta = \arctan r_i$, which is derived from

$$r_i = \sqrt{x_1^2 + y_1^2} = \frac{\sqrt{x^2 + y^2}}{z} \tag{8.69}$$

, where indeed $x_1 = x/z$ and $y_1 = y/z$. Note that the equality holds as

$$\vartheta = \arctan \frac{\sqrt{x^2 + y^2}}{z} = \arccos \frac{z}{\sqrt{x^2 + y^2 + z^2}} \tag{8.70}$$

.

The idea is to consider all camera models as a manipulation of the incident angle ϑ into an angle ϑ', or it is often referred to as a transformation $r(\vartheta) = f\vartheta'$ that converts the angle of the incident ray into a ray in pixel coordinates.

The various fish-eye lenses follow slightly different equations, among which we can highlight the ideal models (thus without distortion):

- linear: $r = f\vartheta$
- orthographic: $r = f \sin \vartheta$
- constant solid angle: $r = 2f \sin(\vartheta/2)$
- stereographic: $r = 2f \tan(\vartheta/2)$.

The pin-hole model can be viewed as a particular case since the ratio between the incident light angle ϑ and the pixel coordinates follows the rule $r = f \tan \vartheta$.

Having absorbed in r also the focal length (thus it is a ray in pixels), the equations above allow us to transform the angle of incident light on the optics into a ray projected onto the sensor:

$$u = \frac{x}{\sqrt{x^2 + y^2}} r + u_0 \qquad v = \frac{y}{\sqrt{x^2 + y^2}} r + v_0 \tag{8.71}$$

since the phase angle remains constant between the incident ray and the projected ray in the absence of tangential distortion.

The class of ideal models above does not account for potential nonlinearities in optics. The Kannala-Brandt model [KHB09] generalizes these equations by parameterizing a generic *fish-eye* lens in the term $r(\vartheta)$ using the classical Taylor series expansion, which allows for the incorporation of both the various lens models presented and any distortions introduced by the optics.

I recall that even in fish-eye cameras, all optical rays converge at the same point (therefore, the rays always pass through a pin-hole). However, all the equations that can be formulated (and several will be seen in the next chapter) in linear form using homogeneous coordinates with a fish-eye model as listed here can no longer be applied. We must transition to non-linear equations, even in homogeneous coordinates.

Chapter 9

Multi-Camera Vision

This chapter generally discusses algorithms that involve the analysis of images from multiple cameras, with particular emphasis on the case of stereoscopic vision.

Stereoscopic vision (stereopsis) is the process through which it is possible to estimate distances and positions of objects observed by two visual sensors, and through this information, reconstruct the observed scene. This concept can be easily extended to the case where the scene is observed not by two but by multiple cameras (multiple view geometry).

These views can be temporally coincident (for instance, in the case of a pair of cameras that form a stereo camera) or they may observe the scene from different points in space and time, as occurs when processing images from the same camera that is moving through space (*motion stereo*, *structure from motion*).

Stereoscopic analysis can be primarily implemented through two techniques:

- *Feature Matching*, where distinctive points between two images are compared without constraints, except for those that will be shown later, allowing the identification of corresponding pairs of points but resulting in a *sparse* reconstruction of the scene;
- *Rectified Stereo*, where points between images from aligned cameras (either in hardware or software through rectification) lie on the same row in both cameras, simplifying the point search problem and enabling *dense* reconstructions of the observed scene.

A necessary condition for achieving a complete three-dimensional reconstruction of the observed scene, through the analysis of multiple images acquired from different viewpoints, is the knowledge of the intrinsic parameters of the involved cameras and their relative poses.

If the relative pose is unknown, it can be estimated through the analysis of the images themselves. However, as will be shown later, the distance between the cameras will be derived up to a multiplicative factor, and consequently, the three-dimensional reconstruction will also be known up to that factor.

If the intrinsic parameters are not known, it is still possible to relate corresponding points between the two images. This process can accelerate the comparison between *KeyPoints*, but it will not be possible to make any statements about the three-dimensional reconstruction of the observed scene (the reconstruction is known up to an affine transformation).

9.1 Camera Coordinate Transformation

When discussing stereoscopic coordinates, it is essential to provide a brief introduction to coordinate transformations between camera systems. This section is, in fact, a continuation of the general discussion regarding sensors presented in section 1.9.

When the sensors involved are video sensors, the rotation matrices involved in the camera equations are matrices that convert from world coordinates to camera coordinates, rather than, as previously indicated, to sensor coordinates.

In the case of the *pin-hole* camera, the generic world point $\mathbf{x}$ is rotated and translated to the point ${}^{i}\mathbf{m}$, expressed in the camera coordinates of the i-th sensor, through the relationship:

$$ {}^{i}\mathbf{m} = {}^{i}\mathbf{R}\left(\mathbf{x} - \mathbf{t}_i\right) = {}^{i}\mathbf{R}\mathbf{x} - {}^{i}\mathbf{R}\mathbf{t}_i \tag{9.1} $$

which involves the rotation and permutation matrix ${}^{i}\mathbf{R}$, expressed in the form of the *pin-hole* camera, that is, a matrix that converts from world coordinates to camera coordinates. The inverse of this transformation always exists and is given by

$$ \mathbf{x} = {}^{i}\mathbf{R}^{-1}\,{}^{i}\mathbf{m} + \mathbf{t}_i \tag{9.2} $$

Therefore, given a point in camera coordinates ${}^{1}\mathbf{m}$ observed in the reference frame of the first video sensor, this point is represented in the reference frame of the second camera as

$$ \begin{aligned} {}^{2}\mathbf{m} &= {}^{2}\mathbf{R}\left({}^{1}\mathbf{R}\right)^{-1}\,{}^{1}\mathbf{m} - {}^{2}\mathbf{R}(\mathbf{t}_2 - \mathbf{t}_1) \\ &= {}^{2}\mathbf{R}_1\,{}^{1}\mathbf{m} + {}^{2}\mathbf{t} \end{aligned} \tag{9.3} $$

having defined

$$\begin{aligned} {}^2\mathbf{R}_1 &= {}^2\mathbf{R}\left({}^1\mathbf{R}\right)^{-1} \\ {}^2\mathbf{t} &= -{}^2\mathbf{R}(\mathbf{t}_2 - \mathbf{t}_1) \end{aligned} \tag{9.4}$$

with the matrices ${}^1\mathbf{R}$ and ${}^2\mathbf{R}$ still defined as in the *pin-hole* model. In this case, the matrix $\mathbf{R}$ is a matrix that converts a point from the camera coordinates of the first reference system to the camera coordinates of the second system.

The relationships connecting points between two sensors depend solely on their relative positioning. Consequently, the coordinates ${}^1\mathbf{m}$ and ${}^2\mathbf{m}$ of the same world point $\mathbf{x}$, observed by the two video sensors, must always satisfy the equation (9.3).

9.1.1 Relative Positioning in Camera Coordinates and Sensor Coordinates

As already mentioned several times, to transform a camera reference system into a sensor reference system, it is sufficient to apply the transformation

$${}^c\mathbf{R}_w = {}^c\mathbf{\Pi}_b \, {}^w\mathbf{R}_b^{-1} \Leftrightarrow {}^w\mathbf{R}_b = {}^c\mathbf{R}_w^{-1} \, {}^c\mathbf{\Pi}_b \tag{9.5}$$

where ${}^c\mathbf{R}_w = \mathbf{R}$ is the rotation matrix used in the equations of the pin-hole camera model.

For example, through these relationships, it is possible to obtain the relationships that connect the relative poses expressed in camera coordinates with those expressed in sensor coordinates as given by equation (1.66):

$$\begin{aligned} {}^{2b}\mathbf{R}_{1b} &= {}^c\mathbf{\Pi}_b^{-1} \, {}^2\mathbf{R} \, {}^1\mathbf{R}^{-1} \, {}^c\mathbf{\Pi}_b \\ {}^{2b}\mathbf{t}_{2,1} &= -{}^c\mathbf{\Pi}_b^{-1} \, {}^2\mathbf{R} \left(\mathbf{t}_2 - \mathbf{t}_1\right) \end{aligned} \tag{9.6}$$

with parameters ${}^1\mathbf{R}$, $\mathbf{t}_1$, ${}^2\mathbf{R}$, and $\mathbf{t}_2$ defined as in the *pin-hole* model, that is, matrices that transform from world to camera and vectors expressed in world coordinates. As can be seen, this result is fully compatible with that obtained in equation (9.4).

These relationships allow for the determination of the parameters of the relative pose $(\mathbf{R}, \mathbf{t})$ starting from the parameters of the pin-hole camera. These relationships enable the conversion of coordinates from sensor 1 to coordinates in sensor 2, as expressed in equation 1.64.

9.2 The Epipolar Plane

In the previous chapters, it has been repeatedly emphasized that it is not possible to obtain the world coordinates of the points that make up an image from a single image alone, without additional information.

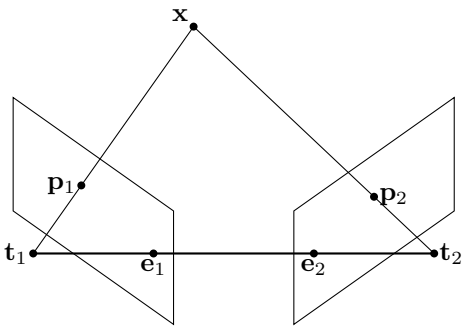


Figure 9.1: Epipolar geometry between two cameras: $\mathbf{t}_1$ and $\mathbf{t_2}$ are the *pin-holes*, $\mathbf{e}_1$ and $\mathbf{e}_2$ are the epipoles, and the world point $\mathbf{x}$ is projected onto the two image points $\mathbf{p}_1$ and $\mathbf{p}_2$ respectively. All points involved belong to the same plane.

The only thing that a generic point of the image $\mathbf{p}$ can provide, given the equation (8.16) of the pin-hole camera, is a relationship between the (infinite) world coordinates $\mathbf{x}$ underlying the image point, that is, the locus of world coordinates that, when projected, would yield exactly that particular image point. This relationship is the equation of a line passing through the *pin-hole* $\mathbf{t}$ and the point on the sensor corresponding to the image point $\mathbf{p}$.

By rewriting the equation (8.16), it is easy to see what the dependency is between the parameters of the i-th camera, the image point $\mathbf{p}_i$, and the line that represents all possible world points $\mathbf{x}$ underlying $\mathbf{p}_i$:

$$\mathbf{x} = \lambda(\mathbf{K}_i\mathbf{R}_i)^{-1}\mathbf{p}_i + \mathbf{t}_i = \lambda\mathbf{v}_i(\mathbf{p}_i) + \mathbf{t}_i \tag{9.7}$$

where $\mathbf{v}_i$ has the same meaning it had in equation (8.17), the direction vector from the *pin-hole* to the sensor point.

As can be inferred both from experience and from the linear relationship that connects these points, it can be stated that the underlying point $\mathbf{x}$ is known up to a scale factor λ.

In the case of stereo vision, we have two sensors, and therefore we need to define two reference systems with parameters $\mathbf{K}_1\mathbf{R}_1$ and $\mathbf{K}_2\mathbf{R}_2$, respectively, and the positions of the pinholes $\mathbf{t_1}$ and $\mathbf{t_2}$, which are always expressed in world coordinates.

The line (9.7), the locus of world points associated with the image point $\mathbf{p}_1$ observed in the first reference frame, can be projected into the view of the second camera:

$$\begin{aligned}\mathbf{p}_2 &= \lambda\mathbf{K}_2\mathbf{R}_2(\mathbf{K}_1\mathbf{R}_1)^{-1}\mathbf{p}_1 - \mathbf{K}_2\mathbf{R}_2(\mathbf{t}_2 - \mathbf{t}_1) \\ &= \lambda\mathbf{K}_2\mathbf{R}\mathbf{K}_1^{-1}\mathbf{p}_1 + \mathbf{K}_2\mathbf{t} \\ &= \lambda\mathbf{K}_2\mathbf{R}\mathbf{K}_1^{-1}\mathbf{p}_1 + \mathbf{e}_2\end{aligned} \tag{9.8}$$

where a variable component appears, which depends on the point being considered and the value λ, and a vector $\mathbf{e}_2$ that remains constant and does not depend on the point in question.

This constant point is the epipole. The epipole is the intersection point of all epipolar lines and represents the projection of the pinhole of one camera onto the image of the other, or the "vanishing point" of the epipolar lines.

Given two cameras, the projections of the coordinates of the *pin-hole* $\mathbf{t}_1$ and $\mathbf{t}_2$ onto the opposite image are

$$\begin{aligned}\mathbf{e}_1 &= \mathbf{P}_1\mathbf{t}_2 = \mathbf{K}_1\mathbf{R}_1(\mathbf{t}_2 - \mathbf{t}_1) \\ \mathbf{e}_2 &= \mathbf{P}_2\mathbf{t}_1 = \mathbf{K}_2\mathbf{R}_2(\mathbf{t}_1 - \mathbf{t}_2)\end{aligned} \tag{9.9}$$

where $\mathbf{P}_1$ and $\mathbf{P}_2$ are the projection matrices. The points $\mathbf{e}_1$ and $\mathbf{e}_2$ are the *epipoles*. If we substitute the definitions of relative pose expressed in (9.4) into equation (9.9), the image coordinates of the epipoles, understood as the projection onto one image of the *pin-hole* of the other camera, are

$$\begin{aligned}\mathbf{e}_1 &= \mathbf{K}_1\mathbf{R}^\top\mathbf{t} \\ \mathbf{e}_2 &= \mathbf{K}_2\mathbf{t}\end{aligned} \tag{9.10}$$

which are solely functions of the relative pose between the two cameras.

The matrix $\mathbf{R}$ is designed to convert from camera 1 coordinates to camera 2 coordinates, and $\mathbf{t}$ represents the position of the pin-hole of camera 1 expressed in the reference frame of camera 2.

The lines generated by the points in the first image all converge at a single point formed by the projection of the pin-hole $\mathbf{t}_1$ onto the second image: in fact, the point in world coordinates and the two epipoles create a plane (the epipolar plane) where the possible solutions, the points in camera coordinates, of the three-dimensional reconstruction problem reside (figure 9.1).

Epipolar geometry is the geometry that connects two images captured from different viewpoints. The relationships between the images, however, do not depend on the observed scene but solely on the intrinsic parameters of the cameras and their relative poses.

For each observed point, the epipolar plane is the plane formed by the point in world coordinates and the two optical centers.

The epipolar line is the intersection between the epipolar plane and the image plane in the second image. In fact, the epipolar plane intersects the plane in both images along the epipolar lines and defines the correspondences between the lines.

In the following sections, we will discuss both how to derive the line along which a point belonging to one image must be located in another image, and how to obtain the corresponding three-dimensional point given two (or more) homologous points.

9.3 Three-Dimensional Reconstruction

The objective of stereoscopic vision (and, in general, multi-ocular vision) is to enable the three-dimensional reconstruction of the observed scene. By identifying the same world point projected from different viewpoints, for which the relative pose is known, it is possible to perform the three-dimensional reconstruction of the observed point.

9.3.1 Triangulation

Observing figure 9.2, it is easy to infer that the solution to the triangulation problem is the intersection point of the epipolar lines generated by the two images. This problem can be easily extended to the case of n cameras where the relative pose among them is known. In the absence of knowledge about the absolute pose, this could be obtained directly from the images themselves using techniques such as the Essential matrix (section 9.4).

Due to the inaccuracies in identifying homologous points (a separate discussion could be made regarding calibration errors), the lines formed by the optical rays are generally skewed. In this case, it is necessary to derive the closest solution under some cost function: the least squares solution is always possible with $n \geq 2$, either using techniques such as *Forward Intersections* or *Direct Linear Transfer* (DLT).

Every optical ray subtended by the image pixel (u_i, v_i), with $i = 1, \ldots, n$ being the i-th view, must satisfy the equation (9.7). The intersection point (*Forward Intersections*) of all these rays is the solution to a potentially overdetermined linear system, with $3 + n$ unknowns in $3n$ equations:

$$\begin{cases} \mathbf{x} &= \lambda_1\mathbf{v}_1 + \mathbf{t}_1 \\ \cdots & \\ \mathbf{x} &= \lambda_n\mathbf{v}_n + \mathbf{t}_n \end{cases} \tag{9.11}$$

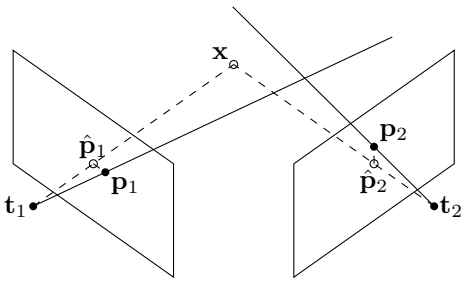


Figure 9.2: Example of triangulation. With the knowledge of camera calibration, the world point $\mathbf{x}$ can be derived from the observation of its projection in at least two images ($\mathbf{p_1}, \mathbf{p_2}, \ldots$). However, due to noise, the resulting lines do not pass through point $\mathbf{x}$ and may not intersect each other. The maximum likelihood solution requires minimizing the sum of the squared errors between the observed point $\mathbf{p}_i$ and the predicted point $\hat{\mathbf{p}}_i$.

where $\mathbf{v}_i = \mathbf{R}_i^{-1}\mathbf{K}_i^{-1}\begin{pmatrix} u_i & v_i & 1 \end{pmatrix}^\top$ indicates the direction of the optical ray in world coordinates. The unknowns are the world point to be estimated $\mathbf{x}$ and the distances along the optical axis λ_i.

The closed-form solution, limited to the case of only two lines, is available in section 1.5.8. This technique can be applied to the case of a camera aligned with the axes and the second positioned relative to the first according to the relationship (9.3).

Exploiting the properties of the cross product, one can arrive at the same expression using perspective projection matrices and image points, expressed in homogeneous coordinates:

$$\begin{cases} [\mathbf{p}_1]_\times \mathbf{P}_1 \mathbf{x} = 0 \\ \ldots \\ [\mathbf{p}_n]_\times \mathbf{P}_n \mathbf{x} = 0 \end{cases} \tag{9.12}$$

with $[\cdot]_\times$ the cross product written in matrix form. Each of these constraints provides three equations, but only two are linearly independent. All these constraints can ultimately be rearranged into a homogeneous system in the form

$$\mathbf{A}\mathbf{x} = 0 \tag{9.13}$$

where $\mathbf{A}$ is a matrix $2n \times 4$ with n being the number of views in which the point $\mathbf{x}$ is observed. The solution to the homogeneous system (9.13) can be obtained using singular value decomposition. This approach is referred to as *Direct Linear Transform* (DLT) by analogy with the calibration technique.

Minimization in world coordinates, however, is not optimal from the perspective of noise minimization. In the absence of further information about the structure of the observed scene, the optimal estimate (*Maximum Likelihood Estimation*) is always the one that minimizes the error in image coordinates (*reprojection*), but it requires a greater computational burden and the use of nonlinear techniques, as the cost function to be minimized is

$$\arg\min_{\mathbf{x}} \sum_{i=1}^{n} \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|^2 \tag{9.14}$$

with $\hat{\mathbf{p}}_i \equiv \mathbf{P}_i\mathbf{x}$ where $\mathbf{P}_i$ is the projection matrix of the i-th image (see figure 9.2).

It is a non-convex nonlinear problem: there are potentially multiple local minima, and the linear solution must be used as the starting point for the minimization.

Another class of techniques, which leverage the information derived from epipolar constraints and thereby allow for the estimation of the positions of noise-free points without the need to derive the three-dimensional point, is presented in section 9.4.4.

9.3.2 Reconstruction with Rectified Cameras

If the corresponding points between the two images of a stereo pair were on the same row of the image (i.e., the same coordinate v), it would be possible to leverage highly optimized code to search for correspondences [LZ99] and obtain dense disparity maps.

There exists a particular configuration of two cameras in which this condition is satisfied, namely when the intrinsic parameters are equal and the optical axes are oriented perpendicularly to the vector connecting the pinholes. For instance, in the case where the vector connecting the pinholes lies along the axis y, the stereo pair configuration that allows for the acquisition of corresponding points on the same row is one that has rotation angles $\rho = 0$ and $\gamma = 0$ with an equal *pitch* angle.

The software procedure to achieve this configuration, when the hardware does not meet such constraints, involves rectification (see 8.3.4). Specifically, starting from an image acquired with a set of parameters (hardware), a new view of the same scene is obtained, but with the desired intrinsic parameters: *yaw*, *pitch*, and *roll*.

Through this consideration, the problem of three-dimensional reconstruction can always be reduced to a pair of cameras perfectly aligned with each other and with the axes, along with a rigid transformation to convert the world coordinates from this sensor system to the actual real-world system.

In the following sections, we will present the specific cases of both perfectly aligned cameras with respect to the axes, as well as cameras that are aligned but tilted (with a non-zero pitch angle), and finally, cameras that are arbitrarily oriented.

9.3.3 Aligned Chambers

In the case of perfectly aligned cameras with respect to the axes and having identical intrinsic parameters (same focal length and same *principal point*), the equations for three-dimensional reconstruction simplify significantly.

In this condition, the equations of perspective projection reduce to

$$\begin{aligned} u_i &= -k_u \frac{y - y_i}{x - x_i} + u_0 \\ v_i &= -k_v \frac{z - z_i}{x - x_i} + v_0 \end{aligned} \tag{9.15}$$

with (x, y, z) being a point in "world" coordinates (see the next section) and (u_i, v_i) the coordinates of the point projected onto the i-th image. The point (u_0, v_0) is the *principal point*, which must be the same for all the cameras involved, and it is assumed that the cameras are all perfectly aligned with the coordinate axes.

Let us now focus solely on the stereoscopic case: for simplicity, we will denote the left camera with the subscript *1* and the right camera with *2*. The alignment constraints impose $x_1 = x_2 = 0$, $y_1 = b$, $y_2 = 0$, and $z_1 = z_2 = 0$, having placed, without loss of generality, the right camera at the center of the reference system. The quantity $b = y_1 - y_2$ is defined as the *baseline*.

The difference $d = u_1 - u_2$ in the horizontal coordinates of the projections of the same point as viewed in the two images of the stereo pair is defined as *disparity*. This value is obtained by incorporating the alignment constraints into equation (9.15), resulting in

$$u_1 - u_2 = d = k_u \frac{b}{x} \tag{9.16}$$

By inverting this simple relation and substituting it into equation (9.15), it is possible to derive the world coordinates (x, y, z) corresponding to a point (u_2, v_2) in the right camera with disparity d:

$$\begin{aligned} x &= k_u \frac{b}{d} \\ y &= -(u_2 - u_0) \frac{b}{d} \\ z &= -(v - v_0) \frac{k_u}{k_v} \frac{b}{d} \end{aligned} \tag{9.17}$$

It is clear that it must be $d \geq 0$ for world points located in front of the stereo pair.

As can be observed, each element is determined by the multiplicative factor b of the *baseline*, the true scaling factor of the reconstruction, and by the inverse of the disparity $1/d$.

World Coordinate Triangulation

The coordinates (x, y, z) obtained are sensor coordinates, referring to a specific stereoscopic configuration where orientation and positioning are aligned and coincide with the axes of the system. To transition from sensor coordinates to the generic case of world coordinates, with arbitrarily oriented cameras, a transformation must be applied that converts the coordinates from sensor to world, specifically the rotation matrix ${}^{w}\mathbf{R}_b$ and the translation $(x_i, y_i, z_i)^\top$ of the pin-hole coordinate, allowing us to express

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = {}^{w}\mathbf{R}_b \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} + \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{9.18}$$

By combining equation (9.17) with equation (9.18), it is possible to define a matrix $\mathbf{M}$ such that the conversion between image point-disparity (u_i, v, d) and world coordinate (x, y, z) can be expressed in a very compact form as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{1}{d} \mathbf{M} \begin{bmatrix} 1 \\ u_i - u_0 \\ v - v_0 \end{bmatrix} + \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{9.19}$$

where i can represent either the left or the right camera interchangeably.

9.3.4 Aligned Cameras and Triangulation in Camera Coordinates

The equations presented earlier refer to a "sensor" or "world" reference frame. For completeness, and to introduce relationships that will be used later, the equations in the case of a "camera" reference frame are now provided.

To keep the sign of the *baseline* positive, let us consider now $b = \tilde{x}_2 - \tilde{x}_1$, $\tilde{y}_1 = \tilde{y}_2 = 0$, and $\tilde{z}_1 = \tilde{z}_2 = 0$. In this case, it is the left camera (with subscript *1*) that is at the center of the reference system.

In camera coordinates, the relationships between the two images can be expressed as

$$d = u_1 - u_2 = k_u \frac{b}{\tilde{z}} \tag{9.20}$$

for the disparity and

$$\begin{aligned} \tilde{x} &= (u_1 - u_0)\frac{b}{d} \\ \tilde{y} &= (v - v_0)\frac{k_u}{k_v}\frac{b}{d} \\ \tilde{z} &= k_u \frac{b}{d} \end{aligned} \tag{9.21}$$

for the equation of the three-dimensional point projected onto the left camera point (u_1, v) with a disparity of d.

9.3.5 Three-Dimensional Reconstruction and Homography

The equation (9.21) can be easily expressed in homogeneous form. The matrix that allows the reconstruction of the three-dimensional point coordinates from image-disparity coordinates directly in the camera reference frame is

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{k_u} & 0 & 0 & -\frac{u_0}{k_u} \\ 0 & \frac{1}{k_v} & 0 & -\frac{v_0}{k_v} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{k_u b} & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} = \mathbf{Q} \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} \tag{9.22}$$

while its inverse

$$\begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & 0 & u_0 & 0 \\ 0 & k_v & v_0 & 0 \\ 0 & 0 & 0 & k_u b \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ 1 \end{bmatrix} = \mathbf{Q}^{-1} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ 1 \end{bmatrix} \tag{9.23}$$

is the matrix that enables the projection of a point from camera coordinates to image-disparity coordinates (these are matrices known up to a multiplicative factor, hence they can be expressed in various forms). The three-dimensional reconstruction of the image-disparity point in the world reference frame, as given by equation (9.17), is equivalent. The matrix $\mathbf{Q}$ is referred to as the *reprojection matrix* [FK08].

In real conditions, since the camera is rotated and translated with respect to the ideal conditions, it is sufficient to multiply the matrix $\mathbf{Q}$ by the matrix 4×4, which represents the transformation from camera coordinates to world coordinates, in order to obtain a new matrix that allows the conversion from disparity coordinates to world coordinates and vice versa.

The use of this formalism allows for the transformation of disparity points acquired from pairs of cameras positioned at different viewpoints (for example, a stereo pair that moves over time or two stereo pairs rigidly connected to each other). In this case, the relationship that links disparity points acquired from the two viewpoints is also represented by a matrix 4×4:

$$\mathbf{H}_{2,1} = \mathbf{Q}_1^{-1} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \mathbf{Q}_2 = \mathbf{Q}_1^{-1} \begin{bmatrix} {}^1\mathbf{R}_2 & {}^1\mathbf{t}_{2,1} \\ 0 & 1 \end{bmatrix} \mathbf{Q}_2 \tag{9.24}$$

which enables the transformation of (u_2, v_2, d_2) into (u_1, v_1, d_1) (this is a homographic transformation in 4 dimensions, quite similar to those in 3 dimensions discussed so far). It is noteworthy that we have used as pose $(\mathbf{R}, \mathbf{t})$ the syntax of equation (1.64), aiming to express the point from reference frame 2 in reference frame 1. Since all the points involved are expressed in camera coordinates, if there are transformations between sensors expressed in world coordinates, as is typically the case, it is necessary to include the change of reference frame. This class of transformations is commonly referred to as *3D Homographies*.

9.3.6 Inclined Chambers with Respect to a Plane

Let us examine the particular case in which the cameras are aligned with respect to the axes, have identical intrinsic parameters, exhibit no relative rotation, and are inclined at a pitch angle with respect to the plane $z = 0$.

In this particular condition, the projection matrix simplifies slightly, taking the form

$$\mathbf{KR} = \begin{bmatrix} u_0 \cos\vartheta & -k_u & -u_0 \sin\vartheta \\ -k_v \sin\vartheta + v_0 \cos\vartheta & 0 & -k_v \cos\vartheta - v_0 \sin\vartheta \\ \cos\vartheta & 0 & -\sin\vartheta \end{bmatrix} \tag{9.25}$$

It is worth noting that the RPY angle system (appendix A.1) has been used for the matrix $\mathbf{R}$.

The horizontal coordinate u of a generic point (x, y, z) in world coordinates is therefore given by:

$$u = u_0 - \frac{k_u(y - y_0)}{\cos\vartheta(x - x_0) - \sin\vartheta(z - z_0)} \tag{9.26}$$

With the assumptions of rectified cameras discussed previously, namely the same orientation and identical intrinsic parameters, a condition that can always be achieved through rectification or by considering appropriate rows of the image, The projection matrix (9.25) remains the same in the two different reference frames, and by examining equation (9.26), the only difference between different cameras is found in the numerator due to the varying position of the *pin-hole* along the y axis.

It follows that the difference in coordinates u in the two images $d = u_1 - u_2$ (disparity) is given by

$$d = u_1 - u_2 = \frac{k_u b}{\cos\vartheta(x - x_0) - \sin\vartheta(z - z_0)} \tag{9.27}$$

, having redefined $b = y_1 - y_2$.

Using the relationship (9.26) in equation (9.27) yields the remarkable result

$$u_i = u_0 - d\frac{y - y_i}{b} \tag{9.28}$$

, from which we can finally derive the coordinate y of the point. It seems you've entered "

$$y = -b\frac{u_i - u_0}{d} + y_i \tag{9.29}$$

". Please provide the specific text or content you would like me to translate from Italian to English, and I'll be happy to assist you! In the case where the cameras are perfectly aligned, the only calibration parameter that affects the coordinate y is the sole b.

The coordinate v of the point can instead be expressed as

$$v - v_0 = -\frac{k_v}{bk_u}(\sin\vartheta(x - x_0) + \cos\vartheta(z - z_0))d \tag{9.30}$$

From which the system of equations follows:

$$\begin{aligned} \cos\vartheta(x - x_0) - \sin\vartheta(z - z_0) &= \frac{bk_u}{d} \\ \sin\vartheta(x - x_0) + \cos\vartheta(z - z_0) &= -\frac{v - v_0}{k_v}\frac{bk_u}{d} \end{aligned} \tag{9.31}$$

the solution that allows us to obtain the remaining two three-dimensional coordinates of the given point is

$$\begin{aligned} x - x_0 &= \frac{bk_u}{d}\left(\cos\vartheta - \frac{v - v_0}{k_v}\sin\vartheta\right) \\ z - z_0 &= -\frac{bk_u}{d}\left(\frac{v - v_0}{k_v}\cos\vartheta + \sin\vartheta\right) \end{aligned} \tag{9.32}$$

V-Disparity

A particular case of disparity occurs when observing a plane, that of the ground, which, due to the number of points, predominates in the image. In the case where the *baseline* is along the axis y, the disparity of the plane $z = 0$ is solely a function of v, and this equation turns out to be that of a straight line.

The disparity relation from the coordinate v can be derived from the value of x from the second equation and substituting it into the first of the equations (9.31):

$$\begin{aligned} x - x_0 &= \tan\vartheta(z - z_0) + \frac{k_u}{d\cos\vartheta}b \\ v - v_0 &= -k_v\tan\vartheta - d\frac{k_v}{k_u}\frac{z - z_0}{b\cos\vartheta} \end{aligned} \tag{9.33}$$

From the first of the equations (9.33), it can be seen that the expression for disparity depends solely on the distance x when the height z is fixed (for example, at ground level). From the second equation, it is evident that the disparity d increases linearly with the coordinate v following a known slope

$$d = \cos\vartheta\frac{b}{z_0}(v - v_{d=0}) \tag{9.34}$$

in the classical case where $k_u \approx k_v$ (square pixel). The point of zero disparity $v_{d=0}$, as mentioned earlier, is located at

$$v_{d=0} = v_0 - k_v\tan\vartheta \tag{9.35}$$

and depends only on the vertical aperture and the *pitch* (which is obviously the same coordinate as the *vanishing point*).

9.4 Essential Matrix and Fundamental Matrix

In 1981, Christopher Longuet-Higgins [Lon81] was the first to observe that a generic point expressed in world coordinates, the corresponding points in camera coordinates, and the *pin-hole* must be coplanar. The geometric derivation of the relationships among the points is omitted, but the analytical presentation is provided directly.

It has been repeatedly stated that a point in an image subtends a line in the world, and the line in the world projected onto another image, captured from a different viewpoint, represents the epipolar line where the corresponding point of the first image lies. This equation, which relates points in one image to lines in the other, can be expressed in a matrix form.

To follow Higgins' reasoning, the intrinsic parameter matrix will be implicit, and the coordinates used will be those of the normalized camera.

Without loss of generality, consider a system consisting of two cameras, the first positioned and oriented with respect to the second with projection matrix $\mathbf{P}_1 = [\mathbf{R}|\mathbf{t}]$, while the second is placed at the origin of the reference system aligned with the axes, that is, with projection matrix $\mathbf{P}_2 = [\mathbf{I}|\mathbf{0}]$: one can arrive at the same result starting from two generic calibrated cameras, arbitrarily oriented and positioned with respect to a third system, through the relations $\mathbf{R} = {}^2\mathbf{R}_1 = \mathbf{R}_2^{-1}\mathbf{R}_1$ and $\mathbf{t}$, which represent the position of camera 1 with respect to system 2.

A generic point $\mathbf{x} \in \mathbb{R}^3$ has coordinates $\mathbf{x}_1$ and $\mathbf{x}_2$ in the two different reference systems and is projected onto sensors 1 and 2 at the points with camera coordinates $\mathbf{m}_1$ and $\mathbf{m}_2$, respectively.

These image points are known to span a subspace of $\mathbb{R}^3$ defined by the equation, for instance, $\lambda\mathbf{m}_2$, which passes through the pin-hole of the second sensor (here set to be at $\mathbf{0}$), namely

$$\mathbf{x}_2 = \lambda_2\mathbf{m}_2 + \mathbf{0} \tag{9.36}$$

A generic point $\mathbf{x}_1 = \lambda_1\mathbf{m}_1$ expressed in coordinates of sensor 1 and observed by that sensor can be projected into coordinates of sensor 2 according to the equation

$$\mathbf{x}_2 = {}^2\mathbf{R}_1\mathbf{x}_1 + \mathbf{t} \tag{9.37}$$

The equation of the epipolar line, a line in camera coordinates of the second sensor and the locus of points where $\mathbf{m}_2$ must lie, associated with the point $\mathbf{m}_1$ (observed and therefore expressed in camera coordinates of the first sensor), is given by

$$\lambda_2\mathbf{m}_2 = \lambda_1\,{}^2\mathbf{R}_1\mathbf{m}_1 + \mathbf{t} \tag{9.38}$$

The locus of homogeneous points $\mathbf{m}_2$ is obtained by varying the parameter λ_1, and this line in $\mathbb{R}^3$ remains a line even in $\mathbb{R}^2$. If two points are indeed homologous, the system is solvable, and it is possible to derive the parameters λ_1 and λ_2 (this is an example of three-dimensional reconstruction through triangulation, as discussed in section 9.3.1).

However, there exists a relationship that connects the points of the two cameras by eliminating the parameters λ, but more importantly, it allows for the reverse reasoning, that is, to derive the relative pose between the two cameras $(\mathbf{R}, \mathbf{t})$ given a list of corresponding points.

If both sides of the equation (9.38) are first multiplied vectorially by $\mathbf{t}$, and then scalarly by $\mathbf{m}_2^\top$, one obtains

$$\lambda_2\mathbf{m}_2^\top(\mathbf{t}\times\mathbf{m}_2) = \lambda_1\mathbf{m}_2^\top(\mathbf{t}\times\mathbf{R}\mathbf{m}_1) + \mathbf{m}_2^\top(\mathbf{t}\times\mathbf{t}) \tag{9.39}$$

. On this relation, it is possible to apply the properties of the vector product $\mathbf{t}\times\mathbf{t} = \mathbf{0}$ and the scalar product $\mathbf{m}_2\cdot(\mathbf{t}\times\mathbf{m}_2) = 0$.

This passage has a physical significance: first of all, the coplanarity constraints (all expressed, for example, in reference 2) are introduced among the points $\mathbf{0}$ (the pinhole of camera 2), $\mathbf{m}_2$, ${}^2\mathbf{R}_1\mathbf{m}_1 + \mathbf{t}$, $\mathbf{x}_2 = {}^2\mathbf{R}_1\mathbf{x}_1 + \mathbf{t}$, and $\mathbf{t}$ (the pinhole of camera 1 in system 2), along with the fact that the body is rigid.

Through this formula, it is possible to express the relationships between the corresponding points $\mathbf{m}_1$ and $\mathbf{m}_2$, represented in the form of homogeneous camera coordinates, in a very compact form

$$\mathbf{m}_2^\top(\mathbf{t}\times\mathbf{R}\mathbf{m}_1) = 0 \tag{9.40}$$

Finally, denoting with $[\mathbf{t}]_\times$, the antisymmetric matrix, the vector product in matrix form (see section 1.7), it is possible to collect the various contributions in the form of a matrix

$$\mathbf{E} = [\mathbf{t}]_\times\mathbf{R} = \mathbf{R}\left[\mathbf{R}^\top\mathbf{t}\right]_\times \tag{9.41}$$

. It is important to recall the significance of matrices by comparing them with equation (9.37). In this way, a linear relationship can be defined that connects the camera points of the two views:

$$\mathbf{m}_2^\top\mathbf{E}\mathbf{m}_1 = 0 \tag{9.42}$$

The matrix $\mathbf{E}$ is defined as the *Essential Matrix*.

Finally, one must pay close attention to the indices because there is no unique convention for indicating points 1 and 2: assuming the convention in (9.41) is satisfied, what needs to be remembered is that matrix $\mathbf{E}$ encodes the relative pose of

the camera of the points on the right (in our case $\mathbf{m}_1$) with respect to the camera of the points on the left (in our case $\mathbf{m}_2$) of the matrix.

The matrix $\mathbf{E}$, relating homogeneous points, is also homogeneous and therefore defined up to a multiplicative factor.

The Essential matrix has the following properties:

- the transpose of the Essential matrix for the ordered pair of cameras (1,2) is the Essential matrix for the pair (2,1);
- $\mathbf{E}$ is a rank 2 matrix with 5 degrees of freedom (it indeed represents a relative pose, thus 3 angles and the direction between the epipoles, which accounts for 2 degrees of freedom);
- The two singular values of the matrix $\mathbf{E}$ must be equal, and the third must be zero.

The Essential matrix establishes relationships in camera coordinates and, therefore, to utilize it from a practical standpoint, it is necessary to have points expressed in this particular reference system. In other words, it is essential to know the intrinsic parameters of the cameras involved.

The equation

$$\mathbf{m}_2^\top (\mathbf{E}\mathbf{m}_1) = 0 \tag{9.43}$$

can also be interpreted as the equation of a plane in space 2 that passes through $\mathbf{0}$, specifically the epipolar plane formed by the two epipoles and the world point, a plane to which the point $\mathbf{m}_2$ must belong.

It is indeed possible to introduce an additional relationship between the points of the images, completely disregarding the intrinsic parameters of the cameras themselves.

If we apply the definition of homogeneous camera coordinates $\mathbf{p} = \mathbf{K}\mathbf{m}$ in relation (9.42), we obtain

$$\mathbf{m}_2^\top \mathbf{E}\mathbf{m}_1 = \mathbf{p}_2^\top \mathbf{K}_2^\top \mathbf{F}\mathbf{K}_1\mathbf{p}_1 = \mathbf{p}_2^\top \mathbf{F}\mathbf{p}_1 \tag{9.44}$$

The Fundamental matrix is defined (Faugeras and Hartley, 1992) as:

$$\mathbf{p}_2^\top \mathbf{F}\mathbf{p}_1 = 0 \tag{9.45}$$

where $\mathbf{p}_1$ and $\mathbf{p}_2$ are the homogeneous coordinates of corresponding points in the first and second images, respectively.

If two points on the two images of the stereoscopic pair represent the same point in the world, the equation (9.45) must be satisfied.

The fundamental matrix allows us to narrow down the search range for correspondences between the two images because, due to the point-line duality, from the relationship (9.45), we can specify the location of points in the second image where we should search for points from the first image.

Indeed, the equation of a line where the points $\mathbf{m_2}$ and $\mathbf{m_1}$ must lie is described by

$$\begin{aligned} \mathbf{l}_2 &= \mathbf{F}\mathbf{m}_1 \\ \mathbf{l}_1 &= \mathbf{F}^\top \mathbf{m}_2 \end{aligned} \tag{9.46}$$

, where $\mathbf{l}_1$ and $\mathbf{l}_2$ are the parameters of the epipolar line, belonging to the first and second images respectively, expressed in implicit form.

The relationship between the Fundamental matrix and the Essential matrix is given by equation (9.44),

$$\mathbf{E} = \mathbf{K}_2^\top \mathbf{F}\mathbf{K}_1 \tag{9.47}$$

or vice versa. It seems that you've entered a placeholder for a mathematical block. Please provide the specific content or equations you would like to have translated, and I will assist you accordingly.

The Essential matrix collects in $\mathbf{E}$ the relative poses between the cameras, while the Fundamental matrix conceals both the intrinsic parameters and the relative pose.

The Essential matrix introduces constraints that are equivalent to those of the Fundamental matrix; however, although it was historically introduced before the Fundamental matrix, it is a special case because it expresses the relationships with respect to camera coordinates.

$\mathbf{F}$ is a matrix 3×3 of rank 2, and it can be determined with just 7 points, as the degrees of freedom amount to exactly 7 (a multiplicative factor and the zero determinant reduce the dimensionality of the problem). The relationship that connects the Fundamental matrix to the 7 degrees of freedom is a nonlinear relationship (one that is not easily expressible through any algebraic representation). With (at least) 8 points, however, it is possible to obtain a linear estimate of the matrix, as described in the following section.

The Fundamental matrix has the following properties:

- the transpose of the Fundamental matrix of the ordered pair of cameras (1,2) is the Fundamental matrix of the pair (2,1);
- $\mathbf{F}$ is a rank 2 matrix with 7 degrees of freedom (the homogeneous matrix $\mathbf{F}$ has 8 degrees of freedom, to which the constraint $\det \mathbf{F} = 0$ is added);

- $\mathbf{l}_2 = \mathbf{F}\mathbf{p}_1$ and $\mathbf{l}_1 = \mathbf{F}^\top \mathbf{p}_2$ are the epipolar lines in image 2 of a point from image 1 and in image 1 of a point from image 2, respectively;
- since the epipoles must satisfy the relations $\mathbf{F}\mathbf{e}_1 = 0$ and $\mathbf{F}^\top \mathbf{e}_2 = 0$, respectively, it follows that they are the "left" and "right" kernels of the matrix $\mathbf{F}$;
- $\mathbf{F}$ is a "quasi-correlation," meaning it is a transformation that converts points into lines but is not invertible.

Figure 9.3: The Fundamental matrix allows for the identification of epipolar lines, shown in the right image, on which the corresponding points of the left image reside.

The Fundamental and Essential matrices can be used to narrow down the search space for corresponding points between two images and/or filter out potential outliers (for example, in RANSAC). The Essential matrix, when decomposed, allows for the extraction of the relative pose between the two cameras and, as such, provides an approximate idea of the motion experienced by a camera moving through the world (motion stereo) or the relative pose of two cameras in a stereoscopic pair (Auto-Calibration).

The use of the Essential matrix allows for the derivation of the relative pose between two views. However, it is not possible to determine the length of the *baseline* connecting the two *pin-holes*, but only its direction. Nevertheless, with the Essential matrix at hand, it is always possible to perform a three-dimensional reconstruction of the observed scene up to a multiplicative factor: the ratios between distances are known, but not their absolute values.

This, however, enables a coherent three-dimensional reconstruction when observing the same scene from more than two different views, where the unknown multiplicative factor remains consistent across all views, thus allowing the merging of all individual reconstructions into a single reconstruction known up to the same scale factor.

9.4.1 Determination of Matrices

The Essential matrix can be derived in closed form by knowing the relative poses between the sensors, and with the knowledge of the intrinsic parameters of the involved cameras, it is possible to obtain the Fundamental matrix.

The most widespread application of the Essential (or Fundamental) matrix is to derive the relative pose between cameras, given a set of corresponding points. By knowing the intrinsic parameters, one can obtain the Essential matrix (and from this, derive the Fundamental matrix), or alternatively, one can derive the Fundamental matrix without any knowledge of the camera parameters.

Eight-Point Algorithm

The criterion for obtaining the matrix $\mathbf{F}$ can be formalized as a minimization of a cost function

$$\min_{\mathbf{F}} \sum_{i=1}^{n} \left(\mathbf{p}_{2,i}^\top \mathbf{F} \mathbf{p}_{1,i}\right)^2 \tag{9.48}$$

under additional constraints that pertain this time to the structure of $\mathbf{F}$.

It can be observed that the epipolar constraint (9.45) can also be rewritten as

$$(\mathbf{p}_1 \otimes \mathbf{p}_2)^\top \mathrm{vec}(\mathbf{F}) = 0 \tag{9.49}$$

by utilizing the compact syntax provided by the Kronecker product $\otimes$ or alternatively

$$\mathbf{u}_i \mathbf{f} = 0 \tag{9.50}$$

in a more explicit form by defining

$$\begin{array}{rl}(\mathbf{p}_1 \otimes \mathbf{p}_2)^\top = & \mathbf{u}_i = (x_1x_2, y_1x_2, x_2, x_1y_2, y_1y_2, y_2, x_1, y_1, 1) \\ \mathrm{vec}(\mathbf{F}) = & \mathbf{f} = (f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}, f_{3,1}, f_{3,2}, f_{3,3})^\top\end{array} \tag{9.51}$$

with $\mathbf{p}_{1,i} = (x_1, y_1)$ and $\mathbf{p}_{2,i} = (x_2, y_2)$. With this formalism, it is possible to highlight a technique that allows the derivation of the elements of $\mathbf{F}$ as the solution to a homogeneous linear system formed by constraints as in equation (9.50).

Collecting all the constraints $\mathbf{u}_i$, we obtain a homogeneous system of the form

$$\mathbf{U}\mathbf{f} = 0 \tag{9.52}$$

consisting of n equations in 9 unknowns.

To derive the Essential matrix, the discussion is analogous and it is the solution of a system $\mathbf{n}_i\mathbf{e} = 0$ in the form

$$\begin{array}{l}\mathbf{n}_i = (x_1x_2, y_1x_2, z_1x_2, x_1y_2, y_1y_2, z_1y_2, x_1z_2, y_1z_2, z_1z_2) \\ \mathbf{e} = (e_{1,1}, e_{1,2}, e_{1,3}, e_{2,1}, e_{2,2}, e_{2,3}, e_{3,1}, e_{3,2}, e_{3,3})\end{array} \tag{9.53}$$

with $\mathbf{m}_{1,i} = (x_1, y_1, z_1)$ and $\mathbf{m}_{2,i} = (x_2, y_2, z_2)$. With all the constraints $\mathbf{n}_i$, this is a homogeneous system of the type

$$\mathbf{N}\mathbf{e} = 0 \tag{9.54}$$

. In fact, by using homogeneous coordinates, the systems (9.51) and (9.53) are algorithmically equivalent.

To the constraints expressed in these homogeneous systems, an additional one must always be added, for example $\|\mathbf{f}\| = 1$, which is typically already satisfied by linear solvers of homogeneous systems. This algorithm is therefore referred to as the *eight-point algorithm* since the solution to the problem requires at least 8 points to be determined. Additional constraints, in order to achieve the actual degrees of freedom of the matrices, cannot be expressed in linear form.

Strengthening of Constraints

Due to noise, the matrices obtained from the linear system typically do not meet the requirement of having rank 2 (and in the case of the Essential matrix, therefore having a large number of degrees of freedom, which do not belong precisely to the subspace of Essential matrices). A possible solution to this problem is to search for the matrix that is closest to the one returned by the linear system while still satisfying the rank constraint. This result can be achieved, for example, by using an SVD decomposition followed by a composition, as suggested by Tsai, Huang, and Hartley:

$$\begin{array}{rl}\mathbf{F} & = \mathbf{U}\,\mathrm{diag}(r, s, t)\mathbf{V}^\top \\ \mathbf{F}' & = \mathbf{U}\,\mathrm{diag}(r, s, 0)\mathbf{V}^\top\end{array} \tag{9.55}$$

This procedure is referred to as *constraint enforcement*. The Essential matrix, in contrast to the Fundamental matrix, has the additional constraint of having its two non-zero singular values equal:

$$\begin{array}{rl}\mathbf{E} & = \mathbf{U}\,\mathrm{diag}(r, s, t)\mathbf{V}^\top \\ \mathbf{E}' & = \mathbf{U}\,\mathrm{diag}(1, 1, 0)\mathbf{V}^\top\end{array} \tag{9.56}$$

If the singular values (following an SVD) of the matrix are 1, the matrix is referred to as a normalized essential matrix. The Essential matrix obtained by setting $\mathbf{D}' = \mathrm{diag}(1, 1, 0)$ is the normalized essential matrix closest to the given one, in accordance with the Frobenius norm. The Essential matrix generated through equation (9.56) satisfies the *cubic trace-constraint* (Demazure, 1988)

$$\mathbf{E}\mathbf{E}^\top\mathbf{E} - \frac{1}{2}\,\mathrm{trace}\left(\mathbf{E}\mathbf{E}^\top\right)\mathbf{E} = 0 \tag{9.57}$$

. This constraint is a necessary condition for the matrix under analysis to be truly Essential.

The matrices obtained through this reinforcement procedure satisfy all the requirements to be considered Fundamental or Essential matrices, but they do not represent an algebraic minimization, nor a geometric one, of the original constraints.

Seven-Point Algorithm

Algorithms that utilize fewer than 8 points to extract an Essential or Fundamental matrix are largely based on the same principle: the multidimensional kernel of $\mathbf{U}$ or $\mathbf{N}$ is extracted, since the Fundamental or Essential matrix must belong to an element of this space, and certain typical constraints of the problem at hand are enforced.

In a nonlinear manner, it is relatively easy to obtain a Fundamental matrix with only 7 points, considering that the matrix $\mathbf{U}$, formed by the elements of equation (9.51), must be of rank 7, as there are indeed 7 degrees of freedom in the Fundamental matrix.

By solving the system (9.51) formed by (at least) 7 points, a subspace of dimension 2 is obtained, consisting of two bases $\mathbf{f}_1$ and $\mathbf{f}_2$, to which two matrices $\mathbf{F}_1$ and $\mathbf{F}_2$ are associated: in the space of possible solutions, it is necessary to find a matrix $\mathbf{F} = \alpha\mathbf{F}_1 + (1-\alpha)\mathbf{F}_2$ such that it has rank 2, which is achieved by imposing $\det\mathbf{F} = 0$, a nonlinear third-degree equation in α.

In this case, the real solutions of α can be either 1 or 3: in the case of 3 real solutions, all three must be evaluated against the data to identify the most plausible one.

Five-Point Algorithm

With fewer than 7 points, there are only algorithms available to determine the Essential matrix. The Essential matrix is indeed composed of only 5 degrees of freedom and can theoretically be estimated through the analysis of correspondences among just 5 points [Nis04]. The five-point algorithm is essentially the standard for estimating the essential matrix; however, its implementation is extremely complex.

Utilizing only 5 correspondences, the matrix $\mathbf{N}$ of the system (9.54) has a rank deficiency of 4. Therefore, the Essential matrix must be formed as a linear combination of the last 4 columns of the matrix V obtained from the SVD, namely:

$$\mathbf{E} = x\mathbf{X} + y\mathbf{Y} + z\mathbf{Z} + \mathbf{W} \tag{9.58}$$

where $(\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{W})$ represents the last 4 columns of eigenvectors of the matrix V and (x, y, z) are the unknowns. To obtain these unknowns, it is necessary to satisfy the constraints

$$\begin{aligned} &\det \mathbf{E} = 0 \\ &\mathbf{E}\mathbf{E}^\top\mathbf{E} - \tfrac{1}{2}\,\mathrm{trace}\left(\mathbf{E}\mathbf{E}^\top\right)\mathbf{E} = 0 \end{aligned} \tag{9.59}$$

which is equivalent to a problem of 10 cubic polynomials in the 3 unknowns.

The request to solve a non-linear system, however, diminishes the advantages compared to the solutions proposed in section 9.4.2.

System Conditioning

The generation of the Essential and Fundamental matrices through the SVD technique, followed by the regularization of these matrices by enforcing the singular values to be equal, is a process that is highly sensitive to noise.

The matrix (9.51) is ill-conditioned: this occurs when attempting to solve a linear system where the known terms consist of numbers with differing orders of magnitude. The method proposed by Hartley [Har95] suggests improving the solution by normalizing the coordinates of the points.

The coordinates $\mathbf{p}_1$ and $\mathbf{p}_2$ are translated separately to achieve a zero centroid and rescaled to have an average value of 1 (or $\sqrt{2}$ as the average value of the modulus) in the new coordinate systems $\tilde{\mathbf{p}}_1$ and $\tilde{\mathbf{p}}_2$, respectively.

We therefore define two transformation matrices $\mathbf{T}_1$ and $\mathbf{T}_2$ such that

$$\begin{aligned} \tilde{\mathbf{p}}_1 &= \mathbf{T}_1\mathbf{p}_1 \\ \tilde{\mathbf{p}}_2 &= \mathbf{T}_2\mathbf{p}_2 \end{aligned} \tag{9.60}$$

.

In this way, it is possible to determine the compatible fundamental matrix $\tilde{\mathbf{F}}$

$$\mathbf{p}_2^\top\mathbf{F}\mathbf{p}_1 = \tilde{\mathbf{p}}_2^\top\mathbf{T}_2^{-\top}\mathbf{F}\mathbf{T}_1^{-1}\tilde{\mathbf{p}}_1 = \tilde{\mathbf{p}}_2^\top\tilde{\mathbf{F}}\tilde{\mathbf{p}}_1 = 0 \tag{9.61}$$

from which the original matrix $\mathbf{F} = \mathbf{T}_2^\top\tilde{\mathbf{F}}\mathbf{T}_1$ can then be derived.

9.4.2 Maximum Likelihood Estimation

When using SVD decomposition to strengthen the constraints, the resulting Fundamental (or Essential) matrix fully meets the requirements to be considered Fundamental (or Essential). However, it is merely a matrix that is more similar under a specific norm (in this case, Frobenius) to the one obtained from the linear system.

Therefore, this solution is not optimal either, as it does not account for how the error propagates from the input points within the transformation: it is still fundamentally an algebraic solution rather than a geometric one.

A preliminary technique that minimizes geometric error involves leveraging the distance between points and the epipolar lines generated through the Fundamental matrix (*epipolar distance*).

Even intuitively, the distance between a point $\mathbf{p}_2$ and the epipolar line $\mathbf{F}\mathbf{p}_1$ can be used as a metric to estimate the geometric error:

$$d\left(\mathbf{p}_2, \mathbf{F}\mathbf{p}_1\right) = \frac{\left|\mathbf{p}_2^\top\left(\mathbf{F}\mathbf{p}_1\right)\right|}{\sqrt{\left(\mathbf{F}\mathbf{p}_1\right)_1^2 + \left(\mathbf{F}\mathbf{p}_1\right)_2^2}} \tag{9.62}$$

where $(.)_i$ denotes the i-th component of the vector (see section 1.5.3 for the equation of the point-line distance). The lower the distance, the more the matrix $\mathbf{F}$ effectively serves as the matrix that relates the corresponding points.

Since it is possible to compute this error for both the first and the second image, it is appropriate to minimize both contributions together. Through this metric, it is possible to define a cost function that minimizes the error symmetrically (*symmetric transfer error*) between the two images:

$$\min_{\mathbf{F}} \sum_i \left(d\left(\mathbf{p}_{1,i}, \mathbf{F}\mathbf{p}_{2,i}\right)^2 + d\left(\mathbf{p}_{2,i}, \mathbf{F}^\top\mathbf{p}_{1,i}\right)^2 \right) \tag{9.63}$$

In this case as well, one can seek a solution with 8 unknowns, but to find a robust solution, it is necessary to constrain $\mathbf{F}$ to have rank 2.

Alternatively to the *Symmetric Transfer Error*, the first-order approximation of the distance between the points and the function is often used in the literature (*Sampson error*, section 3.3.7). It is possible to define an approximate distance between the homologous image points $(\mathbf{p}_1, \mathbf{p}_2)$ and the variety $\hat{\mathbf{p}_2}^\top \mathbf{F} \hat{\mathbf{p}_1} = 0$ through the metric

$$r(\mathbf{p}_1, \mathbf{p}_2, \mathbf{F}) \approx \frac{\left|\mathbf{p}_2^\top \mathbf{F} \mathbf{p}_1\right|}{\sqrt{(\mathbf{F}\mathbf{p}_1)_1^2 + (\mathbf{F}\mathbf{p}_1)_2^2 + (\mathbf{F}^\top \mathbf{p}_2)_1^2 + (\mathbf{F}^\top \mathbf{p}_2)_2^2}} \tag{9.64}$$

where $(.)_i$ again denotes the i-th component of the vector. Using this approximate metric, while always maintaining the additional constraint $\det \mathbf{F} = 0$, it is possible to minimize

$$\min_{\mathbf{F}} \sum_{i=1}^{n} r(\mathbf{p}_{1,i}, \mathbf{p}_{2,i}, \mathbf{F})^2 \tag{9.65}$$

Both the *Symmetric Transfer Error* and the Sampson distance, although superior metrics compared to the algebraic estimate, do not yield the optimal estimator. The Maximum Likelihood Estimation (MLE) for the Fundamental matrix would be obtained by using a cost function of the form

$$\min_{\mathbf{F}} \sum_{i} \|\mathbf{p}_{1,i} - \hat{\mathbf{p}}_{1,i}\|^2 + \|\mathbf{p}_{2,i} - \hat{\mathbf{p}}_{2,i}\|^2 \tag{9.66}$$

denoting by $\hat{\mathbf{p}}_{1,i}$ and $\hat{\mathbf{p}}_{2,i}$ the exact points and by $\mathbf{p}_{1,i}$, $\mathbf{p}_{2,i}$ the corresponding measured points affected by zero-mean white Gaussian noise. The cost function (9.66) needs to be minimized under the constraint

$$\hat{\mathbf{p}}_{2,i}^\top \mathbf{F} \hat{\mathbf{p}}_{1,i} = 0 \tag{9.67}$$

and with additional constraints due to the nature of $\mathbf{F}$. In this case, the exact points $\hat{\mathbf{p}}_{1,i}$ and $\hat{\mathbf{p}}_{2,i}$ become part of the problem (auxiliary variables, *subsidiary variables*). However, introducing the points $\hat{\mathbf{p}}_{1,i}$ and $\hat{\mathbf{p}}_{2,i}$ as unknowns makes the problem unsolvable, as there would always be more unknowns than constraints.

To solve this problem, it is necessary to combine the issue of calculating the Essential or Fundamental matrix with that of three-dimensional reconstruction, and to set the three-dimensional coordinate of the observed point $\hat{\mathbf{x}}_i$ as the auxiliary variable, rather than the projections.

The Essential matrix can be obtained given the knowledge of the intrinsic parameters of the two sensors. In this case, it is indeed possible to exploit the nonlinear system that projects the auxiliary variable $\hat{\mathbf{x}}_i$ onto the respective observations from the two sensors:

$$\begin{aligned} \hat{\mathbf{p}}_{1,i} &\equiv \mathbf{K}_1 \hat{\mathbf{x}}_i \\ \hat{\mathbf{p}}_{2,i} &\equiv \mathbf{K}_2 (\mathbf{R}\hat{\mathbf{x}}_i + \mathbf{t}) \end{aligned} \tag{9.68}$$

where the matrix $\mathbf{R}$ can be expressed through a parameterization with 3 variables (see section A), while the vector $\mathbf{t}$ must be represented through a parameterization with 2 (the scale remains an unknown factor). By inserting the constraints (9.68) into equation (9.66), the objective of deriving the Essential matrix is transformed into that of directly obtaining the parameters relating to the two sensors. Finally, if required, once the relative pose between the sensors is obtained, it is possible to derive the Essential matrix by directly applying the definition (9.41).

When intrinsic parameters are not available, in the case of estimating the Fundamental matrix, it is not possible to perform a true three-dimensional reconstruction of the scene due to the lack of these parameters. However, it is possible to exploit fictitious perspective projections by setting $\mathbf{K}_1 = \mathbf{I}$ and obtaining constraints of the form:

$$\begin{aligned} \hat{\mathbf{p}}_{1,i} &\equiv \hat{\mathbf{x}}_i \\ \hat{\mathbf{p}}_{2,i} &\equiv \mathbf{P}\hat{\mathbf{x}}_i \end{aligned} \tag{9.69}$$

using the auxiliary variable $\hat{\mathbf{x}}_i$ directly, a coordinate that will therefore be known up to an affine transformation $\mathbf{K}_1^{-1}$, namely the intrinsic parameters of camera 1.

By incorporating the constraints (9.69) into the equation (9.66), the objective of deriving the Fundamental matrix is once again transformed into that of extracting the parameters of the projective matrix $\mathbf{P}$. Through the camera matrix $\mathbf{P} = [\mathbf{R}'|\mathbf{t}']$, a fictitious camera matrix, it is finally possible to derive $\mathbf{F}$ by directly applying the definition (9.41), where, however, the matrix $\mathbf{R}'$ is not a rotation matrix.

The maximum likelihood estimation of the fundamental matrix, corrected from a probabilistic standpoint, nonetheless requires a substantial amount of resources: in addition to the 12 global unknowns necessary to estimate $\mathbf{P}$ (compared to the 5 of the essential matrix), for each pair of points to be minimized, 3 additional unknowns are incorporated into the problem.

Finally, as a final warning, for the optimal estimation of matrices in the presence of potential *outliers* in the scene, techniques such as RANSAC are widely employed (see section 3.12).

9.4.3 Factorization of the Essential Matrix

In the previous sections, it has been demonstrated how, by utilizing at least 5 correspondences between homologous points in two images, it is possible to obtain the Essential matrix that encodes the relative pose between the two cameras. The Essential matrix can be factorized again into rotation and translation. This allows for the retrieval of the relative parameters of the involved cameras and, through this information, enables the execution of a three-dimensional reconstruction of the observed scene.

As suggested by Trivedi, from the definition of the essential matrix (9.41), it is easy to show that the symmetric matrix $\mathbf{E}\mathbf{E}^\top$ is independent of the rotation vector:

$$\mathbf{E}\mathbf{E}^\top = [\mathbf{t}]_\times [\mathbf{t}]_\times^\top = \begin{bmatrix} t_y^2 + t_z^2 & -t_x t_y & -t_x t_z \\ -t_y t_x & t_z^2 + t_x^2 & -t_y t_z \\ -t_z t_x & -t_z t_y & t_x^2 + t_y^2 \end{bmatrix} \tag{9.70}$$

From the matrix $\mathbf{E}\mathbf{E}^\top$, the translation vector $\mathbf{t}$ can be derived, keeping in mind that this vector is known up to a multiplicative factor (and therefore a sign), which can then be used to obtain $\mathbf{R}$.

The Essential matrix can also be directly factored through Singular Value Decomposition. Let $\mathbf{U}\mathbf{D}\mathbf{V}^\top$, where $\mathbf{D} = \text{diag}(1, 1, 0)$, the SVD of $\mathbf{E}$ (if this were not the case, it is still possible to project the matrix $\mathbf{E}$ into the space of Essential matrices, as described in section 9.4.1). Through this decomposition, the generating components of $\mathbf{E}$ can be extracted:

$$[\mathbf{t}]_\times = \mathbf{U}\left(\mathbf{R}_z^\top \mathbf{D}\right)\mathbf{U}^\top \quad \mathbf{R} = \mathbf{U}\mathbf{R}_z\mathbf{V}^\top | \mathbf{U}\mathbf{R}_z^\top\mathbf{V}^\top \tag{9.71}$$

where

$$\mathbf{R}_z^\top \mathbf{D} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{R}_z = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.72}$$

with $\mathbf{R}_z$ rotation around the axis z by an angle of $\frac{\pi}{2}$. It should be noted that $[\mathbf{t}]_\times \mathbf{t} = 0$ for every possible $\mathbf{t}$. It can be demonstrated that this is only possible when $\mathbf{t} = \mathbf{U}(0, 0, 1)^\top = \mathbf{u}_3$, the last column of the matrix $\mathbf{U}$.

The rotation matrix $\mathbf{R}$ thus presents two possible solutions that are rotated 180° with respect to the axis connecting the two pinholes. Since the vector $\mathbf{t}$ is known up to a multiplicative factor and the constraint $|\mathbf{t}| = 1$ does not allow us to determine the sign of the translation, there are also two additional alternatives for the factorization due to an ambiguity regarding the sign that $\mathbf{t}$ can assume. Therefore, there are 4 different plausible factorizations of an Essential matrix, and among these, the one that projects all points (or the majority) frontally with respect to both cameras must be selected.

9.4.4 Chirality and Reconstruction with Relative Poses

From the decomposition of the Essential matrix, up to a multiplicative factor, there are therefore 4 possible configurations (the two rotation matrices and the associated translation vectors) that can be recombined to obtain the original Essential matrix once again. To determine which decomposition is the correct one, the only method is to find the configuration that reconstructs the majority of the three-dimensional points appropriately, or more simply, the configuration that results in the majority of points having the camera coordinate z being positive.

Let $(\mathbf{R}, \mathbf{t})$ be a decomposition of the Essential matrix, and let $\mathbf{m}_1$ and $\mathbf{m}_2$ be the camera coordinates of two corresponding points. Define $\tilde{\mathbf{m}}_1 = (\tilde{u}_1, \tilde{v}_1, 1)$ and $\tilde{\mathbf{m}}_2 = (\tilde{u}_2, \tilde{v}_2, 1)$ such that

$$\tilde{\mathbf{m}}_1 = 1/z_1 \mathbf{m}_1 \qquad \tilde{\mathbf{m}}_2 = 1/z_2 \mathbf{m}_2 \tag{9.73}$$

represents the normalized camera coordinates of a pair of corresponding points, that is, It seems that you have provided a placeholder for a mathematical block but did not include any specific content to translate. Please provide the text or equations you would like translated, and I will be happy to assist you!

The objective is to derive the coordinates z_1 and z_2 through which, by evaluating their positivity, it can be inferred that the corresponding points are frontal with respect to the observer, thus allowing us to deduce the correctness of the decomposition of the Essential matrix. Utilizing the formalism (9.73), the equation (9.38) becomes

$$z_2 \tilde{\mathbf{m}}_2 = z_1 \mathbf{R} \tilde{\mathbf{m}}_1 + \mathbf{t} \tag{9.74}$$

. By solving the equation with z_1 as the unknown, we ultimately obtain

$$z_1 = \frac{t_x - \tilde{u}_2 t_z}{(\tilde{u}_2 \mathbf{r}_3 - \mathbf{r}_1) \cdot \tilde{\mathbf{m}}_1} = \frac{t_y - \tilde{v}_2 t_z}{(\tilde{v}_2 \mathbf{r}_3 - \mathbf{r}_2) \cdot \tilde{\mathbf{m}}_1} \tag{9.75}$$

having indicated with $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ the 3 rows of the matrix $\mathbf{R}$. The first equation uses the coordinate $\tilde{u}_2$ to derive z_1, while the second utilizes the coordinate $\tilde{v}_2$. In this way, the coordinate $\mathbf{m}_1$ is obtained, from which $\mathbf{m}_2$ can be immediately derived through equation (9.38), particularly

$$z_2 = \mathbf{r}_3 \cdot \mathbf{m}_1 + t_z \tag{9.76}$$

to assess the frontality of the other element of the pair.

It is worth noting that the solution to the problem could be obtained by directly solving the system (9.74) as if it were an overdetermined linear system with 2 unknowns and 3 equations (a similar approach to what has been discussed in section 9.3.1).

In both cases, an algebraic quantity is optimized, and therefore it will not be the maximum likelihood estimate of the three-dimensional point: unlike the algorithms discussed in section 9.3.1, this approach is indeed somewhat unsuitable for deriving precise world coordinates but is sufficient to verify that the choice of decomposition is correct.

It should always be noted that since the vector $\mathbf{t}$ is extracted from the Essential matrix, it is known up to a multiplicative factor; thus, the estimated points are known up to a multiplicative factor as well.

It is worth noting that this discussion is clearly generic and can be applied to the case of three-dimensional reconstruction when the relative positioning between sensors is known.

9.5 Noise Removal under Epipolar Constraints

As seen in section 9.3.1, triangulating points affected by noise leads to non-concurrent lines whose intersection does not minimize the residual in image coordinates (for instance, under the metric of Euclidean distance). We have also observed that the best estimate of the noise-free points minimizes the quantity in equation 9.66 under the epipolar constraint 9.67. However, so far, given the knowledge of the Essential/Fundamental matrix, this minimization has required the three-dimensional point as an auxiliary variable and an (iterative) optimization technique initialized, for example, by leveraging the triangulation with skew lines of the noise-affected points.

There exists a global nonlinear technique that allows for optimal triangulation (the estimation of image points) through a polynomial method [HS97] that requires finding the roots of a 6th degree polynomial. As more clearly discussed in [Lin10], optimal triangulation can be viewed as the following minimization problem:

$$\min \left(\delta\mathbf{m}_2^\top \delta\mathbf{m}_2 + \delta\mathbf{m}_1^\top \delta\mathbf{m}_1 \right) \tag{9.77}$$

subject to the epipolar constraint

$$\hat{\mathbf{m}}_2^\top \mathbf{E} \hat{\mathbf{m}}_1 = (\mathbf{m}_2 - \mathbf{S}^\top \delta\mathbf{m}_2)^\top \mathbf{E} (\mathbf{m}_1 - \mathbf{S}^\top \delta\mathbf{m}_1) = 0 \tag{9.78}$$

having defined $\delta\mathbf{m}_1 = \mathbf{S}(\mathbf{m}_1 - \hat{\mathbf{m}}_1)$ and $\delta\mathbf{m}_2 = \mathbf{S}(\mathbf{m}_2 - \hat{\mathbf{m}}_2)$ where

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{9.79}$$

is used to extract only the non-homogeneous components from the vector. As previously seen, the points $(\hat{\mathbf{m}}_1, \hat{\mathbf{m}}_2)$ are the estimates of the noise-free points while $(\mathbf{m}_1, \mathbf{m}_2)$ are the observed points.

This constrained minimization problem can be solved using the method of Lagrange multipliers:

$$\mathcal{L}(\delta\mathbf{m}_1, \delta\mathbf{m}_2, \lambda) = \delta\mathbf{m}_1^\top \delta\mathbf{m}_1 + \delta\mathbf{m}_2^\top \delta\mathbf{m}_2 - 2\lambda (\mathbf{m}_2 - \mathbf{S}^\top \delta\mathbf{m}_2)^\top \mathbf{E} (\mathbf{m}_1 - \mathbf{S}^\top \delta\mathbf{m}_1) \tag{9.80}$$

The gradient of the Lagrangian is set to zero at

$$\begin{aligned} &\hat{\mathbf{m}}_2^\top \mathbf{E} \hat{\mathbf{m}}_1 = (\mathbf{m}_2 - \mathbf{S}^\top \delta\mathbf{m}_2)^\top \mathbf{E} (\mathbf{m}_1 - \mathbf{S}^\top \delta\mathbf{m}_1) = 0 \\ &\delta\mathbf{m}_1 = \lambda \mathbf{S}\mathbf{E}^\top \left(\mathbf{m}_2 - \mathbf{S}^\top \delta\mathbf{m}_2 \right) = \lambda \mathbf{S}\mathbf{E}^\top \hat{\mathbf{m}}_2 = \lambda \mathbf{n}_1 \\ &\delta\mathbf{m}_2 = \lambda \mathbf{S}\mathbf{E} \left(\mathbf{m}_1 - \mathbf{S}^\top \delta\mathbf{m}_1 \right) = \lambda \mathbf{S}\mathbf{E}\hat{\mathbf{m}}_1 = \lambda \mathbf{n}_2 \end{aligned} \tag{9.81}$$

from which we obtain 5 constraints in 5 unknowns (the differential coordinates and λ). These constraints can be parameterized as a function of an auxiliary variable, leading to the famous sixth-degree equation. This approach is exactly the same whether using image points with the Fundamental matrix or camera points with the Essential matrix.

In [Lin10], sub-optimal, iterative techniques with low computational cost are also proposed, where the epipolar constraint is still satisfied at each iteration.

Once the image points unaffected by noise are obtained, it is possible to derive the three-dimensional point using any triangulation technique (skew lines in section 1.5.6 or the DLT in section 9.3.1). An alternative formulation [KK95], given two homologous points expressed in camera coordinates $\hat{\mathbf{m}}'$ and $\hat{\mathbf{m}}$, defines the three-dimensional point formed by the intersection of the optical rays as

$$\mathbf{x} = \frac{(\mathbf{t} \times \mathbf{R}\hat{\mathbf{m}}') \cdot \mathbf{z}}{\|\mathbf{z}\|^2} \tag{9.82}$$

where $\mathbf{z} = \hat{\mathbf{m}} \times \mathbf{R}\hat{\mathbf{m}}'$.

9.6 Homologous Points in Alternative Image Spaces

Through an appropriate *Warp-Table*, it is possible to transform the input image into an alternative form that still preserves the ability to reconstruct the scene in three dimensions. To also maintain the concept of disparity, meaning having homologous points along the same vertical coordinate between the two images of the stereoscopic pair, some additional constraints are necessary: the horizontal image coordinate must be a function of the x camera coordinate, while the vertical image coordinate must not be a function of the x. The function of the horizontal coordinate must be monotonic (typically increasing) in x, while the function of the vertical coordinate must be monotonically increasing in y.

A widespread parameterization is the polar one, where, to avoid confusion between the axes, the equation has been chosen as

$$\begin{aligned} x' &= \operatorname{atan} \frac{x}{\sqrt{y^2+z^2}} \\ y' &= \operatorname{atan} \frac{y}{z} \end{aligned} \tag{9.83}$$

which projects a camera point (x, y, z) onto an image point (x', y') (for comparison, I recall that the perspective equation has equations $x' = x/z$, $y' = y/z$). Given this projection, it is possible to write the inverse equation as

$$\begin{aligned} x &= \sin(x') \\ y &= \cos(x')\sin(y') \\ z &= \cos(x')\cos(y') \end{aligned} \tag{9.84}$$

This parametrization allows for the projection of all coordinates of a hemisphere (up to 180 degrees) into an image space, which the pin-hole model does not permit. Therefore, this parametrization is convenient for remapping Fish-Eye cameras. Through this parametrization, it is possible to write an equation similar to that in (9.21) to triangulate two image points.

9.7 Visual Odometry and Bundle Adjustment

Visual Odometry aims to determine the relative pose of a camera (or a stereo pair) moving through space by analyzing two sequential images. The problem of visual odometry for a single camera is typically solved by calculating the essential matrix and subsequently decomposing it. In this case, as previously mentioned, it is not possible to ascertain the scale of the motion, but only to relate the various movements to one another. The situation is different when a stereo pair is available.

Given a series of temporal observations of world points obtained from the three-dimensional reconstruction $(\mathbf{x}_i, \mathbf{x}'_i)$, it is possible to linearly derive a rigid transformation $(\mathbf{R}, \mathbf{t})$ that transforms the world points at time t to time t' such that they can be expressed with an equation of the form:

$$\mathbf{x}'_i = \mathbf{R}\mathbf{x}_i + \mathbf{t} \tag{9.85}$$

This approach is general and does not depend on the specific sensor used to obtain the points.

The rigid body transformation performed by the pair of sensors can be derived by minimizing the quantity:

$$\sum_i \|\mathbf{x}'_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}\|^2 \tag{9.86}$$

The 12-parameter solution, derived from overdetermined data, will find an absolute minimum but is not the optimal estimator, as it minimizes an algebraic quantity and does not guarantee that the rotation matrix is orthonormal. Starting from the linear solution, the use of a nonlinear minimizer (for example, Levenberg-Marquardt, see section 3.3.5) on the cost function given by equation (9.86) allows for a more precise determination of the 6 parameters (3 rotations and 3 translations). This algorithm is referred to as *3D-to-3D* because it derives the motion from pairs of three-dimensional points. As an alternative to the linear solution, a closed-form solution is also possible [Hor87].

The approach presented now is general but poorly suited for the case of world points obtained from a three-dimensional reconstruction from images. The cost function shown indeed optimizes quantities in world coordinates rather than in image coordinates: the noise on the image points propagates non-linearly during the triangulation phase, and therefore it is only in image coordinates that one can assume the noise in point detection to be Gaussian with zero mean. It is thus not possible to create a maximum likelihood estimator using only the points in world coordinates. A more refined approach is the one referred to as *3D-to-2D*, where the goal is to minimize the reprojection of a point from the past in image coordinates:

$$\sum_i \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|^2 \tag{9.87}$$

where $\hat{\mathbf{p}}_1$ is the rotated and translated projection of the three-dimensional point $\mathbf{x}_i$ obtained from the previous frame. This problem is also known as *perspective from n points* (*PnP*) as it is very similar to the previously discussed problem of camera calibration in a static environment.

Clearly, this approach is also affected by the fact that the three-dimensional point $\mathbf{x}_i$ is not a given of the problem but is known with a certain amount of error. For this reason, it is necessary to take an additional step by minimizing both errors

in image coordinates (this is the *Maximum Likelihood Estimation*):

$$\sum_i \|\mathbf{p}_1 - \hat{\mathbf{p}_1}\|^2 + \|\mathbf{p}_2 - \hat{\mathbf{p}_2}\|^2 + \|\mathbf{p}'_1 - \hat{\mathbf{p}'_1}\|^2 + \|\mathbf{p}'_2 - \hat{\mathbf{p}'_2}\|^2 \tag{9.88}$$

with the constraints set as $\hat{\mathbf{p}_1} = \mathbf{K}_1\mathbf{R}_1(\hat{\mathbf{x}}_i - \mathbf{t}_1)$, $\hat{\mathbf{p}_2} = \mathbf{K}_2\mathbf{R}_2(\hat{\mathbf{x}}_i - \mathbf{t}_2)$, $\hat{\mathbf{p}'_1} = \mathbf{K}_1\mathbf{R}_1(\hat{\mathbf{x}'}_i - \mathbf{t}_1)$, and $\hat{\mathbf{p}'_2} = \mathbf{K}_2\mathbf{R}_2(\hat{\mathbf{x}'}_i - \mathbf{t}_2)$, to which the constraint from equation (9.85) is added, while keeping the unknown regarding the actual position of point $\hat{\mathbf{x}}_i$ in both reference frames. In this way, both the displacement performed by the cameras and the three-dimensional coordinate of each individual *feature* in the world are minimized. Even in this case, the solution to the maximum likelihood problem requires solving a nonlinear problem of considerable size. In the case of a rectified stereo pair, the cost function can be significantly simplified.

Visual odometry is a dead-reckoning algorithm and is therefore subject to drift. It is possible to extend these considerations to the case where multiple time instances are involved in the minimization process rather than just two. In this scenario, the discussion becomes complex as one attempts to minimize drift errors when composing the various transformations. A tutorial that addresses these topics is [SF11].

When addressing the problem from a Bayesian perspective, utilizing equation (9.88), and intending to process all frames simultaneously, the term *Bundle Adjustment* is preferred over visual odometry.

The concept of *Bundle Adjustment*, initially introduced in photogrammetry and later adopted by *Computer Vision* (see the excellent survey [TMHF00]), refers to a multivariable minimization aimed at simultaneously achieving a three-dimensional reconstruction, the relative poses of the cameras in a sequence of images, and potentially the intrinsic parameters of the cameras themselves.

This is an extension of the non-linear techniques that estimate parameters by minimizing a suitable cost function based on the reprojection errors of the identified points, in the same form as equation (9.88).

Since the same *feature* can be observed from different images, the estimation process conditions all poses, and consequently, the problem cannot be decomposed into n separate visual odometry problems: all images in the sequence must be minimized simultaneously. For this reason, the *Bundle Adjustment* problem is a high-dimensional problem, certainly non-convex, which requires non-trivial optimization and employs sparse minimization to conserve memory and enhance accuracy.

An alternative approach to *Bundle Adjustment*, which is certainly not the best maximum likelihood estimator but introduces fewer unknowns, is *Pose Graph Optimization* [GKSB10]. This method utilizes information from the same pose obtained from multiple paths, or by identifying *Loops*, allowing for the optimization of only the poses relative to those obtained from visual odometry. Let $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a parameter vector where the element $\mathbf{x}_i$ represents the pose of the i-th node. Let $\mathbf{z}_{ij}$ and $\mathbf{\Omega}_{ij}$ denote the measurement and the precision matrix of the *virtual* observation of the relative pose between nodes i and j. The objective is to obtain an estimate of the parameters $\mathbf{x}$ given the virtual observations $\mathbf{z}_{ij}$. Since the relative poses are obtained by comparing two absolute poses, the parameters to be estimated can be defined through the cost function

$$\mathbf{e}_{ij} = \mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}(\mathbf{x}_i, \mathbf{x}_j) \tag{9.89}$$

, which measures the error between the measured virtual relative pose $\mathbf{z}_{ij}$ and the predicted one $\hat{\mathbf{z}}_{ij}(\mathbf{x}_i, \mathbf{x}_j)$ given the configurations $\mathbf{x}_i$ and $\mathbf{x}_j$ to be evaluated for nodes i and j, respectively. By leveraging the information on the precision of the estimate of the individual relative pose, it is possible to define a global cost function

$$F(\mathbf{x}) = \sum_{<i,j>} \mathbf{e}_{ij}^\top \mathbf{\Omega}_{ij} \mathbf{e}_{ij} \tag{9.90}$$

, which is essentially the sum of the individual Mahalanobis distances between all pairs (i, j) for which a relative pose measurement has been made. The function $F(\mathbf{x})$, minimized with respect to $\mathbf{x}$, provides the best estimate of the absolute poses of the problem, all without involving the individual elements that comprise the single real observation.

9.8 Reconstruction, Representation, and Rendering of Three-Dimensional Environments

Point clouds or meshes are the most common three-dimensional primitives used for the memory representation of environments and objects. These approaches had the advantage or disadvantage of distinctly separating the reconstruction phase from the rendering phase.

Recently, both the *Neural Radiance Field* (NeRF) [MST+20] and, more notably, the *3D Gaussian Splatting* methods have provided a significant boost in this field, harmonizing the reconstruction and rendering processes.

9.8.1 Spherical Harmonics

A key point in color representation comes from the use of Spherical Harmonics (SH). The spherical harmonics are solutions to Laplace's equation in spherical coordinates, orthogonal and forming a complete basis for functions defined on a sphere.

This means that any function $L(\theta, \phi)$ can be expanded into a series of spherical harmonics:

$$L(\mathbf{d}) = L(\theta, \phi) = \sum_{l=0} \sum_{m=-l}^{m=l} k_l^m Y_l^m (\theta, \phi) \tag{9.91}$$

This entire class of functions can be generated by a single formula, selecting $l \geq 0$ for the degree of the harmonic and $-l \leq m \leq l$ for the order:

$$Y_l^m(\theta, \phi) = \frac{(-1)^l}{2^l l!} \sqrt{\frac{(2l+1)(l+m)!}{4\pi(l-m)!}} e^{im\phi} P_l^m (\cos\theta) \tag{9.92}$$

where $P_l^m (\cos\theta)$ are the associated Legendre polynomials [YLT+21]. In the case of $l = 0$, the first harmonic is a constant on the sphere and is equal to $Y_0^0 = \frac{1}{2}\sqrt{\frac{1}{\pi}} \approx 0.282$.

In computer graphics, they are used to represent lighting information in a compact and efficient manner. Spherical Harmonics decompose the incident light into a set of coefficients, each associated with a different harmonic. These coefficients capture the characteristics of light, such as intensity and color, along different directions of the spherical surface.

The idea is to select a maximum degree of l and express each color component (red, green, blue) as a linear combination of spherical harmonics, using (θ, ϕ) as the optical radius that connects the point to the observer.

9.8.2 Rendering

In many respects, point-based α-blending, volumetric rendering *NeRF-style*, and *Gaussian Splatting* share the low-level component for rendering the scene: the color at a pixel in the image is approximated by integrating samples along the ray that passes through this pixel. The final color is a weighted sum of the colors of the 3D points sampled along this ray, weighted according to the transmittance.

The color C of a pixel is the integral of the various densities encountered along the optical ray $\mathbf{r} = \mathbf{o} + t\mathbf{d}$ over the interval $[t_n, t_f]$:

$$C = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\,\mathbf{c}(\mathbf{r}(t), \mathbf{d})\,dt \tag{9.93}$$

where

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))\,ds\right) \tag{9.94}$$

The function $T(t)$ denotes the accumulated transmittance along the ray from t_n to t, that is, the probability that the ray travels from t_n to t without hitting any other particles.

Integrals can be transformed into sums of various contributions. The color C of a pixel can therefore be viewed as a summation of different contributions:

$$C = \sum_{i=1}^{N} T_i \left(1 - \exp(-\sigma_i \delta_i)\right) \mathbf{c}_i \tag{9.95}$$

where $T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$ and $\delta_i = t_{i+1} - t_i$.

This representation reduces to the classic α-blending using $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$. In this way, the transmittance can also be expressed as $T_i = \prod_{j=1}^{i-1}(1 - \alpha_i)$.

In typical neural point-based approaches, the color C of a pixel is obtained by blending the $\mathcal{N}$ ordered points that underlie the pixel itself:

$$C = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{9.96}$$

9.8.3 Neural Representation and Radiance Fields

9.8.4 Gaussian Splattering in 3D

The concept of 3D Gaussian splattering is to represent the image as a mixture of three-dimensional Gaussians. The 3D Gaussians are based on the three-dimensional extension of one-dimensional Gaussians. Three-dimensional Gaussians are defined by a covariance matrix Σ (in world coordinates) and centered at the point (mean) μ:

$$G(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1} (\mathbf{x}-\mu)} \tag{9.97}$$

To be drawn, this Gaussian must first be transformed into camera coordinates through a rigid transformation $\mathbf{W}$ and finally projected into image coordinates. However, one can consider an approximation by drawing a two-dimensional Gaussian in image space. In 2D space, the covariance Σ' becomes

$$\Sigma' = \mathbf{J}\mathbf{W}\Sigma\mathbf{W}^\top\mathbf{J}^\top \tag{9.98}$$

where $\mathbf{W}$ is the only rotational part of the transformation, and using, as an approximation, the Jacobian $\mathbf{J}$ of the perspective projection calculated at the rotated and translated point in camera $(x, y, z)^\top$. For example, in the case of a pinhole camera projection:

$$\mathbf{J} = \begin{bmatrix} k_u/z & 0 & -\frac{k_u x}{z^2} \\ 0 & k_v/z & -\frac{k_v y}{z^2} \end{bmatrix} \tag{9.99}$$

The matrix Σ' therefore has a dimensionality of 2×2 [ZPvBG01] and is comparable to the matrix of a 2D Gaussian.

In [KKLD23], a further step is taken: since it is challenging to parameterize a covariance matrix (which is positive semi-definite), it is based on the fact that the matrix Σ represents an ellipsoid, allowing for a minimal parameterization instead of using all the terms of the matrix as unknowns. The idea is to utilize a scaling matrix $\mathbf{S}$ (3 DOF) and a rotation matrix $\mathbf{R}$ (another 3 DOF, typically represented by a quaternion, see section A.3):

$$\Sigma = \mathbf{RSS}^\top \mathbf{R}^\top \tag{9.100}$$

thus parameterizing each Gaussian with 6 DOF. It is noteworthy that $\mathbf{SS}^\top = \text{diag}\left(s_x^2, s_y^2, s_z^2\right)$.

Finally, each point can be associated with an RGB color or spherical harmonics (*Spherical Harmonics SH*), in addition to the opacity parameter α, which is similar to that of NeRF. Practically, the Gaussians are rendered from the nearest to the farthest until the opacity reaches saturation.

9.8.5 2D Gaussian Splattering

The splattering of 2D Gaussians can be seen as a simpler alternative to 3D Gaussians, and from a historical perspective, they had already been introduced earlier.

2D Gaussians are represented by a central point $\mathbf{p}_k$, two unit tangent vectors $(\mathbf{t}_u, \mathbf{t}_v)$, and a scaling factor $\mathbf{S} = (s_u, s_v)$ that controls the variance in two dimensions of the Gaussian.

The orientation of the 2D Gaussian can be organized in a rotation matrix 3×3 (parameterizable as a classical rotation in 3D) after defining $t_w = t_u \times t_v$ and the scaling factors in a diagonal matrix $\mathbf{S} = \text{diag}(s_u, s_v, 0)$.

The 2D Gaussian is therefore defined on a tangent plane described by the equation

$$P(u, v) = \mathbf{p}_k + s_u \mathbf{t}_u u + s_v \mathbf{t}_v v = \mathbf{H}(u, v, 1, 1)^\top \tag{9.101}$$

after defining the homographic matrix $\mathbf{H} = \begin{bmatrix} s_u \mathbf{t}_u & s_v \mathbf{t}_v & 0 & \mathbf{p}_k \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{RS} & \mathbf{p}_k \\ 0 & 1 \end{bmatrix}$. To each point (u, v) in the plane coordinates, there is clearly associated a Gaussian described by the equation $e^{\frac{u^2+v^2}{2}}$.

Appendix A

Rotation Matrices

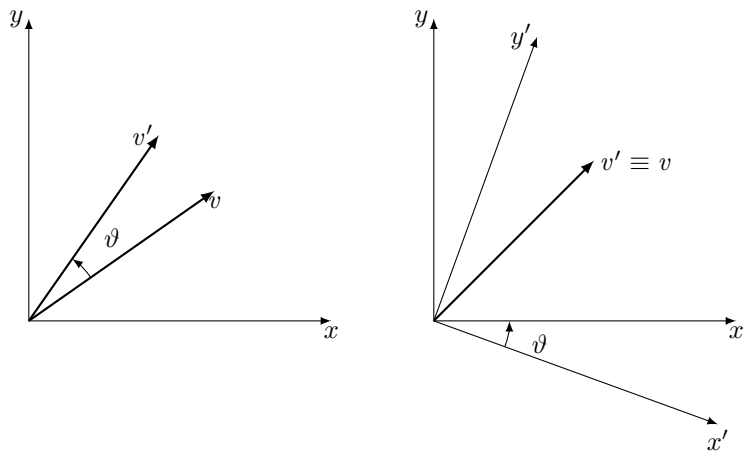


Figure A.1: Representation of the application of a rotation to a vector and a coordinate system: on the left, an *Inner/Active/Alibi Transformation*, and on the right, an *Outer/Passive/Alias Transformation*.

Rotations are isometric transformations of Euclidean space, meaning they transform vectors while preserving their length and leave a set of points unchanged in space, corresponding to a hyperplane (the *center of rotation* in the two-dimensional case or an *axis of rotation* in the three-dimensional case).

The set of all Rotation Matrices $SO(n)$ in $\mathbb{R}^n$ is defined as *Special Orthogonal*

$$SO(n) = \left\{ \mathbf{R} \in \mathbb{R}^n : \mathbf{R}\mathbf{R}^\top = \mathbf{R}^\top\mathbf{R} = \mathbf{I}, \det \mathbf{R} = +1 \right\} \tag{A.1}$$

or $\mathbf{R}^{-1} = \mathbf{R}^\top$.

There are two possible conventions for defining a rotation matrix: some authors prefer to write the matrix that transforms from sensor coordinates to world coordinates, while others prefer the opposite. The rotation matrix itself serves a dual purpose: it can indicate a rotation within a reference frame (*Active* or *Alibi*), or it can represent the transformation of coordinates from one reference frame to a second reference frame (*Passive* or *Alias*).

In this book, matrices are primarily used to represent changes of basis, and whenever possible, the source and destination reference systems are clearly highlighted.

To discuss rotation matrices and make some interesting observations, it is convenient to start by analyzing the two-dimensional case, as illustrated in Figure A.1.

It can be verified that $SO(2)$ has a single degree of freedom. The matrix $\mathbf{R}_\vartheta$, which represents a two-dimensional rotation, can be expressed in the form

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{bmatrix} \tag{A.2}$$

.

As can be seen from Figure A.1, when discussing a rotation by an angle ϑ, the same transformation can be perceived in different ways, depending on the reference frame in which the observer is positioned. The matrix $\mathbf{R}_\theta$ allows for the counterclockwise rotation of a vector (with respect to the origin of the reference frame) by an angle ϑ (left figure in A.1)[1]. The matrix in the form of (A.2) not only rotates a vector counterclockwise but also allows for the determination of the so-called "world" coordinates of a point, given the "sensor" coordinates and knowing that this sensor is rotated by an angle

[1] As previously mentioned, it is important to note that the inverse/transposed transformation, namely the matrix generated by angle $-\vartheta$, may be referred to in the literature as the "rotation matrix" and is also denoted by the letter $\mathbf{R}$.

ϑ (right-hand rule) in the "world" reference frame. Therefore, the matrix (A.2) facilitates the conversion from "sensor" coordinates to "world" coordinates, while the inverse of this matrix enables the transition from "world" coordinates back to "sensor" coordinates.

The distinction between *Inner/Active/Alibi Transformation* and *Outer/Passive/Alias Transformation* is another way to describe the difference between rotations. These terms are often used in mathematical and physical contexts to clarify whether a transformation acts on the reference frame itself (the reference frame is rotated or translated while the objects remain fixed in space, hence *alias* or *passive*) or on the objects within a fixed reference frame (the objects are rotated or translated while the reference frame remains unchanged, hence *alibi* or *active*).

The rotation matrix is also referred to as the Direction Cosine Matrix (DCM) because the columns of the transformation matrix correspond to the matrices of the coefficients of the old basis vectors expressed in terms of the new basis.

In this book, by working primarily with sensors (rather than robotic arms), all matrices are effectively change of basis matrices, as the main objective is to determine the coordinates of a point as perceived by a sensor in the higher reference frame, or vice versa.

Transitioning to the three-dimensional case complicates matters further: there are infinite parametrizations to express a rotation starting from 3 parameters $\mathbf{so}(3)$.

A rotation can be defined, for instance, as a composition of three elementary rotations around one of the three axes. However, since matrix multiplication is non-commutative, there are 24 distinct ways to combine these three matrices. The combinations of matrices are referred to as Euler sequences, followed by three numbers indicating the order of combination of the rotations: 1 for the axis x, 2 for the axis y, and 3 for the axis z.

In robotics, the representation of Euler angles (sequence ZYZ) or Tait-Bryan angles (Euler sequence 321 or ZYX) is widely used; see the following section A.1 for details[2]. In Italian literature, the six groups (XYZ, YZX, ZXY, XZY, ZYX, YXZ) are referred to as Cardano angles.

However, this system of angles presents some singularities that limit its use. Alternatively, the syntax proposed by Rodrigues (section A.2) or quaternions (section A.3) can be employed to overcome this issue.

Due to the non-commutative nature of matrix multiplication, there is an additional level of ambiguity in three-dimensional space stemming from the order in which Euler angles are described, as rotations can be defined as extrinsic or intrinsic:

Extrinsic Rotations These refer to rotations about fixed axes that coincide with the initial reference frame of the rotating object. The reference axes remain unchanged during the rotations. Extrinsic rotation sequences are specified using notations such as x-y-z.

Intrinsic Rotations These refer to rotations about moving axes that are attached to the rotating object. These axes change position after each rotation. Intrinsic rotation sequences are specified using notations such as x-y'-z", where the primes indicate the new subsequent reference frames.

An extrinsic rotation is equivalent to an intrinsic rotation (and vice versa) but with the order of composition of the transformations reversed (e.g., an extrinsic transformation z-y-x is equivalent to an intrinsic transformation x-y'-z").

Regardless of the geometric interpretation assigned to the rotation matrix, it is still possible to make several observations.

As previously mentioned, the definition of the matrix $\mathbf{R}$ in the pin-hole camera equation has been established, both for convenience and tradition, in such a way that it does not rotate a vector (which would represent a conversion from "sensor" coordinates to "world" coordinates). Instead, it inversely removes the rotation of world points by knowing the orientation of the camera itself, thus allowing for the conversion from "world" coordinates to "camera" coordinates.

Deriving an expression for the matrix $\mathbf{R}$ in the form expressed in the pin-hole camera model means finding a matrix that transforms a point from "world" coordinates to "camera" coordinates. This requires the use of the inverse of the rotation matrices that will be discussed in the following sections.

Let us consider a generic rotation ${}^{w}\mathbf{R}_{b}$ that transforms from local, moving, "sensor" coordinates (referred to as *body coordinates* in the general case) to global, fixed, "world" coordinates: the matrix $({}^{w}\mathbf{R}_{b})^{-1} = {}^{b}\mathbf{R}_{w}$ will thus be the matrix that converts from world coordinates to sensor coordinates.

Since the camera/image reference system is a *Left-Bottom-Front* system (with X increasing to the right, Y increasing downward, and Z representing depth as shown in figure 8.3), which differs from the *Front-Left-Up* sensor/world reference system (with Z increasing upward, X representing depth, and Y increasing to the left as illustrated in figure 8.4), it is necessary to define a matrix

$$ {}^{c}\mathbf{\Pi}_{b} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \tag{A.3} $$

for axis permutation. The permutation matrix has a determinant of $+1$, thus it is still a rotation that preserves the chirality of space (transforming right-handed systems into right-handed systems).

[2]Roll-Pitch-Yaw (RPY) angles, or simply Roll-Pitch-Yaw, are often utilized in robotics and aeronautics; however, to create confusion, this is an intrinsic representation and is equivalent to what are sometimes referred to as Euler angles ZYX, where an extrinsic representation is used.

When working in the aerospace or naval fields, it may be necessary to transition from a camera/image system to a *Front-Right-Down* system (for example, the NED). In this situation, the permutation matrix is

$$ {}^{c}\mathbf{\Pi}_{b'} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{A.4} $$

Under these considerations, the matrix $\mathbf{R}$ that converts from "world" to "camera," a formalism typically used in the pin-hole camera equation, is expressed as

$$ \mathbf{R} = {}^{c}\mathbf{R}_{w} = {}^{c}\mathbf{\Pi}_{b} \left({}^{w}\mathbf{R}_{b} \right)^{-1} \tag{A.5} $$

A.1 Tait-Bryan Angles

One way to define the rotation matrix in three dimensions is by composing rotations about the three principal axes of the reference frame.

Let us define ϑ as the pitch angle, γ as the yaw angle, and ρ as the roll angle, which are the orientation angles of the sensor with respect to the world reference frame[3] These angles and this nomenclature are referred to as *Tait-Bryan Angles*, *Cardan Angles* (after Gerolamo Cardano), or *nautical angles*.

Below, the matrices will be presented (as referenced in example [LaV06]) that convert a vector from sensor coordinates to world coordinates through angles that represent the orientation of the sensor with respect to the world itself. These are the same matrices that rotate a vector in a counterclockwise direction with respect to the various axes of the reference system.

The axes of this reference system are those shown in figure 8.4. However, it should be noted that for terrestrial vehicles and ships, a reference system different from the aeronautical one is preferred.

The rotation matrix for the *roll* angle ρ (around the X axis) is given by:

$$ \mathbf{R}_{x} = \mathbf{R}_{\rho} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\rho & -\sin\rho \\ 0 & \sin\rho & \cos\rho \end{bmatrix} \tag{A.6} $$

The rotation matrix for the *pitch* angle ϑ (around the Y axis) is given by:

$$ \mathbf{R}_{y} = \mathbf{R}_{\vartheta} = \begin{bmatrix} \cos\vartheta & 0 & \sin\vartheta \\ 0 & 1 & 0 \\ -\sin\vartheta & 0 & \cos\vartheta \end{bmatrix} \tag{A.7} $$

The rotation matrix for the *yaw* angle γ (around the Z axis) is given by:

$$ \mathbf{R}_{z} = \mathbf{R}_{\gamma} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.8} $$

(La Valle [LaV06], pp. 80-81).

As stated in the previous section, the composition of rotations is not commutative, and it is necessary to make a choice.

In the aeronautical field, the *Roll-Pitch-Yaw* (RPY) convention is suggested. Under this particular convention, the change of basis matrix (alias) is constructed as ${}^{w}\mathbf{R}_{b} = \mathbf{R}_{z}\mathbf{R}_{y}\mathbf{R}_{x}$[4] that is, by performing the multiplications, It should be noted that this matrix transforms points from the moving coordinates "sensor" (body coordinates in the generic case) to the fixed coordinates "world".

In the specific case where the sensor is a pin-hole camera, using this convention and considering equation (A.5), the rotation matrix $\mathbf{R}$ of the pin-hole camera that converts from "world" coordinates Front-Left-Up to "camera" coordinates can be expressed as a product of

$$ {}^{c}\mathbf{R}_{w} = {}^{c}\mathbf{\Pi}_{b}\mathbf{R}_{\rho}^{-1}\mathbf{R}_{\vartheta}^{-1}\mathbf{R}_{\gamma}^{-1} \tag{A.9} $$

that is

$$ \begin{bmatrix} -\cos\gamma\sin\theta\sin\rho + \sin\gamma\cos\rho & -\sin\gamma\sin\theta\sin\rho - \cos\gamma\cos\rho & -\cos\theta\sin\rho \\ -\cos\gamma\sin\theta\cos\rho - \sin\gamma\sin\rho & -\sin\gamma\sin\theta\cos\rho + \cos\gamma\sin\rho & -\cos\theta\cos\rho \\ \cos\gamma\cos\theta & \sin\gamma\cos\theta & -\sin\theta \end{bmatrix} \tag{A.10} $$

It should be emphasized that the matrix ${}^{c}\mathbf{R}_{w}$, expressed as in formula (A.9), is the matrix that "removes" the rotation of a sensor at those specific positioning angles and thus transforms from "world" coordinates to "camera" coordinates. In

[3] Note that there is no universally accepted notation for the Greek letters associated with the three angles. For example, one might find ϕ for the yaw angle and ψ for the roll angle.

[4] The intrinsic z-y'-x" sequence (the use of primes emphasizes this type of transformation) would instead generate $\mathbf{R} = \mathbf{R}_{x}\mathbf{R}_{y}\mathbf{R}_{z}$. To add further confusion, the x-y'-z" sequence is known as Roll-Pitch-Yaw (or Roll-Pitch-Yaw XYZ), while the z-y'-x" (intrinsic) sequence is commonly referred to as Yaw-Pitch-Roll (or Roll-Pitch-Yaw ZYX).

contrast, it is common in the literature to refer to the rotation matrix as the one that converts from "sensor" coordinates to "world" coordinates.

It is interesting to note that from a purely graphical perspective, the columns of the inverse/transposed matrix of matrix (A.10), which allows for the transformation of points from camera coordinates to world coordinates, facilitate the easy drawing of the axes and thus graphically represent the orientation of the camera.

A.1.1 Euler Angles

The Euler sequence (ZYZ) is based on the succession of three elementary rotations:

$$\mathbf{R}_{ZYZ}(\rho, \vartheta, \gamma) = \mathbf{R}_z(\rho)\mathbf{R}_y(\vartheta)\mathbf{R}_z(\gamma) \tag{A.11}$$

A.2 Axis-Angle Parameterization

Every rotation is equivalent to a rotation around an axis (of rotation) by a certain amount of angular displacement. From this premise, the Rodrigues rotation formula or Axis-Angle Parameterization is derived. The Rodrigues formulation aims to address the intrinsic singularity issues present in the Tait-Bryan and Euler formulations (where different combinations of values represent the same rotation matrix), while also providing a geometric and concise representation of rotation.

The rotation formula proposed by Rodrigues is composed of a unit vector $\mathbf{k}$ and an angle ϑ, which together allow for the representation of a rotation of points in space by an angle ϑ, around the axis defined by the vector $\mathbf{k}$, with a positive direction according to the right-hand rule.

It is possible to convert an axis and angle into a rotation matrix using a compact equation proposed by Rodrigues:

$$\mathbf{R} = \mathbf{I} + \sin\vartheta[\mathbf{k}]_\times + (1 - \cos\vartheta)(\mathbf{k}\mathbf{k}^\top - \mathbf{I}) \tag{A.12}$$

(this is one of the many representations available in the literature) which, when the terms are expanded, corresponds to the rotation matrix

$$\mathbf{R} = \begin{bmatrix} c + k_x^2(1-c) & k_xk_y(1-c) - k_zs & k_ys + k_xk_z(1-c) \\ k_zs + k_xk_y(1-c) & c + k_y^2(1-c) & -k_xs + k_yk_z(1-c) \\ -k_ys + k_xk_z(1-c) & k_xs + k_yk_z(1-c) & c + k_z^2(1-c) \end{bmatrix} \tag{A.13}$$

where $s = \sin\vartheta$ and $c = \cos\vartheta$. When $\vartheta = 0$, that is, in the absence of rotation, the matrix reduces to the identity.

The inverse formulation is also extremely compact and is given by:

$$\begin{aligned} \vartheta &= \cos^{-1}\left(\frac{\operatorname{trace}\mathbf{R} - 1}{2}\right) \\ \mathbf{k} &= \frac{1}{2\sin\vartheta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \end{aligned} \tag{A.14}$$

Since $\mathbf{k}$ and ϑ are essentially 4 parameters, a generic vector $\mathbf{w} = \vartheta\mathbf{k}$ is typically used to represent a rotation in the Rodrigues formulation, and the substitutions are made as follows:

$$\begin{aligned} \mathbf{k} &= \frac{\mathbf{w}}{\|\mathbf{w}\|} \\ \vartheta &= \|\mathbf{w}\| \end{aligned} \tag{A.15}$$

to accurately represent the transformation from $\mathbf{so}(3)$ to $SO(3)$.

A.2.1 Infinitesimal Rotations

The compact definition $\mathbf{w} = \vartheta\mathbf{k}$ combined with Rodrigues' formula allows for the expression of infinitesimal rotations in a manner that is very straightforward to compute.

If we send ϑ to infinitesimals, the formula (A.12) can be approximated as

$$\mathbf{R} \approx \mathbf{I} + \sin\vartheta[\mathbf{k}]_\times \approx \mathbf{I} + [\mathbf{w}]_\times = \begin{bmatrix} 1 & -w_z & w_y \\ w_z & 1 & -w_z \\ -w_y & w_z & 1 \end{bmatrix} \tag{A.16}$$

A.3 Quaternions

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))

Quaternions are an attempt to extend complex numbers to a higher dimension. This formulation was first proposed by Sir William Rowan Hamilton. They are represented by a vector of $\mathbb{R}^4$ in the form of

$$\mathbf{q} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} = q_w + q_x i + q_y j + q_z k \tag{A.17}$$

, sometimes also referred to as $\mathbf{q} = \begin{pmatrix} q_0 & q_1 & q_2 & q_3 \end{pmatrix} = \begin{pmatrix} q_1 & q_2 & q_3 & q_4 \end{pmatrix}$. Quaternions have different properties compared to ordinary four-dimensional vectors (such as homogeneous coordinates). The quaternion (A.17) can be viewed as composed of a vector part $\mathbf{v} \in \mathbb{R}^3$ and a scalar part q_w:

$$\mathbf{q} = \begin{bmatrix} q_w \\ \mathbf{v} \end{bmatrix} \tag{A.18}$$

q_w is defined as the scalar part (or real component), while q_x, q_y, q_z are the vector components (or imaginary parts). A quaternion with only the scalar part is called *real*, while a quaternion with only the vector part is termed *pure*.

The product of quaternions, for example, is not commutative (but it is still associative).

It is possible to create an augmented vector (*augmented vector*) of a vector $\mathbf{r} \in \mathbb{R}^3$ in quaternion space as follows:

$$\bar{\mathbf{r}} = \begin{bmatrix} 0 \\ \mathbf{r} \end{bmatrix} \tag{A.19}$$

The conjugate of a quaternion $\mathbf{q}^*$ is

$$\mathbf{q}^* = \begin{bmatrix} q_w \\ -\mathbf{v} \end{bmatrix} \tag{A.20}$$

The norm $|\mathbf{q}|$ is

$$|\mathbf{q}| = \sqrt{\mathbf{q}^* \mathbf{q}} = \sqrt{q_w^2 + \mathbf{v}^2} \tag{A.21}$$

A quaternion $|\mathbf{q}| = 1$ is called a *unit quaternion*. The inverse of a unit quaternion is its complex conjugate $\mathbf{q}^{-1} = \mathbf{q}^*$.

The most important property of a quaternion is that it represents a rotation in $\mathbb{R}^3$.

A rotation $\mathbf{R} = e^{\vartheta \hat{\mathbf{u}}}$, expressed in axis-angle representation, can be written in quaternion form as follows:

$$\mathbf{q} = \exp(\vartheta \hat{\mathbf{u}}) = \begin{bmatrix} \cos(\vartheta/2) \\ \hat{\mathbf{u}} \sin(\vartheta/2) \end{bmatrix} \tag{A.22}$$

with ϑ representing a rotation angle and $\hat{\mathbf{u}}$ denoting a three-dimensional unit vector. In this case, it is a unit quaternion and represents the rotation of an angle ϑ around the axis $\hat{\mathbf{u}}$. It is noteworthy that a rotation of $-\vartheta$ with respect to $-\hat{\mathbf{u}}$ yields the same quaternion as the rotation of ϑ around $\hat{\mathbf{u}}$, thus resolving the singularity of the axis/angle representation. Similarly, it is possible to define the "logarithm" of a quaternion:

$$\vartheta \hat{\mathbf{u}} = \log \mathbf{q} = \begin{cases} 2 \dfrac{\mathbf{v}}{\|\mathbf{v}\|} \arccos q_w & \mathbf{v} \neq 0 \\ 0 & \mathbf{v} = 0 \end{cases} \tag{A.23}$$

which returns the standard axis-angle representation of a rotation given a quaternion.

Rotations are represented by unit-length quaternions $\mathbf{q}^\top \mathbf{q} = 1$.

It is possible to rotate a point using quaternions directly $\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}$, or a unit quaternion can be converted into a rotation matrix (*directional cosine matrix*):

$$\mathbf{R} = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2q_x q_y - 2q_w q_z & 2q_x q_z + 2q_w q_y \\ 2q_x q_y + 2q_w q_z & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2q_y q_z - 2q_w q_x \\ 2q_x q_z - 2q_w q_y & 2q_y q_z + 2q_w q_x & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{A.24}$$

or equivalently:

$$\mathbf{R} = \begin{bmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y - q_w q_z) & 2(q_x q_z + q_w q_y) \\ 2(q_x q_y + q_w q_z) & 1 - 2(q_x^2 + q_z^2) & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_y q_z + q_w q_x) & 1 - 2(q_x^2 + q_y^2) \end{bmatrix} \tag{A.25}$$

to subsequently compute $\mathbf{p}' = \mathbf{R}\mathbf{p}$.

It is noteworthy that $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation matrix $\mathbf{R}$.

Conversely, from the rotation matrix it is possible to derive the quaternion, for example, through

$$\begin{aligned} q_w^2 &= (r_{11} + r_{22} + r_{33} + 1)/4 \\ q_x &= (r_{32} - r_{23})/(4q_w) \\ q_y &= (r_{13} - r_{31})/(4q_w) \\ q_z &= (r_{21} - r_{12})/(4q_w) \end{aligned} \tag{A.26}$$

(operationally, one looks for the largest component and calculates the other components with respect to that one).

The product of two quaternions ultimately represents the composition of rotations:

$$\mathbf{q} \times \mathbf{t} = \begin{bmatrix} t_w q_w - t_x q_x - t_y q_y - t_z q_z \\ t_w q_x + t_x q_w - t_y q_z - t_z q_y \\ t_w q_y + t_x q_z + t_y q_w - t_z q_x \\ t_w q_z - t_x q_y + t_y q_x + t_z q_w \end{bmatrix} \tag{A.27}$$

Appendix B

Nomenclature

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. (8.12));

E Essential Matrix (see eq. (9.41));

F Fundamental Matrix (see eq. (9.45));

P Camera Matrix (see eq. (8.15));

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.

Bibliography

[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.