Pianificazione parallela dei movimenti mediante competizione casuale

Monica Reggiani Stefano Caselli
Dipartimento di Ingegneria dell'Informazione
Università di Parma - 43100 Parma - Italy
e-mail: {reggiani,caselli}@ce.unipr.it

Sommario

Questo lavoro descrive una tecnica parallela di pianificazione del moto di sistemi robotici a molti gradi di libertà. La tecnica si basa sulla competizione per la determinazione di una soluzione da parte di processi che eseguono un pianificatore probabilistico su ciascuno dei nodi dell'architettura. I risultati ottenuti su un'architettura parallela a basso costo di tipo NOW (Network of Workstations) hanno evidenziato l'efficacia della tecnica nella pianificazione del moto di sistemi complessi.

1 Introduzione

La pianificazione dei movimenti (motion planning) si pone l'obiettivo di rendere i robot capaci di determinare autonomamente i moti da eseguire per raggiungere una configurazione obiettivo, evitando contemporaneamente la collisione con ostacoli fisici presenti nell'ambiente. L'importanza di questo tema nella robotica, nell'automazione industriale e, più recentemente, nella realtà virtuale, ha spinto la ricerca a sviluppare diversi approcci alla soluzione. Rassegne dei numerosi contributi proposti in questo campo si trovano nel libro di Latombe [Lat91] e in [HA92], mentre lo "stato dell'arte" della ricerca sarà prossimamente disponibile in [GdP98].

La formulazione più generale del problema di pianificazione dei movimenti, quando il robot si muove in un ambiente dinamico o quando più robot operano nel medesimo ambiente di lavoro, esprime un'alta complessitá computazionale. La ricerca di soluzioni efficaci fa riferimento ad una versione "base" del problema che considera un solo robot che si muove in un ambiente di lavoro statico e perfettamente conosciuto.

È stato dimostrato che il problema di base del motion planning richiede un tempo esponenziale nelle dimensioni dello spazio delle configurazioni, cioè nel numero di gradi di libertà del robot (N_{dof}) [Lat91, HA92]. Per questo motivo gli algoritmi che stanno alla base dei pianificatori *completi* non sono in grado di gestire problemi con $N_{dof} > 4$. Questo vincolo preclude il loro impiego nelle applicazioni più comuni, quali la pianificazione del moto di oggetti rigidi che ruotano e traslano in ambienti tridimensionali o di manipolatori a sei gradi di libertà.

2 Tecniche di pianificazione euristiche

La complessità esponenziale nel numero di gradi di libertà connaturata al problema del motion planning ha stimolato la ricerca di algoritmi euristici di complessità inferiore. La minore complessità computazionale, che porta ad un guadagno in termini di efficienza, viene scontata in termini di completezza: i pianificatori che si basano su questi algoritmi sono infatti solo *probabilisticamente completi*, garantendo di trovare in tempi finiti una soluzione, se essa esiste, solo con un'alta probabilità.

2.1 Random Path Planning

Nei primi anni '90, Barraquand e Latombe [BL91] proposero un algoritmo probabilisticamente completo che utilizzarono con successo in problemi non trattabili con algoritmi completi. Tale algoritmo, denominato Random Path Planning (RPP), è il punto di partenza del presente lavoro. L'algoritmo RPP costruisce un grafo i cui nodi sono i minimi locali di una funzione potenziale U definita nello spazio discretizzato delle configurazioni del robot e ricerca all'interno di esso il percorso che congiunge le configurazioni iniziale e finale. Due minimi locali tra cui il pianificatore costruisce un percorso durante l'esplorazione dello spazio delle configurazioni risulteranno connessi da un arco nel grafo.

La funzione potenziale U che guida l'esplorazione dello spazio delle configurazioni è costruita combinando un potenziale che attrae il robot verso l'obiettivo (collocato in coincidenza con il minimo globale di U) e un potenziale repulsivo nell'intorno degli ostacoli (Figura 1). A partire dalla configurazione iniziale q_{init} del robot, l'algoritmo esegue un *moto a gradiente*, seguendo il gradiente negato del potenziale U fino a quando raggiunge un minimo locale q_{loc}. Se il potenziale in q_{loc} lo identifica come minimo globale, l'algoritmo termina con successo restituendo il percorso; in caso contrario l'algoritmo cerca di uscire dal minimo locale nel quale è intrappolato. Per riuscire a sfuggire dal dominio di attrazione del minimo locale l'algoritmo compie una serie di *moti casuali* a partire da q_{loc}. Ogni moto casuale è immediatamente seguito da un moto a gradiente che raggiungerà un nuovo minimo. L'algoritmo procede quindi alla costruzione incrementale del grafo fino a quando non raggiunge la configurazione obiettivo.

Per limitare la complessità computazionale dell'algoritmo in alcune fasi vengono compiute scelte casuali. Durante il moto a gradiente si dovrebbe esplorare tutto il vicinato della configurazione scegliendo come successore la configurazione che presenta potenziale minore. Purtroppo il numero di vicini (pari a $3^{N_{dof}} - 1$) rende il vicinato esplorabile in modo completo solo quando i gradi di libertà sono pochi. Ad esempio giá con $N_{dof} = 10$ ogni configurazione ha approssimativamente

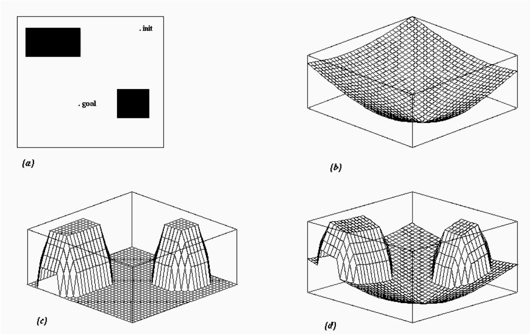

Figura 1. Costruzione del campo potenziale del C-space: a partire dal C-space bidimensionale contenente due C-obstacle (a) il campo potenziale (d) è ottenuto dalla combinazione di un campo attrattivo verso il goal (b) e di un campo repulsivo (c).

60000 vicini. Per migliorare l'efficienza, in questa fase l'esplorazione è compiuta su un vicinato ristretto, scelto casualmente. A motivo di questa approssimazione non è assicurato che l'algoritmo segua la direzione del gradiente negato di U per l'esplorazione dello spazio delle configurazioni. Un'altra più rilevante fonte di casualità è costituita dal comportamento dell'algoritmo quando tenta di sfuggire ai domini di attrazione dei minimi locali compiendo dei moti casuali.

Anche se la formulazione dell'algoritmo ne è indipendente, la funzione potenziale U influenza significativamente l'efficacia di RPP. In particolare, se la funzione U presenta pochi minimi locali si hanno maggiori possibilità che il moto a gradiente conduca verso il minimo globale e, se ogni minimo locale ha un dominio di attrazione limitato, è più probabile riuscire a sfuggire dal minimo locale con i moti casuali [KL94].

2.2 Ricerche correlate

L'algoritmo RPP è stato uno dei primi in grado di fornire, in tempi "ragionevoli", la soluzione di un problema di motion planning caratterizzato da $N_{dof} \geq 6$. La chiave del successo di questo algoritmo, cioè la rinuncia ad un algoritmo computazionalmente completo a favore di un algoritmo probabilisticamente completo ma più veloce, sta alla base di altri algoritmi proposti successivamente in letteratura. Nell'algoritmo proposto in [KS+96] il processo di ricerca di una soluzione viene suddiviso in una fase di *learning* ed una fase di *query*. Nella fase di *learning* vengono costruite e memorizzate in un grafo le mappe dei percorsi tra configurazioni scelte casualmente nello spazio libero con l'uso di pianificatori locali, semplici e veloci. Nella fase di *query* le configurazioni iniziale e finale vengono connesse a due nodi del grafo costruito nella fase precedente, per poi determinare un percorso fra questi due nodi. La capacità dell'algoritmo di fornire una risposta veloce ad una interrogazione dipende dalla scelta delle configurazioni che diventeranno nodi per il grafo della mappa dei percorsi. Tra le tecniche proposte si può segnalare il campionamento uniforme dello spazio seguito da un ricampionamento nelle zone complesse [KS+96], il campionamento sui bordi dello spazio libero [AW97] e il campionamento in uno spazio libero "dilatato" che nasconde gli spazi stretti con successivo ricampionamento nelle zone in cui il percorso è penetrato negli ostacoli [HK+98]. Questo approccio, relegando l'onere del calcolo alla fase di *learning*, consente risposte "veloci" alle richieste formulate al pianificatore. D'altra parte anche in caso di singola richiesta è necessario sobbarcarsi il carico oneroso della costruzione di mappe che esplorano l'intero spazio libero, con conseguente inefficienza. In [HLM97] viene proposto un nuovo algoritmo che cerca di campionare solo la porzione di spazio rilevante per la richiesta corrente, evitando il costo della precomputazione della mappa dei percorsi per l'intero spazio delle configurazioni.

Recentemente alcuni autori hanno applicato al problema del motion planning tecniche parallele di risoluzione. Henrich [Hen97] presenta un'ampia analisi di questi approcci paralleli. Fra essi vanno ricordati, perché più vicini alla proposta di questo articolo, il lavoro di Gini et al. [CGK93, CB+95], che presenta un pianificatore di moto basato su una formulazione parallela dell'algoritmo proposto da [BL91], e quello di [QH96], il cui pianificatore parallelo genera diversi sotto-obiettivi casuali che cerca di connettere in parallelo con la configurazione iniziale e l'obiettivo finale.

3 Pianificazione parallela del moto

Il problema del motion planning è in generale molto complesso in funzione dei gradi di libertà del robot e spesso risulta intrattabile anche da una macchina parallela. D'altra parte, lo scopo della proposta parallela non è quello di ridurre la complessità del problema ma piuttosto di ridurre i tempi necessari per la soluzione di un dato problema o di migliorare la qualità della soluzione.

Un punto di partenza consiste nel riprogettare gli algoritmi sequenziali già esistenti, che spesso si prestano ad un semplice processo di parallelizzazione. In particolare, in questo lavoro si è analizzato l'algoritmo RPP presentato nella sezione precedente. La fase fondamentale dell'algoritmo RPP è la costruzione e la ricerca nel grafo dei minimi locali. Ragioni di efficienza hanno già richiesto l'uso di tecniche di ricerca di tipo "quasi best first" con sequenze di moti a gradiente e casuali per poter costruire ed esplorare solo la parte di grafo necessaria al raggiungimento di una soluzione. La parallelizzazione dell'algoritmo mantiene queste caratteristiche di RPP cercando di suddividere il carico della ricerca del percorso nello spazio di lavoro tra i diversi processori.

I diversi approcci alla ricerca parallela su grafo proposti dalla letteratura possono essere classificati in due categorie fondamentali: nella prima lo spazio di ricerca viene suddiviso tra i processori, che comunicano attivamente tra loro per scambiarsi la conoscenza acquisita dall'esplorazione [KG+93]. Nella seconda, basata sul parallelismo di tipo OR, i processori sono indipendenti e ricercano una soluzione nell'intero grafo. Entrambi i modelli introducono un overhead dovuto alla programmazione parallela. Nel primo modello la maggiore sorgente di overhead è costituita dal tempo speso per trasferire i dati tra i processori. L'efficienza dell'algoritmo è quindi influenzata dall'efficacia dei protocolli di comunicazione disponibili. Nel secondo modello la maggiore sorgente di overhead è costituita dalle computazioni in sovrappiù che vengono compiute quando processori in-

Figura 2. Funzionamento del processo master.

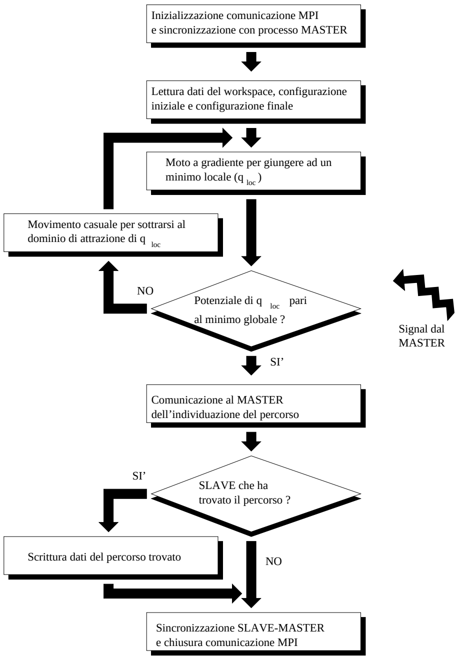

Figura 3. Funzionamento di un processo slave.

dipendenti esplorano il medesimo settore del grafo. La scelta tra i due modelli deve essere compiuta cercando di minimizzare l'overhead, e sarà quindi influenzata dall'architettura su cui viene realizzato l'algoritmo e dal costo delle risorse computazionali.

3.1 Competizione casuale

La tecnica di pianificazione parallela descritta in questo lavoro è rivolta principalmente a piattaforme NOW a basso costo costitute da nodi PC in rete. Tali sistemi rendono disponibile una notevole potenza computazionale a basso costo grazie anche ai supporti software per la comunicazione tra i nodi. Malgrado i recenti sviluppi della tecnologia di rete, un serio ostacolo per l'uso di queste architetture è ancora rappresentato dall'alta latenza di comunicazione esibita da librerie standard, come MPI o PVM, che utilizzano i procolli industriali di comunicazione, quali TCP o UDP.

Questi limiti hanno motivato la scelta di una tecnica di ricerca basata su parallelismo OR in cui ciascun nodo esegue una pianificazione probabilistica sull'intero spazio di lavoro. Questa tecnica, denominata *competizione casuale* [Ert91], si basa sull'idea di utilizzare un insieme di processori paralleli indipendenti (*slave*), ed eseguire su ognuno una ricerca che opera scelte casuali distinte. La elaborazione termina quando uno dei processori trova la soluzione. In questo modello il compito da risolvere non viene quindi suddiviso tra i processori, bensí ogni processore riceve una copia identica dell'intero problema.

L'esecuzione della pianificazione è coordinata da un processo *master* il cui funzionamento è descritto in Figura 2. L'attività dei processi *slave* residenti su ciascun nodo dell'architettura è invece descritta in Figura 3. La pianificazione parallela del percorso è costituita quindi dalle seguenti fasi:

1. **startup phase**: il processo principale (*master*) trasmette l'intero compito di pianificazione a tutti i processi esecutori (*slave*), avendo cura di impostare la casualità dell'algoritmo per differenziare probabilisticamente lo spazio di ricerca di ogni *slave*;

2. **working phase**: ogni *slave* avvia la risoluzione dell'intero problema secondo l'algoritmo RPP, e quindi compie azioni casuali nel corso della esplorazione dello spazio delle configurazioni; ciascun processo *slave* interrompe la propria ricerca solo in caso di successo o in risposta ad un segnale ricevuto dal master (Figura 3);

3. **termination**: se un processo identifica una soluzione del problema o fallisce nella esecuzione (non riuscendo a trovare una soluzione nel tempo predefinito), esso comunica al *master* il risultato della propria esecuzione. Il *master*, arbitrando eventualmente tra i messaggi ricevuti, interrompe tutti i processi *slave* e restituisce il risultato finale.

Nella competizione casuale la comunicazione tra i processori si limita alle fasi di startup e di termination, introducendo quindi un overhead di comunicazione minimo. Tale situazione risulta ideale per l'architettura di sviluppo scelta.

Il tipo di parallelismo adottato porta ad un miglioramento nei tempi di risoluzione del problema quando si utilizzano algoritmi casuali o quando si hanno a disposizione diversi algoritmi la cui gerarchia nei tempi di risposta è determinata dalle caratteristiche problema. Infatti in queste situazioni si hanno tempi di esecuzione variabili senza poter stabilire in anticipo quale sarà l'algoritmo o il "seed" di inizializzazione del processo casuale migliore.

4 Risultati sperimentali

In questo paragrafo vengono presentati e discussi i risultati ottenuti utilizzando il sistema ParMA^2. ParMA^2 è una piattaforma NOW costituita (al momento delle prove) da sei PC, di cui due PPro200 e quattro Pentium100, interconnessi ad un Hub Fast-Ethernet a 12 porte. In questo lavoro sono stati utilizzati solo i quattro Pentium 100, per poter valutare prestazioni e speed up su un insieme di macchine omogenee.

Come supporto software allo scambio di messaggi si è utilizzato l'ambiente di programmazione LAM/MPI [MPI96], che oltre ad offrire un supporto completo allo standard MPI mette a disposizione un pacchetto per la gestione dei segnali in stile UNIX. Il meccanismo dei segnali consente la sospensione dell'attività dei processi slave da parte del processo master in ogni momento dell'esecuzione, eliminando il carico computazionale derivante dal monitoraggio sullo stato della ricerca che i processi slave dovrebbero altrimenti compiere.

Poiché, per la casualità dell'algoritmo, il tempo necessario per risolvere un problema può variare molto da un'esecuzione all'altra, tutti i dati presentati in questa sezione sono stati calcolati considerando il tempo medio di 100 esecuzioni. Per simulare il comportamento temporale quando il programma viene eseguito su un numero n di processori maggiore di quelli a disposizione, sono state compiute un numero di prove con singolo slave sufficienti a fornire un idoneo numero di campioni tra qui sceglierne casualmente n. Il tempo minore tra i campioni scelti rappresenta adeguatamente il tempo necessario a n slave che operano concorrentemente per trovare una soluzione, poiché nella forma di parallelismo scelta ogni processore ricerca la propria soluzione autonomamente e poiché i tempi per la comunicazione costituiscono una parte minima nel tempo complessivo.

In figura 4 è rappresentato l'ambiente di lavoro con il robot al suo interno. L'ambiente viene discretizzato in un array di 128^3 celle. Modificando il numero dei gradi di libertà del robot è possibile incrementare la difficoltà del problema ed analizzare l'impatto dell'approccio parallelo al variare della complessità del compito di pianificazione.

dof	1	2	3	4	8	12	16	20
7	20.5	14.75	11.9	12.2	11.5	10.9	10.7	10.6
9	265.9	129.7	99.4	73.6	56.8	42.6	34.6	35.5
11	532.6	302.5	247.5	194.3	104.4	73.4	70.0	54.1

Tabella 1. Tempi medi su 100 esecuzioni al variare del numero di processori slave (in secondi).

Una prima analisi dei tempi medi (Tabella 1) mostra che l'aumento del numero di processori coinvolti come slave nella ricerca porta ad una sostanziale diminuzione nei tempi medi necessari per ottenere una soluzione. Naturalmente, oltre un certo numero di processori, che varia in funzione della complessità del problema, il tempo medio raggiunge un valore asintotico da cui pressoché non si discosta. Si perviene, infatti, ad un punto in cui accrescere il numero dei processori non aumenta la probabilità di esplorare una zona dello spazio non ancora coinvolta nella ricerca e porta solo a compiere lavoro ridondante. Questo comportamento può essere osservato nei valori dello speed up (Tabella 2). Gli andamenti confermano l'analisi teorica di Ertel per il parallelismo con competizione casuale [Ert91], che indica in $\lim_{k\to\infty} S(k) = \frac{E(T_1)}{t_0}$, con $E(T_1)$ tempo medio per la soluzione seriale e t_0 tempo minimo necessario per la soluzione, il valore asintotico a cui lo speed up tende. Dalla Figura 5 si rileva che il numero di processori necessari per raggiungere questo valore asintotico è funzione della complessità del problema da affrontare. Infatti l'aumento dei gradi di libertà del robot, e quindi delle dimensioni dello spazio delle configurazioni, richiede un numero maggiore di processori per la sua esplorazione in un tempo medio prossimo al minimo teorico t_0.

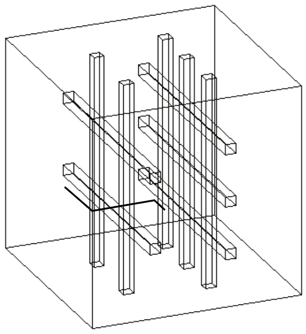

Figura 4. Il workspace tridimensionale utilizzato negli esperimenti.

dof	1	2	3	4	8	12	16	20
7	1.09	1.51	1.86	1.83	1.95	2.04	2.09	2.1
9	0.99	2.02	2.64	3.56	4.61	6.15	7.57	7.39
11	1.24	2.19	2.68	3.41	6.35	9.04	9.47	12.25

Tabella 2. Speed up medio per 100 esecuzioni al variare del numero di processori slave.

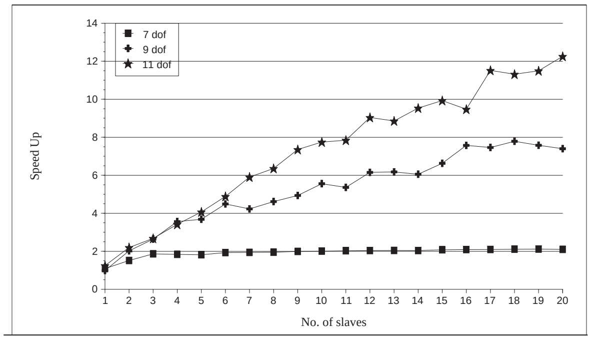

Figura 5. Andamenti dello speed up al variare della complessità del problema.

Anche dopo aver raggiunto il valore asintotico dello speed up, un eventuale incremento del numero di slave può portare un beneficio in termini di riduzione della deviazione standard dalla media dei tempi di risposta (Figura 6). Inoltre, si è sperimentalmente osservato (Figura 7) che anche la qualità delle soluzioni migliora all'aumentare del numero di processori, raggiungendo un valore asintotico per un numero di processori maggiore rispetto al numero necessario per il valore asintotico dello speed up. La possibilità di ottenere una soluzione

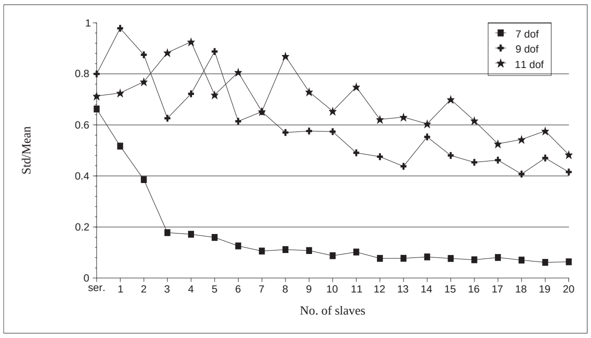

Figura 6. Andamenti del rapporto tra deviazione standard dalla media e valore medio al variare della complessità del problema.

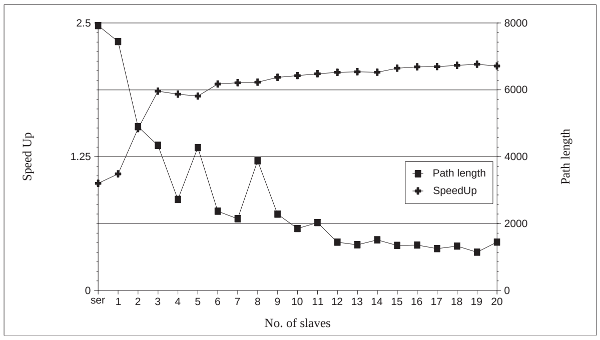

Figura 7. Andamenti delle dimensioni del percorso e dello speed up con un robot a 7 gradi di libertà.

di migliore qualità e dei tempi meno variabili può quindi suggerire l'utilizzo di un numero di processori superiore a quello necessario per ottenere il valore asintotico dello speed up.

5 Conclusioni

Il lavoro ha descritto una tecnica parallela di pianificazione del moto di sistemi robotici basata sulla replicazione di un pianificatore probabilistico sui nodi dell'architettura, che competono casualmente nella determinazione di una soluzione del problema. I risultati ottenuti su un'architettura parallela di tipo NOW a basso costo hanno evidenziato una apprezzabile riduzione dei tempi di pianificazione ed un miglioramento della qualità del percorso individuato. La efficienza del pianificatore parallelo risulta maggiore nei sistemi più complessi, caratterizzati da molti gradi di libertà.

Bibliografia

[AW97] N. Amato, Y. Wu. A randomized roadmap method for path and manipulation planning. In *IEEE International Conference on Robotics and Automation*, Minneapolis, MN, April 1997.

[BL91] J. Barraquand, J. Latombe. Robot motion planning: a distributed representation approach. *International Journal of Robotics Research*, 10(6), December 1991.

[CB+95] D. Challou, D. Boley, M. Gini, V. Kumar. A parallel formulation of informed randomized search for robot motion planning problems. In *IEEE International Conference on Robotics and Automation*, Nagoya, Japan, May 1995.

[CGK93] D.J. Challou, M. Gini, V. Kumar. Parallel search algorithms for robot motion planning. In *IEEE International Conference on Robotics and Automation*, Atlanta, GA, May 1993. IEEE.

[Ert91] W. Ertel. Performance analysis of competitive OR-parallel theorem proving. Technical Report FKI-16291, Technische Universitat Munchen, Germany, November 1991.

[GdP98] K. Gupta, A. P. del Pobil. *Pratical Motion Planning in Robotics: Current Approaches and Future Directions.* Addison Wesley, 1998.

[HA92] Y. K. Hwang, N. Ahuja. Gross motion planning - a survey. *ACM Computing Surveys*, 24(3):219–291, September 1992.

[Hen97] D. Henrich. Fast motion planning by parallel processing - a review. *Journal of Intelligent and Robotic Systems*, 20(1):45–69, September 1997.

[HK+98] D. Hsu, L. Kavraki, J. Latombe, R. Motwani, S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Workshop on the Algorithmic Foundations of Robotics*, 1998.

[HLM97] D. Hsu, J. Latombe, R. Motwani. Path planning in expansive configuration spaces. In *IEEE International Conference on Robotics and Automation.* Albuquerque, NM, April 1997.

[KG+93] V. Kumar, A. Grama, A. Gupta, G. Karypis. *Introduction to Parallel Computing - Design and Analysis of Algorithms.* Benjamin/Cummings Pub. Co., 1994.

[KL94] L. Kavraki, J. Latombe. Randomized preprocessing of configuration space for fast path planning. In *IEEE International Conference on Robotics and Automation*, San Diego, CA, April 1994.

[KS+96] L. Kavraki, P. Svestka, J. Latombe, M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4), August 1996.

[Lat91] J.C. Latombe. *Robot Motion Planning.* Kluwer Academic Press, 1991.

[MPI96] Ohio Supercomputer Center. MPI primer/Developing with LAM. Technical report, The Ohio State University, November 1996.

[QH96] C. Qin, D. Henrich. Randomized parallel motion planning for robot manipulation. Computer Science 5/96, University Karlsruhe, Germany, Februray 1996.