Politecnico di Milano
Facoltà di Ingegneria dell'Informazione

POLO REGIONALE DI COMO

Corso di Laurea Specialistica in Ingegneria Informatica

Sciami di robot volanti per la pianificazione distribuita di percorsi per il movimento di oggetti a terra

Relatore: **Prof. Marco Colombetti**

Correlatori: **Prof. Luca Maria Gambardella**
Dr. Gianni Di Caro
(Istituto Dalle Molle di Studi sull'Intelligenza Artificiale - IDSIA)

Tesi di laurea di: **Andreagiovanni Reina**
matricola **725624**

Anno Accademico 2009/2010

Politecnico di Milano
Facoltà di Ingegneria dell'Informazione

POLO REGIONALE DI COMO

Master of Science in Computer Engineering

Swarms of flying robots performing distributed path planning for objects on the ground

Supervisor: **Prof. Marco Colombetti**

Assistant Supervisors: **Prof. Luca Maria Gambardella**
Dr. Gianni Di Caro
(Istituto Dalle Molle di Studi sull'Intelligenza Artificiale - IDSIA)

Master Graduation Thesis by: **Andreagiovanni Reina**
Student Id. number: **725624**

Academic Year 2009/2010

ai miei genitori, in ogni momento al mio fianco.
Il vero unico punto di riferimento della mia vita.

RINGRAZIAMENTI

Ringrazio i ricercatori dell'IDSIA di Lugano:
il Prof. Luca Gambardella ed il Dr. Gianni Di Caro,
le due figure di riferimento durante questo progetto.
Un grazie speciale va
all'ing. Frederick Ducatelle,
compagno di scrivania
e dispensatore di importanti consigli
per la realizzazione della presente tesi.

Ringrazio la Zeno Karl Schindler Foundation
per l'importante aiuto economico fornito in
supporto a questa tesi.

Un ringraziamento speciale va
a Gianni Brenna,
per avermi aiutato a proseguire nei miei studi.

ABSTRACT

Motion planning is one of the important, classical problems in algorithmic robotics. Another emerging branch is the swarm robotics which offers the potential of parallelism, redundancy of resources, and heterogeneity of structures and functions. In this project we study a distributed approach to the path planning problem, with the aim of discovering advantages and problems of this novel architecture. We focus on holonomic kinematic motion in a plane with static obstacles. The problem consists in planning the path of a rigid object that has to be transported from an initial to a final location through a constrained path. The planner observes the environment from above through a visual system. We consider the case in which the path covers a large area, such that the planner architecture consists of a wireless network of observer nodes, each one able to see a portion of the area. A centralized solution, in which all the observers send their visual data to a sink node that fuses the information, calculates the path, and sends the local navigation instructions back to the individual nodes, requires complex information flux management based on ad hoc routing algorithms, and is neither robust nor scalable. Moreover, it needs precise calibration and alignment of the cameras. To overcome these difficulties we propose a fully distributed cooperative approach: each observer node locally calculates the part of the path relative to the area that it sees, and then it communicates to its neighbour the information which allows the distributed execution of the path planning.

Our goal is to calculate effective paths in a way that is scalable, resource efficient, and robust to calibration and alignment errors.

As reference models for the planner nodes, we consider the eye-bot robots, that are being developed in the Swarmanoid project [1]. These small flying robots that can attach to the ceiling are equipped with an omnidirectional video camera and an infrared system for measuring the relative bearing and distance between two robots (*IrRB*) and for wireless communications.

We followed a top-down approach. We first designed a planning algorithm for a centralized, single camera system with full view. Then we developed a fully distributed version of it with various adaptations. The single camera planner is based on the numerical potential field technique, computed using the *wave front expansion with skeleton* on a bidimensional uniform cell partitioning of the environment. This solution first spreads the potential over a subset of the free space, called skeleton, which corresponds to the Voronoi diagram; then the potential is computed in the rest of the map. The potential descent is performed using the A-Star algorithm.

The distributed system architecture is composed of three phases: neighbour detection, potential field diffusion, and path calculation. During the neighbour detection phase, each robot builds a neighbour table, in which the relative positions of other nearby robots are stored (e.g., using the IrRB system) with some estimation error.

The second phase concerns the potential field diffusion; each robot expands the potential on its part of the map, and sends to its neighbours the frontier values. The robots that can see the goal position begin the process and then each new message triggers an update of the neighbour's values. The system minimizes the communication overhead, transmitting only the information of the skeleton cells near the frontier.

Finally, the robot above the start position begins the third phase, which is the path calculation; when the trajectory exits from the area of view, the robot sends the object coordinates to its neighbour which continues the process. We assume that the partial map overlaps with the neighbour's map, in order to allow the object position sharing. However, the overlapping is subject to errors deriving from errors in camera calibrations and the measures from the IrRB. The process ends when the object reaches the goal configuration (success) or when the exploration tree of the configurations is completely expanded (failure).

As it is shown in this thesis, in the majority of the cases the system produces successful results with an efficient resource exploitation. However, given the distributed nature of the approach, the path found can present loops. Moreover, the well-known problem of attraction toward local minima in the potential field is possibly amplified. To overcome these limitations, and at the same time reduce the related communication overhead and computational effort, we propose a set of heuristics:

Smart Loop Avoidance: during path calculation a robot is able to detect loops and coordinates with others in order to step back to the loop start situation and avoid it.

Blocked Cell List: the robot maintains trace of the area in which the neighbour produces a failure, and avoids to send again the object in those nearby areas.

Skeleton Pruning: pre-pruning of the skeleton during the potential field diffusion. The passages narrower than the smallest object size are blocked.

Narrow Passage Detection: during path calculation the robot detects a narrow passage, blocks it and repeats the potential field diffusion step.

We studied the performance of the different planners in terms of effectiveness, efficiency, scalability and robustness to alignment errors, through a set of experiments, in which we varied the scenario's characteristics, like the size of the environment, the position and the number of robots, the moving object shape and the obstacles positions.

Experiments show a reduction of resources exploitation, in terms of time and communication, and an increase of path quality (we assume shorter paths are better) in the system with the heuristics active.

The proposed systems are able to cooperate, producing effective paths. They operate in a way that is scalable, resource efficient, and quite robust to calibration and alignment errors.

This work studied a new approach to the problem of holonomic motion planning for rigid objects on a bidimensional plane. We propose a set of prototypes of fully distributed cooperative planners, analysing advantages and drawbacks of this novel approach. Since it is a wide area of study, this work can be considered as the initial step towards a more extensive study.

A part of this work has been published with the paper *"A distributed approach to holonomic path planning"* in the workshop *"Motion Planning: From Theory to Practice"* on June 27, 2010, during the international conference *Robotics: Science and Systems (RSS)* in Zaragoza, Spain (Attachment A).

SOMMARIO

La pianificazione di movimento rappresenta una delle principali aree della robotica. Il problema, nella sua accezione originale, consiste nel calcolo di un percorso di un oggetto, o un robot, da una posizione iniziale ad una finale, evitando collisioni con ostacoli. Con questa tesi si vuole studiare un approccio distribuito al problema della pianificazione di percorso, con l'obiettivo di individuare eventuali vantaggi e svantaggi della nuova architettura proposta. In particolare, è studiato il movimento di oggetti olonomici su un piano con ostacoli statici.

Si considera il caso in cui il pianificatore operi in ambienti vasti, senza una conoscenza pregressa della mappa, pertanto si rende necessaria un'acquisizione sensoriale dell'ambiente attraverso una rete wireless di robot, in cui ogni nodo ha una visione parziale dell'ambiente dall'alto. Una soluzione centralizzata, in cui tutte le acquisizioni sensoriali sono inviate ad un nodo centrale che combina le informazioni, ricostruisce la mappa globale e calcola il percorso, richiede un'accurata gestione dei flussi informativi, ottenibile con algoritmi di routing creati ad hoc per l'architettura del sistema, pertanto presenta problemi di scalabilità e robustezza. Inoltre, per il corretto funzionamento è necessaria una precisa calibrazione ed allineamento delle telecamere. Per superare questi limiti, è stato proposto un approccio cooperativo completamente distribuito in cui ogni robot calcola localmente la parte di percorso relativa al proprio campo visivo, e poi invia al vicino le informazioni per la prosecuzione del calcolo del percorso.

L'obiettivo è progettare un pianificatore di percorsi efficaci, che soddisfi le proprietà di scalabilità, utilizzo efficiente delle risorse e robustezza agli errori di calibrazione e allineamento tra robot.

Come modello di riferimento per i nodi sensoriali del sistema sono stati adottati gli eye-bot, piccoli robot volanti, sviluppati nel progetto Swarmanoid, in grado di attaccarsi al soffitto ed equipaggiati di sistemi di visione, di comunicazione wireless e di rilevamento delle posizioni relative fra due robot.

Il pianificatore distribuito è derivato da un pianificatore a camera singola basato sulla tecnica di discesa del potenziale con segmentazione dello spazio in celle bidimensionali uniformi. Questa soluzione prima diffonde il potenziale su un sottoinsieme di celle libere, chiamato skeleton, il quale corrisponde al diagramma di Voronoi; quindi il campo di potenziale viene diffuso sul resto della mappa. La discesa di potenziale è effettuata utilizzando l'algoritmo di A-star.

L'algoritmo distribuito si compone di tre fasi: rilevamento dei vicini, diffusione del potenziale, e calcolo del percorso. Durante la fase di rilevamento dei nodi vicini, ogni robot costruisce una tabella dei vicini, in cui memorizza la posizione relativa dei robot che ha intorno, la quale può essere soggetta ad errori di misurazione.

La seconda fase riguarda la diffusione di potenziale: ogni robot espande il campo di potenziale sulla mappa parziale, ed invia ai vicini i valori sulla frontiera. I robot che vedono la destinazione iniziano il processo ed ogni nuovo messaggio scatena un aggiornamento dei valori dei vicini. Il sistema minimizza le comunicazioni inviando solamente i valori di potenziale dello skeleton vicino alla frontiera.

Infine, il robot sopra la posizione di partenza inizia la terza fase di pianificazione del cammino; quando il percorso esce dal campo visivo, il robot invia

le coordinate dell'oggetto al vicino, il quale prosegue la pianificazione. Si presume che i campi visivi dei robot vicini siano sovrapposti, al fine di permettere la condivisione della posizione dell'oggetto. Tuttavia, a causa di errori nella misurazione delle posizioni relative tra robot, le aree condivise sono soggette ad errori, e quindi risultano disallineate. Il processo termina quando viene raggiunta la posizione finale (successo) oppure quando l'albero di esplorazione è totalmente espanso (fallimento).

I risultati mostrano che nella maggior parte degli esperimenti l'architettura proposta produce risultati soddisfacenti, con un utilizzo delle risorse efficiente. Tuttavia, a causa della natura distribuita dell'approccio, il percorso può presentare cicli. Inoltre l'attrazione verso i minimi locali è amplificata, provocando problemi di efficienza.

Per risolvere i limiti del sistema, ed allo stesso tempo ottimizzare l'utilizzo delle risorse, sono state proposte le quattro seguenti euristiche:

- *Blocked Cell List*: il sistema tiene traccia delle aree di passaggio dell'oggetto in cui i vicini hanno risposto con un fallimento, ed evita ulteriori passaggi in quella zona;
- *Smart Loop Avoidance*: durante il processo di pianificazione il robot è in grado di individuare i cicli nel percorso, e riportare il sistema allo stato precedente l'inizio del ciclo, ed evitare che si ripeta.
- *Skeleton Pruning*: "potatura" dello skeleton prima della fase di espansione del potenziale. Vengono eliminati i passaggi più stretti di una soglia predefinita.
- *Narrow Passage Detection*: rileva automaticamente, in fase di pianificazione, la presenza di passaggi troppo stretti per l'oggetto mobile. Le strade impercorribili vengono bloccate e viene ripetuto l'intero processo a partire dalla diffusione di potenziale.

I diversi sistemi sono stati testati con un insieme di esperimenti in simulazione, con lo scopo di misurare e confrontare le prestazioni delle differenti architetture proposte. I test effettuati variano le caratteristiche degli scenari al fine di studiare l'efficacia, l'efficienza, la scalabilità e la robustezza del sistema al crescere dell'errore di allineamento.

Dai risultati, emerge come le architetture proposte raggiungano gli obiettivi prefissati, con pianificatori distribuiti in grado di generare efficacemente percorsi, operando con un utilizzo delle risorse efficiente e scalabile.

In conclusione, con questo lavoro è stato investigato un nuovo approccio al problema di pianificazione di percorso per oggetti olonomici in un piano bidimensionale. In particolare, sono stati proposti alcuni innovativi prototipi di architetture di pianificazione cooperativa totalmente distribuita, e ne sono stati analizzati vantaggi e svantaggi. Questa vasta direzione di studio è stata poco investigata in letteratura, perciò le soluzioni proposte si possono considerare un primo passo di studio verso un lavoro di indagine più ampio ed approfondito.

Questo lavoro ha portato alla pubblicazione dell'articolo *"A distributed approach to holonomic path planning"* ed alla presentazione di un poster al workshop *"Motion Planning: From Theory to Practice"* del 27 giugno 2010, durante la conferenza internazionale *Robotics: Science and Systems (RSS)* a Saragozza, Spagna.

Indice

Capitolo 1

Introduzione

1.1 Obiettivi

La pianificazione di movimento e traiettorie rappresenta una delle principali aree della robotica. Il problema, nella sua accezione originale, consiste nel calcolo di un percorso di un oggetto, o un robot, da una posizione iniziale ad una finale, evitando collisioni con ostacoli. Negli ultimi anni questo tema è stato ampiamente studiato, e sono stati proposti molteplici algoritmi per applicazioni in uno svariato numero di aree, come l'animazione digitale, la robotica industriale, la chirurgia assistita o la progettazione di farmaci. Il costante aumento di campi di impiego ha spinto la comunità scientifica a trovare un vasto numero di soluzioni con algoritmi in grado di risolvere il problema di pianificazione di percorsi in maniera efficace rispetto al contesto d'applicazione.

Un'altra area della robotica che riscuote notevole interesse scientifico è la *Swarm robotics*, la quale studia il comportamento di sciami di robot. Lo sciame è composto da entità semplici, con limitate capacità elaborative e sensoriali, che spesso non sono sufficienti a risolvere il problema per cui sono state progettate. I robot dovranno quindi coordinarsi e cooperare per raggiungere gli obiettivi finali. Questi sistemi godono di elevato parallelismo dei processi, ridondanza di risorse, ed eterogeneità di strutture e funzioni. L'interazione tra robot e l'auto-organizzazione sono gli aspetti chiave di questo tipo di sistemi e permettono di creare sinergie che generano comportamenti collettivi in grado di incrementare le capacità dei singoli robot.

Con questa tesi si vuole studiare un approccio distribuito al problema della pianificazione di percorso, con l'obiettivo di individuare eventuali vantaggi e svantaggi della una nuova architettura proposta. È trattata la situazione in cui i robot debbano esplicitamente cooperare e coordinarsi al fine di risolvere il problema. Il caso di studio analizzato consiste nella pianificazione di percorso per il movimento di un oggetto da una posizione iniziale ad una destinazione finale. L'ambiente è vincolato da ostacoli e l'oggetto può avere dimensioni elevate e forme complesse, per questo motivo le traiettorie devono spesso comprendere una sequenza articolata di movimenti e rotazioni al fine di permettere il districarsi dell'oggetto fra gli ostacoli. Ogni robot ha una visione della scena dall'alto, ma le capacità sensoriali di ogni singola entità sono ridotte rispetto alle dimensioni totali dell'ambiente quindi ognuno ha una conoscenza parziale della mappa.

Nessun robot nel sistema è in grado di calcolare in modo indipendente l'intero percorso; solamente grazie alla cooperazione ed allo scambio di informazioni i robot possono individuare la traiettoria completa dell'oggetto.

Lo scenario proposto è sufficientemente ampio da essere considerato una fattispecie della vasta classe di problemi di trasporto che richiedono una acquisizione della mappa da fonti sensoriali distribuite. Con questo lavoro di tesi si mira a sviluppare un sistema in grado di affrontare il problema in modo completamente distribuito, e risolvere efficientemente il problema. Con *approccio distribuito* si indica l'aspetto di frammentazione della conoscenza e del processo fra le diverse entità, le quali hanno una conoscenza parziale dell'ambiente, limitata alla propria zona di competenza. Questo tipo di approccio, oltre che in fase sensoriale, è mantenuto anche durante l'effettiva pianificazione, pertanto nessuna entità ha conoscenza totale sull'intero processo. L'architettura proposta è esplicitamente strutturata senza alcun supervisore globale, ma al contrario il sistema è composto da un insieme di entità semplici tutte uguali con conoscenza parziale e locale. Grazie alla cooperazione ed ad un limitato scambio di informazioni l'intero sistema è in grado di trovare, se presente, un percorso libero da ostacoli dall'inizio all'arrivo.

1.2 Motivazioni

Con questo lavoro si intende studiare la pianificazione di oggetti in ambienti vincolati da ostacoli senza una conoscenza pregressa della mappa. Ciò presuppone che vengano acquisite informazioni sull'ambiente tramite l'utilizzo di sensori. Per motivi di natura tecnica, come la visione in ambienti interni occlusa dai muri che separano le stanze, oppure la mappatura di ambienti molto vasti, è necessario utilizzare una molteplicità di dispositivi sensoriali distribuiti. Per operare con fonti sensoriali distribuite è possibile affrontare il problema con due approcci distinti: quello centralizzato o quello distribuito.

Il primo approccio comporta una centralizzazione di tutte le acquisizioni, la loro fusione e la ricostruzione della mappa globale, infine la pianificazione viene eseguita dall'unica entità con conoscenza totale, chiamata *sink*. Questa strategia comporta notevoli problemi e limitazioni, poichè per una corretta fusione delle acquisizioni sono necessari una precisa calibrazione ed allineamento delle camere. Tali requisiti risultano spesso di complicata realizzazione e possono essere molto costosi sia in sistemi statici che soprattutto in sistemi dinamici, nei quali i sensori sono posizionati nell'ambiente senza una precisa installazione e calibrazione, o tanto più quando si tratta di robot autonomi che provvedono ad un autoposizionamento nell'ambiente.

Gli aspetti riguardanti la centralizzazione dei dati provenienti dalla molteplicità di fonti sensoriali rappresenta un ulteriore limite di scalabilità. È possibile individuare due principali approcci a tale problema: (i) connessione diretta di tutti i dispositivi con il Sink, oppure (ii) gestione del flusso di dati mantenendo una comunicazione locale. Il primo caso richiede che sia presente un mezzo di comunicazione capace di connettere tutti i nodi ad un unico nodo centrale, ciò comporta un elevato utilizzo del canale, con possibile saturazione al crescere dei nodi. Quando la connessione diretta non è attuabile, è possibile adottare il secondo approccio basato sulla comunicazione locale tra nodi vicini. Quest'approccio necessita una attenta gestione delle risorse, poichè al crescere del numero

di nodi è possibile incorrere nel problema di saturazione del canale comunicativo. Tale problema può essere risolto tramite un'accurata gestione dei flussi informativi, ottenibile con algoritmi di routing creati ad hoc per l'architettura del sistema. Pertanto risulta naturale che soluzioni di questo genere possano difficilmente adattarsi in modo automatico al variare dinamico delle dimensioni e degli scenari.

Per superare i limiti di un'architettura centralizzata, è stato studiato un approccio distribuito al problema di path planning, con l'obiettivo di sviluppare un sistema efficiente dal punto di vista dell'utilizzo delle risorse e della scalabilità, ed allo stesso tempo robusto ad errori di allineamento e calibrazione. Il sistema proposto intende utilizzare come nodi sensoriali dei robot volanti autonomi (nello specifico gli eye-bot [vedi sezione 2.2]), è pertanto necessario che il sistema sia in grado di adattarsi a cambiamenti dinamici di scenario, come il variare del numero di nodi, la struttura dello scenario e la disposizione dei robot stessi. In particolare, nello scenario considerato, i robot volanti, essendo entità autonome, possono posizionarsi sul soffitto scegliendo la posizione desiderata (per esempio, utilizzando l'algoritmo descritto in [19]). Questo permette una buona velocità di copertura dell'intera area e fornisce anche un sistema che si può facilmente adattare a modifiche dell'ambiente. Tuttavia una elevata dinamicità del sistema può introdurre un errore di allineamento fra le diverse acquisizioni, perciò anche questo aspetto è stato preso in considerazione nella progettazione del sistema proposto.

1.3 Contributi originali

Questo lavoro di tesi studia gli effetti positivi e negativi di un approccio distribuito al problema di path planning per un oggetto di forma arbitraria con movimenti olonomici. Le prestazioni del sistema sono valutate utilizzando come riferimento di paragone i risultati di una soluzione centralizzata con conoscenza perfetta e globale.

Durante il progetto sono state sviluppate molteplici versioni del pianificatore, ognuna con differenti caratteristiche e prestazioni. Il progetto ha seguito un cammino evolutivo del pianificatore, partendo da una soluzione centralizzata con conoscenza perfetta e muovendosi incrementalmente verso un approccio distribuito.

La prima architettura proposta intende rappresentare lo stato dell'arte attuale per questo tipo di problemi, rappresenta il punto di partenza dal quale il progetto è partito e si è sviluppato, allo stesso tempo è utilizzato come punto di riferimento per valutare e confrontare le prestazioni dei successivi pianificatori sviluppati. L'architettura è composta da un unico robot, capace di vedere l'intera mappa in modo perfetto, senza rumore, ed è quindi in grado di calcolare l'intero percorso autonomamente.

La seconda architettura proposta introduce una visione distribuita, ma mantiene un approccio globale nella ricerca. Dal punto di vista sensoriale: ogni robot ha una visione parziale dell'area del problema, e durante il processo mantiene questa informazione per pianificare solamente la parte di percorso di sua competenza. Mentre, per l'aspetto di pianificazione, tutti i robot possiedono una conoscenza completa del livello di espansione dell'albero di ricerca di tutti gli altri robot. In questo modo è possibile pianificare il percorso ottimo a partire

dalla conoscenza frammentata di partenza. Questa soluzione non soddisfa gli obiettivi di scalabilità prefissati, inoltre presenta sprechi di risorse sia comunicative sia computazionali. Infatti, questo pianificatore non mira ad essere una soluzione finale al problema, piuttosto rappresenta un passaggio di studio verso una soluzione totalmente distribuita.

L'architettura successiva soddisfa effettivamente i requisiti di un sistema distribuito, utilizzando comunicazione locale e una limitata conoscenza locale del problema. Questa architettura, in alcuni ambienti particolari, presenta alcune limitazioni.

Per migliorare le prestazioni del pianificatore distribuito, sono state aggiunte alcune euristiche in grado di eliminare alcuni dei principali problemi emersi dal precedente approccio. In particolare, le euristiche sono state progettate con l'obiettivo di migliorare la qualità dei risultati, l'utilizzo delle risorse e la scalabilità del sistema.

Come modello di riferimento per i nodi sensoriali del sistema sono stati adottati gli eye-bot (Sezione 2.2), piccoli robot volanti sviluppati nel progetto Swarmanoid [1]. Tuttavia il lavoro è stato interamente realizzato in simulazione, usando il simulatore ARGoS, il quale fornisce modelli di simulazione molto precisi degli eye-bot.

Durante questo progetto di tesi sono stati progettati e sviluppati in simulazione tutti i pianificatori sopraelencati. Il primo pianificatore è stato implementato in seguito ad uno studio accurato dello stato dell'arte al fine di individuare il sistema migliore, presente in letteratura, che potesse risolvere lo specifico problema di pianificazione di questa tesi.

I successivi pianificatori sono stati interamente progettati e sviluppati per questo progetto. Per ogni problema incontrato sono state studiate nel dettaglio le differenti possibilità, e sono state selezionate le soluzioni capaci di rispondere meglio agli obiettivi fissati per questo progetto di tesi.

Infine, è stato creato un insieme di esperimenti con lo scopo di misurare e confrontare le prestazioni delle architetture proposte. I test effettuati variano nelle caratteristiche degli scenari, al fine di studiare l'efficacia, l'efficienza, la scalabilità e la robustezza del sistema.

Inoltre, questo lavoro ha portato alla pubblicazione dell'articolo *"A distributed approach to holonomic path planning"* (Allegato A) al workshop *"Motion Planning: From Theory to Practice"* del 27 giugno 2010, durante la conferenza internazionale *Robotics: Science and Systems (RSS)* a Saragozza, Spagna. Durante il workshop è stato presentato un poster relativo alla pubblicazione (Allegato B).

1.4 Struttura della tesi

Il testo della tesi è strutturato come segue.

Nel capitolo 2 sono illustrate le caratteristiche generali del problema. In particolare nel paragrafo 2.1 è specificato il contesto in cui si inserisce il problema. I paragrafi 2.2 e 2.3 illustrano le caratteristiche rispettivamente dei robot utilizzati e dell'ambiente. Il paragrafo 2.4 elenca le assunzioni di partenza ed i vincoli iniziali del problema. Il paragrafo 2.5 descrive il modello di rumore utilizzato, mentre il paragrafo 2.6 presenta e descrive l'ambiente di sviluppo del progetto: il simulatore ARGoS.

Il capitolo 3 contiene lo stato dell'arte dei temi su cui verte principalmente la tesi, con espliciti riferimenti ai lavori precedenti.

Nel capitolo 4 è presentato il primo sistema sviluppato, il quale rappresenta lo stato dell'arte dal quale il progetto parte. Sono illustrate le diverse fasi per la pianificazione di percorso di un oggetto rigido, mostrando nel dettaglio i problemi incontrati e le soluzioni adottate.

Il capitolo 5 descrive l'architettura del sistema distribuito. Mostra i passaggi effettuati per raggiungere un'architettura totalmente distribuita e descrive dettagliatamente la struttura del sistema.

Il capitolo 6 mostra i risultati ottenuti dagli esperimenti effettuati, evidenziando vantaggi e svantaggi della nuova architettura progettata.

Il capitolo 7 chiude la tesi, presentando le conclusioni ed i possibili sviluppi futuri del progetto.

Capitolo 2

Caratteristiche generali del problema

Il seguente capitolo inquadra la configurazione iniziale del progetto e presenta le principali caratteristiche dello scenario in cui è stato effettuato lo studio di pianificazione. In particolare sono elencati i dettagli tecnici e le assunzioni di partenza sulle quali si basa il sistema. Nel paragrafo 2.1 è descritto il contesto in cui il progetto in cui è inserito. Il paragrafo 2.2 descrive le caratteristiche hardware dei robot coinvolti e le relative specifiche tecniche dei sensori ed attuatori utilizzati. Nei paragrafi 2.4 e 2.5 vengono descritti rispettivamente i vincoli di posizione iniziale dei robot ed il modello del rumore adottato. Nel paragrafo 2.6 è presentato l'ambiente di sviluppo del progetto: il simulatore ARGoS.

2.1 Contesto del progetto

Questo progetto di tesi rappresenta la prima parte del lavoro di un innovativo caso di studio per il trasporto di oggetti da parte di sciami di robot eterogenei, che a sua volta si inserisce nel contesto più ampio del progetto Europeo Swarmanoid. Qui di seguito sono brevemente presentati i due progetti.

2.1.1 Swarmanoid

Questa tesi fa parte del progetto *Swarmanoid (IST-022888)* [1] finanziato dalla Comunità Europea (6th Framework Programme - FP6-IST) nell'ambito del progetto *Future and Emerging Technologies (FET-OPEN).*

Il principale obiettivo scientifico del progetto di ricerca Swarmanoid è la progettazione, l'implementazione ed il controllo di un innovativo sistema robotico distribuito. Il sistema è composto da piccoli robot autonomi eterogenei e dinamicamente connessi fra loro. Collettivamente, questi robot formano quello che viene chiamato uno swarmanoid. Lo swarmanoid che si intende costruire è composto da un elevato numero (circa 60) di robot autonomi di tre tipi differenti: eye-bot, hand-bot, e foot-bot.

- hand-bot [14] hanno tenaglie e braccia per afferrare e tirare oggetti. Si possono muovere lungo il piano verticale.

- foot-bot [14] sono robot con ruote che si muovono sul terreno. Sono equipaggiati di pinze in grado di afferrare gli hand-bot e muoverli per l'ambiente.
- eye-bot [14, 15, 16] sono robot volanti, in grado di attaccarsi al soffitto utilizzando magneti.

Swarmanoid è il successore del progetto Swarm-bots; lo prosegue ed è costruito sui risultati ottenuti da quest'ultimo.

2.1.2 Trasporto di un oggetto

Questo progetto di tesi è la prima parte di uno scenario più completo per lo studio della navigazione di sciami di robot. L'obiettivo consiste nel produrre un sistema composto da eye-bot e foot-bot per il trasporto da una posizione iniziale ad una destinazione finale di un oggetto con dimensioni maggiori a quelle dei foot-bot. Quest'ultimi saranno gli agenti che effettivamente compieranno il trasporto dell'oggetto; il percorso e la traiettoria verranno calcolati dagli eye-bot, i quali comunicheranno poi le istruzioni agli attuatori (i foot-bot) nel momento in cui entreranno nella loro zona di competenza (area visiva). Il progetto, quindi, si compone principalmente di due parti:

- *calcolo del percorso dell'oggetto.* Questa operazione è svolta dagli eye-bot, essi rappresentano l'aspetto sensoriale del sistema e possono supervisionare il processo con una migliore prospettiva essendo attaccati al soffitto. In questa fase si incontra il problema della conoscenza limitata della mappa, il quale viene affrontato attraverso la cooperazione dei robot.
- *attuazione della traiettoria* calcolata nel punto precedente, ed effettivo trasposto dell'oggetto. Questa parte comprende tutti i problemi relativi al coordinamento dei robot al fine di apprendere in che modo muovere l'oggetto per seguire il percorso.

Questa tesi affronta la prima fase, proponendo un'efficiente soluzione al problema, mentre la seconda parte non verrà trattata.

2.2 Caratteristiche dei robot

Come modello di riferimento per i nodi sensoriali del sistema sono stati adottati i robot *eye-bot* [14, 15, 16], i quali sono stati prodotti per il progetto Swarmanoid [1] presso la EPFL di Losanna. Gli eye-bot sono piccoli robot volanti in grado di attaccarsi al soffitto utilizzando un magnete (la progettazione assume soffitti di materiale ferro-magnetico), e sono equipaggiati di una camera pan-and-tilt, la quale permette libertà di orientamento per ottenere una più ampia visione dell'ambiente. Nel nostro studio la camera è utilizzata per osservare l'ambiente dall'alto inquadrando una porzione limitata dell'intera area del problema. In questo lavoro il campo visivo è schematizzato in una regione quadrata, le cui dimensioni dipendono dall'altezza del soffitto, pertanto nell'ambiente di simulazione questo valore è un parametro configurabile.

I robot possono, inoltre, scambiarsi localmente messaggi wireless con limitata larghezza di banda tramite il sistema infrarossi di *range-and-bearing* (IrRB) [17].

Figura 2.1: Un'immagine di un prototipo dei robot volanti presi come riferimento: gli eye-bot.

In Figura 2.1 è mostrata un'immagine di un prototipo dei robot presi come riferimento. È possibile ottenere maggiori dettagli sulle caratteristiche e sull'hardware degli eye-bot in [14], [15] e [16].

Il pianificatore proposto, pur riferendosi a questi specifici robot, mira ad essere sufficientemente generico e rimanere indipendente da una specifica architettura con un determinato equipaggiamento sensoriale. Pertanto il sistema proposto è adattabile a qualsiasi insieme di entità con visione dell'ambiente dall'alto e capaci di comunicare localmente.

2.3 Ambiente e ostacoli

I robot hanno una visione dello spazio dall'alto e gli spostamenti avvengono sul pavimento. Perciò si assumere che i movimenti dell'oggetto si possano modellizzare in un ambiente bidimensionale, rappresentato come un piano. Alcune zone non sono percorribili, poichè occluse da ostacoli. Si assume che gli ostacoli abbiano un'altezza limitata e quindi interferiscano in maniera molto ridotta nella visione della mappa. Pertanto nel progetto viene trascurata la zona nascosta dietro agli ostacoli, la quale potrebbe risultare occlusa in determinate angolazioni, ma si suppone di avere una visione svincolata da questo genere di problemi.

I robot suddividono il piano, e quindi l'ambiente, in due categorie: aree occluse dagli ostacoli e aree libere (*Free-Space*) nelle quali l'oggetto si può muovere liberamente. L'area libera che può essere individuata da un sistema visivo dall'alto corrisponde ad un sottoinsieme del reale spazio percorribile dall'oggetto. Per esempio, in questo sistema un tavolo viene rappresentato come una zona

rettangolare interamente occupata dall'ostacolo, in realtà l'area effettivamente occupata sul terreno è più limitata, pertanto oggetti sufficientemente bassi potrebbero avrete un'area di movimento maggiore rispetto a quella individuata dal sistema visivo utilizzato. In questo progetto è stata privilegiata l'efficienza temporale del sistema a discapito della precisione della mappa. Il sistema adottato, composto da molteplici camere dall'alto, è in grado di acquisire velocemente le informazioni relative alla conformazione dell'ambiente e la disposizione degli ostacoli nella mappa. L'alternativa, in assenza di una conoscenza pregressa della struttura dell'ambiente (raramente presente in ambienti dinamici), corrisponde alla creazione di una mappa con tecniche di SLAM a terra [5, 6, 12], il quale ha un costo temporale molto maggiore rispetto all'acquisizione di un'immagine dall'alto. Inoltre, il nostro sistema ha le potenzialità di reagire velocemente a cambiamenti strutturali di ambienti dinamici, aggiornando con rapidità la nuova disposizione degli ostacoli nella mappa.

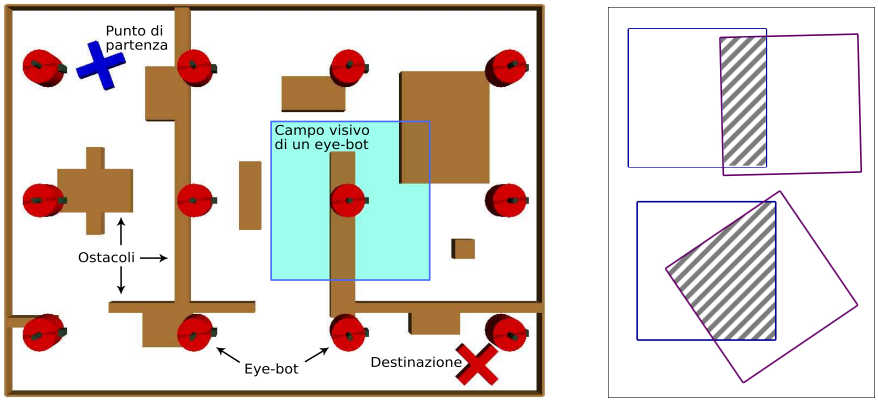
2.4 Formazione iniziale dei robot

Il sistema proposto è in grado di funzionare correttamente e restituire una soluzione solo se opera rispettando determinati vincoli iniziali. Gli eye-bot devono posizionarsi in una formazione che copra l'area compresa fra la posizione iniziale e la destinazione finale. Il campo visivo di ogni robot è stato schematizzato in un'area di forma quadrata e deve sovrapporsi con quello dei suoi vicini, questa zona condivisa deve avere una dimensione pari almeno a quella dell'oggetto. Solo con tale configurazione è possibile permettere agli eye-bot di condividere la visione dell'oggetto in uscita dal campo visivo di un robot ed in ingresso in quello del vicino. Quando la traiettoria esce dall'area di competenza di un eye-bot, quest'ultimo può comunicare la posizione dell'oggetto al proprio vicino, il quale prosegue nel calcolo del cammino. In [19] è descritto un algoritmo che permette ai robot di posizionarsi in una formazione con i vincoli sopracitati. Un esempio di formazione iniziale è presentato in Figura 2.2(a).

Lo sciame mantiene invariato il proprio assetto durante l'intero processo; gli eye-bot sono quindi appesi al soffitto con una posizione fissa, caratterizzata da due coordinate *(x,y)* e un angolo di rotazione. Diversi orientamenti dei campi visivi possono produrre forme particolari e complesse delle aree in condivisione coi vicini (due esempi sono proposti nella Figura 2.2(b)). Questa area comune viene calcolata durante le fasi iniziali, così che ogni entità ha conoscenza della parte di mappa che condivide con in propri vicini. Questa informazione è utilizzata per scambiarsi informazioni sul percorso, ma non sempre è precisa poichè come ogni sistema può essere soggetto ad errori (maggiori dettagli nelle sezioni 2.5 e 5.1).

2.5 Sensori, attuatori e rumore

L'intero progetto è stato sviluppato ed implementato in simulazione; nell'ambiente virtuale è possibile modellare e utilizzare sensori ed attuatori con prestazioni ideali. Quando l'architettura verrà successivamente trasferita sui robot reali, sarà certamente soggetta ad errori provenienti dall'imprecisione intrinseca nei dispositivi sensoriali adottati. Il tipo di errore è strettamente dipendente



(a) (b)

Figura 2.2: (a) Un possibile scenario del problema, in cui si può vedere la disposizione dei robot (cerchi rossi) e l'ambiente in cui si dovrà muovere l'ostacolo - (b) Due esempi di sovrapposizione di campi visivi. I quadrati rappresentano il campo visivo, mentre l'area colorata è la parte di ambiente che i due robot condividono.

dalla tecnologia dei sensori e degli attuatori con cui i robot saranno equipaggiati. Poichè il settore tecnologico è in continua e rapidissima crescita, e poichè l'applicazione creata non vuole essere strettamente legata alla specifica architettura di un determinato robot la quale può essere rapidamente aggiornata, la soluzione proposta è *platform idependent*. Ciò significa che non sono stati esaustivamente trattati i problemi relativi alle limitazioni tecnologiche e non sono state indagate le modalità con cui ottenere informazioni dalle acquisizioni sensoriali. Durante la tesi questo approccio si è tradotto in una generica acquisizione di informazione senza illustrare nel dettaglio le modalità con cui questa sia stata ottenuta. Allo stesso tempo si desidera rendere il progetto il più realistico possibile, e lontano dall'utopica situazione di assenza di rumore. Per questa ragione il sistema ha la possibilità di modellare un generico errore, il quale può essere introdotto e configurato tramite uno specifico modulo dedicato. Rimanendo coerenti con le scelte fatte ed in conseguenza alla mancanza di effettivi sensori ed attuatori specifici: la fonte del rumore non è simulata, ma viene direttamente introdotto un generico errore nell'informazione acquisita.

In questo progetto sono stati modellati tre diversi dispositivi a bordo dei robot, utilizzati per percezioni e azioni nell'ambiente esterno. Qui di seguito sono elencate le tre operazioni principali per le interazioni del robot con l'ambiente, a seguire è illustrato in che modo sono state modellate nel sistema.

- visione: acquisizione della mappa. Posizioni dell'oggetto mobile e degli ostacoli nello spazio;
- *Range and Bearing* (*RB*): rilevazione della posizione e dell'orientamento dei robot vicini;
- comunicazione radio/wireless: scambio di informazioni tra i robot.

La **visione** è utilizzata per acquisire informazioni relative alla mappa, in particolare dai dati e dalle immagini ricevute dalla camera si estraggono le posizioni degli ostacoli. Il processo di estrazione di informazione dall'immagine ed i relativi algoritmi di visione artificiale non sono stati applicati durante questo progetto. Si suppone che il robot sia in grado di estrarre queste informazioni tramite l'utilizzo della camera di cui è equipaggiato. Questa acquisizione sarà probabilmente soggetta ad errore; tale aspetto non è preso in considerazione in questo progetto di tesi. Si assume una visione della mappa perfetta.

Il **sistema *Range and Bearing*** permette di ottenere informazioni sulla posizione dei robot vicini, in particolare è misurata la distanza e l'angolo relativo all'orientamento del robot. L'architettura proposta ignora i mezzi con cui viene calcolata la posizione: a fronte di un generico metodo di RB, il simulatore restituisce al robot le coordinate polari della posizione dei robot vicini con un rumore configurabile. L'accuratezza del sistema RB influisce sulla qualità della soluzione trovata ed in alcuni casi uno scorretto rilevamento della posizione dei vicini può portare ad un fallimento del processo anche quando è presente una soluzione. Il rumore è introdotto in maniera stocastica, utilizzando una distribuzione Gaussiana a media nulla; tramite un file di configurazione è possibile impostare il valore della deviazione standard (sia per la misurazione della posizione, che dell'orientamento). I robot usati come riferimento, gli eye-bot, utilizzano un sistema a infrarossi per il rilevamento della posizione relativa [17]. Questa tecnologia permette di individuare le posizioni dei robot vicini collegati da una linea visiva (*line of sight*) entro un limitato raggio d'azione.

La **comunicazione** è un aspetto chiave dell'intero sistema ed è presente in ogni fase del processo poichè rappresenta il principale strumento di collaborazione dei robot per superare i limiti introdotti dall'approccio distribuito. Ormai la totalità dei robot è equipaggiata di un sistema di comunicazione Wi-Fi, che permette la comunicazione su ampie distanze. Tuttavia, nell'architettura presentata le comunicazioni avvengono in accordo con il sistema RB, il quale definisce l'insieme dei vicini con i quali cooperare. Durante l'intero processo, lo scambio di messaggi è limitato localmente a questo insieme di robot. Questa scelta progettuale è stata fatta in conseguenza ai requisiti di scalabilità prefissati. In questo modo, con l'utilizzo locale di messaggi, si evita il sovraccarico dei canali trasmissivi al crescere dei nodi. L'effettiva comunicazione potrà poi essere effettuata sia con la tecnologia infrarossi, che con il collegamento Wi-Fi. In ogni caso, trattandosi di comunicazione wireless è possibile che ci siano disturbi, o momentanee interruzioni di connessione fra le entità, le quali possono produrre rumore o una mancata ricezione di alcuni messaggi. In letteratura sono stati proposti numerosi ed efficaci metodi per garantire il corretto scambio di informazioni, spesso strettamente dipendenti al tipo di tecnologia utilizzata. Durante questo progetto non si è voluto trattare le tematiche relative ai problemi comunicativi; si suppone di avere una comunicazione efficiente con corretto invio e ricezione di ogni messaggio.

2.6 ARGoS - L'ambiente di sviluppo

Il pianificatore proposto in questa tesi è stato sviluppato e testato nel simulatore ARGoS [13]: un ambiente di sviluppo continuous-time per la simulazione contemporanea di un molteplice numero di robot. Questo potente strumento

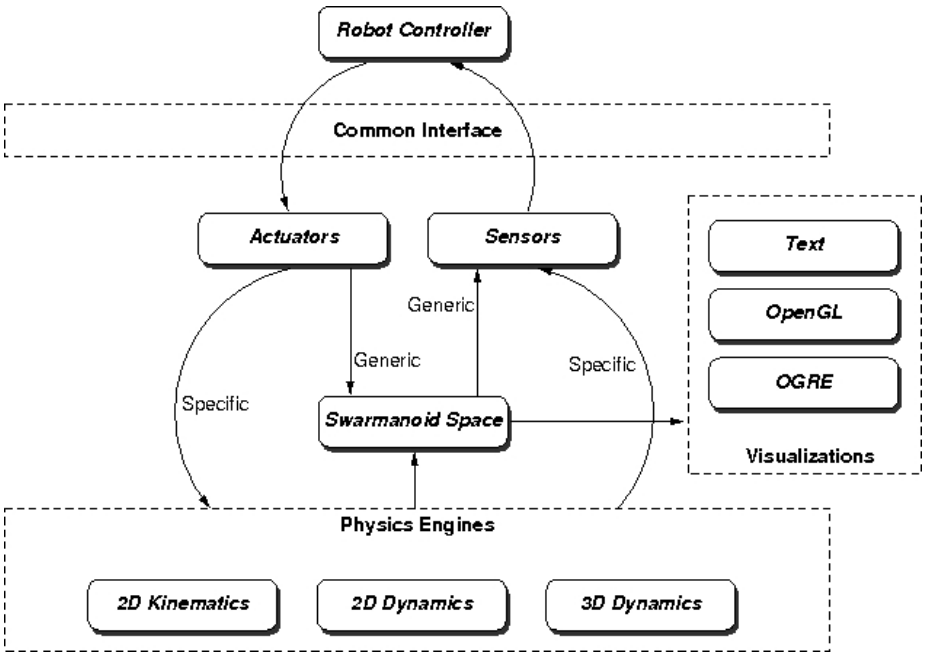


Figura 2.3: Le caratteristiche architetturali generali del simulatore ARGoS.

permette di implementare il controllore del robot, testarlo rapidamente e quando completo consente di trasferirlo direttamente sul robot reale. Il simulatore ARGoS è stato sviluppato con lo scopo di simulare i tre differenti tipi di robot che compongono lo swarmanoid (eye-bot, hand-bot e foot-bot) ed il loro ambiente. Tuttavia può essere utilizzato anche per la simulazione di altri tipi di robot, come ad esempio l'e-puck e l's-bot il cui modello è già integrato. Ogni robot è caratterizzato da una struttura fisica, un insieme di sensori ed attuatori, e un modulo di controllo. L'architettura del simulatore è organizzata intorno ad un singolo componente, lo Swarmanoid Space. Si tratta di sistema di riferimento centrale che ad ogni passo rappresenta l'attuale stato di simulazione: contiene informazioni circa la posizione e l'orientamento di ogni entità simulata. Gli altri componenti del simulatore per il loro funzionamento interagiscono con lo Swarmanoid Space. Il motore fisico calcola i movimenti e le interazioni fisiche in conseguenza alle azioni delle diverse entità; i render visualizzano il contenuto dello SS ad ogni passo di simulazione. I sensori e gli attuatori possono interagire con lo SS oppure direttamente con il motore fisico. I controllori dei robot interagiscono con il resto del simulatore attraverso i sensori e gli attuatori, come avviene per i controllori dei robot reali.

Capitolo 3

Stato dell'arte

Il problema della pianificazione di percorso è stato ampiamente studiato dalla comunità robotica. Il problema si riferisce alla ricerca di un percorso senza collisioni in un ambiente con ostacoli statici, oppure dinamici. I movimenti dell'oggetto possono essere soggetti a vincoli ed avere un ridotto numero di gradi di libertà controllabili. Da alcuni anni sono state proposte numerose soluzioni, ognuna con vantaggi e svantaggi in relazione allo scenario di applicazione. Il pianificatore studiato in questa tesi opera in un ambiente a due dimensioni con cinematica non vincolata e ostacoli statici. La letteratura scientifica ha trattato ampiamente questo genere di problemi, e le soluzioni di Latombe [11, 10] e Closet et al. [4] sono dei lavori di riferimento nel settore.

La maggior parte dei più recenti algoritmi di path planning esplora lo spazio in modo stocastico con utilizzo di euristiche che permettono una rapida convergenza verso una soluzione praticabile ma spesso lontana dall'ottimo. Questo approccio è adottato in conseguenza ad una crescita esponenziale dello spazio di ricerca in problemi con un elevato numero di gradi di libertà. Tale esplosione dimensionale dello spazio rappresenta una delle principali difficoltà tuttora irrisolta per l'intera comunità scientifica. Il problema affrontato in questo lavoro mantiene dimensioni dello spazio di ricerca ridotte poichè riguarda movimenti con 3 gradi di libertà, pertanto è possibile adottare una soluzione più rapida e precisa. In particolare è stato utilizzato l'algoritmo di discesa del potenziale su un piano discretizzato in celle a dimensione fissa proposto in [2]: *wave front expansion with skeleton*. Questa soluzione è composta principalmente da due fasi: (i) la creazione del campo di potenziale; (ii) la discesa del potenziale.

Nella prima fase viene inizialmente diffuso il potenziale su un sottoinsieme dello *spazio libero*, chiamato *skeleton*, il quale corrisponde al diagramma di Voronoi [4, 7, 20]; successivamente il potenziale è computato sul resto della mappa. Questo approccio permette di ottenere percorsi robusti e lontani dagli ostacoli.

La seconda fase di discesa del potenziale è effettuata tramite A-star, proposto in [8]; i principali vantaggi che hanno portato a scegliere questo algoritmo sono: la sua rapidità, e la selezione del percorso più breve, mantenendo ogni volta traccia del costo del percorso già effettuato.

Un ulteriore aspetto affrontato è il movimento di un oggetto di grosse dimensioni e con forme particolari attraverso gli ostacoli. La soluzione adottata segue i metodi presentati in [4, 9], in cui l'oggetto viene modellato come un insieme di

control point. Questi punti sono utilizzati per descrivere la posizione, modellare i movimenti nello spazio e calcolare le euristiche di distanza.

Il sistema studiato ha una visione dell'ambiente dall'alto, con la quale le aree coperte da ostacoli vengono classificate come non percorribili da movimenti dell'oggetto a terra. Questa assunzione non è sempre vera, in quanto, con una prospettiva dall'alto, alcuni ostacoli possono essere rappresentati con un'area di occlusione maggiore di quella realmente occupata sul terreno (per esempio un tavolo con quattro gambe). Da ciò si può dedurre che l'area che il sistema classifica come percorribile è un sottoinsieme del reale spazio di movimento a terra. In letteratura sono presenti soluzioni efficaci che permettono di individuare l'intera area libera da ostacoli attraverso metodi di creazione della mappa da parte di robot a terra (SLAM) [5, 6, 12]. Tali robot hanno un sistema sensoriale di percezione dell'ambiente dalla stessa prospettiva dell'oggetto mobile, pertanto è possibile individuare l'intero ambiente libero da ostacoli. I metodi di SLAM richiedono la scansione completa dell'ambiente, percorrendolo interamente ed analizzando ogni parte accessibile. Tale operazione può richiedere un consistente utilizzo di risorse computazionali e soprattutto temporali. Il nostro approccio si muove nella direzione di una maggiore efficienza temporale, in quanto i robot volanti possono acquisire rapidamente la struttura dell'ambiente. Inoltre, questo sistema di visione dall'alto ha le potenzialità di reagire velocemente a cambiamenti strutturali di ambienti dinamici, aggiornando con rapidità la mappa con la nuova disposizione degli ostacoli. É possibile ottenere questa maggiore efficienza temporale, a fronte di una mappa meno precisa; tuttavia si assume che le aree percorribili nascoste siano relativamente limitate. Tra gli sviluppi futuri di questo progetto, si intende raggiungere una cooperazione tra robot volanti e robot attuatori a terra, in modo da integrare le acquisizioni sensoriali da diverse prospettive, e quindi modificare la traiettoria durante la fase di esecuzione a fronte di un incremento delle aree percorribili, che erano prima nascoste.

L'aspetto distribuito del sistema sensoriale si riflette in una suddivisione della mappa completa dell'ambiente in un insieme di mappe parziali parzialmente sovrapposte. Uno degli aspetti critici del sistema studiato è la corretta sovrapposizione delle mappe confinanti. In letteratura sono proposti svariati metodi per la ricostruzione della mappa totale, basati su tecniche di sovrapposizione e correlazione di parti comuni dell'ambiente, come proposto in [3, 6]. Queste soluzioni presuppongono uno scambio della mappa parziale fra nodi vicini, per effettuare la combinazione delle due. La nostra scelta progettuale accentua l'attenzione sulla scalabilità del sistema; pertanto, in favore di minori comunicazioni, non adotta tali tecniche per la ricostruzione di mappe. In questo modo si opera a vantaggio di un minore utilizzo di risorse comunicative e computazionali, ma a discapito di un maggiore errore di allineamento.

Capitolo 4

Pianificazione di percorso

L'argomento principale di questo lavoro di tesi è un pianificatore di percorso distribuito. In questo capitolo sono descritte le soluzioni adottate per la pianificazione del percorso, sono illustrati i dettagli implementativi e descritto l'algoritmo di ricerca utilizzato. Il capitolo successivo si concentra sull'aspetto distribuito, evidenziando l'architettura del sistema proposto.

La tesi sviluppa inizialmente un pianificatore con conoscenza globale, applicando le tecniche e gli algoritmi considerati migliori rispetto alle specifiche caratteristiche del problema. Questa prima implementazione, oltre che rappresentare il naturale approccio iniziale al problema, è stata utilizzata in fase di valutazione dei risultati come strumento di paragone. La soluzione ottenuta con l'approccio globale viene considerata la traiettoria di riferimento, permettendo così uno studio dei risultati significativo. Il pianificatore con conoscenza totale della mappa rappresenta il punto di partenza del lavoro presentato in questa tesi: è l'implementazione dello stato dell'arte del problema di pianificazione di percorso.

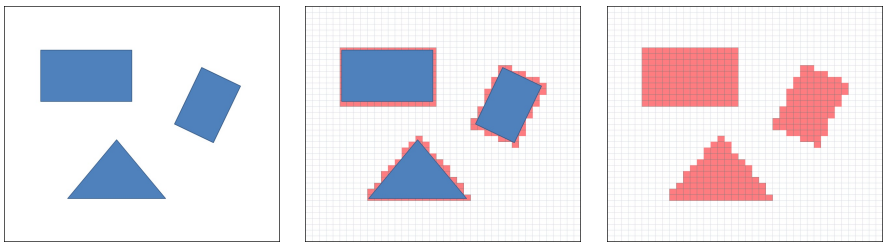
In questo progetto viene proposto un pianificatore di movimenti di oggetti olonomici in un piano bidimensionale. La mappa viene tradotta in una matrice effettuando una segmentazione della mappa di celle uniformi, questa scelta progettuale è stata effettuata al fine di facilitare lo scambio di informazioni tra robot sulla posizione dell'oggetto mobile. Inoltre, permette di rappresentare in modo uniforme i movimenti dell'oggetto nello spazio (traslazioni unitarie).

L'algoritmo utilizzato per il calcolo del percorso è *wave front expansion with skeleton*, il quale è basato sulla tecnica del campo di potenziale numerico, computato con l'utilizzo dell'espansione del fronte con skeleton di Voronoi. Mentre, l'effettiva discesa del potenziale è implementata con il rapido algoritmo di discesa A* (A-star).

Nei seguenti paragrafi sono descritti: la rappresentazione della mappa, il modello dell'oggetto mobile, il campo di potenziale, lo skeleton di Voronoi e l'algoritmo A-star.

4.1 La rappresentazione della mappa

I robot sono attaccati al soffitto e sono equipaggiati di videocamera puntata verso il basso, in questo modo possono acquisire immagini dell'ambiente dall'al-



(a) (b) (c)

Figura 4.1: Segmentazione dello spazio in celle uniformi. Conversione dell'immagine dell'ambiente visto dall'alto con ostacoli in blu e pavimento in bianco in matrice di celle (bianche libere; rosse occluse).

to. Processando l'immagine è possibile rilevare gli ostacoli e creare una mappa bidimensionale dell'ambiente, distinguendo lo spazio libero (pavimento) dalle aree occluse (ostacoli).

L'ambiente viene poi discretizzato: tra le numerose tecniche di partizione della mappa, per questo lavoro è stata adottata la decomposizione in celle uniformi. La mappa è tradotta in una matrice rettangolare M di celle a dimensione fissa d. Ogni cella $M(i, j)$ rappresenta lo spazio compreso fra $(i \cdot d)$ e $[(i+1) \cdot d]$ sull'asse x e fra $(j \cdot d)$ e $[(j+1) \cdot d]$ sull'asse y. In questo modo la mappa può essere rappresentata con una matrice di 1 e 0, memorizzando se una cella è totalmente libera con uno 0 oppure se totalmente o parzialmente occlusa con un 1.

La Figura 4.1 mostra un esempio grafico di segmentazione dello spazio in una matrice di celle uniformi.

La discretizzazione dello spazio semplifica notevolmente il successivo calcolo del campo di potenziale, infatti rende possibile associare un singolo valore ad ogni cella, evitando la creazione e la gestione di una funzione continua. Un ulteriore vantaggio della scomposizione in celle a dimensione fissa riguarda la rappresentazione dell'oggetto nella mappa: è possibile descrivere i movimenti e le posizioni dell'oggetto in maniera uniforme su tutta la mappa e condividere l'informazione con gli altri robot del sistema (maggiori dettagli sulla rappresentazione dell'oggetto mobile sono presentati nella sezione 4.4).

La conversione dello spazio continuo in uno spazio discreto è una semplificazione che velocizza il processo e semplifica l'esecuzione, tuttavia può ridurre lo spazio delle soluzioni e portare ad un fallimento del sistema anche quando una soluzione è presente. Perciò è necessario selezionare con attenzione la dimensione delle celle, in quanto un valore ridotto produce un'accurata rappresentazione dell'ambiente, ma aumenta il tempo computazionale e l'utilizzo della memoria. Dall'altro lato, celle di grosse dimensioni migliorano le prestazioni computazionali, ma non permettono un'accurata rappresentazione della mappa. Questo sistema è stato progettato per operare in modo distribuito con conoscenza di una porzione ridotta della mappa, pertanto è possibile mantenere una fine discretizzazione della mappa con un numero accettabile di celle (per esempio per una porzione di 4×4 metri, e possibile avere una matrice 80×80 con celle di 5 cm).

4.2 Campo di potenziale

Il campo di potenziale è una forza che attrae l'oggetto mobile verso la destinazione, ed allo stesso tempo lo respinge dagli ostacoli. In una mappa a griglia questa forza si traduce in una funzione che associa ad ogni cella un valore che rappresenta la magnitudine della forza. La funzione ha il minimo globale nella destinazione, e valore elevato sugli ostacoli. In tutte le altre celle, la funzione decresce verso la destinazione del percorso, così che il pianificatore può raggiungere la posizione finale seguendo il gradiente negativo del potenziale. Il valore alto del campo di potenziale previene che l'oggetto collida con gli ostacoli.

Sommando la forza repulsiva e quella attrattiva si ottiene il campo di potenziale nella sua accezione originale. Solitamente la principale causa di inefficienza di questo metodo è il minimo locale, poichè l'attrazione verso la destinazione finale può portare l'oggetto in un vicolo cieco, con relativi sprechi di risorse. Questo problema è stato risolto grazie alla tecnica del fronte di espansione [10]. Il funzionamento di questa soluzione è illustrato qui di seguito. Nella posizione finale la forza del campo di potenziale è nulla: non è applicata alcuna forza attrattiva, poichè l'obiettivo è raggiunto. Partendo dalla destinazione, il potenziale è diffuso sull'intera mappa, espandendolo di una cella alla volta. Ad ogni passo viene incrementato il valore di 1, assegnando il nuovo valore a tutte le celle non ancora visitate e confinanti con le celle sul *fronte di espansione*.

Questo algoritmo di diffusione genera un campo di potenziale che mantiene la proprietà attrattiva verso la destinazione, ma perde completamente la componente repulsiva. Infatti, utilizzando questo metodo, si ottengono percorsi che raggiungono la destinazione con la traiettoria più breve, ma spesso generano collisioni con gli ostacoli, o passano molto vicino ad essi. Obiettivo di questo progetto è, al contrario, ottenere percorsi robusti che tendano a rimanere distanti dagli ostacoli almeno in fase di pianificazione, poichè è atteso un certo grado di imprecisione nella successiva fase di realizzazione pratica di quanto pianificato.

L'utilizzo del metodo di *espansione del fronte con skeleton* [2] permette la diffusione del campo di potenziale conservando la proprietà di repulsione dagli ostacoli: valori più bassi sono associati alle celle più distanti dagli ostacoli. Nella prossima sezione è illustrata nel dettaglio la soluzione adottata per il calcolo del campo di potenziale.

4.2.1 Espansione del fronte con skeleton

Questo metodo di espansione del campo di potenziale associa un valore numerico ad ogni cella; il risultato è una mappa con valori che decrementano verso la destinazione del problema. Due importanti caratteristiche di questo metodo sono la circumnavigazione degli ostacoli durante l'espansione e la repulsione dagli ostacoli. La prima permette sia di avere valori veritieri sulla reale distanza dalla destinazione sia di indicare la corretta direzione verso l'obiettivo con un percorso libero da ostacoli. La seconda assegna un potenziale inferiore alle celle più distanti dagli ostacoli, ed incrementa il valore muovendosi verso gli ostacoli.

Il processo si compone di tre fasi:

1. la creazione dello skeleton;
2. la diffusione del potenziale sullo skeleton;

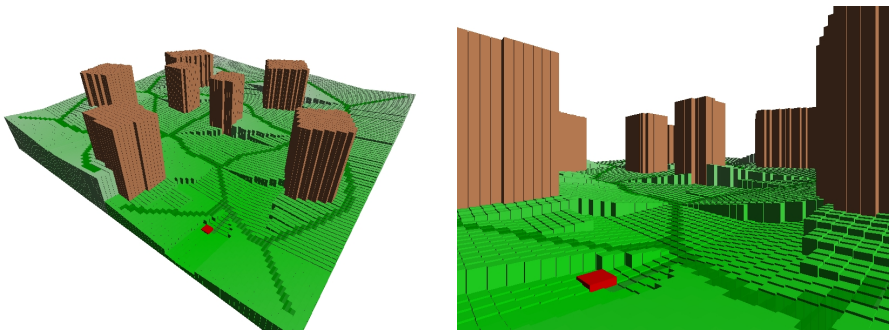


Figura 4.2: Rappresentazione grafica tridimensionale del campo di potenziale. Se gli ostacoli sono considerati come montagne a causa del loro elevato valore di campo di potenziale, lo skeleton di Voronoi può essere considerato come le valli, le quali giacciono fra gli ostacoli.

3. la diffusione del potenziale sulle restanti celle libere.

Durante la prima fase viene generato lo skeleton, si tratta di un grafo non orientato con punti equidistanti dagli ostacoli, che corrisponde al diagramma di Voronoi su un piano a due dimensioni. Un esempio dello skeleton è illustrato dalle celle gialle in Figura 4.3(a). Se gli ostacoli sono considerati come montagne a causa del loro elevato valore di campo di potenziale, lo skeleton di Voronoi può essere considerato come le valli, le quali giacciono fra gli ostacoli (Figura 4.2). Allo skeleton viene poi collegato il punto di arrivo seguendo il percorso diretto più breve.

La seconda fase riguarda la diffusione del potenziale sullo skeleton; il processo inizia dalla destinazione del percorso, alla cui cella è assegnato valore zero: il campo di potenziale è nullo poichè all'arrivo non è applicata alcuna forza attrattiva. L'espansione del potenziale avviene con lo stesso metodo di diffusione illustrato precedentemente: si applica un incremento unitario ad ogni cella non ancora visitata confinate con il fronte di espansione.

Quando la diffusione sullo skeleton è completata, viene eseguita la terza fase, in cui si associa un valore alle restanti celle libere. Partendo dallo skeleton, si espande il campo verso gli ostacoli, incrementando il valore di 1 ad ogni passo; in Figura 4.3 è mostrata una rappresentazione grafica di alcuni passaggi del processo di espansione del potenziale. Alle celle occupate da ostacoli viene assegnato il valore massimo di potenziale.

4.3 La discesa del potenziale con A-star

Completata la fase di diffusione del potenziale, è possibile proseguire con l'effettiva pianificazione di un percorso dal punto di partenza alla destinazione. Partendo dall'origine del percorso, l'oggetto viene spostato seguendo il gradiente negativo del campo di potenziale, fino al raggiungimento della posizione finale. L'algoritmo adottato per la discesa del potenziale è A* (A-star). Questo metodo utilizza le seguenti strutture:

- configurazione: è una posizione univoca nello spazio dell'oggetto mobile;

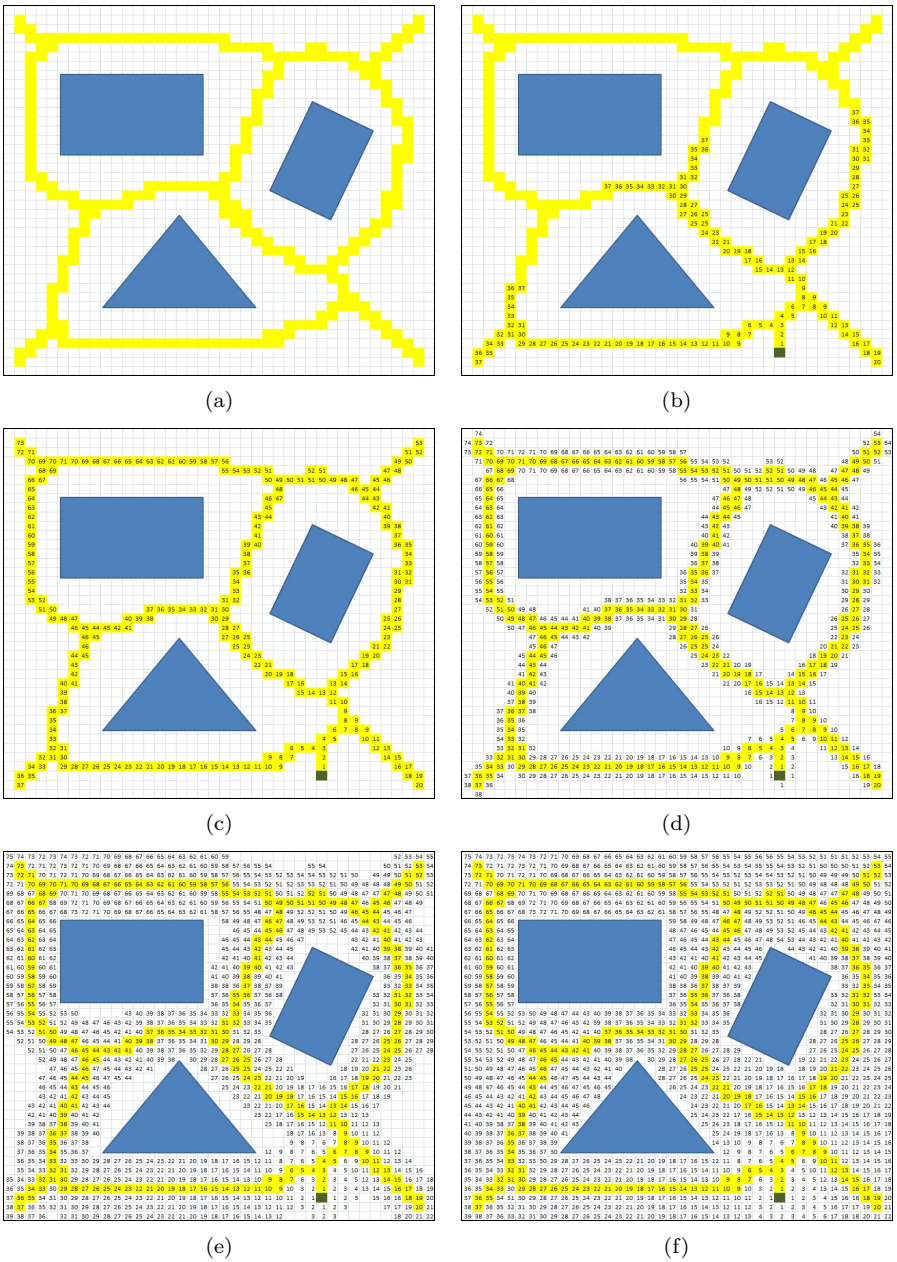


Figura 4.3: Diffusione del campo di potenziale sulla mappa. Bianco: spazio libero. Blu: ostacoli. Giallo: skeleton. Cella verde: destinazione finale. (a) Viene selezionato il sottoinsieme dello spazio libero, chiamato skeleton, il quale corrisponde al diagramma di Voronoi (celle gialle). (b) Viene diffuso il potenziale sulle celle dello skeleton a partire dalla destinazione. (c) Diffusione del potenziale sullo skeleton completata. (d), (e) Diffusione del campo di potenziale sulle restanti celle libere. (f) Diffusione del campo di potenziale completata.

è descritta da una posizione ed un orientamento. Ogni configurazione possiede i seguenti attributi:

  - valore G: il costo del percorso effettuato. Rappresenta il costo degli spostamenti effettuati dalla partenza alla configurazione corrente
  - valore H: una stima euristica della distanza dalla destinazione, la quale non deve sovrastimare la reale distanza dalla destinazione. Per questo valore si utilizza il campo di potenziale.
  - valore F: una stima del costo dell'intero percorso che passa dalla configurazione corrente. Questo valore è calcolato come somma del costo del percorso già effettuato (valore G) e della stima di costo del percorso da eseguire (valore H).
  - genitore: la configurazione precedente a quella corrente. Si tratta della configurazione prima dell'ultimo movimento effettuato.

- open-set: la lista di configurazioni che devono essere ancora visitate.
- closed-set: la lista delle configurazioni che sono state visitate ed analizzate durante la ricerca. È un albero di configurazioni che cresce durante l'esecuzione dell'algoritmo.

L'algoritmo in pseudo-codice di A-star è illustrato in Figura 4.4.

Il processo inizia inserendo la configurazione di partenza nella lista delle configurazioni da visitare: l'open-set. Durante ogni passo viene selezionata la configurazione dell'open-set con valore F minore. Tutte le configurazioni *vicine* a quella corrente (traslazioni e rotazioni) che non sono ancora state visitate e libere da collisioni con ostacoli vengono inserite nella lista di configurazioni da visitare (open-set). La configurazione visitata viene così rimossa dall'open-set ed inserita nel closed-set: in questo modo viene aggiunto un nodo all'albero di ricerca della soluzione. L'esecuzione termina quando si manifesta uno dei due seguenti eventi: (i) viene visitata la configurazione di destinazione; oppure (ii) si esauriscono le configurazioni da visitare, cioè l'open-set è vuoto e sono state pertanto visitate tutte le configurazioni raggiungibili dalla partenza, e nessuna di queste corrisponde alla destinazione. In quest'ultimo caso, il processo ha espanso totalmente l'albero di ricerca e restituisce un messaggio di fallimento, poichè non può trovare alcun percorso valido. Invece, nel caso in cui il pianificatore trovi la destinazione, viene ricostruito il percorso: il pianificatore, a partire dalla configurazione finale, ripercorre a ritroso il cammino seguendo ad ogni passo la *configurazione genitore* fino alla configurazione di partenza.

Questo algoritmo di pianificazione può essere visto come una espansione dell'albero di ricerca, in cui ad ogni passo viene aggiunto un nodo (configurazione) all'albero (closed-set). L'esplorazione è eseguita saltando da un nodo all'altro, selezionando ogni volta il nodo con valore F minore.

4.4 Modello dell'oggetto mobile

L'algoritmo presentato nelle sezioni precedenti è progettato per il movimento di un punto nello spazio. Il problema che si affronta è più complesso e consiste nel pianificare il trasporto di un oggetto rigido di forma arbitraria e generalmente con dimensioni ingombranti da un punto di partenza fino a destinazione.

```
function A-Star(start,goal){
  closedset = the empty_set
  // The set of nodes already evaluated
  openset = set containing the initial node
  // The set of tentative nodes to be evaluated
  g_score[start] = 0
  // Distance from start along optimal path
  h_score[start] = heuristic_estimate_of_distance(start, goal)
  f_score[start] = h_score[start]
  // Estimated total distance from start to goal through y
  while (openset is_not_empty){
    x = the node in openset having the lowest f_score[] value
    if (x == goal)
      return reconstruct_path(came_from[goal])

    remove x from openset
    add x to closedset
    foreach y in neighbor_nodes(x){
      if y in closedset
        continue

      tentative_g_score = g_score[x] + dist_between(x,y)

      if y not in openset
        add y to openset
        tentative_is_better = true
      else if tentative_g_score < g_score[y]
        tentative_is_better = true
      else
        tentative_is_better = false

      if (tentative_is_better == true){
        came_from[y] = x
        g_score[y] = tentative_g_score
        h_score[y] = heuristic_estimate_of_distance(y, goal)
        f_score[y] = g_score[y] + h_score[y]
      }
    }
  }
  return failure
}

function reconstruct_path(current_node){
  if came_from[current_node] is set
    p = reconstruct_path(came_from[current_node])
    return (p + current_node)
  else
    return current_node
}
```

Figura 4.4: L'algoritmo A* (A-star) per la discesa del potenziale - Pseudo codice

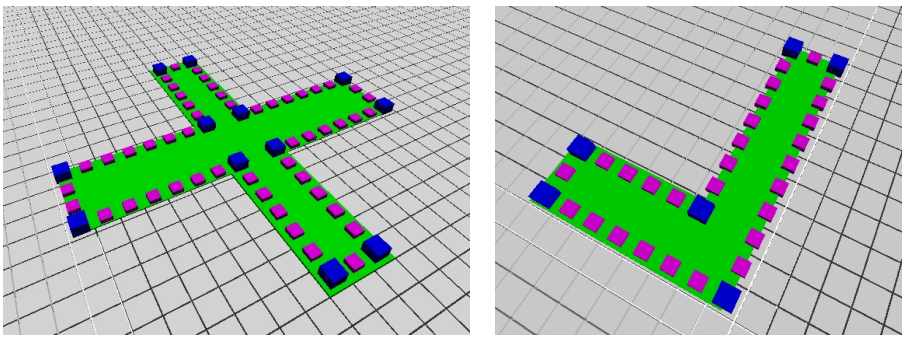


Figura 4.5: Rappresentazione grafica tridimensionale dell'oggetto. Sono evidenziati i due tipi di punti: in blu i *control points* ed in viola i *collision points*.

Nelle due seguenti sottosezioni sono illustrate le modifiche applicate al pianificatore al fine di operare con questo genere di oggetti mobili. In particolare sono descritte la rappresentazione dell'oggetto di grosse dimensioni e le modifiche applicate al processo di pianificazione di percorso.

4.4.1 La rappresentazione dell'oggetto

Date le dimensioni dell'oggetto superiori alle dimensioni unitarie di discretizzazione, l'oggetto in ogni istante occupa più di una cella. Per tale motivo esso viene descritto da una sequenza di punti, chiamati *control points*, posizionati sui lati dell'oggetto. Il pianificatore ricava la forma dell'oggetto attraverso la congiunzione con linee rette dei control points in sequenza, pertanto è importante selezionare accuratamente questi punti, i quali di norma sono posizionati sugli angoli dell'oggetto. I control points rappresentano l'oggetto mobile e sono utilizzati per tutte le operazioni ad esso relativo: per calcolare i suoi spostamenti, per misurare come influisce il campo di potenziale su di esso, o per comunicarne la posizione ai robot vicini. I control points vengono impostati come parametro iniziale, e vengono indicate le coordinate di partenza e di destinazione. Il sistema calcola successivamente in modo automatico un altro insieme di punti: i *collision points*. Questi punti sono generati interpolando i control points, e selezionando un punto per ogni cella attraversata. I collision points sono utilizzati durante gli spostamenti dell'oggetto per controllare eventuali collisioni con ostacoli. I collision points coprono interamente i lati dell'oggetto, e sono in numero maggiore rispetto i control points, ma vengono utilizzati solamente per rilevare possibili collisioni. La Figura 4.5 illustra una rappresentazione grafica tridimensionale dell'oggetto, sul quale sono evidenziati i due tipi di punti: in blu i control points ed in viola i collision points.

4.4.2 Modifiche all'algoritmo di path planning

Data la sopradescritta struttura dell'oggetto, l'algoritmo non può limitarsi al calcolo del percorso per un singolo punto. La pianificazione deve essere modificata e riadattata al fine di operare con oggetti composti da una molteplicità di

punti con posizioni relative vincolate tra loro. Le caratteristiche che necessitano nuove soluzioni sono le tre seguenti:

- diffusione del potenziale
- calcolo dei valori G ed H
- movimenti unitari - i *vicini* di ogni configurazione

Qui di seguito sono elencate le soluzioni adottate per adattare il problema al trasporto di un oggetto di grandi dimensioni.

Diffusione del potenziale

La diffusione del potenziale ha inizio da un singolo punto (il punto di arrivo), dal quale il potenziale si espande incrementando il valore ad ogni passo. Poichè la configurazione finale è composta da un insieme di punti (control points), è necessario selezionare il punto di partenza dal quale iniziare la fase di diffusione. Come punto per l'espansione del potenziale è utilizzato il centro dell'oggetto, calcolato in modo automatico come il punto di mezzo fra i control point più esterni. In questo modo il campo di potenziale rappresenta il valore d'attrazione per il centro dell'oggetto e non per l'intero oggetto; tale valore può essere considerato attendibile quando l'oggetto è distante dalla destinazione. Nel momento in cui l'oggetto si avvicina alla destinazione e la distanza da essa scende sotto una soglia predefinita, si utilizzano mappe di potenziale multiple. Ogni mappa contiene i valori di attrazione per un singolo control point, di conseguenza si hanno N mappe, con N pari al numero di control point. L'utilizzo di mappe multiple può risultare computazionalmente costoso, per questo motivo è stato limitato solo ad una parte dello spazio: l'area vicina alla destinazione. In questo modo è possibile ottenere il corretto orientamento dell'oggetto. In conseguenza a questa soluzione, il pianificatore effettuerà l'allineamento con la posizione obiettivo e quindi le relative rotazioni solo nella parte finale del percorso.

Valori della configurazione in A-star

I valori da utilizzare durante la pianificazione di movimento di un punto sono intuitivi: il valore G incrementa in maniera costante poichè i movimenti sono traslazioni unitarie e il valore H corrisponde al campo di potenziale della cella occupata. Per la pianificazione di percorso di un oggetto di grandi dimensioni i valori sono calcolati nel seguente modo:

- valore G: rappresenta il costo del percorso dalla partenza al punto corrente. La prima configurazione di partenza ha valore 0 (costo nullo), e per ogni successivo movimento il valore viene incrementato opportunamente. Ogni traslazione di una cella ha costo unitario (+1), mentre le rotazioni hanno un costo proporzionale al numero di celle attraversate (+1 ogni cella).
- valore H: è una stima della distanza dal punto corrente alla dalla destinazione. Questa stima è calcolata utilizzando il campo di potenziale, il quale ha un significato molto simile: ad ogni cella è assegnato un valore di potenziale pari al numero di celle necessarie a raggiungere la destinazione circumnavigando gli ostacoli. Quando l'oggetto è composto da più control

point il valore H è la somma del potenziale delle celle su cui giacciono i punti.

I *vicini*

Durante l'esecuzione dell'algoritmo di discesa del gradiente, quando il pianificatore visita una configurazione aggiunge tutte le *configurazioni vicine* non ancora visitate e libere da collisioni con ostacoli alla lista dei nodi da visitare (open-set). Per *configurazioni vicine* (o *vicini*) si intende tutte le configurazioni distanti un movimento dalla configurazione di partenza. Ci sono due tipi di movimenti: la traslazione e la rotazione. Ogni configurazione ha un minimo di 6 *vicini*:

- 4 traslazioni di una cella;
- 2 rotazioni sul centro.

È possibile che alcuni oggetti con forme particolari abbiano ulteriori fulcri di rotazione, i quali possono essere esplicitamente specificati prima dell'esecuzione oppure più normalmente possono corrispondere ai control point dell'oggetto. Un ulteriore parametro da specificare è l'angolo di rotazione; prima della pianificazione è possibile impostare i gradi dei movimenti rotatori. Più il passo di rotazione è piccolo, maggiore saranno i movimenti possibili; ciò significa che alcuni percorsi molto complessi possono essere trovati solo con esecuzioni con passo di rotazione sotto una certa soglia. Un passo di rotazione molto ridotto può generare traiettorie morbide ed aumentare la possibilità di trovare una soluzione. Tuttavia il percorso sarà composto da un numero maggiore di movimenti e quindi di configurazioni, ciò richiede maggiori risorse computazionali per l'esecuzione dell'algoritmo di discesa A-star. Pertanto è necessario selezionare un valore adeguato da permettere un elevato numero di movimenti all'oggetto, ed allo stesso tempo limitare il numero di configurazioni durante l'esplorazione dello spazio. Per gli scenari studiati durante gli esperimenti i gradi di rotazione hanno valori compresi fra un massimo di 30° ed un minimo di 5°.

4.5 Ottimizzazione del percorso

Le traslazioni sono limitate alle quattro direzioni principali (cella sopra, cella sotto, cella a destra e cella a sinistra), permettendo all'oggetto movimenti riconducibili alla *geometria di Manhattan*. Questa assunzione semplifica il processo, ma può produrre movimenti irregolari e poco naturali per traiettorie con traslazioni diagonali dell'oggetto. Questo problema è superato grazie all'introduzione di un processo di ottimizzazione del percorso successivo alla fase di pianificazione. Quando il sistema termina la pianificazione e restituisce il percorso finale viene eseguita l'ottimizzazione della traiettoria. Le coppie di movimenti traslatori che portano ad una posizione diagonale, vengono sostituiti da un unica traslazione dell'oggetto (per esempio la coppia di traslazioni sopra-destra può essere sostituita da un unica traslazione nella cella in alto a destra). Grazie a questo semplice e rapido processo di ottimizzazione, il percorso finale ottiene una traiettoria più naturale, ed allo stesso tempo mantiene una esecuzione semplice in fase di pianificazione e ricerca della soluzione.

Capitolo 5

Pianificazione di Percorso Distribuita

La pianificazione di percorso distribuita è caratterizzata da una frammentazione della conoscenza fra una molteplicità di entità locali (nella fattispecie i robot eye-bot). Ogni robot ha una conoscenza parziale dell'ambiente, limitata alla propria zona di competenza. Questo tipo di approccio, oltre che in fase sensoriale, è mantenuto anche durante l'effettiva pianificazione, pertanto nessun robot in nessuna fase dell'esecuzione ha conoscenza totale sull'intero processo: l'architettura proposta è strutturata senza alcun supervisore globale. Grazie alla cooperazione ed ad un limitato scambio di informazioni l'intero sistema è in grado di trovare, se presente, una percorso libero da collisioni dall'inizio all'arrivo.

Il processo si compone principalmente di tre fasi: una prima fase di configurazione, una seconda fase di acquisizione sensoriale dell'ambiente e creazione della mappa, ed una terza fase di effettiva pianificazione del percorso. In tutte le fasi i robot devono collaborare per superare i limiti introdotti da una conoscenza parziale, le comunicazioni sono limitate localmente, e l'architettura mira ad un elevata scalabilità.

L'idea sviluppata prevede che nella fase di configurazione i robot identifichino le posizioni relative dei propri robot vicini. Successivamente i robot passano alla fase di acquisizione sensoriale dell'ambiente con lo scopo di costruirsi una mappa dell'ambiente sottostante. La mappa è rappresentata da una matrice di celle; ad ogni cella è associato un valore di potenziale. Tale operazione di associazione dei valori avviene attraverso tecniche collaborative di diffusione del campo di potenziale. Infine, i robot passano alla terza fase, in cui ogni robot calcola la parte di percorso relativo alla parte di mappa nel proprio campo visivo. Inizia il robot in grado di vedere la posizione di partenza; quando l'oggetto esce dal campo visivo invia le informazioni necessarie alla prosecuzione del percorso al proprio robot vicino. Il ricevente, a sua volta, calcola la parte di percorso compresa nel suo campo visivo per poi passare il controllo ad un successivo robot vicino. Questo processo si ripete fino al raggiungimento della configurazione di destinazione. L'intero sistema è stato progettato ponendo particolare attenzione ad un limitato utilizzo delle risorse ed alle caratteristiche di scalabilità.

In questo capitolo è presentata l'architettura del pianificatore distribuito e

sono illustrati sia l'approccio generale adottato sia i dettagli tecnici di implementazione e le specifiche soluzioni ai problemi incontrati. Il capitolo è strutturato nel seguente modo: nella sezione 5.1 è descritta la prima fase di configurazione iniziale in cui gli eye-bot individuano i propri robot vicini e rilevano le loro posizioni relative. Nella sezione 5.2 è presentata la seconda fase, la quale consiste nella diffusione del campo di potenziale mantenendo una conoscenza distribuita e locale dell'ambiente. Le successive tre sezioni sono tre diversi pianificatori, la loro successione segue un percorso logico e rappresenta il processo evolutivo verso un sistema completamente distribuito, efficiente e robusto. Nella sezione 5.3 è presentato brevemente un sistema ibrido che combina la conoscenza della mappa distribuita con una conoscenza globale e perfetta in fase di pianificazione. La sezione 5.4 presenta un pianificatore distribuito con ricerca esaustiva della soluzione. Questo sistema presenta alcuni limiti e svantaggi in scenari con determinate caratteristiche; per migliorare il sistema, in modo che affronti efficientemente tutti i possibili scenari, è stato introdotto un insieme di euristiche, le quali sono presentate nella sezione 5.5.

Ogni sezione ha una prima parte di descrizione generale dell'approccio adottato, ed una seconda parte in cui vengono illustrate nel dettaglio le caratteristiche della soluzione.

5.1 Rilevamento robot vicini

Idea generale

La prima operazione effettuata dal sistema consiste nel rilevare le posizioni relative dei robot vicini. Ogni entità memorizza una *tabella dei vicini* contenente informazioni circa il posizionamento relativo dei robot nel proprio raggio di comunicazione. L'informazione che si desidera ottenere in questa fase è, in particolare, la posizione e l'orientamento del nodo vicino; tale informazione permette di localizzare il campo visivo delle entità nel mio raggio di comunicazione. Pertanto nella tabella sono indicati i seguenti dati:

- *coordinate polari relative* del nodo vicino (angolo e distanza relative alla propria posizione e orientamento);
- *orientamento relativo*: orientamento del nodo vicino rispetto al proprio orientamento (l'angolo di rotazione da applicare al proprio orientamento per raggiungere quello del vicino).

Nella Figura 5.1 è presente una rappresentazione grafica dei tre valori sopradescritti.

Dettagli tecnici

Per espletare l'operazione di rilevamento della posizione è possibile sfruttare molteplici strategie, le quali sono fortemente dipendenti dai dispositivi di cui il robot è equipaggiato. Nello scenario considerato gli eye-bot sono dotati di un sistema *Range and Bearing* descritto in sezione 2.5. Con questo sistema RB è possibile individuare l'angolo e la distanza da cui si ricevono i messaggi (coordinate polari). Successivamente il robot comunica al mittente l'angolo relativo da cui è stato ricevuto il messaggio, in questo modo il robot vicino è in grado

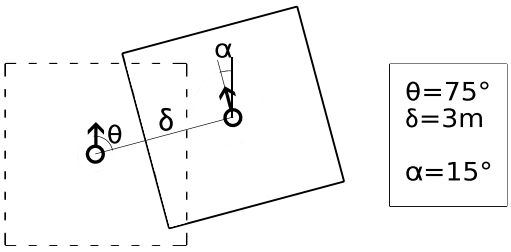


Figura 5.1: Rappresentazione grafica dei campi visivi di due eye-bot vicini e le loro posizioni relative. I valori Θ e δ corrispondono alle coordinate polari della posizione relativa, il valore α è l'orientamento relativo.

di calcolare l'orientamento relativo. Per mantenere questo progetto di tesi indipendente da una specifica tecnologia, questa fase è stata simulata senza uno studio dettagliato delle modalità di acquisizione del segnale e delle strategie di riduzione dell'errore. Bensì i robot effettuano una generica acquisizione del posizionamento relativo; tale informazione è soggetta a sua volta ad un generico errore stocastico. Il livello di errore è configurabile: è possibile indicare due distinti valori di errore sia per la distanza sia per l'angolo. Tali valori corrispondono alla deviazione standard della distribuzione Gaussiana a media nulla che modellizza il rumore dell'acquisizione.

L'errore sull'informazione delle posizioni relative si traduce in assunzioni errate dell'area visiva che due robot vicini condividono. Questo disallineamento può portare a difficoltà durante la fase di pianificazione di percorso, che consiste in una sequenza di passaggi dell'oggetto da un robot all'altro. Quando il percoso porta l'oggetto fuori dal campo visivo, il robot interrompe la pianificazione locale del cammino, e comunica al vicino che giace nella direzione d'uscita di proseguire la pianificazione dal punto finale del proprio percorso locale. Il passaggio delle coordinate dell'oggetto è basato sulla condivisione di aree visive comuni, cioè una sovrapposizione dei campi visivi. Quando il sistema è soggetto ad errore di allineamento, la percezione del campo visivo del vicino è disallineata rispetto alla posizione reale. Pertanto percezioni dell'ambiente errate possono interferire nella corretta prosecuzione del percorso e portare a fallimenti del processo di pianificazione.

La Figura 5.2 illustra graficamente l'effetto dell'errore di rilevamento delle posizioni relative tra vicini. A sinistra è visualizzata la situazione reale, con il punto rosso nella posizione corretta; mentre a destra è mostrato l'effetto dell'errore: una percezione errata della posizione condivisa.

5.2 Diffusione del potenziale nello sciame di robot

Idea generale

La seconda fase del processo consiste nella diffusione del campo di potenziale. Ogni robot ha una conoscenza parziale dell'ambiente, pertanto è necessaria una

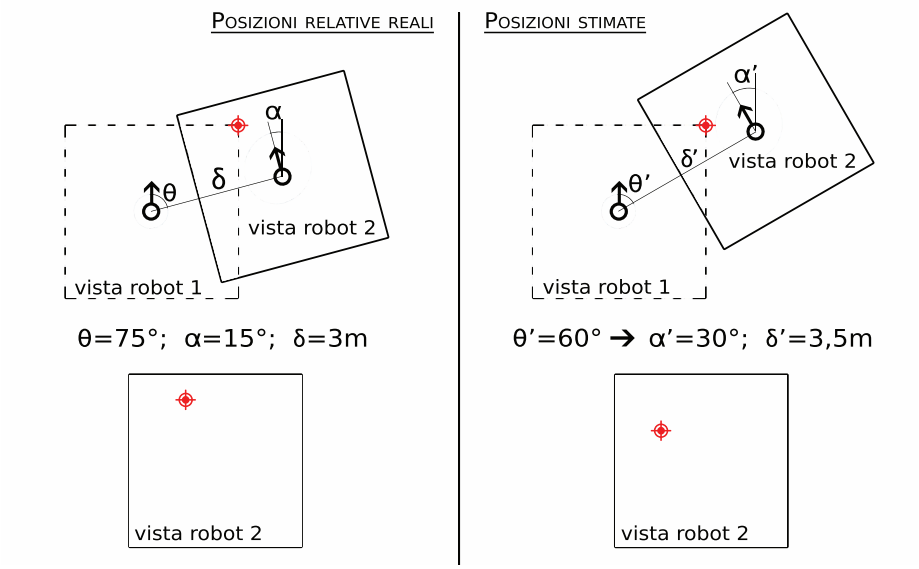


Figura 5.2: Errore causato da una misurazione di allineamento scorretta. Nel caso a destra è stato introdotto errore nelle misurazioni di θ e δ. Si può notare come questo errore produca un disallineamento dell'area condivisa del campo visivo, causando una percezione errata della posizione di un punto condiviso (punto rosso). Il valore errato di α è conseguenza di θ: il robot 2 mantiene in entrambe le immagini un angolo relativo di 90° rispetto al robot 1, ma misurazioni errate di θ influiscono negativamente oltre che sulla posizione relativa anche sull'orientamento relativo.

collaborazione fra i diversi robot per completare la diffusione del potenziale sull'intera mappa.

Si tratta di un processo con un obiettivo finale sovradimensionato rispetto ai limiti sensoriali e comunicativi imposti in questo progetto alle entità che lo espletano. Tali entità possono raggiungere l'obiettivo finale solamente grazie alla collaborazione ed al coordinamento, in questo modo si creano sinergie in grado di incrementare le capacità delle singole entità. Questo aspetto del processo è la caratteristica fondamentale di un sistema distribuito, ed in particolare di uno sciame di robot.

In questa sezione vengono evidenziate le modifiche strutturali rispetto al precedente sistema ed in particolare le operazioni di coordinamento e collaborazione fra i robot per raggiungere una corretta diffusione del campo di potenziale. Il processo di espansione del campo di potenziale si compone di due fasi: (i) creazione dello skeleton, e (ii) espansione del potenziale. Le due fasi sono illustrate qui di seguito.

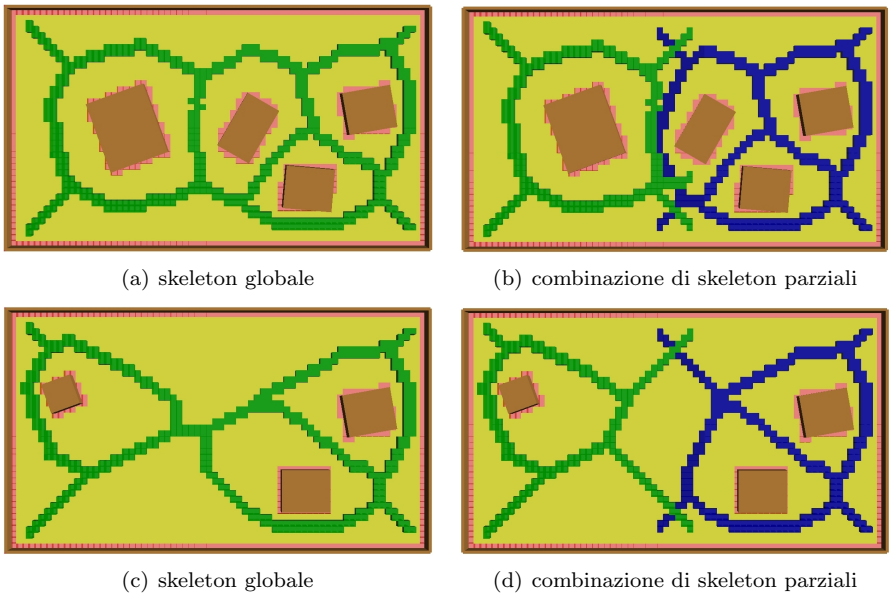
5.2.1 Skeleton parziali

La prima fase di creazione dello skeleton non differisce proceduralmente dal caso centralizzato, tuttavia il risultato è differente, poichè ogni robot crea lo skeleton limitato alla parte di mappa visualizzata. Pertanto ogni robot calcola solamente una parte dello skeleton globale. Così come la mappa totale è suddivisa in molteplici mappe parziali, ognuna acquisita localmente da un robot, allo stesso modo lo skeleton è suddiviso in una molteplicità di skeleton parziali, ognuno relativo ad una mappa parziale. La somma degli skeleton parziali difficilmente corrisponde esattamente allo skeleton risultante da un approccio globale, a causa della proprietà intrinseca dello skeleton in prossimità del confine con la mappa. In mappa rettangolare e ambiente libero da ostacoli lo skeleton tende a biforcarsi e andare verso i gli angoli della mappa; pertanto suddividendo il terreno in mappe parziali si ottengono biforcazioni nelle zone di congiunzione. È possibile vedere una rappresentazione grafica di questo effetto in Figura 5.3: le prime due immagini (a) e (b) mostrano due risultati molto simili fra loro, mentre le due immagini successive, (c) e (d), mostrano un risultato molto differente fra approccio globale e distribuito.

5.2.2 Diffusione del potenziale

Completata la creazione dello skeleton, il processo prosegue con l'esecuzione della seconda fase: la diffusione del campo di potenziale. Come nel caso centralizzato l'espansione del potenziale ha inizio dal punto di destinazione, il quale è l'unico punto che possiede per definizione un valore noto: attrazione nulla. I robot che hanno nel loro campo visivo questo punto possono iniziare il processo; essi diffondono completamente il campo di potenziale nella loro mappa parziale con lo stesso metodo del caso centralizzato: prima sullo skeleton e successivamente sui restanti punti (vedi sezione 4.2).

Quando un robot completa la diffusione sulla mappa locale, comunica ai propri robot vicini i valori di potenziale delle celle che giacciono sulla parte di mappa in condivisione (*frontiera condivisa*). La comunicazione è minimizzata, e vengono trasmessi solamente i valori di un limitato numero di celle. Le celle



(a) skeleton globale (b) combinazione di skeleton parziali

(c) skeleton globale (d) combinazione di skeleton parziali

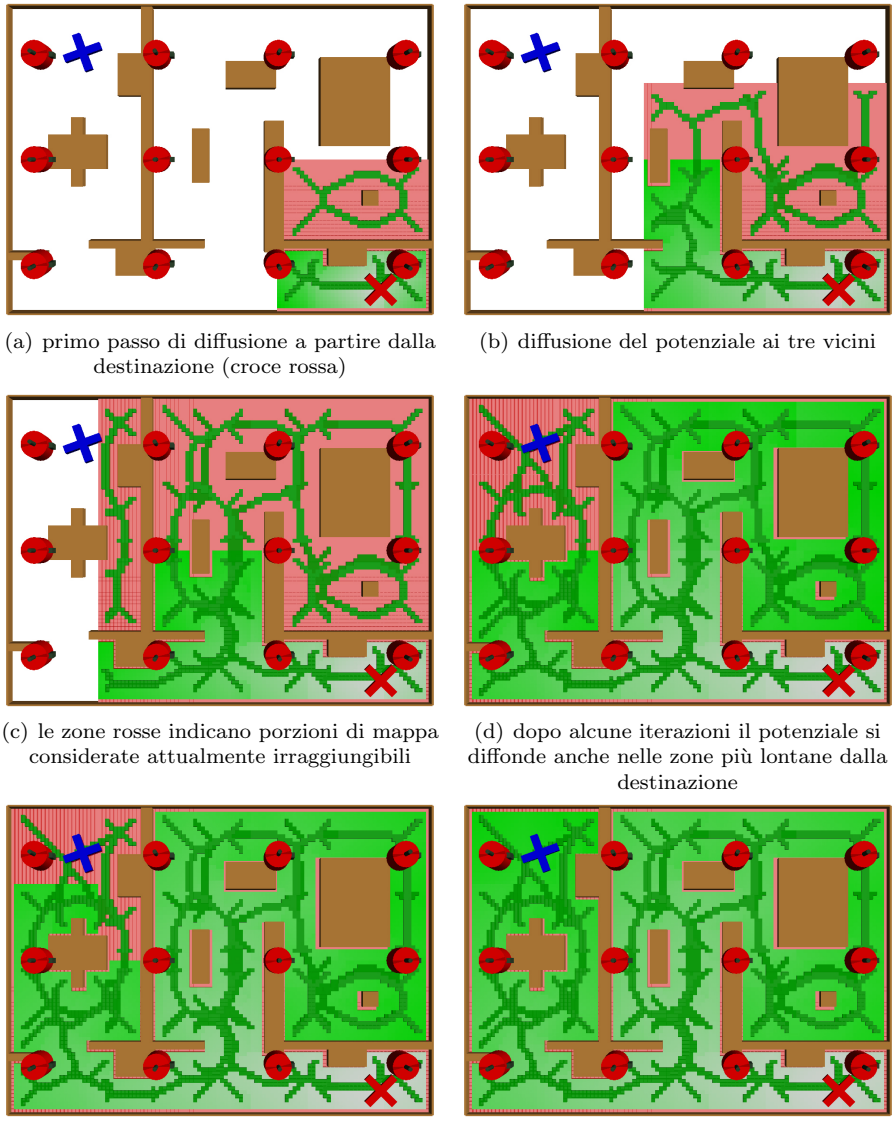
Figura 5.3: Confronto fra skeleton su mappa globale e composizione di skeleton parziali.

selezionate corrispondono alle celle di skeleton che giacciono sulla frontiera del campo visivo del ricevente.

Quando un robot riceve un messaggio con nuovi valori, aggiorna il campo di potenziale sulla propria mappa. Le celle ricevute per definizione giacciono sul bordo della mappa e corrispondo allo skeleton del mittente sulla frontiera condivisa; come prima operazione il robot connette queste celle con il proprio skeleton. Poi, a partire dai valori ricevuti, espande il potenziale sull'intera mappa, come sempre prima sullo skeleton e successivamente sulle restanti celle libere. Il valore di una cella viene aggiornato solamente se il nuovo valore è inferiore al campo di potenziale della medesima cella al passo precedente. Con questa regola l'aggiornamento del campo di potenziale assume un significato specifico: è stato trovato un cammino più breve (potenziale inferiore), pertanto è corretto mantenere sempre informazione circa il percorso più corto. Al termine dell'aggiornamento della propria mappa, se sono stati modificati alcuni valori, il robot comunica i nuovi valori di frontiera ai propri robot vicini. Il nuovo messaggio, andrà ad attivare a sua volta successivi aggiornamenti dei vicini.

In Figura 5.4 è mostrata una rappresentazione grafica dei passaggi del processo di diffusione del campo di potenziale fra i robot del sistema.

Questo processo converge verso una situazione stabile, in cui non sono innescati ulteriori aggiornamenti fra robot vicini; quando viene raggiunto questo stato di stabilità il processo di diffusione del potenziale termina. In pratica, ogni robot non può sapere se il processo di diffusione è realmente completato, o ancora in fase di aggiornamento, pertanto il robot ipotizza che non ci saranno ulteriori cambiamenti nel proprio campo di potenziale dopo un determinato numero di iterazioni senza nuovi messaggi di aggiornamento, in particolare, negli esperimenti, sono stati usati 5 cicli di attesa. La proprietà di convergenza



(a) primo passo di diffusione a partire dalla destinazione (croce rossa)

(b) diffusione del potenziale ai tre vicini

(c) le zone rosse indicano porzioni di mappa considerate attualmente irraggiungibili

(d) dopo alcune iterazioni il potenziale si diffonde anche nelle zone più lontane dalla destinazione

(e) il potenziale si diffonde nella mappa

(f) il campo di potenziale è completamente diffuso sull'intera mappa in sei passi

Figura 5.4: Diffusione del potenziale tra i robot dello sciame. Valori più chiari rappresentano un campo di potenziale più basso, mentre le zone più distanti dalla destinazione hanno un potenziale maggiore e sono contraddistinte da un verde più inteso. I grafi verde scuro sono gli skeleton parziali che si sovrappongono. La croce è l'oggetto mobile nella posizione di partenza (blu) e quella di arrivo (rosso).

verso una situazione di stabilità può essere facilmente dedotta dalle due regole utilizzate per l'aggiornamento:

- durante la diffusione del potenziale nella mappa locale i valori incrementano costantemente (in particolare un incremento unitario ogni passo)
- l'effettivo aggiornamento nella mappa locale avviene solamente quando il nuovo valore di potenziale è inferiore al valore nella medesima cella nell'aggiornamento precedente. Significa che gli aggiornamenti nel campo di potenziale sono eseguiti mantenendo traccia dello stato del campo sulla mappa prima della ricezione del messaggio. Sono memorizzati solo decrementi nel campo di potenziale, poichè si intende utilizzare solamente l'informazione relativa ai cammini più brevi.

Questa procedura evita che si creino cicli e permette una corretta diffusione del potenziale sull'intera area composta da mappe parziali interconnesse. Essendo i robot in numero finito, il processo può non terminare solamente a causa di cicli. Un ciclo è generato da un aggiornamento di un nodo (di partenza) che attiva una sequenza di aggiornamenti, i quali portano ad attivare un ulteriore aggiornamento del nodo di partenza. Questa situazione non può presentarsi poichè ad ogni passaggio i valori possono solamente aumentare (incremento unitario ad ogni passo nella mappa) ed un aggiornamento è attivato solamente se i nuovi valori sono inferiori di quelli precedenti (percorso più breve). Pertanto è impossibile che un aggiornamento attivato da un nodo gli possa ritornare con un valore inferiore a quello inviato.

5.3 Ricerca globale del percorso con conoscenza locale della mappa

Questa prima evoluzione del sistema opera con una struttura sensoriale distribuita, ma mantiene un approccio globale nella ricerca; si tratta di un sistema ibrido che combina una conoscenza parziale nella prima fase di acquisizione con una conoscenza globale in fase di espansione dell'albero di ricerca.

Qui di seguito è presentata brevemente la pianificazione di percorso con questo tipo di architettura ibrida. La sezione si conclude poi con una breve valutazione dell'architettura presentata.

Idea generale

La ricerca nello spazio delle soluzioni avviene sulle mappe locali utilizzando però conoscenza globale. In questa architettura ibrida, ogni robot pianifica la parte di percorso relativa all'area nel suo campo visivo, quando la traiettoria esce dalla propria mappa locale, viene passato il controllo al robot vicino posizionato sul lato di uscita. Il percorso viene generato utilizzando l'algoritmo di ricerca A-star (vedi sezione 4.3), il quale seleziona ad ogni passo la configurazione con il valore inferiore della stima di costo del percorso totale. I robot hanno conoscenza globale dei fronti d'espansione di tutti i robot coinvolti nel calcolo del percorso. Essi mantengono una tabella con l'elenco dei robot attivi nel calcolo ed il relativo valore della configurazione con costo minore. Ad ogni passo viene esaminata la tabella ed il controllo viene dato al robot con la configurazione

con il costo inferiore, il quale effettua un passo, aggiorna i propri valori, ed infine aggiorna la tabella. Se il robot corrente, dopo l'aggiornamento, rimane con la configurazione a costo minore allora mantiene il controllo del processo; altrimenti cede il controllo al successore. I robot comunicano solo localmente e, durante il passaggio di controllo, inviano messaggi in broadcast con la nuova tabella aggiornata a tutti i vicini, i quali ripetono a loro volta l'operazione di invio del messaggio (*flooding*). Ogni robot analizza la nuova tabella ricevuta, la quale contiene l'informazione del robot designato a proseguire la ricerca del percorso.

Il processo termina quando un robot esplora la configurazione corrispondente alla configurazione obiettivo (la destinazione del percorso) oppure quando tutti i robot hanno visitato tutte le configurazioni possibili. Nel primo caso il processo termina con successo e viene quindi ricostruito il percorso finale risalendo le *configurazioni genitore*. Nel secondo caso, invece, è stata effettuata una ricerca esaustiva senza aver raggiunto la destinazione, quindi il processo termina con fallimento: non esiste alcun un percorso libero da ostacoli dalla partenza alla destinazione.

Discussione finale

Il pianificatore presentato studia il comportamento di un sistema di pianificazione con conoscenza dell'ambiente frammentata fra le diverse entità, e con processo di ricerca della soluzione con conoscenza perfetta e globale. È evidente fin dalla fase di pianificazione che questa soluzione ibrida non soddisfa gli obiettivi di scalabilità prefissati, presentando sprechi di risorse sia comunicative sia computazionali. Infatti, questo pianificatore non mira ad essere una soluzione finale al problema, piuttosto rappresenta un passaggio di studio verso una soluzione totalmente distribuita. Il sistema proposto è in grado di individuare il percorso ottimo (in termini di costo, quindi di lunghezza del percorso), a partire dalla conoscenza frammentata di partenza. Tuttavia il risultato non può corrispondere a quello ottenuto da un approccio globale, in conseguenza alle differenze nella struttura dello skeleton descritte precedentemente nella sottosezione 5.2.1.

5.4 Path Planning totalmente distribuito

Idea generale

Questo secondo pianificatore è totalmente distribuito: esso mantiene durante l'intero processo conoscenza parziale sia della mappa sia dello stato di pianificazione. Ogni robot possiede conoscenza limitata alla parte di percorso nel suo campo visivo, nessuna entità possiede informazioni sul percorso totale, la conoscenza è distribuita fra i robot dello sciame. Si tratta di una ricerca esaustiva e completa della soluzione, ciò significa che se il processo fallisce sono state esaminate tutte le possibili combinazioni. In questo caso il sistema restituisce con sicurezza il risultato di assenza di soluzioni a partire dai dati di partenza. Tuttavia è possibile che con differenti dati di partenza, come una riduzione del passo di discretizzazione, oppure dell'angolo di rotazione dell'oggetto, sia presente un percorso valido.

Il processo è simile a quello del pianificatore precedente, ma con conoscenza più limitata. L'idea generale è la seguente: ogni eye-bot calcola la parte di per-

corso nel suo campo visivo; quando l'oggetto esce dai limiti della mappa locale, comunica la posizione dell'oggetto al robot vicino con cui condivide quell'area di mappa, il quale prosegue la pianificazione del percorso. L'esecuzione si ripete fino all'esplorazione della configurazione finale (destinazione raggiunta: successo) oppure fino all'esaurimento delle possibili configurazioni da analizzare (nessun percorso valido: fallimento).

Un robot può assumere due possibili stati: (i) esecuzione della pianificazione oppure (ii) attesa di messaggi dai robot vicini. Il primo stato riguarda l'attività di effettiva esplorazione delle configurazioni e discesa del potenziale con l'algoritmo A-star. Mentre il secondo è uno stato di attesa, in cui il robot è inattivo ed attende specifici messaggi relativi all'operazione da eseguire. Il messaggio può richiedere di proseguire la pianificazione precedente, modificarla oppure iniziare un nuovo percorso parziale.

Dettagli tecnici

Con questa architettura, a differenza della precedente, quando viene passato l'oggetto ad un robot, e questo assume il controllo del processo, lo mantiene per tutta la fase di pianificazione locale. L'eye-bot attivo non ha alcuna informazione circa lo stato di pianificazione degli altri robot; pertanto esso pianifica l'intera parte di percorso nella sua area, fino al punto di uscita dal proprio campo visivo. La pianificazione locale avviene utilizzando l'algoritmo A-star (vedi sezione 4.3) il quale mantiene traccia dei nodi visitati memorizzandoli in un albero di configurazioni (*closed-set*). Questa informazione è mantenuta per l'intero processo, anche dopo aver completato la parte di pianificazione locale di percorso ed aver ceduto il controllo ad un robot vicino. Tale informazione è utile per evitare che il robot continui ad analizzare la medesima parte di percorso. Con questa strategia, se un robot riceve l'oggetto in una posizione già visitata, rifiuta di continuare il processo; in questo modo il mittente è forzato a trovare un percorso alternativo, poichè quello proposto è già stato analizzato.

La pianificazione locale da parte di un robot può terminare in tre differenti situazioni:

- il percorso *esce dal proprio campo visivo*. Quindi, il robot individua il vicino con cui condivide quella parte di mappa, e gli invia le informazioni relative all'ultima configurazione del percorso. Con questi dati il robot vicino potrà proseguire la pianificazione.

- viene esplorata la *configurazione finale*. Il processo di pianificazione termina, e viene inviato un messaggio a tutti i vicini del completamento con successo della pianificazione di percorso (flooding dell'informazione).

- *fallimento locale*. Il robot ha visitato tutte le configurazioni della sua area, senza raggiungere nessuna delle due situazioni sopraelencate. Questa situazione si traduce in un fallimento locale: non c'è alcun percorso percorribile nel campo visivo parziale; il robot risponde al vicino che lo ha attivato comunicando un fallimento locale. Se non c'è alcun robot precedente, ma si tratta del robot di partenza che ha iniziato il processo, allora il fallimento locale si traduce in un *fallimento globale*. Il robot informa tutti i robot del sistema attraverso un protocollo di *flooding* del risultato finale: l'intero processo di pianificazione è terminato con fallimento.

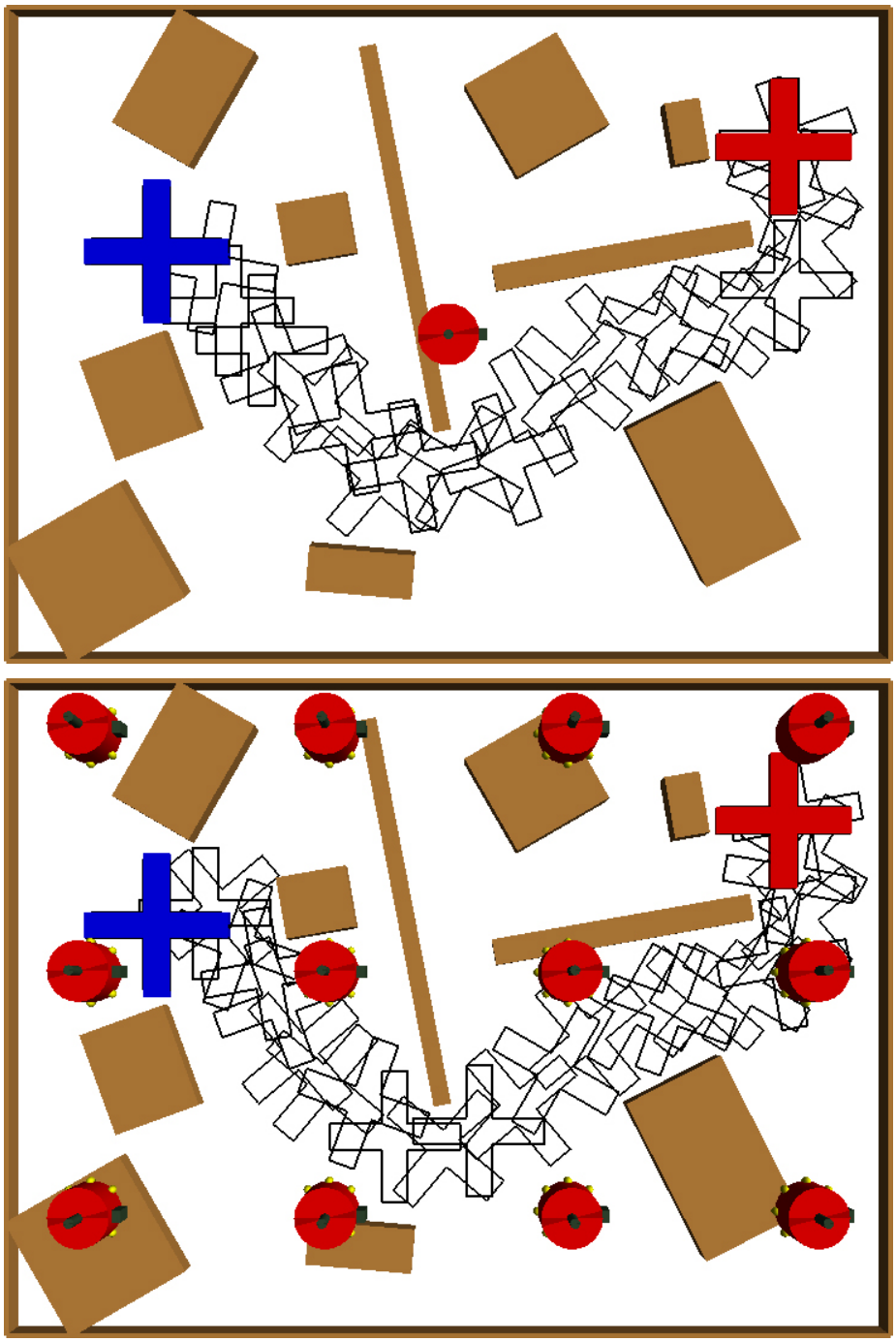


Figura 5.5: Un esempio di percorso finale individuato attraverso il pianificatore distribuito (in basso) a confronto con la soluzione individuata dal pianificatore centralizzato con conoscenza perfetta e globale (in alto).

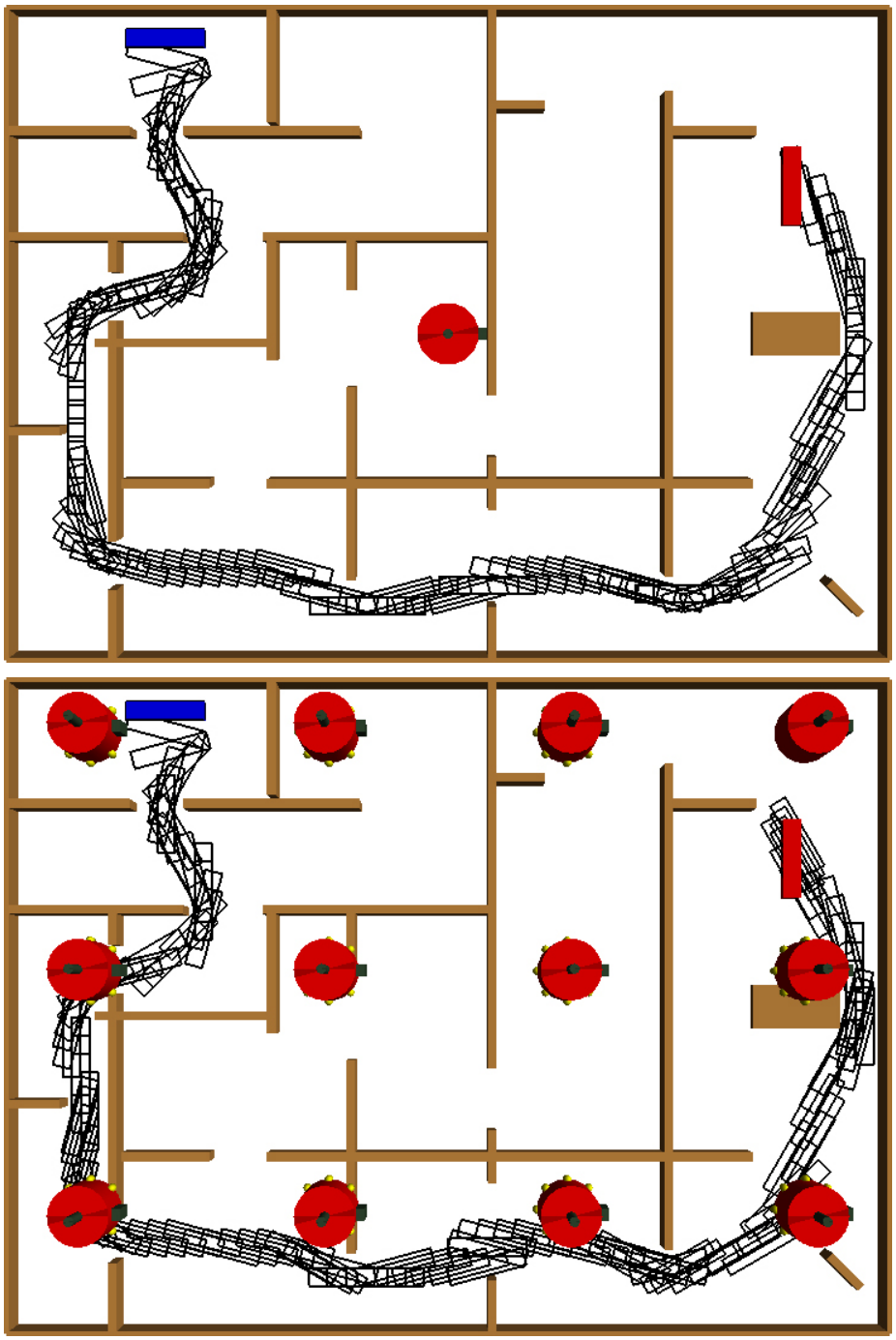


Figura 5.6: Un esempio di percorso finale individuato attraverso il pianificatore distribuito (in basso) a confronto con la soluzione individuata dal pianificatore centralizzato con conoscenza perfetta e globale (in alto).

Quando un robot non sta pianificando il percorso, si trova in uno stato di attesa di messaggi. Nella tabella 5.1 sono elencati i possibili messaggi che un robot può ricevere durante la fase di pianificazione quando si trova in stato di attesa.

Tabella 5.1: Comunicazione fra i robot. Elenco dei possibili messaggi che un robot può ricevere nella fase di pianificazione.

Messaggio	**Significato**	**Azione**
START PATH	Passaggio del processo di pianificazione tra robot vicini. Il messaggio è accompagnato dalle informazioni relative alla configurazione finale dell'oggetto mobile nel percorso locale del mittente, la quale corrisponde alla configurazione iniziale del ricevente.	Il robot effettua un controllo di validità della configurazione ricevuta. Se valida, inizia la pianificazione del percorso locale; altrimenti risponde con un messaggio *Not Acceptable*.
FAILURE	Il mittente ha conseguito un fallimento nella pianificazione del percorso. Questo messaggio può essere ricevuto solamente dal robot a cui è stato precedentemente inviato un messaggio di *Start Path*.	Se è presente un altro robot vicino, nella stessa direzione, allora il robot gli invia un messaggio per la prosecuzione del percorso. Altrimenti, se non sono presenti ulteriori eye-bot vicini, il robot prosegue la pianificazione nella propria mappa dal punto in cui era arrivato prima di sospendere il processo per il passaggio di consegna, il quale è appena terminato con un fallimento locale.
GOAL FOUND	Un robot ha raggiunto la destinazione. Il processo è terminato con successo.	Il robot ricostruisce la parte di percorso effettuata, ripercorrendo a ritroso la sequenza di configurazioni legate dal legame di parentela *configurazione genitore*. Il percorso parziale viene memorizzato. Inoltro a tutti i vicini (eccetto il mittente) il messaggio *Goal Found*.
NA (Not Acceptable)	La configurazione inviata ad un robot vicino per la prosecuzione del percorso non è accettabile. Questo messaggio può essere ricevuto solamente dal robot a cui è stato precedentemente inviato un messaggio di *Start Path*. Può essere ricevuto per due motivi: (i) la configurazione inviata giace su un ostacolo nella mappa del vicino; oppure (ii) la configurazione inviata è già stata esplorata in precedenza dal robot vicino.	Il comportamento è il medesimo di quello conseguente ad un messaggio di *Failure*.

Tabella 5.1: Comunicazione fra i robot. Elenco dei possibili messaggi che un robot può ricevere nella fase di pianificazione.

Messaggio	**Significato**	**Azione**
GLOBAL FAILURE	Il processo termina con un fallimento. Non ci sono soluzioni valide per l'attuale scenario.	Il robot pulisce la memoria e inoltra a tutti i vicini (eccetto il mittente) il messaggio *Global Failure*.

Qui di seguito è presentata più esaustivamente l'azione attivata nel robot che riceve un messaggio di fallimento locale. Questo messaggio può essere ricevuto solamente dal robot vicino a cui è stato precedentemente inviato un messaggio di *Start Path*, contenente le coordinate dell'oggetto nell'ultima configurazione del percorso parziale calcolato. Se il vicino risponde con un fallimento locale nella pianificazione del percorso, l'eye-bot a cui viene restituito il controllo effettua un ulteriore tentativo di prosecuzione, nella stessa direzione, ma passando il controllo ad un differente robot. Per fare ciò, viene controllato se è presente un altro robot vicino nella stessa direzione, con il quale si condivide una parte di mappa che contenga le coordinate dell'ultima configurazione dell'oggetto del percorso locale; se presente allora il robot invia a tale robot un messaggio per la prosecuzione del percorso. Se invece, non sono presenti ulteriori eye-bot vicini in quella direzione, il robot che riceve un messaggio di fallimento locale prosegue la pianificazione nella propria mappa dal punto in cui era arrivato prima di sospendere il processo per il passaggio di consegna.

Quando un robot completa una parte di percorso e cede il controllo ad un vicino, memorizza il percorso calcolato e rimane in attesa di messaggi. È possibile che riceva messaggi di fallimento o rifiuto, oppure nuovi messaggi di prosecuzione del percorso. Nel primo caso il robot rimuove il percorso memorizzato, e prosegue la ricerca dal punto in cui era arrivato prima di sospenderla. Per consentire quest'ultima operazione, è necessario che il robot memorizzi le due liste di configurazioni: l'*open-set* ed il *closed-set*. Invece, nel caso in cui il robot riceva un nuovo messaggio di prosecuzione, è necessario che pianifichi e memorizzi un nuovo percorso, ma allo stesso tempo, non dimentichi quanto pianificato in precedenza. Se successivamente riceverà un messaggio di fallimento, dovrà modificare solamente la seconda pianificazione, e mantenere in memoria la prima. Questo ben noto approccio è chiamato *LIFO - Last In First Out* - ed è stato modellizzato nel pianificatore attraverso una pila (*stack*) con operazioni di *push* e *pop* per inserire e rimuovere percorsi parziali. Ogni percorso parziale è descritto dai seguenti attributi: la sequenza ordinata di configurazioni (l'effettivo percorso), il robot vicino predecessore (colui al quale inviare un eventuale messaggio di fallimento locale del percorso), il robot vicino successore (il quale ha ricevuto il nostro messaggio di prosecuzione), e le due liste di configurazioni (open-set e closed-set) nello stato al momento della sospensione (utili nel caso sia necessario proseguire la pianificazione in conseguenza ad un messaggio di fallimento locale). Una rappresentazione grafica che può aiutare la comprensione del concetto è illustrata nella Figura 5.7.

Discussione finale

L'architettura di pianificazione presentata, basata sull'algoritmo di discesa del potenziale e A-star, presenta un'esecuzione piuttosto rapida e lineare verso la

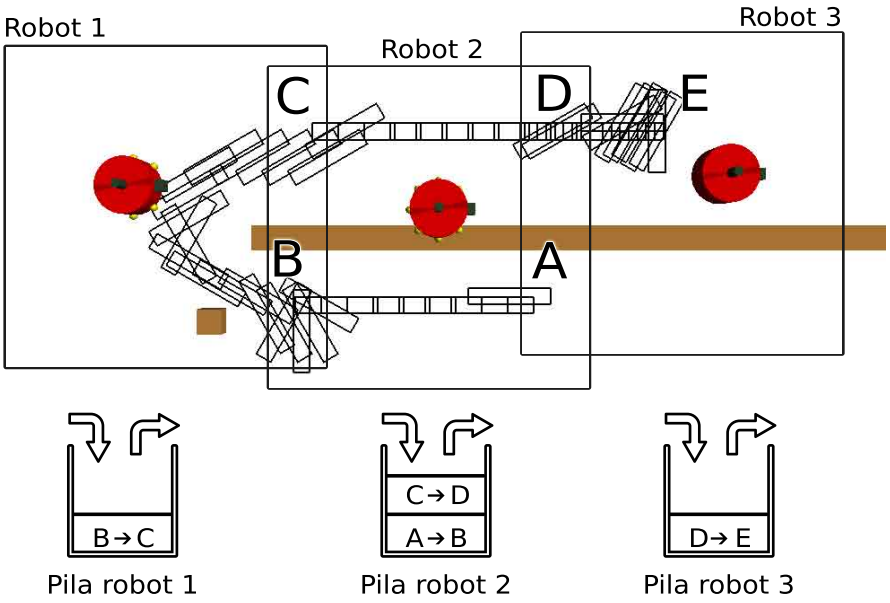


Figura 5.7: Memorizzazione dei percorsi parziali in strutture dati a pila (*stack*). In questa immagine sono mostrate le tre pile *LIFO* dei tre robot: il robot 2 inzia la pianificazione, e memorizza la parte di percorso dal punto A al punto B; poi il robot 1 prosegue la pianificazione e memorizza le informazioni relative al percorso dal punto B al punto C; il robot 2 aggiunge un secondo percorso parziale dal punto C al punto D in cima alla pila; infine il robot 3 memorizza l'ultima parte di percorso da D a E.

soluzione quando lo scenario non presenta minimi locali nel campo di potenziale. Per rapida e lineare si intende un'esecuzione che espande l'albero di ricerca principalmente verso la direzione della soluzione, visitando un limitato numero di nodi non appartenenti al percorso finale.

Quando il sistema opera in scenari complessi o senza soluzione (una definizione di "scenari complessi" è data in seguito nella sottosezione 5.4.1) viene eseguita una ricerca esaustiva della soluzione. Ogni robot esamina tutte le configurazioni possibili all'interno della mappa del proprio campo visivo e di conseguenza invia tutte le diverse configurazioni di confine ai propri vicini, attivandoli molteplici volte. La ricerca attiva tutti gli eye-bot con mappe interconnesse, e ogni robot esplora totalmente il suo spazio di ricerca. In questo modo il sistema esegue una ricerca esaustiva della soluzione: se una soluzione è presente viene individuata, altrimenti il sistema restituisce con sicurezza un risultato di fallimento del processo, e quindi l'assenza di una soluzione. Come indicato nel paragrafo *Idea generale* di questa sezione, è possibile che il sistema fallisca con determinati dati di partenza (livello di discretizzazione della mappa, e angolo di rotazione unitaria dell'oggetto), ma applicando discretizzazioni più accurate sia possibile individuare un percorso valido. Il pianificatore che termina l'esecuzione con un fallimento, indica l'assenza di percorsi liberi da collisioni dalla partenza all'arrivo a partire dalla conoscenza ricavata nelle fasi precedenti. Pertanto un fallimento non prova che non esista una soluzione per il dato scenario, ma piuttosto che non è possibile trovare una soluzione con i dati di partenza utilizzati.

5.4.1 Minimo locale

Alcuni scenari presentano percorsi semplici, nei quali il pianificatore segue l'attrazione verso la posizione finale, individuando senza problemi il percorso dell'oggetto dalla partenza alla destinazione (due esempi di percorsi finali sono mostrati in Figura 5.5 e 5.6). Altri scenari, invece, possiedono ambienti più complessi, in cui il percorso più breve presenta passaggi stretti fra ostacoli che non permettono l'effettivo passaggio dell'oggetto (Figura 5.8). Il campo di potenziale è in grado di diffondersi anche attraverso queste zone di restringimento; in questo modo nella mappa si genera un'attrazione verso questi passaggi, i quali pur essendo i percorsi più brevi non sono percorsi realmente percorribili dall'oggetto mobile. Quando si crea questa situazione, il pianificatore, durante la discesa del potenziale, espande l'albero di ricerca verso il passaggio stretto, senza poi riuscire a proseguire il percorso in tale direzione. A questo punto, il pianificatore effettua una ricerca esaustiva nell'area di attrazione, analizzando singolarmente ogni possibile configurazione. Il processo continua l'espansione dell'albero, con priorità alla zona con valori di potenziale minore (quindi nelle vicinanze del passaggio stretto); solo quando tutte le configurazioni di quell'area sono state analizzate, allora l'albero di ricerca prosegue l'espansione con configurazioni anche in altre direzioni, provando in questo modo percorsi alternativi. Si tratta del problema del *minimo locale*, un problema molto noto in letteratura [10, 4] e spesso ricorrente nel path planning con discesa di potenziale. Quando si manifesta questo genere di problema, il sistema soffre di elevati sprechi di risorse, sia computazionali sia comunicative. Viene analizzato un numero di configurazioni molto superiore rispetto a quelle del percorso finale (sprechi di risorse computazionali), e se il minimo locale è in prossimità di una frontiera fra robot vicini, i due robot si scambiano un elevato numero di mes-

saggi nell'ordine del numero di celle sulla frontiera condivisa (sprechi di risorse di comunicazione).

Un ulteriore problema che può manifestarsi in scenari con minimi locali è la generazione di percorsi con cicli (*loops*). Per percorsi con cicli si intende percorsi che si incrociano con se stessi: la traiettoria passa per gli stessi punti, o punti molto vicini, ad una parte del percorso precedente. Un esempio di un percorso con ciclo è mostrato in Figura 5.8(a). Questo effetto indesiderato è causato dall'assenza di conoscenza della parte di percorso pianificato dagli altri robot; nessun robot può sapere se la corrente posizione è già stata inclusa nel percorso di un altro robot vicino con il quale si condivide parte del campo visivo.

Per migliorare le prestazioni del pianificatore nei casi di scenari con minimi locali è stato introdotto un insieme di euristiche, le quali sono presentate e descritte dettagliatamente nella prossima sezione.

5.5 Euristiche

Il sistema presentato nella sezione precedente, in scenari con minimi locali, soffre di una serie di problemi che possono tradursi in un utilizzo eccessivo di risorse, e in risultati con scarsa qualità. Al fine di fronteggiare tali aspetti critici è stato progettato un insieme di euristiche in grado di risolvere alcuni dei limiti del sistema precedente. L'introduzione delle euristiche nel sistema ha come effetto un miglioramento delle prestazioni del pianificatore, anche in casi affetti da minimi locali; tuttavia una delle conseguenze dell'utilizzo di questi metodi euristici è la riduzione del tasso di successo. Si tratta di una conseguenza intrinseca dell'approccio euristico ai problemi, il quale migliora gli aspetti per cui l'euristica è stata progettata, ma allo stesso tempo riduce lo spazio delle soluzioni poichè si basa su assunzioni che semplificano il processo. Quindi questo approccio può portare il sistema al fallimento anche quando è presente un'effettiva soluzione.

Sono state introdotte quattro diverse euristiche, ognuna indipendente dalle altre e con differenti obiettivi. Qui di seguito sono elencate le quattro euristiche accompagnate da una breve descrizione; i dettagli tecnici sono illustrati nelle sottosezioni successive.

- *Blocked Cell List* - con questa tecnica il sistema tiene traccia delle aree in cui è stato passato l'oggetto ai vicini che hanno poi risposto con un fallimento. In questo modo il robot può accrescere la propria conoscenza a valle dell'esperienza passata, ed utilizzarla per evitare ulteriori tentativi di passaggio dell'oggetto in aree molto vicine a tentativi fallimentari precedenti. Questa tecnica riduce notevolmente le comunicazioni tra robot in presenza di attrazioni verso minimi locali.

- *Smart Loop Avoidance* - permette di individuare e rimuovere i cicli (*loops*) nel percorso. Durante il processo di pianificazione vengono effettuati alcuni controlli che permettono di individuare il generarsi di cicli nella traiettoria. È stata progettata una procedura di collaborazione tra robot che permette, quando viene individuato un ciclo, di riportare il sistema allo stato precedente l'inizio del ciclo, e quindi di modificare il campo di potenziale in modo da evitare che si ripeta la medesima situazione.

- *Skeleton Pruning* - questa tecnica è utlizzata nella fase di espansione del potenziale. Consiste nel "potare" lo skeleton, evitando che esso passi per

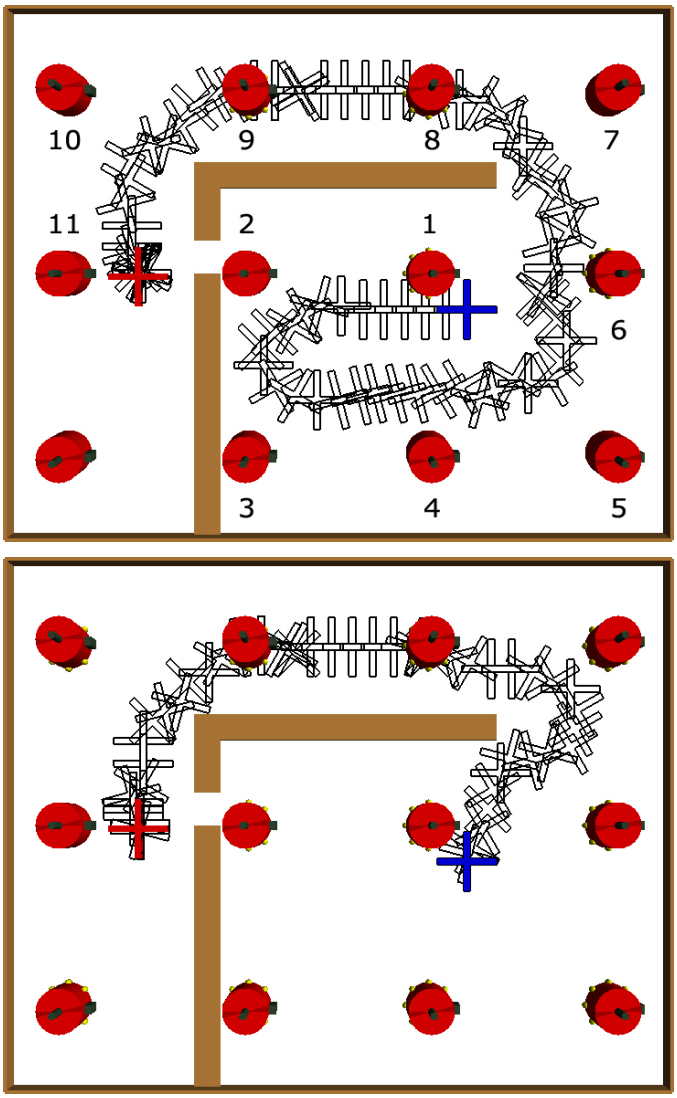


Figura 5.8: Due esempi di percorso in ambienti con minimo locale. Il primo percorso è calcolato dal pianificatore senza euristiche, mentre il secondo dal pianificatore con l'euristica di *Smart Loop Avoidance*, la quale permette di evitare i cicli. Nella prima immagine, è presente un ciclo nel percorso, causato dal minimo locale. Il robot 1 inizia la pianificazione in direzione del robot 2, al quale passa il controllo. Poichè il corridoio verso la destinazione è troppo stretto, il robot 2 prova esaustivamente tutte le configurazioni nel suo campo visivo. Il robot 1 rifiuta nuovi percorsi in ingresso in quella zona, poichè sono direttamente collegati con la prima parte di percorso calcolata. Così l'oggetto passa ai robot 3, 4, 5 e 6 in successione, i quali provano esaustivamente tutte le configurazioni, poichè attratti verso la direzione errata. Solamente il robot 7 inizia a muovere l'oggetto velocemente verso la destinazione, in conseguenza al cammino alternativo imboccato, il quale presenta una differente direzione di attrazione. Il percorso è lineare per i robot 8, 9, 10 e 11.

zone troppo strette, in particolare più strette della dimensione minima dell'oggetto mobile. C'è anche la possibilità di impostare manualmente il parametro di dimensione minima accettabile.

- *Narrow Passage Detection* - rileva automaticamente, in fase di pianificazione, la presenza di passaggi troppo stretti per l'oggetto mobile. Le strade impercorribili vengono bloccate e viene ripetuto l'intero processo a partire dalla diffusione di potenziale.

5.5.1 Blocked Cell List

La base di conoscenza del robot viene incrementata in conseguenza ad eventi precedenti. Ogni robot apprende dalle esperienze passate; in particolare memorizza l'informazione delle zone in cui i robot vicini hanno risposto con un fallimento locale. Il robot aggiorna la propria conoscenza a seguito di ogni messaggio *Failure* o *Not Acceptable*; nel dettaglio è memorizzata la sequenza di celle occupate dall'oggetto all'uscita dal campo visivo (configurazione inviata al vicino). Questo insieme di celle viene aggiunto alla *lista nera* di celle bloccate verso quel preciso robot vicino. In futuro, durante la pianificazione, il robot non effettuerà ulteriori tentativi per passare l'oggetto al medesimo vicino se posizionato su una o più delle celle presenti nella *lista nera*. In questo modo, ogni robot mantiene aggiornato un insieme di liste di celle bloccate, una per ogni suo robot vicino.

Attraverso l'utilizzo di questa tecnica viene notevolemente ridotto il numero di comunicazioni, che nell'architettura precedente esplodeva in caso di minimo locale; il robot non effettua più la sequenza esaustiva di tentativi per tutte le possibili configurazioni sulla frontiera. Una volta bloccata una zona, non saranno effettuate ulteriori prove con differenti orientamenti e leggere traslazioni dell'oggetto. L'effetto positivo è una riduzione di comunicazione durante il processo, ma allo stesso tempo è possibile che vengano rimossi gli unici tentativi capaci di portare alla soluzione. In questo caso si avrebbe il fallimento del sistema pur esistendo una soluzione.

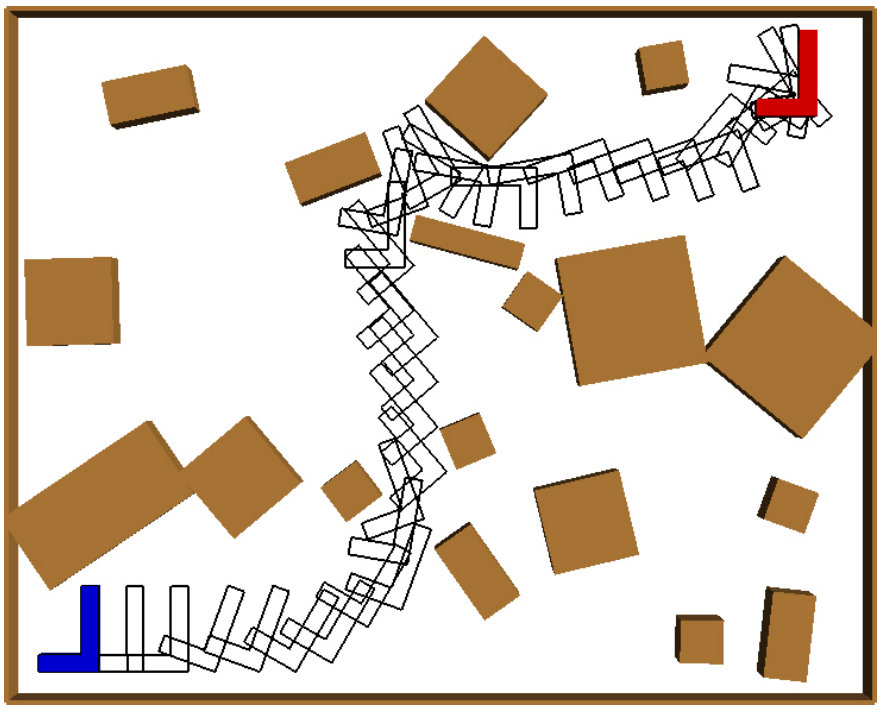
In conclusione: si ottiene un considerevole decremento di comunicazione locale tra robot confinanti, a discapito di una riduzione dello spazio delle soluzioni.

5.5.2 Smart Loop Avoidance

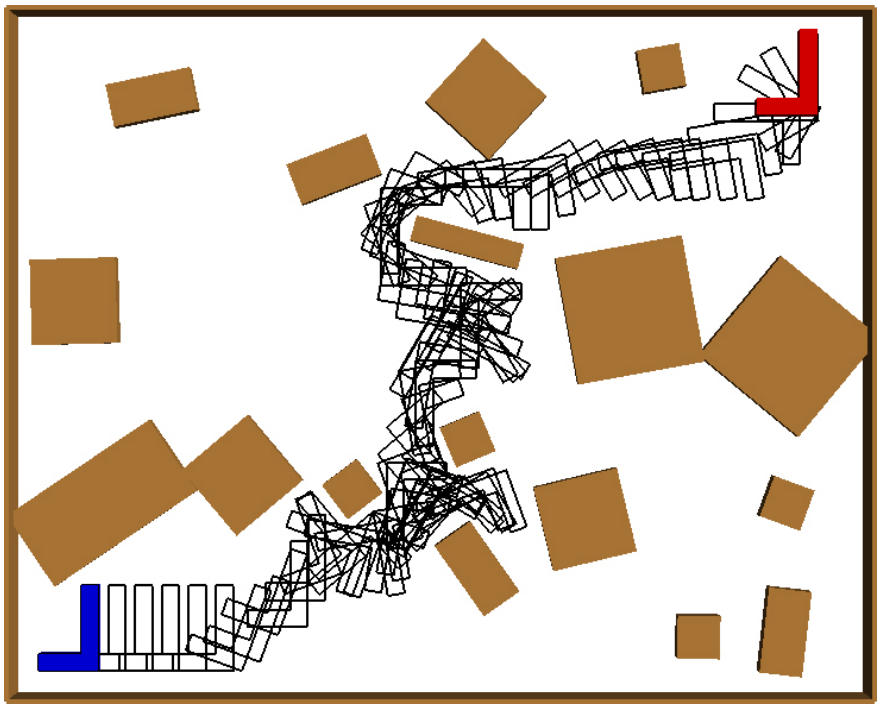
Questa procedura ha lo scopo di migliorare la qualità delle soluzioni trovate. Il principale problema qualitativo evidenziato nel precedente sistema è il crearsi di cicli nel percorso, i quali si tradurranno in un elevato spreco di risorse in una successiva fase di attuazione di quanto pianificato. Per percorso con ciclo, o loop, si intende un percorso che si incrocia con se stesso: la traiettoria forma un anello, tornando a percorrere parti del percorso già attraversate.

Questa tecnica di *Smart Loop Avoidance* si compone di due principali operazioni: (i) rilevazione dei cicli durante la fase di pianificazione nel momento in cui si vengono a formare ed (ii) eliminazione del ciclo. Il sistema viene riportato nello stato precedente all'inizio del loop, e quindi il campo di potenziale viene modificato in modo da evitare che si ripeta il medesimo ciclo.

Per individuare il formarsi di cicli sono state introdotte due regole nel processo di pianificazione:

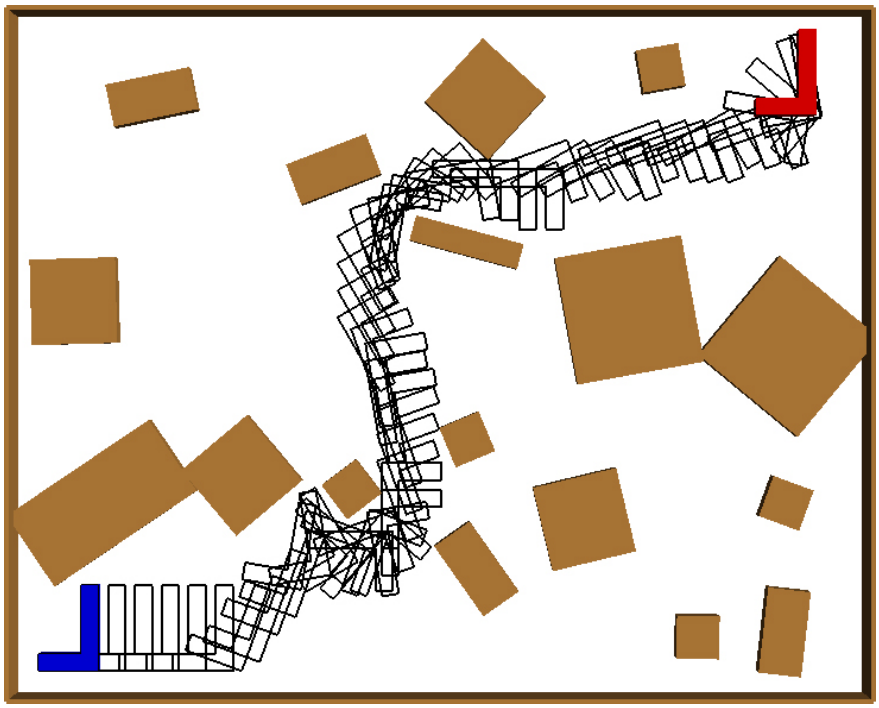


(a) Percorso generato dal **pianificatore centralizzato** con conoscenza perfetta e globale.

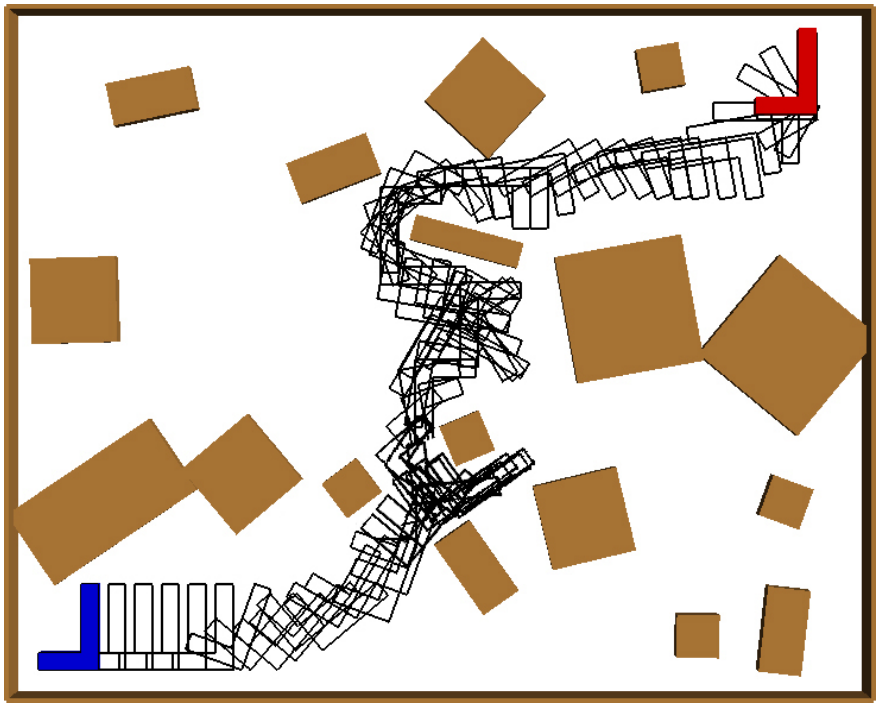


(b) Percorso generato dal pianificatore distribuito **senza euristiche**.

Figura 5.9: Percorsi a confronto



(a) Percorso generato dal pianificatore distribuito con l'euristica **Narrow Passage Detection**.



(b) Percorso generato dal pianificatore distribuito con l'euristica **Skeleton Pruning**.

Figura 5.10: Percorsi a confronto

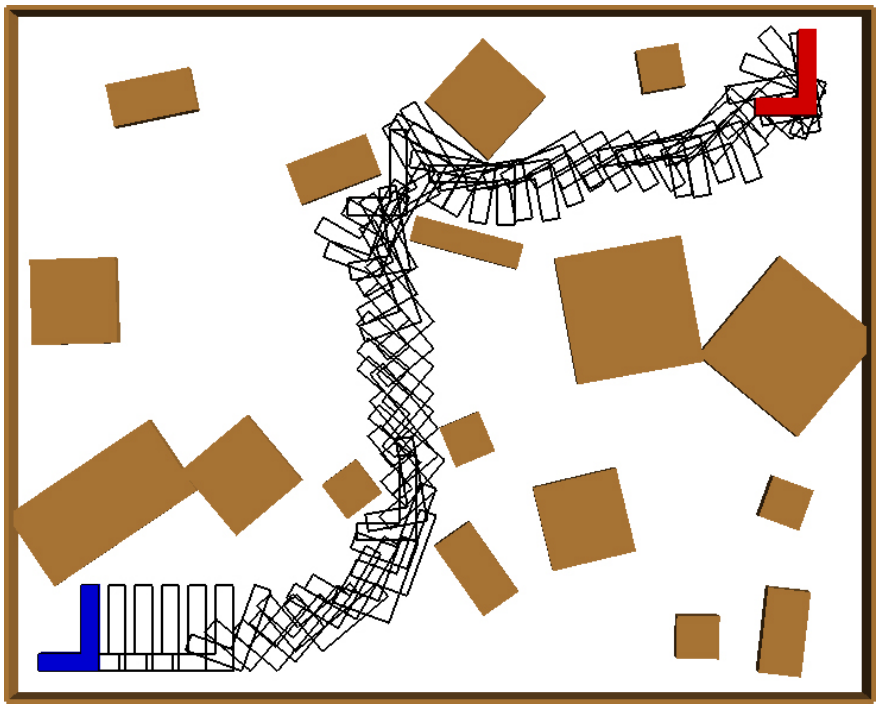


Figura 5.11: Percorso generato dal pianificatore distribuito con la combinazione di **tutte le euristiche**.

- quando ci sono molteplici robot che possono ricevere le informazioni dell'oggetto per la prosecuzione del percorso, la scelta non è più casuale come in precedenza, ma viene data priorità ai robot che hanno già calcolato una precedente parte di percorso.
- Quando un robot riceve un messaggio *Start Path*, prima di procedere con il calcolo del percorso, effettua un controllo per individuare un eventuale ciclo: prova a connettere la nuova posizione iniziale con i precedenti percorsi parziali.

Il primo punto, per poter essere realizzato, necessita una conoscenza maggiore rispetto al sistema precedente. È necessario conoscere la cronologia dei robot che hanno calcolato le parti precedenti di percorso. Perciò questa informazione deve essere trasmessa ad ogni passo congiuntamente al messaggio di *Start Path*. Ad ogni pianificazione parziale, il robot aggiorna la cronologia aggiungendo il proprio identificativo univoco alla fine della sequenza, e quindi inoltra questa informazione aggiuntiva allegata al messaggio per la prosecuzione di percorso.

Il secondo punto consiste nell'operazione di verifica per individuare eventuali cicli; questa è eseguita da ogni robot designato a proseguire la pianificazione prima di iniziare l'effettivo calcolo del percorso. Nel caso il robot corrente non abbia calcolato in precedenza nessuna parte di percorso, la verifica termina con successo: non c'è alcuna possibilità che il robot stia generando un ciclo. Invece, nel caso in cui il robot abbia già calcolato precedenti percorsi parziali, allora prova a connettere la nuova posizione iniziale ricevuta con ognuno dei percorsi già calcolati. Questo test di connessione viene eseguito con un rapido algoritmo di path planning locale, in cui si imposta come partenza il nuovo punto ricevuto e come destinazione il punto del percorso precedente più vicino alla partenza. Se viene individuato un percorso che connette questi due punti, allora significa che tutto il percorso successivo al precedente percorso locale (quello rilevato positivo al test di connessione) è stato inutile, poichè ha portato ad una posizione dell'oggetto raggiungibile anche in precedenza. È stato individuato un *loop*: si interrompe il normale processo di pianificazione e si attiva la specifica procedura per la rimozione del ciclo.

La procedura di rimozione del ciclo riporta il sistema nella posizione precedente alla creazione del loop, e modifica il potenziale in modo da evitare che si ripeta il medesimo ciclo. Per ottenere questo comportamento del sistema, sono stati introdotti due nuovi tipi di messaggi: *Loop* e *Forward Loop*. Qui di seguito è illustrato il processo. Il robot che individua il ciclo risponde al suo robot predecessore con un messaggio di *Loop*; questo messaggio indica il rilevamento di un ciclo. Quando un robot riceve un messaggio *Loop* aggiunge alla lista di celle bloccate (*lista nera* - vedi sottosezione precedente) la zona di uscita dell'oggetto dove è stato rilevato un ciclo. Questa azione è giustificata dal fatto tale zona porterà ad inviare il messaggio di prosecuzione nuovamente al robot che ha rilevato il loop, quindi si ritiene corretto bloccare tale passaggio. Poi, il robot che ha ricevuto *Loop* invia a sua volta un messaggio *Forward Loop* al suo robot predecessore; quando un robot riceve un messaggio *Forward Loop* o *Loop* elimina l'ultimo percorso memorizzato nella sua pila di percorsi parziali ed inoltra a sua volta il messaggio di *Forward Loop* al suo robot predecessore, il quale ripete a sua volta l'operazione. Questo processo itera fino a quando il messaggio viene inoltrato al robot che ha rilevato il ciclo. Questo robot era in attesa del ritorno del messaggio *Forward Loop*; quando lo riceve modifica il proprio campo

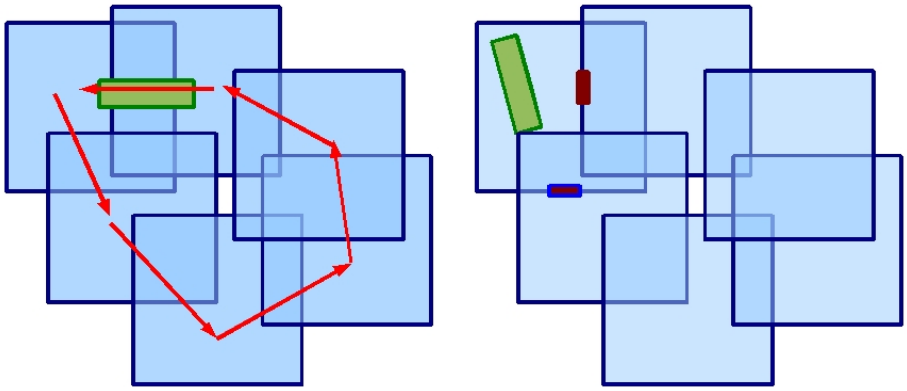


Figura 5.12: Effetto dell'euristica *Smart Loop Avoidance*: (i) viene riportato il sistema nello stato precedente al ciclo, (ii) si blocca la zona d'uscita dall'ultimo robot del ciclo (blocco rosso), e (iii) si assegna un elevato potenziale alla zona d'uscita dal primo robot del ciclo (blocco blu-rosso).

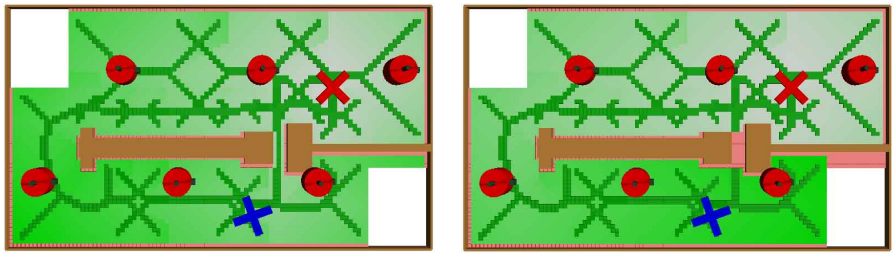
di potenziale in modo da evitare il ripetersi del ciclo. Poi ripete il calcolo del proprio percorso parziale. Il campo di potenziale viene modificato impostando un valore elevato alla zona d'uscita dell'oggetto. Per valore elevato si intende un valore poco inferiore alla soglia di potenziale assegnata agli ostacoli, i quali hanno valore di potenziale massimo. In questo modo la zona modificata è ancora percorribile, tuttavia il pianificatore ne viene respinto a causa dell'elevato valore di potenziale; questa strada verrà scelta solamente quando saranno state esaurite tutte le altre possibilità.

La Figura 5.12 chiarisce i concetti presentati attraverso una rappresentazione grafica.

Questa euristica raggiunge gli scopi per la quale è stata progettata e sviluppata, migliorando notevolmente la qualità dei percorsi pianificati: percorsi più brevi e senza cicli (Figura 5.8). Come svantaggio il processo necessita di una conoscenza maggiore, e riduce, seppure minimamente, la possibilità di trovare una soluzione.

5.5.3 Skeleton Pruning

Questa tecnica ha l'obiettivo di velocizzare il calcolo del percorso, e quindi limitare l'utilizzo delle risorse computazionali. L'idea su cui si basa questa euristica è la creazione di un campo di potenziale che si diffonda solo attraverso le vie che l'oggetto può realmente percorrere. Questa procedura viene eseguita durante la fase iniziale di creazione dello skeleton, durante la quale le parti dello skeleton che attraversano zone più strette della dimensione minima vengono eliminate. Per ogni cella dello skeleton viene effettuato un controllo: se non è presente un numero sufficiente di celle libere sia nella direzione verticale che in quella orizzontale allora la cella è rimossa dallo skeleton. Un valore significativo per la soglia minima di taglio può essere la dimensione minima dell'oggetto mobile. Tuttavia questo valore può essere impostato dall'utente come parametro iniziale.



(a) Skeleton senza alcuna modifica. (b) Skeleton *potato*.

Figura 5.13: Effetto dell'euristica di *Skeleton Pruning*.

La Figura 5.13 mostra l'effetto di questa euristica: nell'immagine di sinistra è mantenuto lo skeleton originale, mentre nell'immagine di destra è stata applicata l'eliminazione dei passaggi stretti. L'oggetto blu rappresenta la partenza, l'oggetto rosso la destinazione, ed il valore del campo di potenziale è descritto dalle tonalità di verde, più il colore è intenso, maggiore è il valore di potenziale. Si può vedere come nella Figura (a) il potenziale si diffonda attraverso il passaggio, mentre nella Figura (b) debba aggirare l'ostacolo, non riuscendo più a diffondersi attraverso le vie troppo strette.

Questa euristica è molto rapida, semplice e poco costosa, tuttavia spesso non è semplice individuare la dimensione minima da mantenere. Alcuni oggetti dalle forme particolari possono attraversare passaggi anche molto stretti attraverso accurati movimenti rotatori, pertanto bloccare uno di questi passaggi potrebbe tradursi in un'azione scorretta. Per esempio, l'oggetto mostrato in Figura 5.14 con forma ad "L" è in grado di attraversare il passaggio della Figura (a), ma non quello di Figura (b), pur trattandosi di due passaggi con la stessa ampiezza. Per questo motivo, per alcuni oggetti, è impossibile individuare la dimensione perfetta: valori troppo elevati chiudono percorsi percorribili, eliminando soluzioni valide; dimensioni troppo piccole lasciano percorsi impraticabili, e quindi non rimuovo competamente il problema del minimo locale.

5.5.4 Narrow Passage Detection

Questa tecnica ha gli stessi obiettivi e funzionalità della tecnica presentata nella sottosezione precedente (*Skeleton Pruning*): individuare e bloccare le vie troppo strette per il passaggio dell'oggetto, al fine di velocizzare il processo di pianificazione, e quindi migliorare l'efficienza del sistema. A differenza della precedente, questa euristica agisce in fase di pianificazione del percorso, individuando quando l'oggetto termina in un minimo locale.

Questa tecnica monitora il processo di pianificazione, in particolare controlla che il pianificatore continui la discesa del potenziale, e quindi analizzi configurazioni sempre più vicine alla destinazione. Quando viene rilevato un cambiamento nel comportamento, ed il sistema inizia ad allontanarsi dalla destinazione, significa che è stata imboccata una direzione scorretta, ed il sistema sta ora analizzando vie alternative. Per individuare se il cambiamento di direzione è stato effettivamente causato da un passaggio stretto, il sistema verifica che nella direzione di discesa del potenziale sia presente un'occlusione generata da ostacoli. Nel caso si tratti di un minimo locale causato da un passaggio

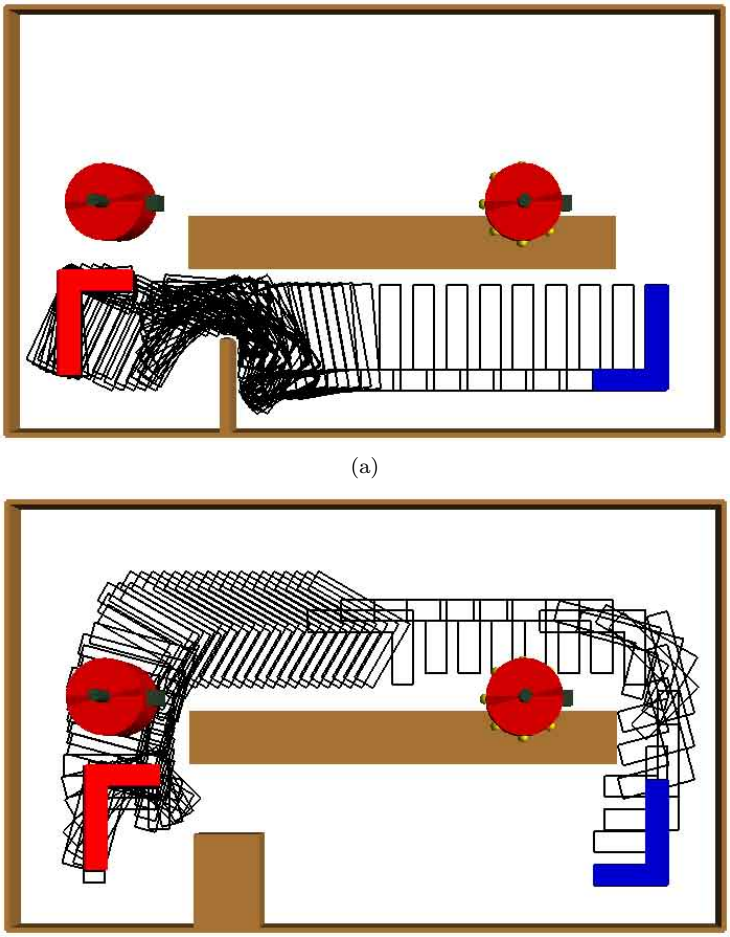
stretto, il robot inserisce un *blocco virtuale* nel passaggio e inoltra a tutti i vicini un messaggio di *Re-Spread*. Questo messaggio viene a sua volta inoltrato a tutti i robot del sistema attraverso *flooding*. L'azione innescata da questi messaggi è un azzeramento dello stato corrente del processo, e la ri-diffusione del campo di potenziale sull'intera mappa, per poi ricominciare nuovamente l'intero processo di pianificazione. Il *blocco virtuale* inserito dal robot ha come unico scopo quello di impedire al nuovo potenziale di diffondersi attraverso il passaggio stretto; in questo modo viene generato un nuovo campo di potenziale privo del minimo locale.

Con questa strategia si preferisce resettare l'intero processo e ricominciare totalmente la pianificazione piuttosto che effettuare una lunga ricerca esaustiva. Il sistema si costruisce un nuovo campo di potenziale che aggira i minimi locali, e permette una rapida pianificazione verso la soluzione. Il vantaggio di una pianificazione rapida è controbilanciato da un costo di un'ulteriore inizializzazione dell'intero sistema, il quale costo spesso è molto inferiore a quello che si sosterrebbe per una fuga dal minimo locale attraverso la ricerca esaustiva.

L'euristica di *Skeleton pruning* richiede che si definisca a priori, prima dell'inizio del processo, la dimensione minima dei passaggi, assumendo che corridoi più stretti di tale soglia non permettano all'oggetto di proseguire il percorso. Ma come mostrato dagli esempi di Figura 5.14, la dimensione non è l'unico parametro incisivo. Questa euristica prova a risolvere il problema in modo differente: solo i passaggi realmente non percorribili vengono bloccati. Questo nuovo approccio richiede che il processo cada in un minimo locale prima che il sistema lo individui; risulta pertanto più costoso (computazionalmente) e meno rapido dello *Skeleton pruning*, a favore di un sistema più *intelligente*, in grado di comprendere se un percorso è realmente percorribile o meno.

5.5.5 Discussione finale

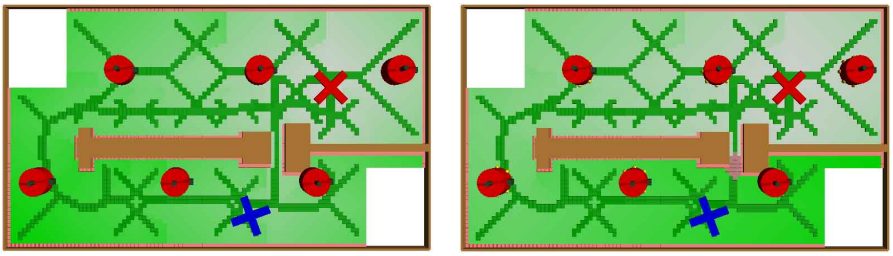
Il sistema con euristiche sopradescritto risolve alcuni dei principali aspetti critici di cui soffriva il sistema presentato nella sezione 5.4. Il nuovo sistema proposto opera con uno sfruttamento più limitato delle risorse e produce risultati qualitativamente migliori (percorsi più brevi). Tuttavia presenta una percentuale di successo inferiore rispetto all'altra architettura, poichè fallisce più spesso anche in presenza di una soluzione valida. I dati dettagliati sulle prestazioni di questo pianificatore, messi a confronto con i risultati del pianificatore precedente, sono illustrati nel capitolo 6.



(a)

(b)

Figura 5.14: Due scenari simili, con un passaggio stretto con la stessa dimensione; l'oggetto a L è in grado di attraversare il passaggio di figura (a), mentre fallisce nello scenario (b).



(a) Situazione iniziale.

(b) Individuato il passaggio stretto, il quale viene bloccato. Situazione dopo le seconda diffusione di potenziale.

Figura 5.15: Effetto dell'euristica di *Narrow Passage Detection*.

Capitolo 6

Risultati sperimentali

Per analizzare le prestazioni dei diversi pianificatori è stato eseguito un insieme di esperimenti, con lo scopo di poter confrontare i risultati ottenuti dai diversi approcci, e quindi valutare vantaggi e svantaggi delle diverse tecniche implementate.

I test sono stati eseguiti in simulazione, ripetendo l'esecuzione del pianificazione su un insieme di differenti scenari, e poi estraendone il comportamento medio. I diversi scenari mantengono una conformazione simile, in modo da poter misurare il reale effetto che i diversi approcci hanno sulle performance del sistema, senza essere influenzati da cambiamenti strutturali degli scenari. Gli scenari di test sono creati automaticamente, modificando in modo casuale le posizioni degli ostacoli, ma mantenendo la struttura di base.

La **qualità** del risultato è misurata con il rapporto tra la lunghezza relativa del percorso generato e la lunghezza del percorso di riferimento. Il percorso di riferimento è quello calcolato dal pianificatore centralizzato con conoscenza perfetta e globale. Si assume che tale pianificatore centralizzato restituisca il percorso desiderato: l'obiettivo che si mira raggiungere con una pianificazione distribuita; per questo motivo è stato utilizzato come riferimento nella valutazione qualitativa dei percorsi. Percorsi più brevi sono qualitativamente migliori, poichè corrispondono ad un notevole risparmio energetico nella successiva fase di attuazione del percorso. Nelle sezioni successive, i grafici e le valutazioni di qualità dei risultati utilizzano come unità di misura l'incremento percentuale della lunghezza del percorso rispetto dal percorso di riferimento.

Un altro parametro preso in esame è la **percentuale di successo** (*success ratio*): questo valore indica il rapporto tra gli esperimenti in cui è stata effettivamente individuata una soluzione (un percorso valido) ed il numero di esperimenti totali. Tutti gli scenari selezionati per i test hanno ambienti strutturati in modo da avere un percorso percorribile dalla partenza alla destinazione, tuttavia alcuni esperimenti falliscono nell'individuarlo a causa di errori di allineamento o eccessive semplificazioni del processo.

Tutti i risultati sulle performance mostrati in questo capitolo sono studiati al variare del livello di **errore di allineamento**, cioè l'errore nella misurazione della posizione relativa dei robot vicini. Nel sistema sono stati modellizzati due differenti tipi di errore: l'errore sulla distanza e quello sull'angolo. Gli effetti di questi due valori sono stati studiati in modo indipendente: mantenendo uno dei due errori nullo, e incrementando gradualmente l'altro. Negli esperimenti

sono stati utilizzati 9 distinti livelli di errore, con un incremento costante fra i livelli. Il valore indicato nei grafici corrisponde alla deviazione standard della distribuzione Gaussiana a media nulla che modellizza l'errore. Poichè l'effettivo valore dell'errore è generato in maniera stocastica, ad ogni esecuzione è possibile ottenere valori di allineamento differenti, con conseguente pianificazione del percorso differente. Per questo motivo ogni scenario è stato eseguito più volte, ed è stato utilizzato il valore medio.

Gli esperimenti sono stati eseguiti su scenari con una percentuale costante di spazio occupato dagli **ostacoli**; in particolare il 17% dell'area è coperta da ostacoli. Nel dettaglio è stato selezionato un insieme di blocchi con diverse dimensioni la cui somma raggiunga il 17% dell'area totale. Ogni scenario è stato generato con i medesimi blocchi in modo da mantenere costante anche la frammentazione degli ostacoli. In ogni scenario sono stati inseriti gli ostacoli con disposizione casuale: posizione e orientamento. Per ogni scenario, utilizzando l'algoritmo di pianificazione globale, è stato testato se fosse presente un percorso libero da collisioni dalla partenza alla destinazione, in modo da eseguire gli esperimenti solamente quando nell'ambiente ci sia realmente un percorso percorribile dalla partenza alla destinazione.

Poichè gli esperimenti sono eseguiti su ambienti generati in modo casuale, è possibile, in alcuni casi, ottenere risultati con valori relativamente molto alti, o molto bassi, rispetto al valore medio. Ciò si traduce in un alta deviazione standard di alcuni valori riportati, per questo motivo si è preferito mostrare nei grafici solamente il risultato medio degli esperimenti.

I pianificatori presentati nelle sezioni precedenti corrispondono a comportamenti (*behaviours*) dei controllori (*controllers*) dei robot eye-bot, i quali sono stati sviluppati nel simulatore ARGoS presentato in sezione 2.6, pertanto anche gli esperimenti sono stati eseguiti nel medesimo ambiente di simulazione.

Gli esperimenti sono stati eseguiti con l'obiettivo di misurare le prestazioni dei nuovi sistemi sotto diversi punti di vista, in particolare la scalabilità e la robustezza all'errore. Questo capitolo presenta i risultati sperimentali suddivisi in base all'aspetto analizzato. Nella sezione 6.1 è mostrata la robustezza del sistema al crescere dell'errore di allineamento, nella sezione 6.2 si analizzano le prestazioni in relazione all'uso delle risorse, mentre nelle sezioni 6.3 e 6.4 sono mostrati i risultati relativi a due differenti aspetti di scalabilità del sistema.

6.1 Robustezza: effetto dell'errore di allineamento

I pianificatori descritti nei capitoli precedenti sono stati testati su un insieme di scenari al fine di valutare le prestazioni, e studiare vantaggi e svantaggi dei differenti sistemi.

Gli esperimenti sono stati eseguiti utilizzando il pianificatore *algoritmico* presentato in sezione 5.4 e quattro differenti pianificatori con le euristiche illustrate nella sezione 5.5; questi quattro pianificatori sono stati progettati con diverse combinazioni di euristiche in modo da individuare gli effetti di ogni componente. Nel dettaglio è stato utilizzato un sistema con l'euristica di *Skeleton pruning*, un sistema con le euristiche di *Blocked Cell List* e *Smart Loop Avoidance* (in questo capitolo chiamato *Basic Heuristics*), un sistema con le caratteristiche di *Basic Heuristics* più l'euristica di *Narrow Passage Detection*, ed infine un sistema con la combinazione di tutte le quattro euristiche.

Attraverso i test presentati in questa sezione sono state valutate le prestazioni in termini di percentuale di successo e qualità della soluzione, in relazione all'incremento dell'errore di allineamento. Cioè l'errore nella misurazione della posizione relativa dei robot vicini, sia nella distanza che nell'angolo.

I casi di test

Gli scenari di test pur mantenendo invariato il rapporto tra area libera e area occupata dagli ostacoli, variano nella dimensione dell'area totale. I robot coprono l'intera area, e sono disposti in una formazione a griglia con una distanza dai propri vicini di 2 metri, ed un campo visivo di 2 metri in ogni direzione. Pertanto ogni coppia di robot vicini condivide una regione di mappa delle dimensioni di 4×2 metri.

Gli scenario differiscono oltre che per la dimensione dell'area e per la disposizione degli ostacoli nell'ambiente, anche per la forma e la dimensione dell'oggetto mobile. Le tre forme di oggetto utilizzate per i test sono: il rettangolo (con lunghezza pari a 1 metro), la forma ad "L" (simile all'oggetto di Figura 5.14), e la croce (simile all'oggetto rappresentato in Figura 5.15).

Poichè l'errore è modellizzato attraverso una distribuzione Gaussiana a media nulla, di cui è possibile impostare la deviazione standard, ad ogni esecuzione si possono ottenere valori di errore differenti. Per questo motivo, ogni scenario, per ogni livello di errore, è stato eseguito 40 volte, in modo da ottenere un insieme rappresentativo di risultati su cui fare la media.

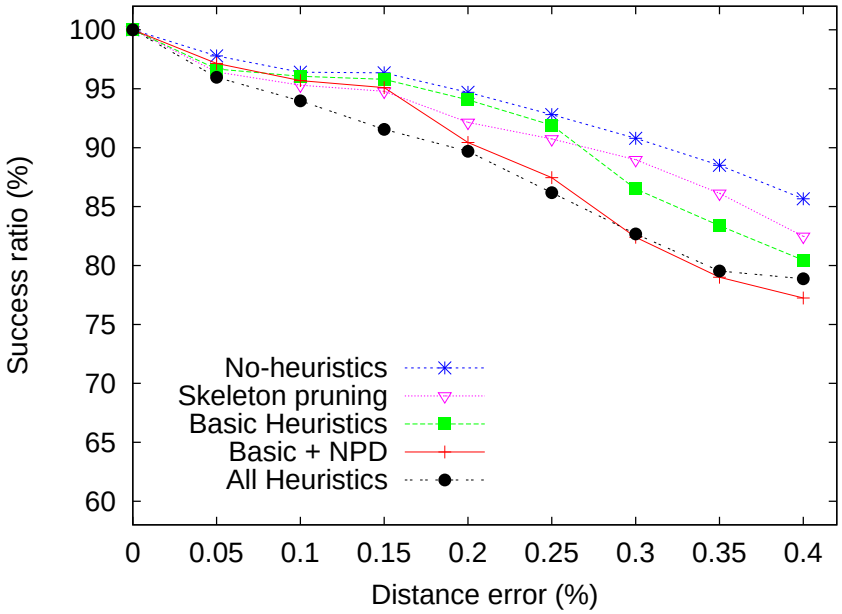
In totale sono stati eseguiti 25 scenari, con 2 tipi di errore con 9 livelli ciascuno e con 40 esecuzioni ogni volta. Quindi, si può dedurre che ogni punto dei grafici dei risultati delle Figure 6.1 e 6.2 corrisponde alla media di 1000 esperimenti.

Risultati

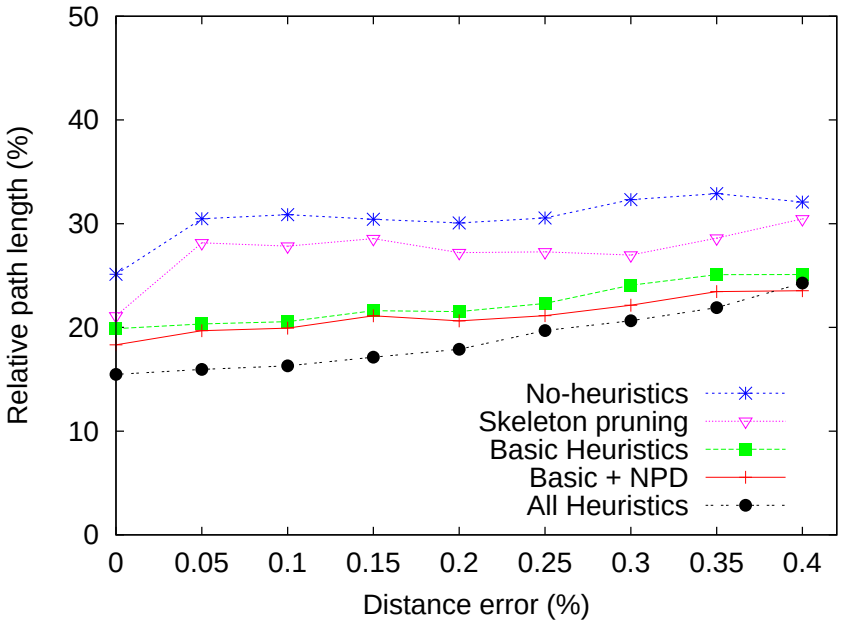
La prima informazione che si può notare dai grafici delle Figure 6.1 e 6.2 è l'andamento generale di tutti i pianificatori per entrambi i tipi di errore: al crescere dell'errore, il sistema deteriora le sue prestazioni con una qualità di percorso e una percentuale di successo in discesa. Il sistema è più sensibile agli errori sull'angolo (6.2), mentre è più robusto agli errori sulla distanza (6.1).

Un altro aspetto che emerge dai risultati è l'effetto delle euristiche nel sistema: il pianificatore senza euristiche peggiora rapidamente in termini di qualità di percorso, ma lentamente nella percentuale di successo. Mentre i pianificatori con euristiche si comportano in maniera opposta: mantenendo una buona qualità del percorso, ma riducendo velocemente la percentuale di successo totale, manifestando questo effetto in modo graduale al crescere del numero di euristiche introdotte. A conferma di questo fatto il sistema che evidenzia maggiormente l'effetto sopracitato è il pianificatore con tutte le euristiche.

Per errori nulli è possibile notare come le euristiche incidano positivamente sulla qualità del percorso, arrivando ad ottenere in media percorsi più lunghi del riferimento di solo il 15.5%, contro un incremento del 25% del pianificatore senza euristiche. Questo secondo valore è piuttosto elevato a causa di alcuni scenari più complicati che possono introdurre cicli nelle traiettorie (per maggiori dettagli su questo effetto vedi la sottosezione 5.4.1).

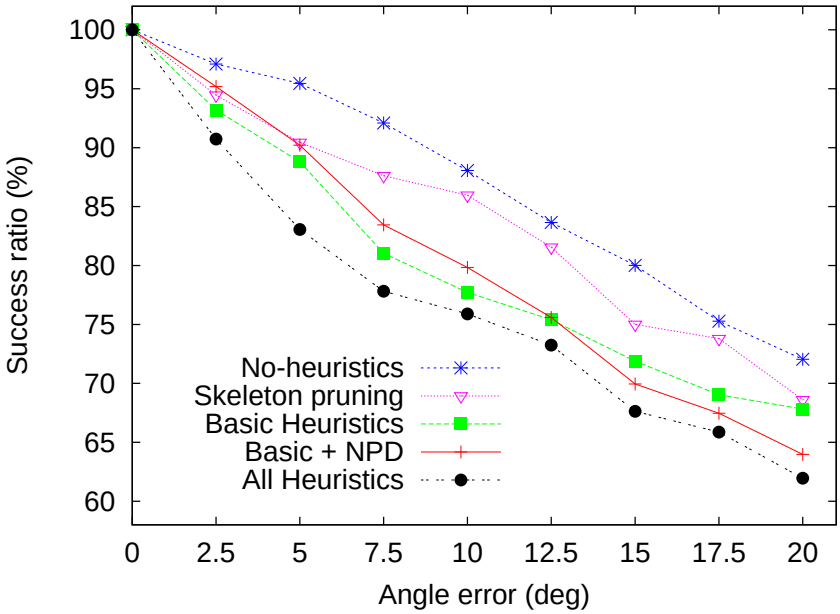


(a) Percentuale di successo vs. errore della distanza

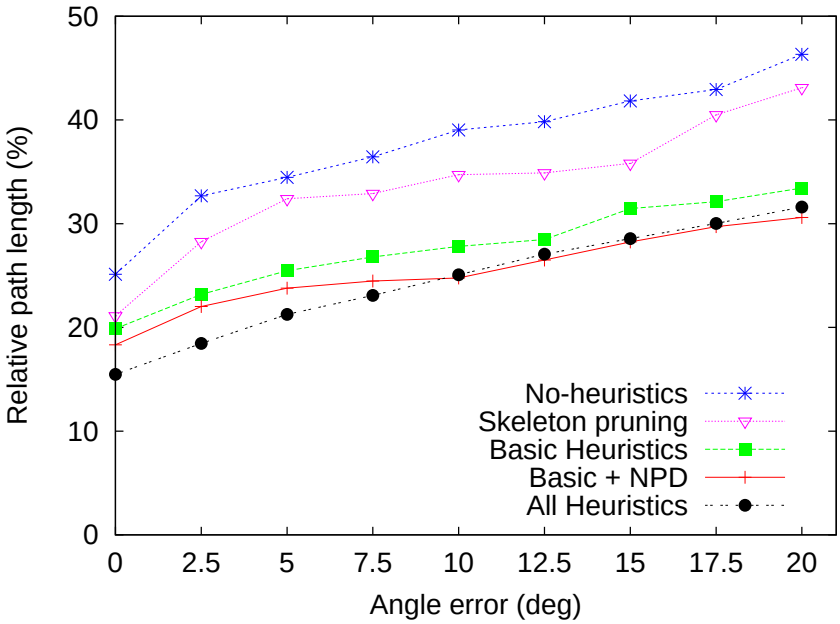


(b) Qualità del percorso vs. errore della distanza

Figura 6.1: Prestazioni dei differenti pianificatori al crescere dell'errore nella misurazione della distanza nella posizione relativa.



(a) Percentuale di successo vs. errore dell'angolo



(b) Qualità del percorso vs. errore dell'angolo

Figura 6.2: Prestazioni dei differenti pianificatori al crescere dell'errore nella misurazione dell'angolo nella posizione relativa.

Effetti dell'errore

I principali motivi per cui i pianificatori al crescere dell'errore producono risultati di qualità inferiore e riducono la percentuale di successo sono i seguenti:

(i) Un errore di allineamento si traduce in assunzioni errate dell'area visiva che due robot vicini condividono. Questo sfasamento può portare a difficoltà durante la fase di pianificazione, poichè il percorso consiste in una sequenza di passaggi dell'oggetto da un robot all'altro. Il passaggio è basato sulla condivisione di aree visive, perciò percezioni dell'ambiente errate possono portare a fallimenti nel processo di pianificazione. La sezione 5.1, ed in particolare la Figura 5.2, descrivono nel dettaglio questo genere di problema.

(ii) L'errore sulla posizione relativa dei robot vicini può rendere impossibile la prosecuzione del percorso in corridoi molto stretti, rendendo impraticabili alcuni passaggi. Ciò si traduce in minimi locali, cioè attrazioni verso direzioni che non permetto la prosecuzione del percorso verso la destinazione. Il problema del minimo locale ed i suoi effetti sono descritti dettagliatamente nella sottosezione 5.4.1. Una delle principali conseguenze è l'aumento dell'utilizzo di risorse, e, soprattutto per il pianificatore senza euristiche, un incremento della lunghezza del percorso.

(iii) Errori elevati possono portare ad un *fallimento locale* (tabella 5.1) nella pianificazione del percorso da parte di un robot incapace di trasmettere la posizione corretta ai robot vicini. Ciò si traduce in un fallimento dell'intero processo se il robot in questione copre una parte "*insostituibile*" del percorso; nel caso in cui ci siano strade alternative, il sistema è in grado di rilevarle, creando così un percorso che passa per altri robot. Nella maggioranza dei casi, questo processo si riflette in percorsi più lunghi al fine di aggirare il robot problematico.

Euristiche a confronto

Il pianificatore con ***Skeleton Pruning*** ha lo scopo di eliminare i minimi locali, e quindi creare percorsi più brevi e con un migliore utilizzo delle risorse. Questa euristica è molto rapida, semplice e poco costosa, tuttavia spesso non è semplice individuare la dimensione minima da mantenere. Alcuni oggetti dalle forme particolari possono attraversare passaggi anche molto stretti attraverso accurati movimenti rotatori, pertanto bloccare uno di questi passaggi potrebbe tradursi in un'azione scorretta. Un esempio è mostrato dalla traiettoria dell'oggetto a "L" della Figura 5.14. Per questo motivo è stata selezionata una soglia ridotta, che ha bloccato solamente i passaggi per i quali sicuramente l'oggetto non è in grado di passare, ma ha lasciato aperte alcune strade che non è sicuro permettano il passaggio dell'oggetto. Come si può notare in entrambi i grafici di qualità 6.1(b) e 6.2(b), questo pianificatore ha un andamento molto simile al pianificatore senza euristiche, ma di qualche punto percentuale inferiore (percorso più breve). L'andamento è simile poichè mantiene il medesimo comportamento in fase di pianificazione, tuttavia produce risultati migliori eliminando alcuni minimi locali attraverso il blocco di alcuni dei percorsi troppo stetti.

In assenza di errore, il pianificatore ***Basic Heuristics*** presenta un miglioramento nella qualità media del percorso rispetto al caso senza euristiche, con una riduzione della lunghezza relativa del percorso di 5 punti percentuali. Inoltre, dai grafici dei risultati 6.1(b) e 6.2(b) emerge come questo sistema sia robusto all'errore in termini di qualità delle soluzioni, presentando una crescita limi-

tata della lunghezza relativa del percorso a fronte dell'incremento dell'errore. Tuttavia, questo aspetto positivo è controbilanciato da una maggiore sensibilità all'errore di allineamento in termini di percentuale di successo, soprattutto per valori elevati dell'errore. La percentuale di successo minore è la conseguenza delle semplificazioni introdotte dalle euristiche utilizzate, le quali portano il vantaggio di una migliore qualità dei risultati (e, vedremo in futuro, un migliore utilizzo delle risorse), ma hanno lo svantaggio di eliminare alcune soluzioni.

Il pianificatore ***BH+NPD*** mantiene le caratteristiche del pianificatore *Basic Heuristics* ed aggiunge la funzionalità di *Narrow Passage Detection*, la quale rileva automaticamente durante la discesa di potenziale la presenza di passaggi troppo stretti per il reale passaggio dell'oggetto mobile, ed elimina il minimo locale. Questo pianificatore ha il medesimo andamento qualitativo di BH, traslato di circa 2 punti percentuali in basso (percorso più breve).

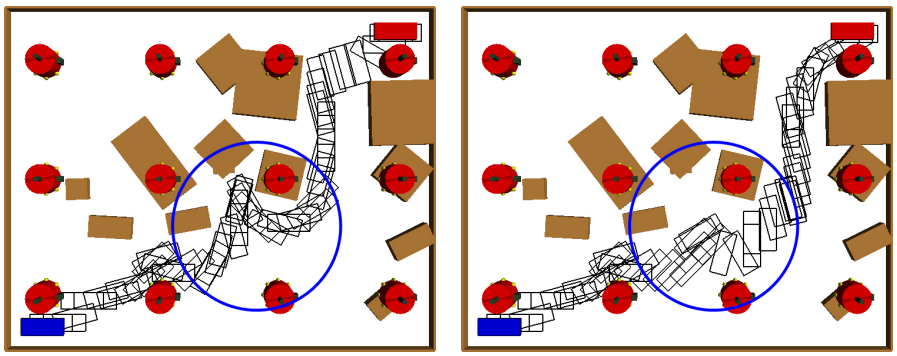
Per valori di errore sotto una certa soglia ($\leq 15\%$ in distanza e $\leq 12.5°$ in angolo), questo pianificatore ha una percentuale di successo paragonabile, o in alcuni casi perfino migliore di *BH*; mentre per valori di errore superiori il sistema è in media meno robusto, fallendo un numero maggiore di volte. Questi fallimenti sono causati da una diffusione imprecisa del campo di potenziale che porta a bloccare alcuni passaggi che si sarebbero potuti superare con una ricerca esaustiva della soluzione.

Il pianificatore con **tutte le euristiche** corrisponde a quello precedente con l'aggiunta di *Skeleton Pruning*, la quale porta a buoni risultati per valori di errore sotto a una certa soglia. Per esempio per errore nullo, questo pianificatore ha una lunghezza di percorso di circa il 15% contro il 18% del pianificatore precedente. Questo miglioramento è causato dalla chiusura di alcuni passaggi troppo stretti che l'euristica NPD non fa in tempo ad individuare poichè sono velocemente schivati (un esempio di un percorso che presenta questa casualità è mostrato in Figura 6.3). Si tratta di piccoli miglioramenti nella traiettoria, che al crescere dell'errore, e quindi della lunghezza del percorso, vanno a ridurre la loro influenza, fino a sparire. Infatti lo scarto di miglioramento si riduce con l'aumentare dell'errore fino ad ottenere prestazioni migliori con il precedente pianificatore. Per esempio per valori di errore sull'angolo superiori ai 10°, il sistema precedente presenta percorsi in media più brevi e con un tasso di successo dal 4% al 2% maggiore. Pertanto questa combinazione di euristiche produce buoni risultati principalmente in scenari con limitati valori di errore.

6.2 Efficienza: utilizzo delle risorse

In questa sezione è stato misurato l'utilizzo delle risorse computazionali e comunicative a disposizione, il quale rappresenta un importante strumento di misura di efficienza del sistema proposto. L'efficienza è un parametro di fondamentale importanza per questo progetto, poichè ha un impatto diretto sulla scalabilità del sistema.

Per questi esperimenti si è mantenuta una dimensione della mappa costante, poichè le risorse computazionali (stimate attraverso il tempo di esecuzione dell'intero processo) non corrispondono ai valori che si otterrebbero con un'esecuzione su robot reali poichè nel simulatore è presente un'unica macchina che deve eseguire le operazioni di tutti i robot, con suddivisione temporale delle risorse computazionali. Pertanto è ovvio che al crescere del numero di robot,



(a) Percorso finale del pianificatore *Basic Heuristics + Narrow Passage Detection*.

(b) Percorso finale del pianificatore con la combinazione di tutte le euristiche.

Figura 6.3: Le immagini dei due percorsi evidenziano (cerchio blu) come alcuni passaggi troppo stretti non vengano individuati dall'euristica di *Narrow Passage Detection* (figura (a)), poichè in tempi molto ridotti viene individuata una strada alternativa, mentre con l'euristica *Skeleton Pruning* attiva la deviazione di percorso non è presente (figura (b)).

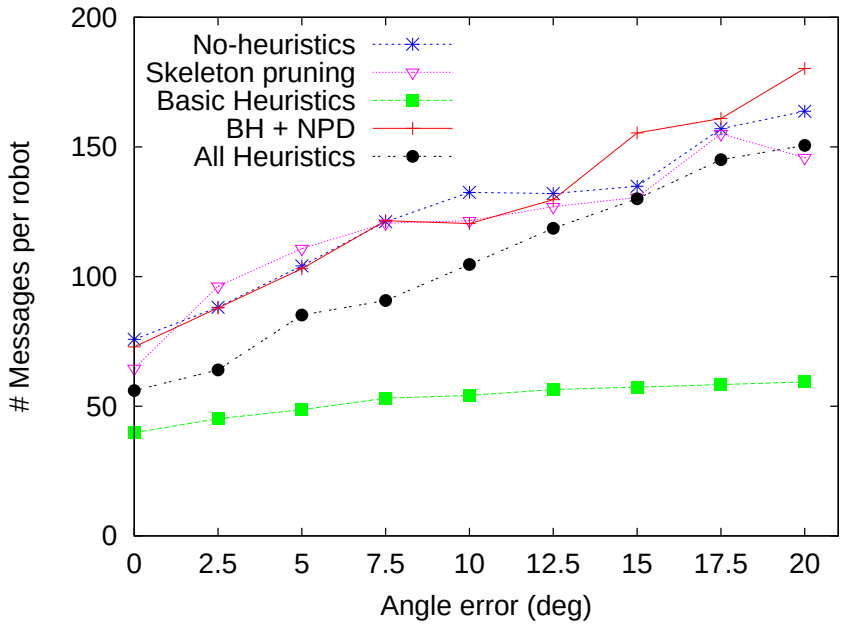
aumenti anche il tempo di esecuzione in simulazione. Per eliminare, o almeno limitare, questo effetto indesiderato in questa sezione sono stati usati scenari con dimensioni della mappa e numero di robot costanti.

In questa sezione sono stati confrontati gli stessi pianificatori utilizzati per i test della sezione 6.1. I parametri presi in esame sono le risorse comunicative, misurate attraverso il numero medio di messaggi che ogni robot invia (e riceve), e le risorse computazionali, misurate attraverso il tempo totale di esecuzione. Quest'ultime mirano a mostrare un andamento qualitativo delle prestazioni, e non hanno l'obiettivo di esprimere risultati quantitativi. Pertanto i valori indicati mostrano il tempo in secondi del processo in simulazione, che non è paragonabile al reale tempo d'esecuzione sui robot reali per le motivazioni sopraelencate. L'obiettivo è quello di confrontare i risultati dei diversi pianificatori, individuando i pianificatori più rapidi di altri.

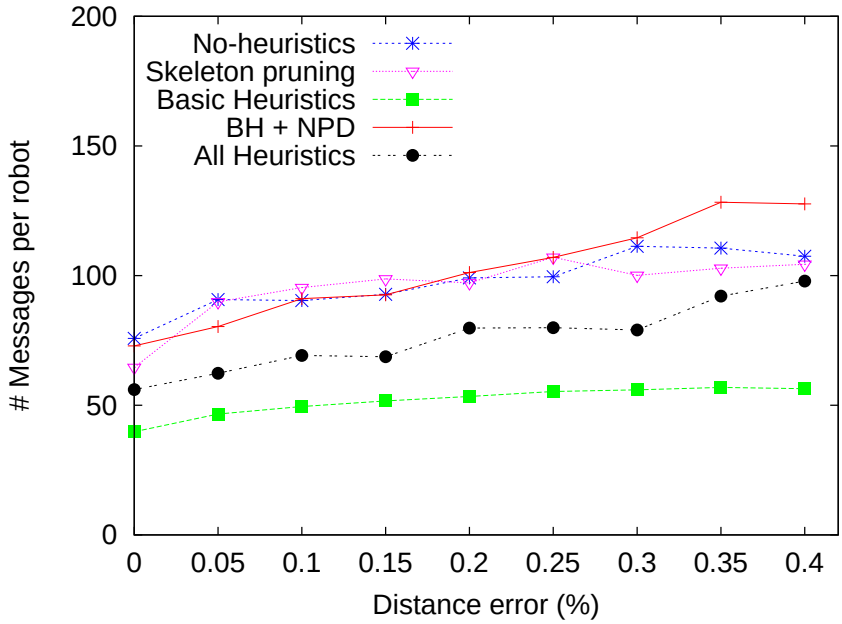
L'unità di misura dell'utilizzo di risorse comunicative è la media dei messaggi che ogni robot riceve (e quindi invia) durante l'intero processo di pianificazione, dalla fase di rilevamento delle posizioni dei vicini al *flooding* del messaggio con il risultato finale del processo.

Risultati

Il primo aspetto che è possibile notare dai grafici di Figura 6.4 è la presenza di tre diversi andamenti principali nell'utilizzo delle risorse di comunicazione. Il pianificatore con *Basic Heuristics* presenta un andamento costante, con un piccolo coefficiente di incremento, e con un numero di messaggi molto basso: una media di circa 50 messaggi per robot, in scenari con un numero medio di robot vicini pari a 4.8, corrisponde a 10 messaggi ad ogni vicino, più pochi altri messaggi con un destinatario preciso. Se si considera che in tutti gli scenari sono necessari per ogni vicino due messaggi durante la prima fase di rilevamento delle posizioni, ed un messaggio finale con il risultato del processo, si intuisce che il

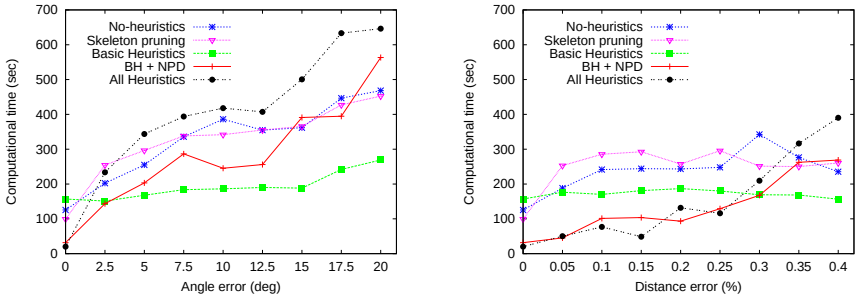


(a) Numero medio di messaggi per robot vs. errore dell'angolo



(b) Numero medio di messaggi per robot vs. errore della distanza

Figura 6.4: Confronto dei differenti pianificatori nell'utilizzo delle **risorse comunicative** al crescere dell'errore di allineamento (in (a) errore nell'angolo e in (b) errore nella distanza).



(a) Tempo medio di esecuzione vs. errore dell'angolo

(b) Tempo medio di esecuzione vs. errore della distanza

Figura 6.5: Confronto dei differenti pianificatori nell'utilizzo delle **risorse computazionali** al crescere dell'errore di allineamento (in (a) errore nell'angolo e in (b) errore nella distanza).

numero di messaggi utilizzati per la diffusione del potenziale e la pianificazione di percorso è molto limitato.

Un andamento differente è quello dei pianificatori senza euristiche, con l'euristica *Skeleton Pruning*, e con la combinazione di euristiche *BH+NPD*. Questi tre pianificatori sono molto sensibili all'errore, ed aumentano notevolmente il numero di messaggi al crescere dell'errore, arrivando a valori molto elevati (180 messaggi per robot).

Il pianificatore con la combinazione di tutte le euristiche ha il terzo andamento, il quale è simile al precedente, è quindi sensibile all'incremento di errore, tuttavia presenta valori di un gradino inferiori (circa 25 messaggi in meno).

Pianificatori a confronto

Il **pianificatore senza euristiche** mostra un elevato utilizzo di risorse poichè quando lo scenario presenta minimi locali, il pianificatore prova esaustivamente tutte le configurazioni nella direzione di errata attrazione, con spreco di risorse sia temporali che comunicative.

Il pianificatore con ***Skeleton Pruning*** elimina alcuni dei minimi locali, in alcuni casi migliorando l'esecuzione, ma nel complesso non incide significativamente nell'utilizzo delle risorse.

Il pianificatore con le ***Basic Heuristics*** mostra il migliore utilizzo delle risorse con un comportamento molto robusto all'errore. Sia le comunicazioni che il tempo di esecuzione rimangono pressochè costanti, con incrementi minimi all'aumentare dell'errore.

Il pianificatore con la combinazione ***BH+NPD*** introduce la tecnica *Narrow Passage Detection* presentata nella sottosezione 5.5.4, con la quale il sistema reagisce al problema del minimo locale con il riavvio dell'intero processo e ripetendo la fase di diffusione del potenziale. In alcuni casi questo approccio può velocizzare l'esecuzione, tuttavia il costo di re-inizializzazione e ri-diffusione del campo di potenziale è pagato con un costo di circa $7 \times N$ messaggi ogni minimo locale individuato, con N pari al numero di robot vicini. Negli esperimenti effettuati questa tecnica porta ad un utilizzo notevole di risorse comunicative

e computazionali. È possibile che in scenari con conformazioni differenti, il costo di re-inizializzazione sia meno influente, e migliori le prestazioni totali del sistema.

Il pianificatore con la combinazione di **tutte le euristiche**, è molto simile al pianificatore precedente con l'aggiunta dell'euristica di *Skeleton Pruning*. Questa tecnica individua, e blocca, prima dell'inizio della reale pianificazione i passaggi molto stretti, in questo modo il numero di minimi locali viene ridotto, ed in conseguenza sono ridotte anche le re-inizializzazioni scatenate dall'euristica di *Narrow Passage Detection*. Ciò si riflette in un minor numero medio di messaggi, poichè il sistema viene re-inizializzato un numero inferiore di volte.

6.3 Scalabilità: dimensioni dell'ambiente

Uno degli principali obiettivi di questo progetto è la scalabilità: i pianificatori sono stati progettati e sviluppati con particolare attenzione alle caratteristiche relative alla scalabilità del sistema. In questa sezione è stato eseguito un insieme di esperimenti al fine di misurare le performance dei differenti sistemi al crescere delle dimensioni dell'ambiente. I pianificatori studiati sono il pianificatore senza euristiche, il pianificatore con le *Basic Heuristics* (*Blocked Cell List* e *Smart Loop Avoidance*), ed il pianificatore con la combinazione di tutte le euristiche.

Perchè il sistema sia scalabile deve essere in grado di incrementare le proprie prestazioni attraverso l'aggiunta di nuove risorse. In particolare il pianificatore deve essere in grado di calcolare il percorso in un ambiente più vasto a fronte dell'introduzione di nuovi robot in numero proporzionale alla crescita della mappa. Inoltre, ci si aspetta che il sistema, a fronte di questo incremento di robot, mantenga un utilizzo unitario (di ogni singolo robot) delle risorse indipendente dalla dimensione totale della mappa e dal numero di robot totali.

Per misurare quanto i pianificatori siano in grado di risolvere il problema in ambienti più vasti, è stata misurata la percentuale di successo a fronte di un incremento della mappa, e quindi della griglia di robot. L'utilizzo di risorse è misurato attraverso il numero medio di messaggi trasmesso (o ricevuto) da ogni robot del sistema. Nel caso in cui l'utilizzo delle risorse cresca in misura esponenziale rispetto alle dimensioni dell'ambiente, oppure le prestazioni subiscano un drastico peggioramento all'aumentare dei robot, allora il sistema non può essere considerato scalabile.

I casi di test

Sono stati eseguiti i test su 21 scenari, incrementando le dimensioni dell'ambiente e modificando la posizione degli ostacoli nella mappa. In tutti gli scenari è mantenuto costante il rapporto tra area libera e area occupata dagli ostacoli, pari al 17%.

I robot sono disposti in formazione a griglia con una distanza costante di 2 metri tra un robot e l'altro, e condividono con i vicini un'area di 4×2 metri. Lo scenario base è di $80m^2$ ($10m \times 8m$) con una copertura di 12 robot in una formazione a griglia 4×3, come mostrato in Figura 6.8(a). I test sono stati effettuati raddoppiando e triplicando la dimensione della griglia per ambienti con dimensioni rispettivamente pari al 175% e al 250% della mappa iniziale. La configurazione di partenza dell'oggetto mobile è mantenuta in un angolo

della mappa, mentre la destinazione è costantemente posizionata nell'angolo opposto, in modo che la distanza fra le due configurazioni incrementi insieme alle dimensioni della mappa. Un esempio di tre scenari è mostrato in Figura 6.8, nella quale è possibile vedere l'incremento delle dimensioni dell'ambiente, la copertura totale dei robot e le posizioni delle configurazioni di partenza e di destinazione dell'oggetto mobile.

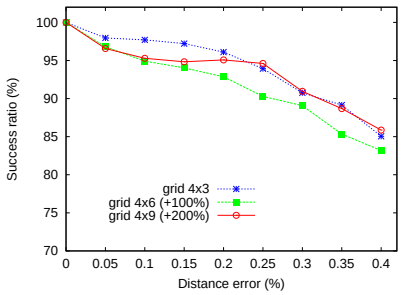
Risultati

In generale, **per errori di allineamento nulli**, o molto bassi, i percorsi restituiti dal sistema hanno una **lunghezza** maggiore del percorso di riferimento in conseguenza a piccole deviazioni sul percorso poco prima del passaggio dell'oggetto da un campo visivo all'altro. Questo fenomeno è riconducibile alla frammentazione degli skeleton presentata nella sottosezione 5.2.1; in assenza di errore si tratta dell'unica causa di crescita della lunghezza del percorso. Analizzando i grafici della qualità dei risultati (Figure 6.6 e 6.7) è possibile vedere come in percorsi più lunghi, questi movimenti aggiuntivi in prossimità dei confini dei campi visivi, e le lievi deviazioni dovute alla combinazione di skeleton parziali, abbiano un'incidenza minore. In assenza di errore, il rapporto tra lunghezza del percorso individuato e lunghezza del percorso di riferimento è in media minore per i percorsi più lunghi; da ciò si deduce che quei movimenti aggiuntivi causati dalla conoscenza distribuita crescono in rapporto inferiore rispetto alla crescita del percorso totale. Per esempio, con errore nullo, il pianificatore senza euristiche con una griglia di 4×9 robot trova percorsi in media il 20% più lunghi del percorso di riferimento, invece lo stesso pianificatore con un terzo dei robot restituisce percorsi con un incremento medio di lunghezza pari al 27% (grafico 6.6(b)).

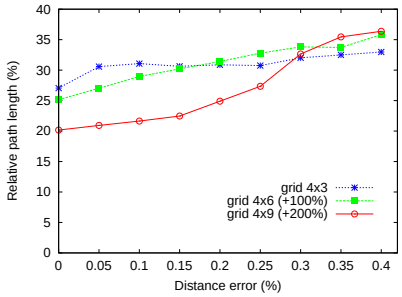
Al crescere dell'errore di allineamento i robot hanno percezioni errate della realtà, le quali generano problemi durante l'operazione di passaggio dell'oggetto da un robot all'altro. Durante la fase di pianificazione, quando l'oggetto esce dal campo visivo locale, il robot passa il controllo ad un suo vicino, perchè quest'ultimo prosegua la parte di pianificazione locale. I due robot, che si susseguono nella pianificazione distribuita, condividono un'area dell'ambiente, cioè i loro campi visivi si sovrappongono. Quando il sistema è soggetto ad errore di allineamento, la percezione del campo visivo del vicino è disallineata rispetto alla posizione reale. Ciò si traduce in un maggior numero di spostamenti rispetto al caso ottimale di allineamento perfetto (assenza di errore). Per questo motivo, la **lunghezza relativa del percorso** cresce con l'errore di allineamento.

Negli esperimenti di questa sezione è stata incrementata gradualmente la dimensione dell'ambiente, e con essa anche la lunghezza del percorso totale, di conseguenza anche il numero di passaggi dell'oggetto da un campo visivo all'altro ha subito un incremento. L'errore di allineamento influisce sul passaggio dell'oggetto, ed è stato illustrato come gli spostamenti "superflui al percorso finale" generati durante il passaggio da un campo visivo al successivo siano maggiori con il crescere dell'errore. In conseguenza a ciò, la lunghezza relativa di percorso in relazione all'errore presenta un coefficiente di crescita proporzionale al del numero di robot, e quindi alle dimensioni dell'ambiente (seconda colonna di Figure 6.6 e 6.7).

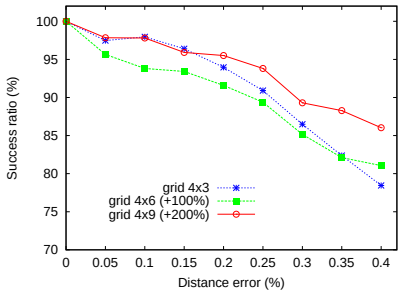
In conclusione, la **qualità dei risultati** in scenari più ampi è migliore per errori bassi, ma decresce più rapidamente all'aumentare dell'errore di allinea-



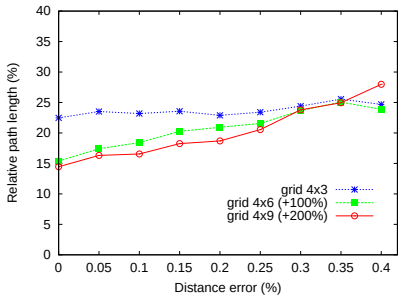
(a) **Pianificatore senza euristiche**
Success ratio vs. errore della distanza



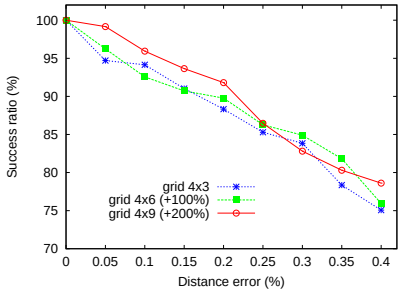
(b) **Pianificatore senza euristiche**
Qualità vs. errore della distanza



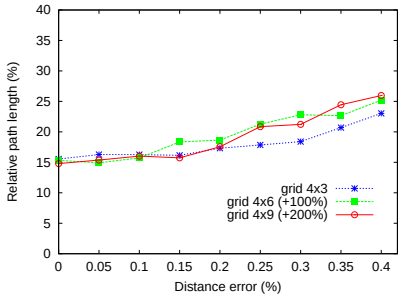
(c) ***Basic Heuristics***
Success ratio vs. errore della distanza



(d) ***Basic Heuristics***
Qualità vs. errore della distanza

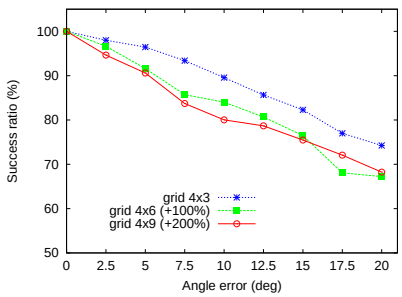


(e) **Pianificatore con tutte le euristiche**
Success ratio vs. errore della distanza

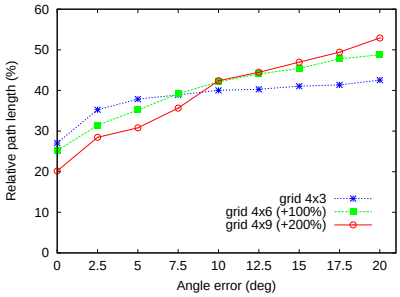


(f) **Pianificatore con tutte le euristiche**
Qualità vs. errore della distanza

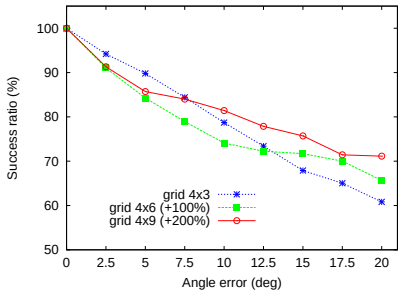
Figura 6.6: Scalabilità: incremento delle dimensioni dell'ambiente in relazione all'errore sulla distanza. Differenti pianificatori a confronto. In particolare, il pianificatore senza euristiche, con le *Basic Heuristics* e con la combinazione di tutte le euristiche.



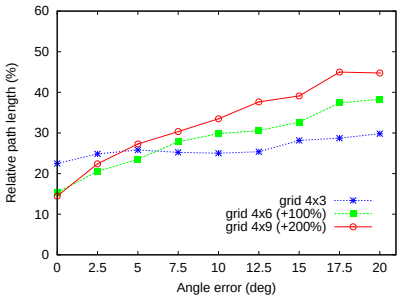
(a) **Pianificatore senza euristiche**
Success ratio vs. errore dell'angolo



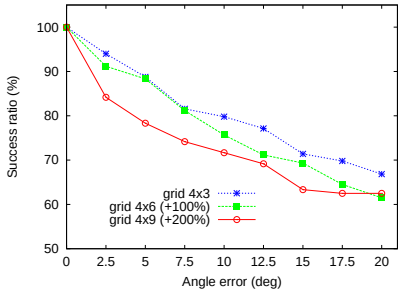
(b) **Pianificatore senza euristiche**
Qualità vs. errore dell'angolo



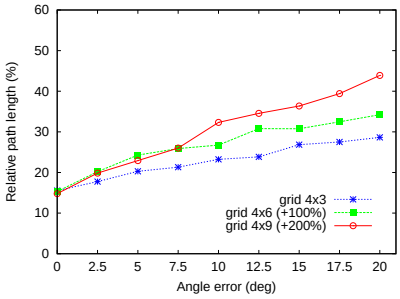
(c) ***Basic Heuristics***
Success ratio vs. errore dell'angolo



(d) ***Basic Heuristics***
Qualità vs. errore dell'angolo

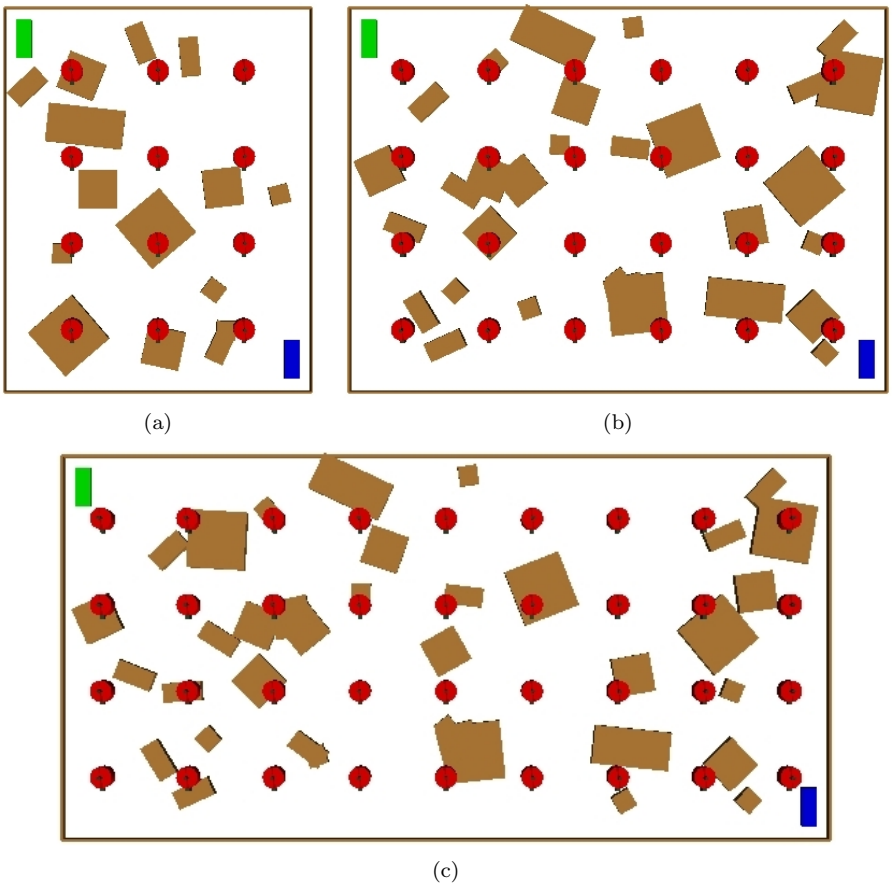


(e) **Pianificatore con tutte le euristiche**
Success ratio vs. errore dell'angolo



(f) **Pianificatore con tutte le euristiche**
Qualità vs. errore dell'angolo

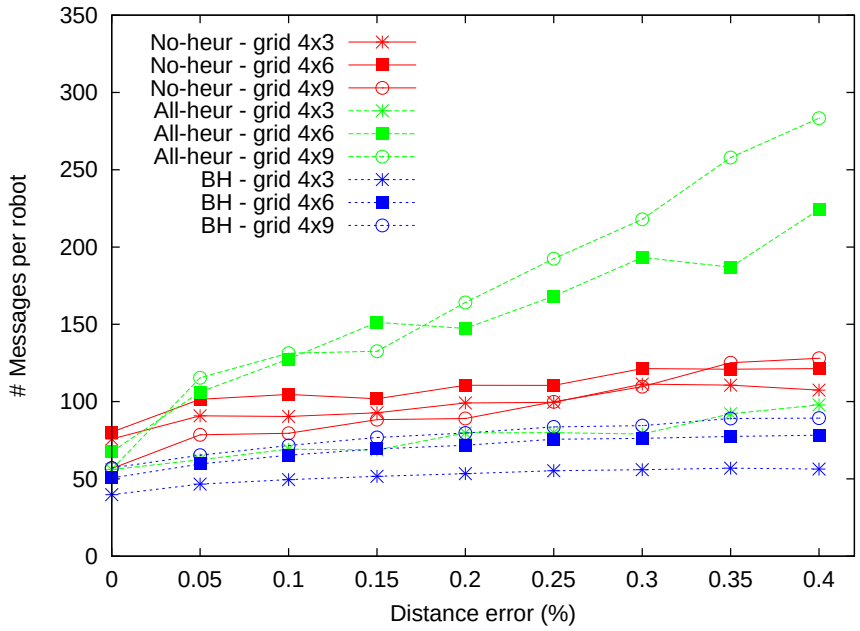
Figura 6.7: Scalabilità: incremento delle dimensioni dell'ambiente in relazione all'errore sull'angolo. Differenti pianificatori a confronto. In particolare, il pianificatore senza euristiche, con le *Basic Heuristics* e con la combinazione di tutte le euristiche.



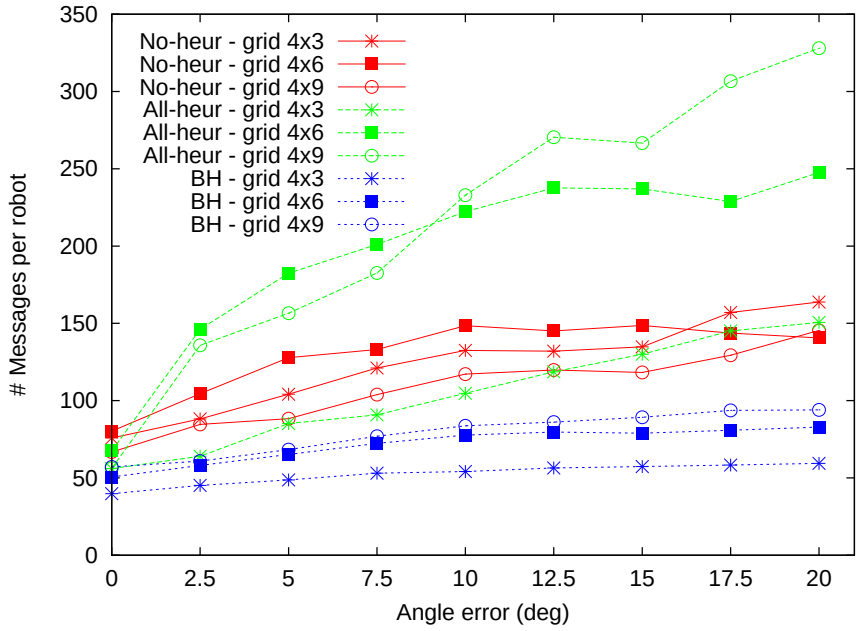
(a) (b)

(c)

Figura 6.8: Scalabilità: incremento delle dimensioni dell'ambiente.



(a) Numero medio di messaggi per robot vs. errore della distanza



(b) Numero medio di messaggi per robot vs. errore dell'angolo

Figura 6.9: Scalabilità: utilizzo delle risorse comunicative in relazione all'incremento delle dimensioni della mappa ed al crescere dell'errore di allineamento.

mento. Il sistema si dimostra più sensibile all'errore sull'angolo (Figura 6.7), ed è più robusto all'errore sulla distanza (Figura 6.6).

Un risultato molto positivo corrisponde ad una **percentuale di successo** con differenze poco significative e molto limitate (prima colonna di Figure 6.6 e 6.7). I valori peggiori sono in maggioranza per gli scenari più ampi, i quali hanno maggiori probabilità di avere un robot con errori di allineamento così elevati da non permettere la prosecuzione del percorso, e che quindi blocchi l'unica strada percorribile. Tuttavia i grafici con le percentuali di successo non mostrano cambiamenti significativi al crescere delle dimensioni della mappa; l'andamento dei risultati è molto simile, ed i coefficienti angolari delle tre curve di ogni grafico hanno cambiamenti minimi.

I grafici di Figura 6.9 mostrano l'**utilizzo di risorse** dei diversi pianificatori al crescere delle dimensioni della mappa, e quindi dei robot. L'aumento dei robot avviene raddoppiando la lunghezza della griglia, in particolare si hanno 4×3, 4×6 e 4×9 robot per scenario, come mostrato in Figura 6.8. Con questo tipo di incremento ogni robot ha in media rispettivamente 4.8, 5.7 e 5.9 robot vicini, poichè un robot in posizione centrale ha come vicini gli 8 robot intorno a lui, mentre un robot sul lato, o nell'angolo della mappa, ha un numero minore di vicini. Il numero medio di vicini è parametro importante, da tenere in considerazione durante lo studio dell'utilizzo di risorse, in quanto i robot possono comunicare solo localmente ai propri vicini. Dalle curve dei grafici è possibile notare come il pianificatore *Basic Heuristics* presenti un utilizzo delle risorse limitato, molto robusto all'errore e con ottima scalabilità. Mentre il pianificatore con tutte le euristiche ha una crescita più rapida per scenari con un numero maggiore di vicini. L'aumento del numero dei vicini è del 17% e del 23%, mentre il numero dei messaggi subisce un incremento nella velocità di crescita (coefficiente angolare medio della curva) del 275% e del 415% per gli errori sulla distanza e del 139% e del 187% per gli errori sull'angolo. Pertanto, dai risultati emerge come questo pianificatore soffra dell'aumento del numero di vicini con un incremento eccessivo del numero di messaggi in presenza di errore nel sistema. I risultati relativi agli altri due pianificatori portano a conclusioni differenti: tali pianificatori presentano un'architettura scalabile nelle dimensioni dell'ambiente e nel numero di robot vicini con riferimento all'utilizzo di risorse comunicative.

6.4 Scalabilità di carico: incremento della densità di robot

Per scalabilità di carico si intende la capacità di un sistema di incrementare le proprie prestazioni se a tale sistema vengono fornite nuove risorse. In questa sezione è stato studiato come un aumento della densità dei robot (nuove risorse) in una area di dimensione fissa influisca sulle prestazioni del pianificatore al variare dell'errore. Per lo studio di questa caratteristica sono stati confrontati il pianificatore senza euristiche ed il pianificatore con le *Basic Heuristics*, in relazione all'incremento dell'errore. In Figura 6.10 sono mostrati i risultati solamente rispetto all'errore sull'angolo, poichè è l'errore al quale il sistema è più sensibile.

I casi di test

Gli esperimenti sono stati effettuati in un ambiente di $80m^2$ ($10m \times 8m$) con una copertura di 12 robot in una formazione a griglia 4×3, come mostrato in Figura 6.10(e). Sono stati effettuati ulteriori esperimenti incrementando il numero dei robot nello stesso ambiente. In tutti gli esperimenti è mantenuta la griglia di base 4×3, in modo da mantenere una copertura dell'intera area, successivamente sono inseriti i robot aggiuntivi posizionandoli in modo casuale all'interno dell'ambiente. Gli incrementi graduali di eye-bot sono del 50% e del 100% dei robot iniziali; in questo modo si ottengono scenari con rispettivamente 18 e 24 robot. Un esempio di un ambiente con il progressivo aumento di eye-bot è mostrato in Figura 6.10.

Sono stati utilizzati 10 differenti scenari, tutti delle dimensioni di $80m^2$, e con la stessa disposizione di base dei robot. Gli scenari differiscono fra loro per la forma dell'oggetto mobile e per la disposizione degli ostacoli, i quali coprono il 17% dell'area totale.

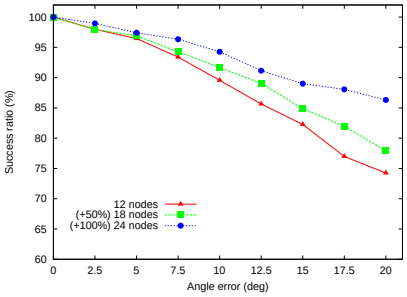
Il livello di errore selezionato corrisponde alla deviazione standard di una distribuzione Gaussiana a media nulla. Per questo motivo l'errore prodotto ad ogni esecuzione è stocastico e produce risultati differenti. Al fine di ottenere un andamento medio, ogni scenario, per ogni livello di errore, è stato eseguito 40 volte.

In conseguenza, ogni punto dei grafici di Figura 6.4 corrisponde al valore medio dei risultati di 400 esperimenti.

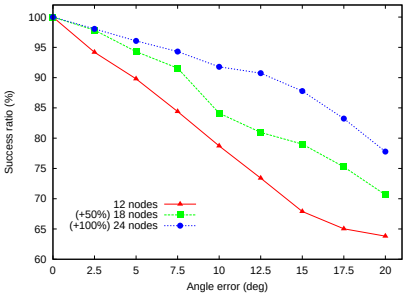
Risultati

Con un approccio distribuito l'imprecisione nella diffusione del campo di potenziale cresce con il numero di nodi, in conseguenza all'effetto della combinazione di skeleton parziali precedentemente descritto nella sezione 5.2.1. Il grafico della **qualità di percorso** (6.10(c)), in corrispondenza di un errore nullo, mostra come l'aumento del numero di robot influenzi negativamente la qualità dei risultati dei pianificatori; l'effetto è più visibile per il pianificatore senza euristiche. I valori sull'asse delle ordinate indicano l'incremento percentuale della lunghezza del percorso con riferimento al percorso calcolato dal pianificatore centralizzato con conoscenza perfetta e globale. Il grafico mostra come, in assenza di errore di allineamento, nel caso base i risultati del pianificatore senza euristiche presentino un incremento del 27% rispetto al caso di riferimento, mentre al crescere del numero di robot aumenti anche la lunghezza del percorso: +30% per il caso con 18 robot, e +32% per quello con 24. Un incremento della lunghezza del percorso è subito anche dal pianificatore con le *Basic Heuristics*, anche se in maniera minore. La causa di questo incremento, anche in assenza di errore, è dovuta alla creazione di skeleton parziali differenti dallo skeleton globale, soprattutto nelle zone di frontiera con gli altri robot. Spesso, questo fenomeno si traduce in un lieve incremento degli spostamenti dell'oggetto in prossimità del passaggio da un campo visivo all'altro. Quando il numero di robot aumenta, anche il numero di passaggi fra di essi aumenta. Una qualità inferiore nel percorso è mantenuta anche con l'incremento dell'errore, il quale accentua i problemi di passaggio dell'oggetto da un robot all'altro.

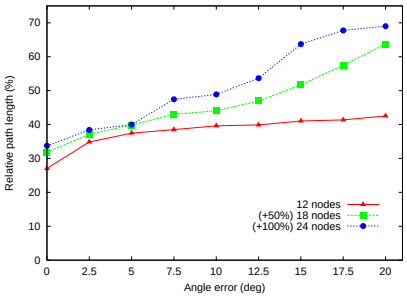
Tuttavia, l'effetto negativo sulla qualità del percorso causato da una ridondanza di robot è controbilanciato da evidenti aspetti positivi. Entrambi i pia-



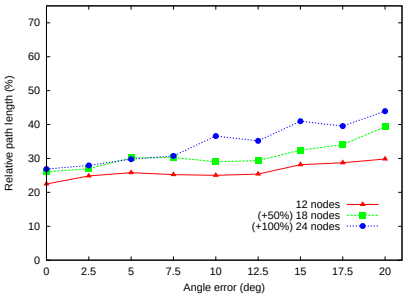
(a) **Pianificatore senza euristiche**
Success ratio vs. errore dell'angolo



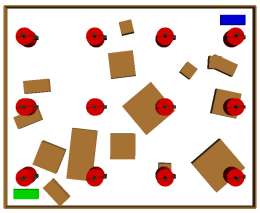
(b) ***Basic Heuristics***
Success ratio vs. errore dell'angolo



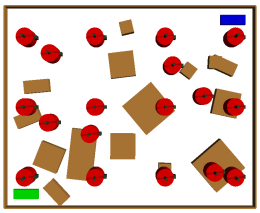
(c) **Pianificatore senza euristiche**
Qualità vs. errore dell'angolo



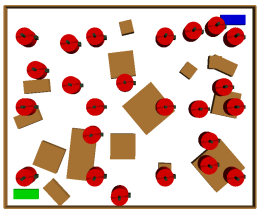
(d) ***Basic Heuristics***
Qualità vs. errore dell'angolo



(e) Griglia 4x3 (12 robot)



(f) Aumento nodi del 50% (18 robot).



(g) Aumento nodi del 100% (24 robot)

Figura 6.10: Incremento della densità dei robot nella mappa. È stata mantenuta la dimensione fissa della mappa, e sono stati aggiunti gradualmente robot nell'ambiente, analizzando il suo effetto sulle prestazioni del pianificatore senza euristiche.

nificatori si dimostrano scalabili, migliorando la propria efficacia all'aumentare delle risorse a disposizione. Nel grafico 6.10(a) è mostrata la **percentuale di successo** degli esperimenti in relazione all'errore di allineamento. Al crescere dell'errore il tasso di successo diminuisce; tuttavia gli scenari con un numero maggiore di robot sono in grado di terminare positivamente più spesso del caso base. Gli errori di allineamento ostacolano il passaggio dell'oggetto fra robot, rendendo così inaccessibili alcune parti della mappa; un numero maggiore di robot nella stessa area aumenta la possibilità di individuare un robot alternativo che riesca a ricevere l'oggetto e quindi proseguire il percorso.

Il pianificatore con le *Basic Heuristics* risulta beneficiare maggiormente dell'incremento di risorse, con un deterioramento più limitato della qualità a fronte di un notevole aumento della percentuale di successo e quindi dell'efficacia del sistema.

Un'ulteriore considerazione, a valle dei risultati ottenuti, riguarda la **relazione fra la qualità del percorso e la percentuale di successo**. Se un pianificatore è in grado di risolvere un maggior numero di scenari, significa che è stato capace di risolvere anche gli scenari più difficili, nei quali gli altri hanno fallito. È plausibile che scenari più difficili richiedano traiettorie più lunghe, pertanto i pianificatori che hanno un tasso maggiore di successo hanno anche una qualità media inferiore rispetto ai pianificatori in grado di risolvere solamente gli scenari più semplici.

Capitolo 7

Conclusioni

L'obiettivo di questo lavoro di tesi è stata la creazione di un pianificatore di percorso distribuito, per il movimento a terra di un oggetto olonomico di forma arbitraria, partendo da una posizione iniziale ad una destinazione finale in un ambiente vincolato da ostacoli. Il sistema doveva essere in grado di operare in ambienti di grandi dimensioni senza una conoscenza pregressa della mappa, perciò è stato necessario acquisire informazioni sull'ambiente tramite l'utilizzo di una molteplicità di dispositivi sensoriali distribuiti. Per ottenere un sistema scalabile (nelle dimensioni dell'ambiente) e dinamico (cioè capace di adattarsi automaticamente a nuovi scenari) è stato studiato un approccio distribuito al problema. Come modello di riferimento per i nodi sensoriali sono stati adottati gli *eye-bot*, piccoli robot volanti in grado di attaccarsi al soffitto equipaggiati di sistemi di visione, di rilevamento delle posizioni relative e di comunicazione wireless.

L'obiettivo generale era di avere un sistema in grado di operare in modo scalabile, dinamico, efficiente dal punto di vista dell'utilizzo delle risorse, e robusto ad errori di allineamento e calibrazione. Pertanto, le scelte progettuali sono state effettuate con particolare attenzione ad un limitato utilizzo delle risorse, soprattutto comunicative.

È stato creato un **primo pianificatore** in grado di calcolare il percorso di oggetti di qualsiasi forma, con il perimetro composto da una sequenza di vertici uniti da linee rette. Il processo di pianificazione individua percorsi composti da una sequenza di rotazioni e traslazioni nell'ambiente, le quali permettono all'oggetto di superare curve e passaggi relativamente stretti, evitando collisioni con ostacoli.

L'architettura proposta è esplicitamente strutturata senza alcun supervisore globale, ma al contrario il sistema è composto da un insieme di entità semplici con visione dell'ambiente dall'alto e conoscenza locale. Tali entità possono raggiungere l'obiettivo finale solamente grazie alla collaborazione ed al coordinamento, in questo modo si creano sinergie in grado di incrementare le capacità delle singole entità. Questo aspetto del processo è la caratteristica fondamentale di un sistema distribuito, ed in particolare di uno sciame di robot. Le comunicazioni sono limitate tra nodi vicini, al fine di migliorare l'efficienza del sistema, e quindi ottenere migliori risultati di scalabilità.

Quando lo scenario non presenta **minimi locali**, l'architettura di pianificazione sviluppata presenta un'esecuzione efficace ed efficiente con un utilizzo delle

risorse molto limitato e risultati molto simili al percorso di riferimento generato dal pianificatore centralizzato. Altri scenari, invece, possiedono ambienti più complessi, in cui il percorso più breve presenta corridoi stretti fra ostacoli che non permettono l'effettivo passaggio dell'oggetto, ciò genera minimi locali nel sistema. L'architettura proposta, in questo tipo di scenari, soffre di una serie di problemi che si traducono in un utilizzo eccessivo di risorse, e in risultati con scarsa qualità. Al fine di fronteggiare tali aspetti critici sono state progettate alcune euristiche in grado di risolvere alcuni dei limiti del sistema.

L'introduzione delle **euristiche** apporta al sistema un miglioramento delle prestazioni, soprattutto nei casi affetti da minimi locali in cui il sistema soffriva di scarsi risultati. Tuttavia uno svantaggio dell'utilizzo di questi metodi euristici è la riduzione del tasso di successo. In alcuni casi, questo approccio può portare il processo al fallimento anche quando è presente un'effettiva soluzione. Si tratta di una conseguenza intrinseca dell'approccio euristico ai problemi, il quale migliora gli aspetti per cui l'euristica è stata progettata, spesso velocizzando il processo, ma allo stesso tempo riduce lo spazio delle soluzioni poichè il suo funzionamento è basato su assunzioni che semplificano il processo.

Sono state progettate e sviluppate quattro diverse euristiche, ognuna indipendente dalle altre e con differenti obiettivi.

Infine, è stato creato un insieme di esperimenti con lo scopo di misurare e confrontare le performance delle differenti architetture proposte. I test effettuati variano le caratteristiche degli scenari, tra cui la dimensione dell'ambiente, la disposizione ed il numero di robot, la disposizione degli ostacoli nella mappa, e la forma dell'oggetto mobile, al fine di studiare l'efficacia, l'efficienza, la scalabilità e la robustezza del sistema al crescere dell' errore di allineamento.

Dai risultati degli esperimenti effettuati non esiste un sistema che si dimostri significativamente migliore degli altri sotto tutti gli aspetti studiati. Ogni pianificatore presenta vantaggi e svantaggi relativi al suo specifico funzionamento.

Come atteso, lo studio di **robustezza** mostra una diminuzione delle prestazioni del sistema al crescere dell'errore di allineamento. Tuttavia, il sistema mantiene risultati accettabili anche per valori di errore relativamente elevati, pertanto soddisfa con successo gli obiettivi di robustezza prefissati. Inoltre, come è stato sottolineato più volte, in questo progetto non sono state studiate tecniche specifiche al tipo di sensori per la riduzione dell'errore di allineamento. Il tipo e la grandezza dell'errore sono fortemente dipendenti dalla tecnologia dei sensori e degli attuatori con cui i robot saranno equipaggiati, in conseguenza anche le relative tecniche per la riduzione dell'errore cambieranno in relazione alla tecnologia dei dispositivi sensoriali.

Alcuni sistemi, come il pianificatore senza euristiche e quello con la combinazione delle euristiche *Blocked Cell List* e *Smart Loop Avoidance* (chiamata *Basic Heuristics*), si dimostrano **scalabili** sia al crescere della dimensione della mappa, sia all'aumentare della densità di robot in un ambiente limitato. Nel primo caso, i sistemi proposti sono in grado di operare in un ambiente con dimensioni maggiori a fronte di un incremento proporzionale delle risorse (cioè dei robot). A tale incremento il sistema risponde mantenendo costante efficacia ed efficienza globale. Nel secondo caso, un incremento della densità di robot nell'ambiente porta il sistema a migliorare la propria efficacia, dimostrando una scalabilità di carico dei pianificatori.

Tutti i sistemi proposti sono in grado di rispondere efficacemente ad una

situazione di disposizione dinamica nell'ambiente dei robot, i quali sono in grado di coordinarsi in modo automatico, rispondendo agli obiettivi di dinamicità.

La scelta finale di quale sistema adottare dipenderà dai vincoli progettuali del sistema reale in cui verrà utilizzato il pianificatore.

In conclusione, con questo lavoro è stato investigato un nuovo approccio al problema di pianificazione di percorso per oggetti olonomici in un piano bidimensionale. In particolare, sono stati proposti alcuni innovativi prototipi di architetture di pianificazione cooperativa totalmente distribuita, e ne sono stati analizzati vantaggi e svantaggi. Questa vasta direzione di studio è stata poco investigata in letteratura, perciò le soluzioni proposte si possono considerare un primo passo di studio verso un lavoro di indagine più ampio ed approfondito.

Questo lavoro ha portato alla pubblicazione dell'articolo *"A distributed approach to holonomic path planning"* (Allegato A) al workshop *"Motion Planning: From Theory to Practice"* del 27 giugno 2010, durante la conferenza internazionale *Robotics: Science and Systems (RSS)* a Saragozza, Spagna. Durante il workshop è stato presentato un poster relativo alla pubblicazione (Allegato B).

7.1 Lavori Futuri

Questo progetto ha riguardato lo studio di un aspetto poco trattato in letteratura, ed in alcune parti perfino completamente nuovo. Si tratta del primo passo verso la creazione di un sistema più completo e robusto, pertanto presenta notevoli direzioni di sviluppo, che toccano aspetti molto differenti fra loro.

Un primo miglioramento del sistema può riguardare la creazione di metaeuristiche per la **riduzione dell'errore di allineamento** tra nodi vicini. Sfruttando la ridondanza di nodi, e combinando le informazioni relative ai vicini in comune con i propri vicini, è possibile effettuare triangolazioni in grado di ridurre notevolmente la misura dell'errore di allineamento, e quindi ottenere prestazioni elevate.

Una naturale prosecuzione del lavoro è la sperimentazione del pianificatore proposto sui **robot reali**. Un interessante caso di utilizzo può riguardare l'implementazione della seconda parte del progetto, presentata nella sottosezione 2.1.2, in cui l'effettivo spostamento e trasporto dell'oggetto avviene da parte di un insieme di robot a terra. Questa estensione affronta i problemi relativi alla conversione del percorso in istruzioni di navigazione da inviare ai robot a terra, ed al coordinamento di quest'ultimi al fine di apprendere in che modo muovere l'oggetto per seguire il percorso pianificato.

In questa situazione, è possibile che, durante la fase di effettiva attuazione del percorso, la navigazione e gli spostamenti dell'oggetto mobile siano soggetti ad errori, ciò si traduce in **deviazioni dalla traiettoria pianificata**. Inoltre, l'attuazione degli spostamenti nell'ambiente reale può rivelare errori della pianificazione, in conseguenza a movimenti verso curve o passaggi troppo stretti che non permettono la continuazione del percorso. Uno sviluppo interessante può riguardare la creazione di tecniche in grado di reagire a queste situazioni, ricalcolando la traiettoria, o modificandola, in modo da consentire all'intero sistema di raggiungere comunque la destinazione finale.

Una funzionalità aggiuntiva, che può essere oggetto di interessanti studi, è il calcolo parallelo dello stesso percorso con differenti **soglie di *"sicurezza"***. A percorsi con passaggi stretti verranno assegnati valori minori, mentre traiet-

torie in strade più larghe, anche se più lunghe, hanno valori di *"sicurezza"* di successo maggiore. L'agente incaricato di eseguire il percorso pianificato potrà così scegliere tra diversi livelli di sicurezza in base alla priorità, o al costo, dell'operazione.

Spesso, gli ambienti del mondo reale sono composti anche da **ostacoli dinamici**, come persone, o altri robot, che si muovono nello spazio. Pertanto, con un estensione del pianificatore per operare anche in ambienti con ostacoli dinamici, si incrementerebbero notevolmente i possibili casi d'utilizzo del pianificatore.

Un'ulteriore estensione del sistema consiste in un architettura in grado di effettuare la **pianificazione parallela di molteplici percorsi** di differenti oggetti/robot. Questo problema affronterebbe le tematiche di percorsi concorrenti, con le relative tecniche di sincronizzazione dei percorsi per evitare collisioni fra oggetti e robot mobili. Si tratta di una fattispecie di notevole utilità in scenari popolati da sciami di robot, poichè una comune soluzione ai problemi consiste nella suddivisione dei compiti fra gruppi di robot, ogni gruppo di lavoro opera in parallelo. In questo caso lo sciame a terra verrebbe coordinato dallo sciame che supervisiona dall'alto.

Bibliografia

[1] Swarmanoid project. `http://www.swarmanoid.org`, from October 1, 2006 to September 30, 2010.

[2] Jerome Barraquand, Bruno Langlois, and Jean Claude Latombe. Numerical potential field techniques for robot path planning. *Man and Cybernetics, IEEE Transactions on*, 22(2):224–241, Mar/Apr 1992.

[3] Andreas Birk and Stefano Carpin. Merging occupancy grid maps from multiple robots. *IEEE Proceedings, special issue on Multi-Robot Systems*, 94(7):1384–1397, 2006.

[4] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A. Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations.* MIT Press, Cambridge, MA, June 2005.

[5] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localisation and mapping (slam): The essential algorithms. *Robotics and Automation, IEEE International Conference on*, 2, 2006.

[6] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, 94(7):1325–1339, July 2006.

[7] Santiago Garrido, Luis Moreno, and Dolores Blanco. Voronoi diagram and fast marching applied to path planning. *Robotics and Automation, IEEE International Conference on*, pages 3049–3054, May 2006.

[8] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, July 1968.

[9] Jean Claude Latombe. A fast path planner for a car-like indoor mobile robot. In *In Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 659–665, 1991.

[10] Jean Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[11] Jean Claude Latombe. Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *International Journal of Robotics Research*, 18:1119–1128, 1999.

[12] J. J. Leonard and H. F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*, pages 1442–1447 vol.3, 1991.

[13] Swarmanoid project. Swarmanoid simulation. `http://www.swarmanoid.org/swarmanoid_simulation.php`, 2009.

[14] Swarmanoid project. Swarmanoid hardware. `http://www.swarmanoid.org/swarmanoid_hardware.php`, 2010.

[15] J. F. Roberts, J. C. Zufferey, and D. Floreano. Energy management for indoor hovering robots. In *Proceedings of the International Conference on Intelligent Robots and Systems, IROS '08*, pages 1242–1247, Piscataway, September 22–26, 2008. IEEE Press.

[16] James Roberts, Timothy Stirling, Jean-Christophe Zufferey, and Dario Floreano. Quadrotor using minimal sensing for autonomous indoor flight. In *European Micro Air Vehicle Conference and Flight Competition, EMAV '07*, September17–21 2007.

[17] James F. Roberts, Timothy S. Stirling, Jean-Christophe Zufferey, and Dario Floreano. 2.5D infrared range and bearing system for collective robotics. In *Proceedings of the International Conference on Intelligent Robots and Systems, IROS '09*, pages 3659–3664, Piscataway, 2009. IEEE Press.

[18] Yi Shang and W. Ruml. Improved MDS-based localization. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2640–2651 vol.4, 2004.

[19] T. Stirling, S. Wischmann, and D. Floreano. Energy-efficient indoor search by swarms of simulated flying robots without global information. *Swarm Intelligence*, 4(2):117–143, February 2010.

[20] Osamu Takahashi and R. J. Shilling. Motion planning in a plane using generalized voronoi diagrams. *Robotics and Automation, IEEE Transactions on*, 5(2):143–150, 1989.

[21] Atsushi Yamashita, Tamio Arai, Jun Ota, and Hajime Asama. Motion planning of multiple mobile robots for cooperative manipulation and transportation. *IEEE Transactions on Robotics and Automation*, 19:223–237, 2003.

Allegati

Allegato A

Articolo *"A distributed approach to holonomic path planning"* pubblicato al workshop *"Motion Planning: From Theory to Practice"* del 27 giugno 2010, a Saragozza (Spagna), durante la conferenza internazionale *Robotics: Science and Systems (RSS)*.

A distributed approach to holonomic path planning

Andreagiovanni Reina
Dept. of Electronics and Information, Politecnico di Milano
Piazza L. da Vinci, 32 - 20133 Milano, Italy
andreagiovanni.reina@mail.polimi.it

Gianni A. Di Caro, Frederick Ducatelle, Luca Gambardella
Dalle Molle Institute for Artificial Intelligence Studies (IDSIA)
Galleria 2, 6928 Manno - Lugano, Switzerland
{gianni, frederick, luca}@idsia.ch

I. Introduction

We study a distributed approach to the path planning problem. We focus on holonomic kinematic motion in a plane with static obstacles. The problem consists in planning the path of a rigid object that has to be transported from an initial to a final location through a constrained path [4, 2]. The planner observes the environment from above through a visual system. We consider the case in which the path covers a large area, such that the planner architecture consists of a wireless network of observer nodes which each can see a portion of the area. A centralized solution is neither robust nor scalable. To overcome these difficulties we propose a fully distributed approach: each observer node locally calculates the part of the path relative to the area that it sees, and communicates to its neighbor the information which permits the execution of the planning. Our goal is to calculate effective paths in a way that is scalable, resource efficient , and robust to calibration and alignment errors. As models for the planner nodes, we consider smart cameras equipped with wireless communications and a system for measuring the relative angle and distance between two cameras.

II. Distributed path planning

Our distributed planner is derived from the numerical potential field technique for a single camera planner, which is computed using the *wave front expansion with skeleton* [1] on a bidimensional uniform cell partitioning. This solution first spreads the potential over a subset of the free space, called *skeleton*, which corresponds to the Voronoi diagram; then the potential is computed in the rest of the map. The potential descent is performed using A*[3].

The distributed algorithm has three phases: *neighbor detection, potential field diffusion*, and *path calculation*. During neighbor detection, each node builds a neighbor table, in which the relative positions of nearby nodes are stored with some estimation error. The second phase concerns the potential field diffusion: each node expands the potential on its part of the map, and sends to its neighbors the frontier values. The nodes that can see the goal position begins the process and then each new message triggers an update of the neighbor's values. The system minimizes the communication overhead, transmitting only the information of the skeleton cells near the frontier. Finally, the node above the start position begins the path calculation phase: when the trajectory exits from the area of view, the node sends the object coordinates to its neighbor which continues the process. We assume that the partial map overlaps with the neighbor's map, in order to permit the object position sharing. However, the overlapping is subject to errors deriving from errors in camera calibrations and relative positioning. The process ends when the object reaches the goal configuration (success) or when the exploration tree of configurations is completely expanded (failure).

Given the distributed nature of the approach, the calculated path can present loops. Moreover, attraction towards potential field local minima is possibly amplified. To overcome these limitations, and at the same time optimize resource usage, we propose three heuristics. (i) *Smart Loop Avoidance*: during path calculation a node can detect loops and coordinates with others in order to step back to the loop start situation and avoid it. (ii) *Skeleton Pruning*: pre-pruning of the skeleton during the potential field diffusion to block the passages narrower than the smallest object size. (iii) *Narrow Passage Detection*: during path calculation a node can detect a narrow passage, block it and repeat the potential field diffusion step.

III. Experimental results

Through simulation, we tested our approach in a set of sample scenarios (see Figure 1 for an example). They differ in obstacle structure and object shape. The camera nodes maintain an 4x3 grid formation.

We studied the effect of the relative positioning error between nodes in terms of angle error (Figure 2) and distance error (Figure 3) for the algorithm with and without the heuristics. We report the percentage of successful runs and the relative length of the path compared to a reference centralized approach. The results show that for relatively low errors the performance is always very close to the reference. For increasing errors, the performance of the algorithm with heuristics degrades rapidly in terms of success rate but slowly for path quality. The algorithm without heuristics behaves in opposite way. In both cases, the system is relatively sensitive to errors on the angle, while it is quite robust to distance errors.

In a distributed approach the imprecision in potential diffusion necessarily increases with the number of nodes. To study how this impacts the performance, in Figure 4 we show the effect of increasing the density of the nodes over a fixed area varying angle errors. The plot of the path quality in correspondence of zero angle error clearly shows how the increase in node number negatively affects performance. However, node redundancy can at the same time counterbalance the increase in the relative positioning error, as is well evidenced by the success rate plot.

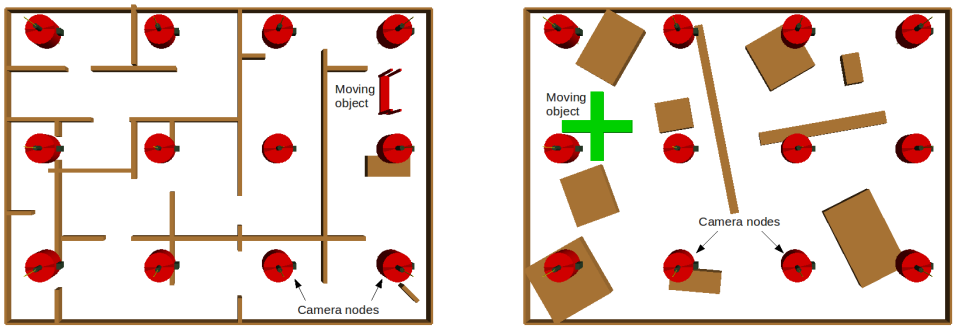


Fig. 1: Examples of the scenarios used for the experiments.

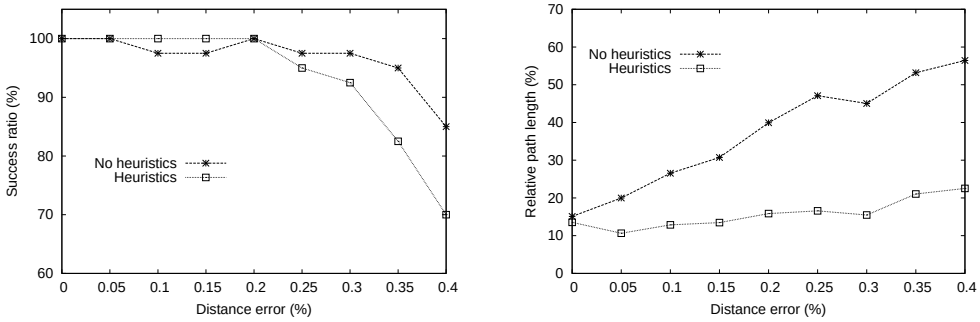


Fig. 2: Performance evaluation vs. increasing percentage error in distance estimation.

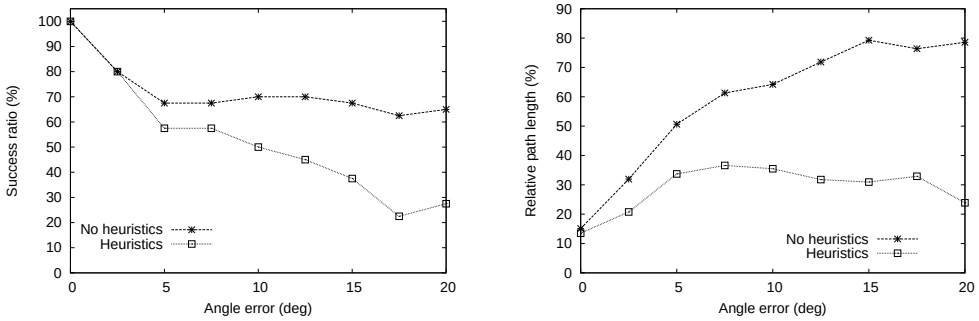


Fig. 3: Performance evaluation vs. increasing error in angle estimation.

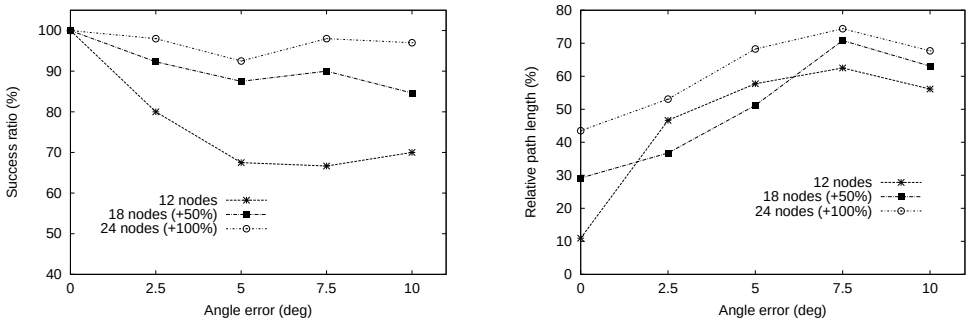


Fig. 4: Effect of node density on performance vs. increasing error in angle estimation.

REFERENCES

[1] Jerome Barraquand, Bruno Langlois, and Jean-Claude Latombe. Numerical potential field techniques for robot path planning. *Man and Cybernetics, IEEE Transactions on*, 22(2):224–241, Mar/Apr 1992.

[2] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A. Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.

[3] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, July 1968.

[4] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

Allegato B

Poster *"A distributed approach to holonomic path planning"* presentato al workshop *"Motion Planning: From Theory to Practice"* il 27 giugno 2010 a Saragozza (Spagna), durante la conferenza internazionale *Robotics: Science and Systems (RSS)*.

A distributed approach to holonomic path planning

Andreagiovanni Reina[+], Gianni A. Di Caro[*], Frederick Ducatelle[*] and Luca M. Gambardella[*]

[+]Department of Electronics and Information (DEI), Politecnico di Milano
Piazza L. da Vinci, 32 - 20133 Milano, Italy
andreagiovanni.reina@mail.polimi.it

[*]Dalle Molle Institute for Artificial Intelligence Studies (IDSIA)
Galleria 2, 6928 Manno - Lugano, Switzerland
{gianni, frederick, luca}@idsia.ch

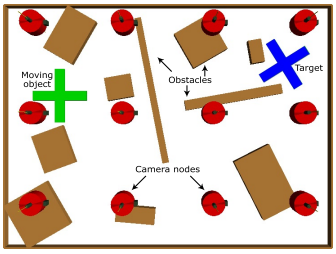
Abstract

We study a distributed approach to the path planning problem. We focus on holonomic kinematic motion in a plane with static obstacles. The problem consists in planning the path of a rigid object of arbitrary shape that has to be transported from an initial to a final location through a constrained path. The planner observes the environment from above through a visual system. We consider the case in which the path covers a large area, such that the planner architecture consists of a wireless network of observer nodes which each can see a portion of the area. A centralized solution is neither robust nor scalable. To overcome these difficulties we propose a fully distributed approach: each observer node locally calculates the part of the path relative to the area that it sees, and communicates to its neighbors the information which permits the cooperative execution of the planning. Our goal is to calculate effective paths in a way that is scalable, resource efficient, and robust to calibration and alignment errors.

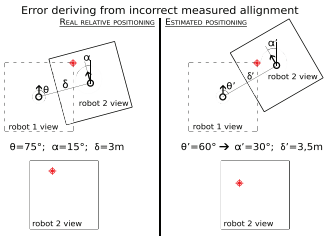
1 The robot model

As reference models for the planner nodes, we consider the *eye-bot* robots, that are being developed in the Swarmanoid project (*http://www.swarmanoid.org*). These are small flying robots that can attach to the ceiling, are equipped with a video camera pointing to the floor, and have an infrared system for measuring the relative bearing and distance between two robots and for wireless communications.

2 Problem description

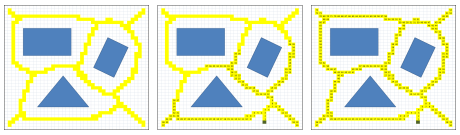


Eye-bots attach to the ceiling and cover the entire area between the start and the goal. Each robot has a limited vision of the environment and we assume that the partial view of neighbor robots overlaps, in order to permit the sharing of the rigid object position. However, the overlapping is subject to errors deriving from errors in camera calibrations and in the measure of robots' relative positioning.

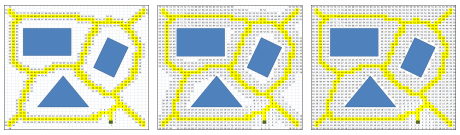


3 Path planning

Our distributed planner is derived from the numerical potential field technique for a single camera planner (Latombe et al., 1991; Choset et al., 2005) which is computed using the *wave front expansion with skeleton* on a bidimensional uniform cell partitioning. This solution first spreads the potential over a subset of the free space, called *skeleton*, which corresponds to the *Voronoi diagram*; then the potential is computed in the rest of the map. The potential descent is performed using A*.



Potential field expansion over the skeleton.



Diffusion of the potential field over the remaining free space.

4 Distributed path planning

The distributed algorithm has three phases:

1. ***Neighbor detection***. Each robot builds a neighbor table, in which the relative positions of nearby nodes are stored with some estimation error.
2. ***Potential field diffusion***. Each robot expands the potential field on its part of the map, and sends to its neighbors the frontier values. The system minimizes the communication overhead, transmitting only the information of the skeleton cells near the frontier.
3. ***Path calculation***. The robot above the start position begins path calculation. When the trajectory exits from its area of view, the robot sends the object coordinates to a selected neighbor. Then the process iterates from robot to robot until the target position is reached.

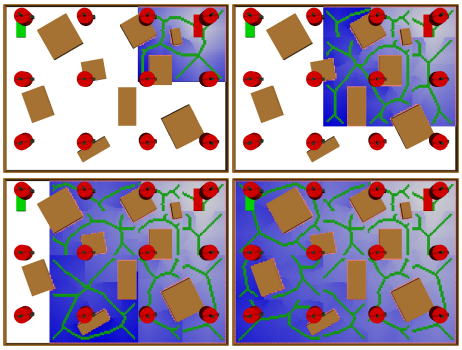
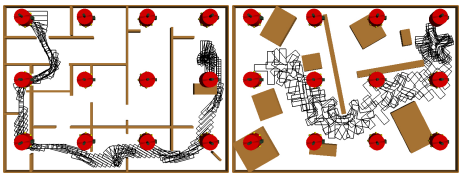


Illustration of the potential field diffusion phase.



Examples of path trajectories.

4.1 Heuristics to escape local minima and loops

Given the distributed nature of the approach, the same local minimum in the potential field can negatively affect the path calculation phase of multiple robots. Moreover, the calculated path can present loops due to distributed sub-path composition. To overcome these issues, which can cause an overhead in computational and communication resource usage, we propose three heuristics:

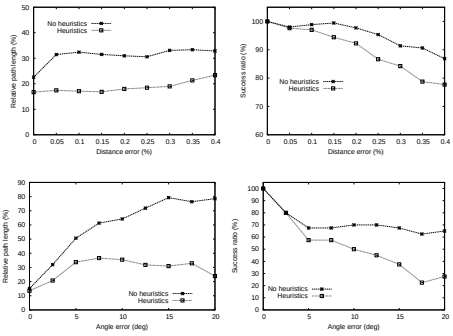
- *Smart Loop Avoidance*: during path calculation a node can detect loops. In this case, it coordinates with all involved nodes to retrace the search back to the beginning of the loop.
- *Skeleton Pruning*: pre-pruning of the skeleton during the potential field diffusion, to block passages that would not let the object passing through.
- *Narrow Passage Detection*: during path calculation a node can detect a narrow passage, block it, and then repeat the potential field diffusion step.

5 Experimental results

Through simulation, we tested our approach in a set of 20 sample scenarios differing in obstacle structure and object shape.

5.1 Effect of heuristics vs. position errors

We studied the effect of the relative positioning error between nodes in terms of angle and distance error for the algorithm with and without the heuristics. The results for different error values are shown in the plots below. In all the experiments the camera nodes maintain a 4x3 grid formation.



Performance vs. errors. The error values on the x-axis indicate the standard deviation of a zero mean Gaussian distribution used to sample the distance/angle error between the robot pairs. For each scenario we ran 40 trials. Each data point represents the average of 20x40 experiments.

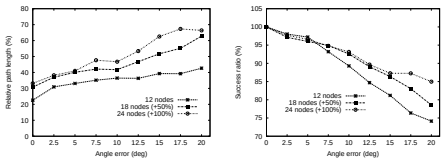
For relatively low errors the performance is always very close to the reference. While for increasing errors:

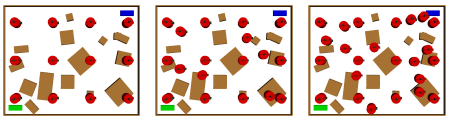
- The algorithm with heuristics degrades rapidly in terms of success rate but slowly for path quality.
- The algorithm without heuristics behaves in opposite way.

In both cases, the system is relatively sensitive to errors on the angle, while it is quite robust to distance errors.

5.2 Scalability performances

In a distributed approach the imprecision in potential diffusion necessarily increases with the number of nodes. We study the effect on performance of increasing the density of the nodes over a fixed area varying angle errors.





12 robots | 18 robots [+50%] | 24 robots [+100%]

The increase in node number negatively affects performance. However, node redundancy can at the same time counterbalance the increase in the relative positioning error, as is well evidenced by the success rate plot.

References

Latombe, Jean-Claude. *Robot Motion Planning*. Kluwer Academic Publishers. Norwell, MA, USA, 1991.

H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, 2005.

Acknowledgments This work has been partially supported by the *Zeno Karl Schindler Foundation* (Genève, Switzerland), and by the *SWARMANOID* FET project, funded by the European Commission.