Predictive Autonomous

Robot Navigation

Predictive Autonomous Robot Navigation

Amalia Foka

Ph. D. Thesis, Department of Computer Science

University of Crete, May 2005

Predictive Autonomous Robot Navigation

Amalia Foka

A thesis submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy in Computer Science

University of Crete

2005

Αυτόνομη Πλοήγηση Ρομπότ με Τεχνικές Πρόβλεψης

Αμαλία Φωκά

Διατριβή για την μερική εκπλήρωση

των απαιτήσεων για το πτυχίο του

Διδάκτωρ στην Επιστήμη Υπολογιστών

Πανεπιστήμιο Κρήτης

2005

UNIVERSITY OF CRETE
DEPARTMENT OF COMPUTER SCIENCE
Predictive Autonomous Robot Navigation
thesis submited by
Amalia Foka
in partial fulfillment of the requirements for
the PhD degree in Computer Science

Author:

Amalia Foka, Department of Computer Science

Heraklion, May, 2005

Acknowledgements

First of all, I would like to thank my supervisor Prof. Panos Trahanias for his guidance, support and encouragement. Throughout all my years at the lab and during this thesis work he has always been a reference of enthusiasm and dedication.

I would also like to express my gratitude to the members of my doctoral committee Assoc. Prof. Panos Tsakalides and Assoc. Prof. Yannis Stylianou. Prof. Stelios Orphanoudakis has been a member of my doctoral committee till the near end of my thesis work. His sudden death forced his replacement in the doctoral committee. Still, his advices and ideas have greatly contributed to my thesis, a fact that is highly appreciated. Many thanks also to Prof. Nikos Koussoulas, Prof. George Stavrakakis, Assoc. Prof. Kostas Kyriakopoulos and Assist. Prof. Nikos Vlassis for travelling to Heraklion to participate to the committee for my thesis defence.

Both the Foundation for Research and Technology–Hellas (FORTH) and the University of Crete, where this thesis was conducted, have provided a wealth of academic stimuli as well as a pleasant and creative environment for which I am heartily thankful. FORTH also provided financial support which is gratefully acknowledged as well.

I would also like to thank my colleges at the Computational Vision and Robotics Laboratory - FORTH for the friendly atmosphere and the useful discussions that have greatly influenced this work.

Special thanks go to all my friends and staff of the ICS-FORTH that participated as human motion models in the experiments I have conducted.

Last but not least, thanks to my family and my friends for their constant support and infinite patience.

Abstract

A primary goal in robotics research is to provide means for mobile platforms to perform autonomously within their environment. Depending on the task at hand, autonomous performance can be defined as the execution by the robot, without human intervention, of certain navigational tasks. In mobile robotics literature, commonly addressed navigation tasks include the localization, mapping, path planning and obstacle avoidance tasks. Solving any of these tasks is a hard problem by itself. The reason stems from the inherent complexity associated with both the robot and its environment, each of them being an extremely complex dynamical system with uncertainty involved.

In this thesis, we propose a probabilistic framework for mobile robot navigation in dynamic environments based on the Partially Observable Markov Decision Process (POMDP) model. The proposed model is able to perform the navigation tasks of localization, path planning and obstacle avoidance. POMDPs are models for sequential decision making where the world in which the robot operates is partially observable, i.e. the true underlying robot's state is not known, and the outcome of actions it executes is modelled probabilistically. As such POMDPs perform localization and path planning in a probabilistic manner.

POMDPs have the major shortcoming of their extreme computational complexity and hence they have been mainly used in robotics as high level path planners only. In this thesis, we propose a novel hierarchical representation of POMDPs, specifically designed for the autonomous

robot navigation problem and termed as the Robot Navigation-Hierarchical POMDP (RN-HPOMDP). The proposed hierarchical POMDP can efficiently model large real-world environments and is amenable to real time solution. This is achieved mainly due to the design choice of modelling the state transition and observation functions dependent only on the robot motion model and not on the environment as it is commonly used in the POMDP literature. Furthermore, the notion of the *reference* POMDP (rPOMDP) is introduced that infers the robot motion model in a very small POMDP and it transfers this information to the hierarchical structure while being solved. The environment specific information is modelled within the reward function of the RN-HPOMDP. The employed model is utilized as a unified probabilistic navigation framework that accommodates for localization, path planning and obstacle avoidance. Hence, real-time solution of the RN-HPOMDP is essential since no other external modules are utilized and paths have to be replanned at each time step.

The RN-HPOMDP has been developed for the application of robot navigation in dynamic real-world environments that are highly populated. Thus, it is desirable for the robot to perform obstacle avoidance in a manner that resembles the human motion for obstacle avoidance. That is, the robot should be able to decide the most suitable obstacle avoidance behavior based on the state of the environment. Therefore, the robot can decide to either perform a detour or follow a completely new path to the goal and also modify its speed of movement (increase it or decrease it) to bypass an obstacle or let it move away respectively. Any of the above four distinct behaviors for obstacle avoidance should be decided well before the robot comes too close to the obstacle. For that reason, future motion prediction of obstacles is employed. Two kinds of prediction are utilized: short-term and long-term prediction. Short term prediction refers to the one-step ahead prediction whereas long-term prediction refers to the prediction of the final destination point of the obstacle's movement. Both kinds of prediction are integrated into the reward function of the RN-HPOMDP and the speed decision is performed through a modified solution of the RN-HPOMDP. As a result, the RN-HPOMDP can decide the optimal obstacle avoidance behavior based on the current and the predicted state of the environment without the intervention of any other external module.

Experimental results have shown the applicability and effectiveness of the proposed framework for the navigation task. The robustness and the probabilistic nature of the RN-HPOMDP as well as the future motion prediction are required to be able to perform efficiently and effectively in dynamic real-world environments that are highly populated.

Περίληψη

Ένας σημαντικός ερευνητικός στόχος στον τομέα της ρομποτικής είναι η ανάπτυξη ρομπότ που έχουν την δυνατότητα να λειτουργούν αυτόνομα. Η αυτόνομη συμπεριφορά ορίζεται σε ορισμένες περιπτώσεις ως η λειτουργία του ρομπότ για την εκτέλεση διαφόρων ενεργειών πλοήγησης χωρίς την ανθρώπινη παρέμβαση. Στη βιβλιογραφία των αυτόνομων κινούμενων ρομπότ αναφέρονται ως συνηθισμένες λειτουργίες πλοήγησης η εύρεση της θέσης του ρομπότ (localization), η χαρτογράφηση του περιβάλλοντος χώρου (mapping), η σχεδίαση διαδρομών μέσα στο χώρο προς αυθαίρετα ορισμένες θέσεις/στόχους (path planning) και η αποφυγή εμποδίων (obstacle avoidance). Η λύση κάθε λειτουργίας πλοήγησης χωριστά είναι ένα δύσκολο πρόβλημα από μόνο του. Αυτό οφείλεται στην έμφυτη πολυπλοκότητα τόσο του χώρου αλλά και του ίδιου του ρομπότ, αφού καθ' ένα από αυτά είναι ένα πολύπλοκο δυναμικό σύστημα που χαρακτηρίζεται από αβεβαιότητα για την κατάστασή του κάθε χρονική στιγμή.

Σε αυτή τη διατριβή προτείνεται ένα πιθανοκρατικό πλαίσιο για την αυτόνομη πλοήγηση ρομπότ βασιζόμενο σε Μερικώς Παρατηρήσιμες Μαρκοβιανές Διαδικασίες Απόφασης - ΜΠΜΔΑ (Partially Observable Markov Decision Processes - POMDPs). Το προτεινόμενο μοντέλο έχει την δυνατότητα να εκτελεί τις λειτουργίες πλοήγησης της εύρεσης της θέσης του ρομπότ, του σχεδιασμού διαδρομών καθώς και αποφυγής εμποδίων. Οι ΜΠΜΔΑ είναι μοντέλα για διαδοχική λήψη αποφάσεων όπου το περιβάλλον στο οποίο λειτουργεί το ρομπότ είναι μερικώς παρατηρήσιμο, δηλαδή η πραγματική κατάσταση του ρομπότ δεν είναι γνωστή. Επιπλέον, το αποτέλεσμα των ενεργειών που εκτελεί το ρομπότ μοντελοποιούνται πιθανοκρατικά. Συνεπώς,

οι λειτουργίες πλοήγησης εκτελούνται με πιθανοκρατικό τρόπο.

Το βασικό μειονέκτημα των ΜΠΜΔΑ είναι η εξαιρετικά μεγάλη υπολογιστική πολυπλοκότητα λύσης τους και για αυτό το λόγο έχουν εφαρμοστεί στην ρομποτική μέχρι σήμερα κυρίως ως μοντέλα για την λήψη αποφάσεων σε υψηλό επίπεδο μόνο. Σε αυτή τη διατριβή προτείνεται μια νέα ιεραρχική αναπαράσταση των ΜΠΜΔΑ ειδικά σχεδιασμένη για το πρόβλημα της αυτόνομης πλοήγησης ρομπότ που αποκαλείται Ιεραρχικές ΜΠΜΔΑ - Ρομποτικής Πλοήγησης (ΙΜΠΜΔΑ-ΡΠ). Οι ΙΜΠΜΔΑ-ΡΠ έχουν τη δυνατότητα να αναπαριστούν με αποδοτικό τρόπο μεγάλα πραγματικά περιβάλλοντα και λύνονται σε πραγματικό χρόνο. Αυτό επιτυγχάνεται κυρίως λόγω της σχεδιαστικής επιλογής να μοντελοποιηθούν οι συναρτήσεις μετάβασης (transition) και παρατηρήσεων (observation) βάση του μοντέλου κίνησης του ρομπότ και ανεξάρτητα από τον περιβάλλον στο οποίο λειτουργεί, όπως γίνεται συνήθως στην βιβλιογραφία των ΜΠΜΔΑ. Επιπλέον, εισάγεται η έννοια της *αναφορικής* ΜΠΜΔΑ (αΜΠΜΔΑ) που διατηρεί το μοντέλο κίνησης του ρομπότ χρησιμοποιώντας μια πολύ μικρή ΜΠΜΔΑ και το μεταφέρει στην ιεραρχική δομή κατά την διάρκεια της λύσης της. Η δομή και κατάσταση του περιβάλλοντος μοντελοποιείται στην συνάρτηση ανταμοιβής (reward) της ΙΜΠΜΔΑ-ΡΠ. Το προτεινόμενο μοντέλο εφαρμόζεται ως ένα ενιαίο πλαίσιο πλοήγησης το οποίο δεν χρησιμοποιεί εξωτερικά συνεργαζόμενες μονάδες. Για την επίτευξη αυτού του στόχου η λύση της ΙΜΠΜΔΑ-ΡΠ σε πραγματικό χρόνο είναι απαραίτητη εφ' όσον πρέπει να σχεδιάζει εκ νέου σε κάθε χρονική στιγμή το μονοπάτι προς τον στόχο και κατά συνέπεια να διεξάγει την λειτουργία της αποφυγής εμποδίων.

Η χρήση της ΙΜΠΜΔΑ-ΡΠ ως ενιαίο μοντέλο πλοήγησης έχει το πλεονέκτημα ότι εκμεταλλεύεται την πιθανοκρατική φύση της σε όλα τα επίπεδα λειτουργίας της πλοήγησης το οποίο είναι εξαιρετικά σημαντικό δεδομένης της αβεβαιότητας που υπάρχει στους αισθητήρες του ρομπότ όπως και στην ίδια του την κίνηση. Αντίθετα, στην μεγαλύτερη πλειοψηφία των μεθόδων πλοήγησης που υπάρχουν στην βιβλιογραφία χρησιμοποιούνται πολλές διαφορετικές μονάδες που εκτελούν κάθε μια λειτουργία πλοήγησης. Σε αυτές τις μεθόδους, συναντάται συχνά πιθανοκρατική λύση του προβλήματος της εύρεσης της θέσης του ρομπότ και του σχεδιασμού μονοπατιών σε υψηλό επίπεδο αλλά σπανίως λύνεται πιθανοκρατικά το πρόβλημα της αποφυγής εμποδίων. Επιπλέον, με την χρήση της ΙΜΠΜΔΑ-ΡΠ το πρόβλημα της αποφυγής εμποδίων λύνεται καθολικά, εφ' όσον ολόκληρο το μονοπάτι προς τον στόχο επανασχεδιάζεται σε κάθε χρονική στιγμή. Οι περισσότερες μέθοδοι αποφυγής εμποδίων που υπάρχουν στην βιβλιογραφία λύνουν το πρόβλημα τοπικά, κυρίως λόγω του μεγάλου υπολογιστικού κόστους λύσης.

Η ΙΜΠΜΔΑ-ΡΠ σχεδιάστηκε για εφαρμογή στο πρόβλημα της πλοήγησης σε δυναμικά πραγματικά περιβάλλοντα στα οποία υπάρχει πυκνή κίνηση ανθρώπων. Έτσι, είναι επιθυμητό το ρομπότ να εκτελεί την αποφυγή εμποδίων με τρόπο παρόμοιο με τον οποίο αποφεύγουν εμπόδια οι ίδιοι οι άνθρωποι. Δηλαδή το ρομπότ θα πρέπει να έχει τη δυνατότητα να αποφασίζει την κατάλληλη συμπεριφορά για αποφυγή εμποδίων δεδομένης της κατάστασης του περιβάλλοντος. Επομένως, το ρομπότ μπορεί να αποφασίσει να εκτελέσει μια παράκαμψη ή να ακολουθήσει ένα εντελώς καινούργιο μονοπάτι προς τον στόχο του και επιπλέον να αυξήσει ή να μειώσει την ταχύτητά του για να προσπεράσει ένα εμπόδιο ή να του επιτρέψει να περάσει, αντίστοιχα. Κάθε μια από τις παραπάνω τέσσερις συμπεριφορές για την αποφυγή εμποδίων θα πρέπει να αποφασιστεί αρκετά πριν φτάσει το ρομπότ κοντά στο εμπόδιο. Γι' αυτό το λόγο, η χρήση πρόβλεψης της κίνησης εμποδίων είναι απαραίτητη. Εφαρμόζονται δύο τύποι πρόβλεψης κίνησης: η βραχυπρόθεσμη και η μακροπρόθεσμη πρόβλεψη κίνησης. Η βραχυπρόθεσμη πρόβλεψη αναφέρεται στην πρόβλεψη της θέσης του εμποδίου την αμέσως επόμενη χρονική στιγμή ενώ η μακροπρόθεσμη αναφέρεται στην πρόβλεψη του τελικού σημείου κίνησης του εμποδίου. Και τα δύο είδη πρόβλεψης ενσωματώνονται στην συνάρτηση ανταμοιβής της ΙΜΠΜΔΑ-ΡΠ. Η μακροπρόθεσμη πρόβλεψη επιτυγχάνεται ορίζοντας τα ενδιαφέροντα σημεία του χώρου (ο ορισμός τους μπορεί να γίνει χειρωνακτικά από τον χρήστη ή μέσω μιας αυτοματοποιημένης διαδικασίας εκμάθησής τους). Στη διαδικασία αυτή θεωρείται ότι οι άνθρωποι δεν κινούνται άσκοπα μέσα σε έναν χώρο αλλά επιθυμούν να κατευθυνθούν προς κάποιο ενδιαφέρον σημείο του χώρου. Συνεπώς, η μακροπρόθεσμη πρόβλεψη κίνησης ορίζεται ως η πρόβλεψη του ενδιαφέροντος σημείου που πρόκειται να προσεγγίσει ένας άνθρωπος ή ένα άλλο κινούμενο εμπόδιο. Και τα δύο είδη πρόβλεψης ανανεώνονται σε κάθε χρονική στιγμή και έτσι πιθανά λάθη μπορούν να διορθωθούν πολύ γρήγορα και να μην επηρεάσουν την απόφαση του ρομπότ για το μονοπάτι το οποίο ακολουθεί για να φτάσει στον στόχο του. Τέλος, η απόφαση για την ταχύτητα με την οποία θα κινηθεί το ρομπότ επιτυγχάνεται με την χρήση μιας τροποποιημένης μεθόδου λύσης της ΙΜΠΜΔΑ-ΡΠ. Συνεπώς, η ΙΜΠΜΔΑ-ΡΠ μπορεί να αποφασίσει την βέλτιστη συμπεριφορά για την αποφυγή εμποδίων βάση της τωρινής και προβλεπόμενης κατάστασης του περιβάλλοντος χωρίς την παρέμβαση κανενός άλλου εξωτερικού στοιχείου.

Το προτεινόμενο πλαίσιο ρομποτικής πλοήγησης δοκιμάστηκε σε πραγματικές συνθήκες και τα αποτελέσματα επιβεβαιώνουν την αποδοτικότητα και αποτελεσματικότητά του. Η πιθανοκρατική φύση καθώς και η ακρίβεια και σταθερότητα της ΙΜΠΜΔΑ-ΡΠ όπως και η πρόβλεψη της κίνησης κρίνονται απαραίτητα στοιχεία για την αποτελεσματική και αποδοτική πλοήγηση σε δυναμικά πραγματικά περιβάλλοντα με πυκνή κίνηση ανθρώπων.

Contents

List of Tables

List of Figures

List of Symbols

$\mathcal{S}$	The finite set of all possible states, s, of the environment an agent might occupy.
s_r	The invariant initial state utilized in the rPOMDP.
s_p	The projected state of the robot utilized for deciding the robot speed.
$\mathcal{A}$	The finite set of actions, a, an agent can execute.
$\mathcal{Z}$	The finite set of observations, z, an agent can observe.
$\mathcal{T}$	The state transition function.
$\mathcal{R}$	The reward function.
$\mathcal{O}$	The observation function.
b_t	The belief state of the robot at time t, representing the discrete probability distribution over the set of all possible states $\mathcal{S}$.
$\mathcal{B}$	The set of all possible belief states.
π	A policy obtained by the solution of an MDP.
π^*	An optimal policy.

$\mathbf{p}$	A POMDP policy tree.
α-vectors	The set of hyperplanes defined by a policy tree $\mathbf{p}$.
Γ	The set of all α-vectors.
γ	A discount factor that determines how important are the future rewards the robot will receive.
V	The value function of an MDP or POMDP.
V^*	The optimal value function.
Q	The Q-functions defined for solving MDPs.
ϕ	The desired discretization step of the orientation and action angles modelled by a POMDP.
d	The desired discretization of the grid map modelled by a POMDP.
L	The number of levels of hierarchy in the RN-HPOMDP.
l	A level of hierarchy in the RN-HPOMDP.
r	The overlapping region defined in bottom level POMDPs of the RN-HPOMDP.
R	The size of the square grid that defines the state space of the rPOMDP.
$\lvert\mathcal{S}^0\rvert$	The state space size of the corresponding flat POMDP.
$\lvert\mathcal{A}^0\rvert$	The action space size of the corresponding flat POMDP.

θ_p	The orientation angle of the state that is decomposed to a POMDP in the RN-HPOMDP structure.
a_p	The action angle that is decomposed to a POMDP action set in the RN-HPOMDP structure.
Λ	The maximum angular distance from the GDO.
λ	The angular distance from the GDO.
Δ	The maximum distance allowed from a cell to be considered as a hot point.
δ	The distance from a candidate hot point.
$Ediff(a)$	The average expected difference of the reward value between adjacent grid cells for an action angle a.

List of Abbreviations

MDP	Markov Decision Process
POMDP	Partially Observable Markov Decision Process
HPOMDP	Hierarchical Partially Observable Markov Decision Process
RN-HPOMDP	Robot Navigation - Hierarchical Partially Observable Markov Decision Process
rPOMDP	Reference Partially Observable Markov Decision Process
MLS	The most likely state heuristic for solving POMDPs.
OGM	Occupancy Grid Map
RGM	Reward Grid Map
IDC	Iterative Dual Correspondence algorithm for scan mathcing.
GDO	Global Direction of Obstacle.
HP	Hot Point.

1

Introduction

This thesis addresses the problem of a robot navigating efficiently and effectively in crowded environments. In this introductory chapter, we define the problem, indicate its scientific and practical significance and outline this thesis work.

1.1 Problem Statement

This thesis concentrates on the problem of a robot navigating in crowded environments in a efficient and effective manner. This problem involves three main navigation tasks:

- localization, the robot must be aware at all times of its true position in the environment it operates;
- path planning, decide what the optimal path to its goal point is;
- obstacle avoidance, avoid humans and/or other objects operating in the environment with manoeuvres that affect the optimality of the path to the goal point the least possible.

The above three tasks are treated in a unified manner in this thesis to obtain an optimal navigation behavior in crowded environments that attempts to simulate as much as possible the human behavior.

Human Navigation

For humans the navigation task is eminent. Humans, at least when walking purposively, can choose the optimal path to reach their destination point without having to put much thought into all the aspects of the task they are about to perform. The optimality of the chosen path by humans is commonly taken for granted. Yet, quantitative and qualitative measures can be employed to assess the optimality of a path executed by a human to reach a destination point. Quantitative measures are easy to define and in most cases they are indisputable. The most common quantitative measures are the time taken to reach the destination point and the distance travelled to the destination position. It is harder to define qualitative measures since depending on the specific environment where the task is executed each qualitative measure can have different importance. For example, the qualitative measure of “politeness”, i.e. whether a human gives way to other humans, can be employed in an office environment but definitely not in a high risk environment. The “elegance” and “awareness” qualitative measures could be employed that are less environment specific. A chosen path to a destination path can be characterized as “elegant”, i.e. well-designed, when it is a smooth and easy-to-follow path that does not direct a human to overly congested areas that eventually might lead to a situation where a human gets completely blocked. Furthermore, a human follows a path that can be characterized as “aware”, when it is a suboptimal path, i.e. a little longer in distance, than the obvious optimal path but it has been estimated that the latter path is becoming congested and might eventually delay him or even completely block him.

When a human wants to reach a destination point he/she makes an instantaneous decision about the path to follow. It is assumed that all humans, most commonly, want to reach a destination point as fast as possible with the minimum effort. If we consider the steps a human follows to decide how to approach its destination point we will see that its not a spontaneous decision but an informed decision. When a human is in an empty and static environment, he simply assesses the time or the distance required to reach the destination position. However, when a human is in a dynamic environment where there are other humans moving within it as well, the two previously mentioned quantitative measures might be proven insufficient. In dynamic congested environments, the time and distance quantitative measures when assessed prior to the start of the human's movement are not guaranteed to be valid throughout the whole movement until the destination point is reached. For that reason, in dynamic and congested environments humans attempt to assess the quantitative measures with a projection to the future, i.e. attempt to predict the state of the environment at sometime in the future. Furthermore, in dynamic and congested environments humans consider alternative paths to the destination position, "elegant" and "aware" paths. These alternative paths might be worse in time or distance measures when assessed prior to the human's decision but in the future they can potentially outperform a currently optimal path, in terms of time or distance measures, due to the changes occurring in the environment. Although, this might seem oxymoron, this is due to the fact that in dynamic environments optimal preplanned paths most commonly cannot be executed as planned because of the unforeseen changes in the environment.

Another important behavior of humans is that they are able to repeatedly reassess the quantitative and qualitative measures of the path they are executing but also of other alternative paths using the information they gather during execution. This allows humans to decide if they should change completely the path they have chosen initially to reach their destination point, depending on the occurring changes in the environment. Hence, humans do not stick to their initial decision and can easily change their initial plan.

In result, although humans are able to decide instantaneously which path they should follow to reach a destination point, they actually perform many tasks so that this decision is an informed one. They assess the time or distance required to reach their destination point, they attempt to predict the future state of the environment and assess the time or distance required to reach their destination point when considering the future state of the environment. Furthermore, they preform these tasks continuously until they have reached their destination point.

On the other hand, currently in robotics the most common approach employed to decide the path a robot should follow to reach its destination point is much different. Most approaches produce an optimal path that is decided as if the robot was operating in a static environment, and then during execution other methodologies are employed to accommodate for the unexpected changes of the environment. More importantly, in most of these approaches the robot sticks to the initial preplanned path no matter what the changes in the environment are and attempts to follow this path without considering alternative paths during execution. This approach however, as explained earlier, leads very often to executing paths that are very poor in performance measures as compared to the optimal preplanned path.

In this thesis, a novel approach to path planning is proposed that has more resemblance to the approach a human would follow to decide the optimal path to his destination point. As such, the proposed approach employs techniques for predicting the future state of the environment and makes use of this prediction to produce “elegant” and “aware” paths that, although they might be sub-optimal at the time of the decision, when considering only the current state of the environment, they end up being optimal during execution. Furthermore, the proposed approach continuously reassesses the state of the environment and is able to change completely the path the robot executes on-line.

1.2 Practical Interest

The robot navigation task has been studied extensively by the robotics research community over the last decades. Today it can be claimed that robotics research has reached a point where the main task of navigation is almost solved.

However, this is completely true only for specific environments and under specific assumptions. There are numerous successful approaches to the robot navigation task but robotics research is still far from the vision of building robots that behave much like humans and can truly cooperate and coexist with them and in a sense acclaim a role in our society - much like the robots appearing in science fiction books and movies.

Nowadays, robotics applications are not confined to industrial purposes where they only have to perform a specific and, most commonly, a repetitive task. The trend in robotics today and for the future is in robots that are intelligent, autonomous and can perform complex tasks in dynamic environments. More importantly, future robots have to be able to perform their assigned tasks in harmony with humans, operating within the same environment.

This thesis attempts to make a step forward towards robots that simulate aspects of human behavior and can truly cooperate and coexist with humans. Specifically, we consider the problem of building robots that can operate in environments that are highly populated and perform the assigned tasks efficiently and effectively while humans operate in the same environment. In this thesis, the task the robot performs is focused in reaching a goal position.

The problem considered in this thesis, a robot reaching a goal position in a time-optimal, “elegant” and “aware” manner, can be applied in many robotics applications. This behavior fits mainly, but is not limited, to personal and service robots. Personal robots have many applications ranging from purely enlightenment purposes (e.g. Sony’s Aibo dog-like robot) to robots that assist the disabled and elderly and robots that perform the dull and fatigue household tasks (e.g. iRobot's vacuum cleaner Roomba). Service robots can be applied to many tasks as tour guides, office robots, surveillance robots, search and rescue robots and many more.

1.3 Scientific Interest

When a robot is given the task of reaching a specific destination point, it is faced with the question "What should I do now?"[71]. This question has to be answered, where:

- "now" is the current state of the robot (e.g. location, orientation) and the current environment state;
- "do" is one of the actions the robot can perform (e.g. move forward, move left);
- "should" is maximize a long-run measure of reward; and
- "I" is an automated planning or learning system (agent).

Actually the answer to the question "What should I do now?" is given by the three main subtasks of the navigation problem: *mapping*, *localization* and *motion planning*.

The *mapping* subtask provides the robot with the knowledge of the environment within which it operates. The representation of this knowledge can be given either in the form of a metric map, that represents the geometric properties of the environment, or in the form of a topological map, that represents the connectivity of detected features of the environment. Hybrid approaches are also possible, where both metric and topological information is used.

The *localization* subtask is responsible for maintaining at all times the robot's exact location within the environment it operates. There have been proposed numerous approaches to solving the localization subtask but the most recent approaches are targeted towards maintaining a probability distribution over all possible robot locations instead of providing a single estimate of the robot's location.

The *motion planning* subtask steers the robot in executing actions that will eventually lead it to its destination position in an optimal manner. The optimality criterion has been discussed previously. Since we are referring to a robot operating in a dynamic environment, the motion planning subtask has to provide sequences of actions that not only lead the robot to its goal position but also ensures that the robot will not collide with any static or moving obstacles. Hence, the motion planning task is subdivided into two subtasks: global motion planning and local motion planning or just motion planning and collision avoidance, respectively. Therefore, the global motion planning module provides the path to the goal position and the local motion planning module directs the robot to execute manoeuvres if necessary to avoid obstacles. Furthermore, recently it has been recognized that there is the need for probabilistic motion planning approaches that provide sequences of actions based on the probability distribution of the robot's location instead of planning based on a single best estimate of the robot's position. Although, probabilistic methods for mapping and localization have been studied extensively and are commonly used, the same does not hold for probabilistic motion planning approaches.

In total, the navigation problem is treated currently by employing separate cooperating modules for each of the above mentioned modules.

1.4 Proposed Approach

In this thesis, we are mainly dealing with the motion planning module and implicitly with the localization module. It is assumed that a metric map of the environment is provided. This thesis proposes a unified model that incorporates the modules for localization, global motion planning and local motion planning. The proposed model is utilized for driving the robot in highly populated environments, as mentioned earlier, and therefore future obstacle motion prediction is also incorporated into the model to obtain paths that are not only time-optimal but also "elegant" and "aware".

As implied earlier, probabilistic methods for mapping, localization and motion planning are recognized currently as the most appropriate navigation approaches. This is due to the fact that there is uncertainty in identifying the true state of the world and the robot [125]. Hence, during the last years robotics research has focused in probabilistic methods that provide a probability distribution over all possible states rather than a single estimate of the true underlying state. Probabilistic methods for mapping and localization have been well studied and are extensively used in many robotics applications. However, probabilistic methods for motion planning are not that commonly used even thought it is now recognized that probabilistic methods offer great advantages over conventional motion planning methods [101]. Conventional motion planners that provide sequences of actions based only on a single best estimate often result to deceptive paths, in the sense that the planned shortest path to the goal is not the one that is actually executed since the planner may have an erroneous estimate of the actual robot location.

In this thesis we employ a probabilistic unified model for motion planning and localization. This model is the Partially Observable Markov Decision Process (POMDP). POMDPs are well suited for robotic applications as they are probabilistic and model explicitly the actions and states of the robot. Since the true underlying state of the robot is never known, POMDPs use observations that assist in determining the probability distribution over the set of possible robot states.

POMDPs are well established models for robotics applications, but they have not been used that extensively for motion planning because of the very large computational overhead. POMDPs have been applied so far mainly as global path planners that provide an abstract trajectory to a destination point and a local motion planning module is usually employed to drive the robot between the points directed by the solution of the POMDP. POMDPs have been originally employed for robot motion planning almost a decade ago [59] but only recently the robotics research community has revived its interest in applying POMDPs for motion planning where many approximation methods for solving POMDPs have been proposed that attempt to harness the extreme computational requirements of solving POMDPs [51].

In this thesis, POMDPs are employed under a completely different perspective. POMDPs are used to drive the robot to its destination position without the intervention of any other module for local obstacle avoidance or localization. To facilitate this task within a POMDP formulation, it is required that the POMDP is solved at each time step on-line to provide the action the robot should perform. Furthermore, the POMDP must be able to model the environment at a fine resolution to produce smooth paths. These requirements are achieved by employing a novel hierarchical representation of POMDPs.

The navigation problem described cannot be modelled with a flat POMDP, even if approximation methods were used to solve it, since the computational requirements are extremely hard to manage in real-time. Hence, a hierarchical representation of POMDPs, specifically designed for the autonomous robot navigation problem, is utilized and is termed as Robot Navigation-HPOMDP (RN-HPOMDP). The RN-HPOMDP is capable of modelling real-world environments at a fine resolution and it is solved in real-time.

As mentioned earlier, the paths the robot follows have to be not only time efficient but also “elegant” and “aware”. This is achieved by incorporating into the POMDP not only the current state of the environment but also prediction about the movement of humans and/or other objects. Motion prediction with conventional techniques is able to provide good estimates only for the immediate next step position of the movement. This kind of prediction does not provide enough information to be able to produce “elegant” and “aware” paths. Hence, we utilize a novel methodology for long-term prediction that attempts to estimate the final destination of a human’s movement.

Finally, a methodology for modifying the robot’s speed of motion by the POMDP is proposed. Increasing or decreasing the robot’s speed, improves further the quality of the paths obtained by the POMDP especially in the cases where there are changes in the environment that are impossible to predict.

1.4.1 Contributions

The main contributions of this thesis are summarized into the following features:

- a new hierarchical representation of POMDPs, specifically designed for the autonomous robot navigation problem, termed as the Robot Navigation-HPOMDP (RN-HPOMDP);
- POMDPs are employed as a unified navigation model that can handle all aspects of the navigation without the intervention of any other external module and also provide the actual actions the robot executes;
- a novel approach for predicting the future motion of humans and/or other obstacles based on estimates of their final destination point;
- a novel motion tracker that utilizes the future motion prediction of humans and/or other obstacles;
- a novel methodology for modifying the robot's speed of motion to avoid humans and/or other obstacles effectively.

1.4.2 Publications

Parts of the work presented in this thesis have already been submitted for publication or published to international scientific journals and conferences as follows:

- A. Foka and P. Trahanias, "Predictive autonomous robot navigation", under preparation for submission to the Autonomous Robots Journal [35].

- A. Foka and P. Trahanias, "Real-time hierarchical POMDPs for autonomous robot navigation", IJCAI-05 Workshop: Reasoning with Uncertainty in Robotics (RUR-05) [37].
- A. Foka and P. Trahanias, "Real-time hierarchical POMDPs for autonomous robot navigation", submitted for publication in the Robotics and Autonomous Systems (RAS) Journal [36].
- A. Foka and P. Trahanias, "Predictive control of robot velocity to avoid obstacles in dynamic environments", presented in the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS03), Las Vegas, USA, 2003 [34].
- A. Foka and P. Trahanias, "Predictive autonomous robot navigation", presented in the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS02), Lausanne, Switzerland, 2002 [33].

1.5 Thesis Structure

This thesis is organized into eight chapters, the first being the current introductory chapter. The rest of the thesis is composed as follows:

Chapter 2 briefly reviews each module for robot navigation problem that will be incorporated into the proposed POMDP model, i.e. motion planning, localization and obstacle avoidance, and also the modules for motion prediction and motion tracking.

In **Chapter 3** the background theory of POMDP models is given.

Chapter 4 describes the hierarchical POMDP (HPOMDP) proposed. The approach for learning and planning with the proposed HPOMDP is given. Finally, a comparison of the proposed HPOMDP with other hierarchical approaches in the literature is given.

Chapter 5 gives the methodology used for object motion prediction and motion tracking.

Chapter 6 describes how robot velocity control is achieved using a modified solution of the POMDP.

Chapter 7 presents the results obtained with the proposed approach in real environments and also provides quantitative evaluation measures.

Finally, in **Chapter 8**, the achievements of this thesis are summarized and directions for future research are indicated.

2

Background and Literature Review

The autonomous robot navigation task is subdivided in the three subtasks of mapping, localization and motion planning. This thesis is mainly concerned with the task of motion planning and implicitly with the task of localization. Hence, in this chapter the current research status for motion planning is presented. Since motion planning is performed in dynamic environments with integrated motion prediction of moving objects, the current research status for the tasks of obstacle avoidance and motion prediction and tracking is first presented. Finally, the localization problem is treated implicitly by the POMDP used for navigation and, therefore, a short survey of localization approaches is given.

2.1 Obstacle Avoidance

Obstacle avoidance, or otherwise stated as local motion planning, is a problem well studied in robotics research. The obstacle avoidance task is also stated as local motion planning since the vast majority of robotic navigation systems make use of global path planning and local path planning modules. The distinction between global and local path planning is made since the first one is responsible for deciding the abstract path the robot should follow to reach its goal and the latter decides how the robot should move between intermediate points of the abstract path without colliding with any static or dynamic obstacles. This distinction has been adopted because of the large computational overhead required to solve the global path planning task. Therefore, it is not possible to solve the path planning problem globally, continuously and on-line to be able to avoid dynamic obstacles and as a result local motion planning has been adopted that solves a small proportion of the general path planning problem.

The most commonly used solution to the obstacle avoidance problem is the artificial potential field method. This approach originally proposed by Khatib [63], has been very popular due to its simplicity and versatility. The robot is treated as a charged particle where it is attracted by the goal point and repelled by the obstacles present in the environment by imaginary forces. The artificial potential field method suffers from two major problems: the robot may become trapped in local minima and the robot may oscillate in narrow passages. Despite these problems, the artificial potential field method has attracted the interest of the robotics research community and many improvisations on the originally proposed method have been proposed that attempt to overcome these problems. More specifically, the use of complex potential fields as harmonic functions [140] and distance transforms [57] treat the problem of local minima whereas potential field methods that also account for the robot dynamics treat the problem of oscillations [47].

A very successful and commonly used alternative approach to potential fields is the Vector Field Histogram (VFH) approach, proposed by Borenstein [12]. The VFH approach has taken elements by the potential fields and the occupancy grids approach. The VFH approach maintains a polar histogram that represents the confidence of the existence of an obstacle as with occupancy grids, only that the update procedure is much faster since it updates only one cell for each range reading. Finally, each cell that is believed to be occupied by an obstacle exerts a virtual repulsive force to the robot whereas the target exerts a virtual attractive force. Improved versions of the VFH approach have been also proposed. The VFH+ [130] method extends the original method by considering the robot dynamics as well. In the most recent improvement on the VFH, the VFH* [129] method, each candidate direction the robot is targeted to is verified by projecting the position of the robot after executing the candidate direction.

A completely different approach to the obstacle avoidance problem that has been adopted mainly after the identification of the problems of the original VFH method, is the class of methods that formulate the obstacle avoidance problem as one of constrained optimization in the velocity space of the robot. Such methods estimate the translation and rotational velocity of the robot in order to avoid an obstacle. This approach is different from the potential field or VFH approaches that provide the direction the robot should follow. Representatives of this class of approaches can be found in [18, 30, 31, 38, 40, 106]. The most successful and widely adopted approach of this class is the Dynamic Window Approach (DWA) [40]. The DWA searches for a pair of translation and rotation velocities that maximizes an objective function but only within a dynamic window, i.e. within the velocities that are reachable within a short time interval. Fox et al. [38] have also proposed an extension to their originally proposed DWA, the model-based dynamic window approach (μDWA). The μDWA is in essence the original DWA integrated with a map of the environment and the Markov localization algorithm. As such, the μDWA performs obstacle avoidance in a probabilistic manner.

The problem of obstacle avoidance has been also treated by fuzzy logic [102] and neural network [20] approaches. However, these methods have not been widely adopted mainly due to the difficulty of training these systems and more importantly due to their difficulty in dealing with conditions that have never been encountered before. Finally, standard mechanics and control theory approaches [25, 88, 128, 131, 137] have also been applied to the obstacle avoidance problem. However, these methods are not appropriate for sensor-based navigation since they assume the robot has a complete world model.

To summarize, all the above mentioned approaches for obstacle avoidance have their advantages and disadvantages, but they all suffer from a very important feature that is inherent in all of them: they treat the problem locally. Although many of the above mentioned approaches avoid obstacles effectively, the local treatment of the problem directs the robot into executing globally suboptimal paths to its destination point. This is due to the fact that all the above mentioned obstacle avoidance methods stick on the initial planned path to the destination point obtained by the global path planning module and do not replan no matter the changes that occur in the environment. This problem has been recognized recently by the robotics research community and some approaches that integrate the path planning and obstacle avoidance modules have been proposed [13, 73, 74, 111].

In [13] an approach that integrates the DWA with a local minima-free potential function that is updated during execution is presented that is termed as the Global DWA. The update of the potential function is performed only for a specific region to be able to replan in real-time. Another approach that integrates path planning and collision avoidance is the one presented in [111]. In this approach, after a path to the goal is computed in the x-y space, a search space is defined that is reachable within a short time interval that also defines the next subgoal of the robot. Following, the path to the subgoal point is computed in the five-dimensional space that includes the orientation of the robot and the translational and rotational velocities.

As implied by the above, the key problem of obstacle avoidance methods is how they affect the global path the robot follows to reach its goal point. This problem is caused because the obstacle avoidance is treated locally. The last two approaches reviewed attempt to alleviate this by solving the problem within an extended area but still not globally. In general we can safely assume that the desirable robot behavior for obstacle avoidance is to simulate the behavior of humans for avoiding obstacles. That is, to continuously assess the current state of the environment but also its estimated future state and compute completely new global paths to the goal point if necessary.

In this thesis, the obstacle avoidance problem is integrated into the global navigation problem. As such, obstacle avoidance is treated globally where the complete path to the goal position is replanned at each time step. Furthermore, obstacle avoidance is performed in a probabilistic manner since it is integrated into the global probabilistic navigation model. Finally, obstacle avoidance is reinforced by performing prediction of the obstacle's movement as it will be explained in the following sections.

2.2 Motion Prediction

Predicting the motion of obstacles can assist greatly in the task of obstacle avoidance. Being able to predict the future position or path of an obstacle allows the robot to decide the actions it should perform to avoid collision before it comes too close to the obstacle. More importantly, knowing the future positions of an obstacle's movement permits the obstacle avoidance module - local or global - to plan more effectively and produce paths that are closer to the corresponding statically planned optimal path to the destination position.

The importance of motion prediction to obstacle avoidance has been recognized early by the robotics research community and there are numerous approaches present in the literature. Until very recently referring to motion prediction was equivalent to referring to one-step ahead prediction of an obstacle's movement, i.e. predict the position of an obstacle at the immediate next time step.

One-step ahead motion prediction has been mainly approached by attempting to model the obstacle's motion by a stochastic process [29, 61, 82, 115, 116, 138, 142] or a Neural Network

(NN) [19, 85, 86]. In the class of methods that use a stochastic process there are major assumptions about the obstacle's motion model. For example, in [61] it is assumed that the obstacle is moving along a straight line; in [142] the obstacle is assumed to obey a specific motion model that can be extracted by visual processing or otherwise it is assumed to be Gaussian; and in [82] obstacles' velocity and direction angles are modelled by a random walk process that it used to calculate the probability density function of the obstacles' position. However, in [29] there are no constraints on the obstacle motion.

Other approaches to one-step ahead prediction have been proposed that include the least-mean-square-error classification method [134], the grey prediction theory [77], the use of an Extended Kalman Filter (EKF) [78] and a polynomial regression model [136]. Furthermore, approaches that extract information from visual processing regarding the obstacle's motion are also present in the literature [54, 66, 118]. Finally, in [80] an approach that maintains a probabilistic distribution of the possibility of the robot and the obstacle meeting on a path belonging to a predefined set of possible paths based on velocity estimates is presented.

In total, all the previously mentioned approaches that perform one-step ahead prediction have their advantages and disadvantages but what really matters is the actual effect of prediction to obstacle avoidance. Although many of the prediction methods present in the literature claim to be able to predict further in the future than for just one-step ahead, this prediction cannot be as accurate to be utilizable for obstacle avoidance. However, even though there are lately prediction methods that perform quite satisfactorily for one-step ahead prediction with relaxed assumptions on the obstacle's motion, the effect they have on the obstacle avoidance module is still questionable. This is due to the fact that the information available about an obstacle's current position and its immediate next time-step position is not indicative about the general obstacle behavior. In result, one-step ahead prediction can potentially assist the obstacle avoidance module locally but cannot be that effective to the final path the robot will execute to reach its destination position. As discussed in the previous section it is of greater importance the optimality of the ultimate path the robot executes to reach the goal position, including any manoeuvres to avoid obstacles, than of examining locally how well it avoided a specific obstacle at a specific occurrence.

To be able to perform obstacle avoidance optimally in a global sense it would be more useful to be able to know the full trajectory of an obstacle and use this information when planning. Very recently methodologies for predicting the whole path an obstacle is following have been proposed in [9, 14, 132] concurrently with our own proposed approach originally presented in [33]. In [9] a methodology for learning motion patterns of humans in an environment is presented. This approach uses the Expectation-Maximization (EM) algorithm to cluster motion trajectories into various classes of motion patterns that are possible in a specific environment. Prediction is performed by maintaining a likelihood of an obstacle's motion trajectory belonging to each class of the learned motion paths. As the obstacle continues its movement elements of the set of hypotheses are eliminated and after some time it converges to a single class of motion paths. Subsequently proposed methods [14, 132] are very similar to the work of Bennewitz et al. [9] only that they use a different methodology for clustering the motion paths.

Our proposed approach is different in the way it performs long-term prediction as it does not attempt to learn or predict whole motion paths but possible destination points. In addition, our prediction module performs also short-term prediction that is utilized by the long-prediction algorithm. Furthermore, our approach is integrated with the motion tracker and the global path planning module.

2.3 Motion Tracking

The task of obstacle avoidance discussed in the previous sections has been defined of high importance for the robot navigation problem in populated environments. Furthermore we have elaborated on the importance of being able to predict the motion of humans or other obstacles present in the environment and use this information for obstacle avoidance. A prerequisite for performing the tasks of obstacle avoidance and motion prediction is to be able to solve the problem of motion tracking, i.e. being able to extract from the obtained sensor measurements at each time step the position of each object correctly.

The problem of motion tracking has been studied originally by the signal processing community, mainly for military applications. Kalman filters have been used extensively in this kind of applications [6]. Additionally the problem of motion tracking has been studied extensively by the vision community [44]. The motion tracking problem in this thesis is considered under the perspective of tracking multiple objects based on laser measurements perceived during navigation. Lately, there is a growing interest for tracking multiple objects with multiple robots [45, 58, 87, 98, 135], a task that is beyond the scope of this thesis.

The problem of motion tracking is actually twofold; it includes a prediction step where the current position of previously tracked objects is estimated and a data association step where it assigns detected features to previously detected objects or new objects.

The prediction step of the motion tracking problem is a difficult task to perform due to the fact that the movement of humans is in general non-linear and non-Gaussian and it is very difficult to obtain a realistic motion model. However, the problem of motion tracking has been originally approached with Kalman filters (KF) where non-linearities cannot be modelled. This eventually led to the use of the Extended Kalman Filter (EKF), that linearizes models with weak non-linearities around the current state estimate, so that a KF recursion can be applied. Following, the Unscented Kalman Filter (UKF) has been proposed that approximates more effectively non-linear models. In total, Kalman filters have been used extensively for motion tracking although it is recognized that the motion model cannot be well approximated in most real-world cases. Recently, methods for the prediction step of motion tracking based on Particle

Filters have been proposed with most influential the work in [103]. Particle Filters represent the dynamic object states by a set of samples (particles) and weight particles based on a likelihood score. Particles are propagated according to a motion model. The complexity of Particle Filters grows exponentially with the number of objects to be tracked but several methods have been proposed that overcome this shortcoming.

The data association step has been originally addressed by the Multiple Hypotheses Tracker (MHT) [95]. The MHT maintains all possible hypotheses over time and this increases dramatically the computational complexity. To reduce the complexity methods that filter out hypotheses have been proposed. The Nearest Neighbor Standard Filter (NNSF) [7] classifies detected objects to the object that is closest in distance. The NNSF and variants of it have been used extensively in motion tracking since it is a simple filter with small computational complexity [65, 70]. However, in many cases the NNSF classifies incorrectly a detected object and prunes away feasible hypotheses. The Joint Probabilistic Data Association Filter (JPDAF) [24] maintains at all times the joint probability distribution that determines which detected feature is assigned to which object. In JPDAF the association variables are considered to be stochastic variables and one needs only to evaluate the association probabilities at each time step.

The most recent approaches to laser-based motion tracking are based on particle filters and JPDAF, mainly influenced by the work presented in [103, 104] by Schulz et al. They have proposed the Sample Based Joint Probabilistic Data Association Filters (SJPDAF) for motion tracking, that use particle filters to predict the state of objects and JPDAFs to assign measurements to objects. This approach assumes that the motion of humans follows a Gaussian distribution.

In [81] conditional particle filters are proposed to perform robot localization and motion tracking. The conditional particle filter estimates the robot pose as well as people pose. People pose estimates are conditioned on the robot pose. Data association is performed by a modified nearest neighbor filter and human motion is assumed to be brownian motion.

In [56] the multi-target particle filter (MTPF) is introduced, where samples are obtained from their joint posterior using a proper Markov Chain Monte Carlo (MCMC) technique, the Gibbs sampler. The Monte Carlo-JPDAF (MC-JPDAF) [133] represents distributions of targets with Monte Carlo samples instead of Gaussians.

Furthermore, to deal with the increasing complexity as the number of tracked objects increases the authors in [133] also propose the Sequential Sampling Particle Filter (SSPF) that samples the individual targets sequentially by utilizing a factorization of the importance weights and the Independent Partition Particle Filter (IPPF) that assumes that associations are independent over the individual targets. Another approach to deal with the computational cost of data association is presented in [41]. It is a hybrid method of couple sample based JPDAF, that offers stabilility in critical situations, and the independent sample based JPDAF, that has low computational cost.

To conclude the particle filter based methods, the multiple hypothesis particle filter presented in [62] that employs nearly independent filters for low computation cost, makes use of interactions between targets to assist in the prediction step. A Markov random field motion model is used to model interactions between targets.

A completely different approach to motion tracking is proposed in [10] that utilizes learned motion patterns of humans, as discussed in the previous section, to track the motion of humans. In this approach, independent Kalman filters and the nearest neighbor filter are utilized.

In this thesis, the motion tracking problem is dealt in conjunction with the motion prediction problem. The motion prediction method described in the previous section will be used not only to provide the estimate of an object's position but also to eliminate data association mistakes due to occlusions by making use of the long-term prediction. Data association is performed in this work by the nearest neighbor filter.

2.4 Robot Motion Planning

The problem of path planning has been studied extensively by the robotics research community. The methods present in the literature can be classified into two categories depending on the model of the environment used. Hence, there are path planning methods that make use of a topological map and methods that make use of a metric map.

Topological maps are in the form of graphs where the nodes represent distinct places and the edges represent the relations between them. Common path planning methods that make use of topological information are the roadmap algorithms including the visibility graph, voronoi diagram and cell decomposition based techniques [15, 42, 67]. Furthermore, there are path planning methods based on the A^* and D^* algorithms that search a graph for an optimal path [105, 112]. Finally, the family of the BUG-algorithms are also based on graph search [60, 76].

Planning methods that make use of metric maps have been dominated till now by potential field methods [63]. In potential field methods the robot is treated as a particle that is under the influence of a potential field that is formed to represent the environment structure. However, potential field methods have the major shortcoming of getting the robot trapped in local minima. To escape local minima there have been proposed heuristics [26] that detect whether the robot is trapped and attempt to escape the local minimum. Another approach for escaping local minima is the use of a global navigation function that is a minimum-free function [22, 97, 140]. An additional popular method for path planning with metric maps is value iteration, that is a dynamic programming algorithm [114, 123].

Other motion planning methods are the ones based on genetic algorithms [2, 79], control theory [139] and more recently there have been proposed methods that make use of a sensor network embedded in the environment [8].

A major issue when planning with any of the above mentioned methods is whether they can deal with the uncertainty of sensor measurements and the uncertainty of the robot's pose. Unfortunately, all the above mentioned planning methods assume that the true robot pose is known and the plans they provide are based on this assumption. However, this is not the case in real-world environments. Lately, it has been recognized that it is crucial to plan in a probabilistic manner, i.e. based on a probability density function over all the possible robot locations rather than based only on a single estimate of the robot's location [125].

Virtually all methods present in the literature that plan in a trully probabilistic manner, i.e. based on the probability distribution of the robot's true state rather than on a single estimate, are implemented with a Partially Observable Markov Decision Process (POMDP). POMDPs are models for decision making that model explicitly uncertainty. As such, POMDPs have been used for planning in many successfull robot platforms [59, 83, 84, 90, 91, 100, 101, 107, 110, 119, 124].

All the above mentioned approaches for path planning with metric maps have the disadvantage that become computationally expensive as the discretization of the modelled environment increases. As a result, most robotic platforms deployed in real-world environments use many path planners of different hierarchy to navigate. For example, the robot Rhino [123] uses value iteration as a global planner and send intermediate sub-goals to the reactive collision avoidance module (DWA). The robot Xavier [107] performs high level path planning with the use of a topological map, then uses another planner for moving the robot from a location to another location and finally makes use of an obstacle avoidance module. Finally, the robot Minerva [124] uses the POMDP based coastal planner to move from one exhibit to another and a low-level reactive collision avoidance (μDWA).

In this thesis, the motion planning problem is treated with use of a POMDP model. However, in the proposed approach the POMDP is not used as a high level mission planner as in the previous implementations of POMDP based planners. The major drawback of using POMDPs as planners is their extreme computational requirements, that has made it impossible to use with fine discretized environments. Although, lately there has been a growing interest by the research community to provide approximation methods for solving POMDPs [51, 91, 110] or hierarchical representations [89, 119], still the allowed discretization of the environment is not that fine to facilitate the use of POMDPS as low level planners.

In this thesis, a hierarchical representation of POMDPs is proposed that can effectively model large real-world environments at a fine resolution and can be solved on-line. Hence, the proposed hierarchical POMDP is used as a low level planner, without the need of any other high level mission planner, that provides the robot with the actual actions it will execute. This approach alleviates the need of using many planners of different discretization. Thus, the probabilistic manner in which POMDPs plan is fully utilized as compared to other architectures where plans are obtained probabilistically only at a certain level, most commonly at a high level. Finally, the proposed hierarchical POMDP is utilized as unified navigation model that incorporates the modules of localization, obstacle movement prediction and obstacle avoidance.

2.5 Localization

The localization problem refers to the problem of deterimining the true location (state) of the robot at each time step. The simplest way to perform localization is by dead-reckoning, that is integrate the velocity history of the robot in order to determine the change in position between two subsequent measurements. However, there is uncertainty in dead reckoning sensor measurements so localization methods are used to update the robot position.

Localization methods utilize sensor measurements or sensed features and a known map of the environment in addition to dead reckoning to estimate the true robot position. As a result, depending on the type of map of the environment and type of features used there are different approaches to localization. In general, localization approaches can be classified into the landmark-based, feature-based and grid-based approaches.

Landmark based approaches usually in conjuction with a topological map attempt to locate a sensed landmark into the topological map and then perform a triangulation procedure to determine the position of the robot [11, 16, 21, 53, 68, 84, 96, 108, 113, 121, 127].

Feature-based methods extract information from sensor measurements (e.g. lines, corner points) and attemp to match them with a known metric map of the environment [23, 27, 43, 46, 109, 117]. Another class of feature-based methods is the scan matching approaches, that is to match directly laser measurements with a known map of laser measurements [75]. Many of the feature-based methods utilize Kalman filters to perform localization [3, 4, 17, 25, 55, 69, 141].

In grid-based methods [16, 39, 122] instantaneous sensory information acquired by the robot is matched with the occupancy grid map and the location of the robot is calculated by considering areas of the map with high correlation. Grid based methods mainly utilize the Markov localization paradigm.

Localization methods can be further subdivided into the methods that utilize Kalman filters and the ones that are based on Markov localization. A comparison between these two main approaches has been conducted in [48, 49]. The results of this comparative study show that Kalman filter approaches are superior with respect to computational efficiency, scalability, and accuracy. On the other hand, Markov-based localization approaches are more robust in the presence of noise and/or unreliable odometry information. Hence, many hybrid localization methods that attempt to exploit the robustness of Markov-based localization methods and the computational efficiency of Kalman-based methods have been proposed [5, 28, 50, 120, 126].

In this thesis, the localization problem is treated implicitly by the POMDP model used for navigation. POMDPs have been used for localization previously in [16, 64, 84, 108]. In

[84] a topological map was used and in [64] topological information is combined with metric information. In our implementation, only metric information is used since the hierarchical structure employed facilitates the use of a fine discretized grid map.

2.6 Conclusions

In this chapter the background and previous work regarding navigation tasks has been illustrated. The tasks of obstacle avoidance, motion prediction and tracking, path planning and localization are treated in a unified manner by the navigation model proposed in this thesis. Hence, this chapter provided an insight on how the robot navigation problem is treated by incorporating separate modules and also elaborated on the advantages of using a unified probabilistic model, the POMDP, that the following chapter presents its theory.

3

Sequential Decision Making

In this chapter the background theory for Partially Observable Markov Decision Processes (POMDPs) is presented. For clarity reasons, first the Markov Decision Processes (MDPs) properties will be outlined and following they will be extended for the case of POMDPs. Then, methods for learning and solving POMDPs will be described.

3.1 Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs) are a model for sequential decision making. MDPs are formally defined as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, where

- $\mathcal{S}$, is a finite set of all possible states of the environment that the agent might occupy.
- $\mathcal{A}$, is a finite set of actions.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$ is the *state transition function*, giving for each state and agent action, a probability distribution over states. $\mathcal{T}(s, a, s')$ is the probability of ending in state s',

given that the agent starts in state s and takes action a, $p(s'|s,a)$. The distribution over the state space depends only on the current state-action pair and not on previous state-action pairs. This requirement ensures the *Markov property* of the process.

- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the *reward function*, giving the expected immediate reward gained by the agent for taking an action a when it is in state s, $\mathcal{R}(s,a)$.

3.1.1 Solving Markov Decision Processes

Solving an MDP amounts to obtaining a sequence of actions that maximize the reward the agent will receive according to the defined reward function, $\mathcal{R}$. Hence, the solution of an MDP is a *policy*, denoted as π, that is a mapping function from states to actions. An optimal policy, π^*, is the policy that maximizes the sum of rewards the agent will receive. The sum of rewards can be computed either for a specific number of steps, the *finite horizon* case, and either until the agent reaches the goal state or until the sum of rewards can not be further increased, the *infinite horizon* case.

However, in a stochastic world the outcome of an action is never known so the sum of rewards is actually the expected sum of rewards according to the MDP transition and reward functions. Therefore, the optimal action to be executed when the agent occupies a state s_t at time t, is the one with the maximum expected accumulated reward,

$$E\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)\right], 0 \leq \gamma \leq 1$$

where γ is a discount factor that determines how important are the future rewards the robot will receive. If γ is zero, the robot will maximize the reward it will receive for the next time step only.

The most common way to find the optimal policy of an MDP is to compute iteratively the

Table 3.1: MDP value iteration algorithm.

```
$t = 0$
**for** all $s \in \mathcal{S}$
    $V_t(s) = 0$
**end**
**while** $\max_{s \in \mathcal{S}} |V_t(s) - V_{t-1}(s)| < \epsilon$
    $t = t + 1$
    **for** all $s \in \mathcal{S}$
        **for** all $a \in \mathcal{A}$
            $Q_t(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V_{t-1}(s')$
        **end**
    $\pi_t(s) = \arg\max_a Q_t(s,a)$
    $V_t(s) = Q_t(s, \pi_t(s))$
    **end**
**end**
```

value function

$$V_t^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V_{t-1}(s') \right]. \tag{3.1}$$

The value function is computed for each state $s \in \mathcal{S}$ at the time step t according to the previously computed value function at $t - 1$. The algorithm for value iteration [94] is given in Table 3.1. The algorithm in Table 3.1 makes use of the auxiliary *Q-functions*,

$$Q_t(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V_{t-1}(s'). \tag{3.2}$$

The value iteration algorithm terminates when the maximum difference between two successive value functions is less than a predefined error bound factor ϵ.

To illustrate how value iteration works a simple navigation example is given below where the world consists of four states that the robot might occupy as shown in Figure 3.1(a). There are two actions the robot can execute; move to the *East* or to the *West*. The goal state the robot has to reach is state 3. The reward the robot will receive is 1 if it ends up in the goal state, -1 if executes an action that leads to a wall and 0 otherwise. The robot executes the actions successfully with probability 0.9. The probability of moving in the opposite direction or remaining to the same state is 0.1. The reward for each action and the transition probabilities are shown graphically in Figure 3.1(b).

The value function at time step $t = 1$ is simply the maximum immediate reward the robot will receive for executing an action since the value function is initialized to zero, i.e. the value function for each state at time step $t = 1$ is equal to:

$$\begin{aligned} V_1(1) &= 0 \\ V_1(2) &= 1 \\ V_1(3) &= 0 \\ V_1(4) &= 1. \end{aligned}$$

The value function at time step $t = 2$, if we set $\gamma = 0.9$, for each state is equal to:

$$V_2(1) = 0 + 0.9[0.9 \times V_1(2) + 0.1 \times V_1(1)] = 0.81$$

$$V_2(2) = 1 + 0.9[0.9 \times V_1(3) + 0.1 \times V_1(1)] = 1$$

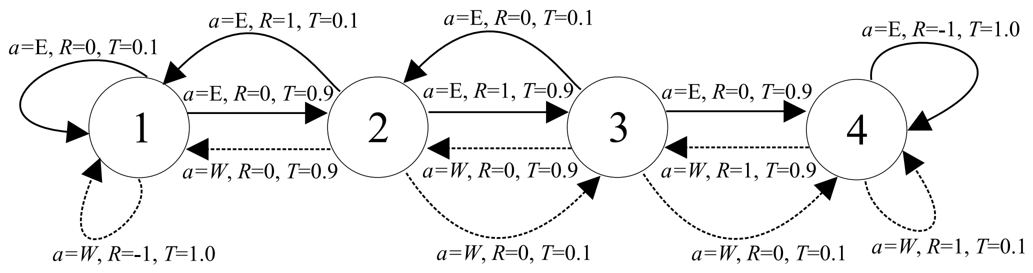
$$V_2(3) = 0 + 0.9[0.9 \times V_1(4) + 0.1 \times V_1(2)] = 0.9$$

$$V_2(4) = 1 + 0.9[0.9 \times V_1(3) + 0.1 \times V_1(4)] = 1.09.$$

In essence, the value function at horizon $t = 2$ computes the reward the robot will receive when it executes two consecutive actions. Therefore, the value function equation contains a term for the immediate reward, $\mathcal{R}(s, a)$, and also the expected reward, $\gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') V_{t-1}(s')$, the robot will receive. The expected reward is computed according to the transition model defined in the MDP. In the above example, to compute the value function at time step $t = 2$ for state 1, we add the immediate reward it will receive for taking action *East*, that is equal to zero, with the expected reward, i.e. the reward the robot is expected to receive after executing

1	2	3	4

(a) The four states that compose the world in which the robot operates.



(b) The transition probabilities and rewards defined for each state and action.

Figure 3.1: Modelling of a simple navigation task with an MDP.

action *East* from state 1. When the robot executes action *East* from state 1, it will end to state 2 with probability 0.9 or it will remain in state 1 with probability 0.1. Thus, the expected reward is equal to the value function at time step $t = 1$ of state 2 plus the value function time step $t = 1$ of state 1 multiplied by 0.9 and 0.1 respectively, that is the probability with which each of the two actions will be executed. The value function at time step $t = 1$, is essentially only the immediate reward the robot will receive for executing action *East* from state 2 and state 1. As a result, the value function of state 1 at time step $t = 2$ is the sum of rewards for executing two actions discounted by the γ factor and according to the transition probabilities defined in the MDP.

With this simple example given above, it is evident that by value iteration at infinite horizon the optimal sequence of actions that will lead the robot to the goal state is evaluated for each starting state. Thus, the value function provides the accumulated reward the robot will receive for executing the whole sequence of actions and not only a single action. However, the performance of the value iteration algorithm is critically dependent on the transition model defined in the MDP, since in a real world environment the outcome of actions is never guaranteed.

3.2 Partially Observable Markov Decision Processes

MDPs discussed in the previous section provide a model for decision making under uncertainty of actions, i.e. the outcome of actions is modelled probabilistically. However, MDPs require that the true state the robot occupies is always known. Yet, this is not the case in real world applications where the robot is not always certain about its true underlying state. When the state of the robot is not always known then the world within it operates is termed as *partially observable*. Partially Observable Markov Decision Processes (POMDPs) are an extension of the MDPs that include a set of observations that assist in determining the state the robot occupies.

Formally, a POMDP is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \mathcal{O} \rangle$, where

- $\mathcal{S}$, is a finite set of all possible states of the environment that the agent might occupy and are partially observable.
- $\mathcal{A}$, is a finite set of actions.
- $\mathcal{Z}$, is a finite set of observations.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$ is the *state transition function*, as defined for MDPs.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the *reward function*, as defined for MDPs.
- $\mathcal{O} : \mathcal{A} \times \mathcal{S} \rightarrow \Pi(\mathcal{Z})$ is the *observation function* giving for each state and agent action, a probability distribution over observations. $\mathcal{O}(s', a, z)$ is the probability of observing z, in state s' after taking action a, $p(z|s', a)$.

3.2.1 Belief State

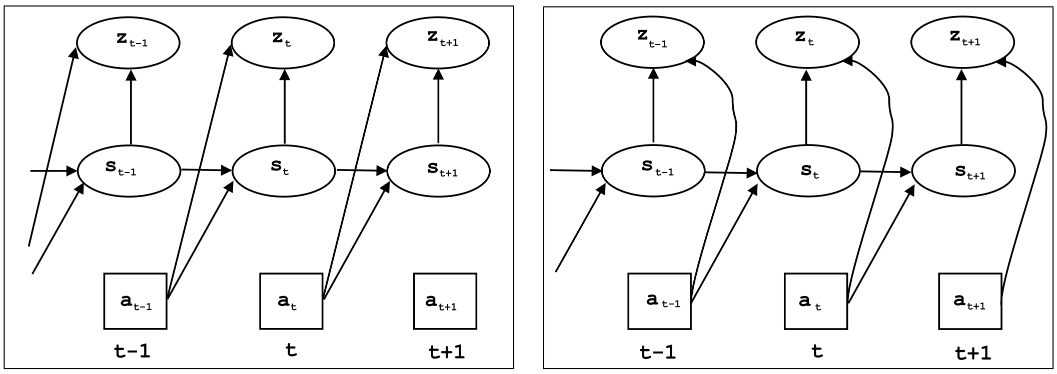
As already stated, in POMDPs the true state the robot occupies is not guaranteed to be known. Therefore, in POMDPs a discrete probability distribution is maintained over the set of environment states, $\mathcal{S}$, representing for each state the robot's belief that is currently occupying that state. This probability distribution is maintained at all times and is termed as the *belief state*, b_t, of the robot. Hence, $b_t(s)$ is the probability of the robot occupying state s at time t, $p_t(s : s \in \mathcal{S})$. The set of all possible belief states is $\mathcal{B}$. The belief state is updated every time the robot executes an action, based on the action it executed and the observation it perceived. Furthermore, when solving a POMDP to obtain an optimal policy, it provides a mapping between actions and belief states rather than actions and states as in the MDP case.

Belief Update

The state estimator component of a POMDP updates the belief state of the agent every time it executes an action. Given the belief state of the agent at time t, b_t, we would like to compute the belief state at time $t+1$, b_{t+1}, after a transition in the process where the agent occupies state s, executes an action a and perceives an observation z. The belief that the agent is in the resulting state s' is derived by:

$$\begin{aligned} b_{t+1}(s') &= p(s'|z,a,b_t) \\ &= \frac{p(z|s',a,b_t)p(s'|a,b_t)}{p(z|a,b_t)} \\ &= \frac{p(z|s',a,b_t)\sum_{s\in S} p(s'|a,b_t,s)p(s|a,b_t)}{p(z|a,b_t)} \\ &= \frac{p(z|s',a)\sum_{s\in S} p(s'|s,a)p(s|b_t)}{p(z|a,b_t)} \\ &= \frac{\mathcal{O}(s',a,z)\sum_{s\in S} \mathcal{T}(s,a,s')b_t(s)}{p(z|a,b_t)} \end{aligned} \tag{3.3}$$

In essence the above equation evaluates the probability of ending up in state s' given that the agent had a belief about its own state b_t, executed an action a and perceived an observation z according to the predefined observation and transition functions of the POMDP, $\mathcal{O}(\cdot)$ and $\mathcal{T}(\cdot)$ respectively. The denominator $p(z|a,b_t)$, is a normalizing factor and is equal to the total probability of perceiving the observation z given the previous belief state of the agent and the



(a) (b)

Figure 3.2: (a) The forward and (b) backward triggered observation model.

action it executed :

$$\begin{aligned} p(z|a,b_t) &= \sum_{s' \in S} p(z|s',a)p(s'|s,a)b_t(s) \\ &= \sum_{s' \in S} \mathcal{O}(z,s',a)\mathcal{T}(s,a,s')b_t(s) \end{aligned}$$

In the literature, it is frequently assumed that observations are forward triggered [52], i.e. the agent executed an action, transited to a new state and then it perceived an observation. Equation 3.3 for belief update is for POMDPs modelled with forward triggered observations. However, most practical applications are modelled by POMDPs with backward triggered observations, where the perceived observation refers to the "before action" state. The agent executes an action, transits to a new state but the observation is perceived from the state the agent occupied before executing the action. The concept of forward and backward observations is illustrated in Figure 3.2. The belief update for POMDPs with backward triggered observations is:

$$\begin{aligned} b_{t+1}(s') &= p(s'|z,a,b_t) \\ &= \frac{p(z|s',a,b_t)p(s'|a,b_t)}{p(z|a,b_t)} \\ &= \frac{\sum_{s \in S} p(s'|s,a)p(z|s,a)b_t(s)}{p(z|a,b_t)} \\ &= \frac{\sum_{s \in S} \mathcal{T}(s,a,s')\mathcal{O}(s,a,z)b_t(s)}{p(z|a,b_t)} \end{aligned}$$

where $p(z|a, b_t)$ is equal to:

$$\begin{aligned} p(z|a, b_t) &= \sum_{s \in S} p(z|s, a) b_t(s) \\ &= \sum_{s \in S} \mathcal{O}(z, s, a) b_t(s) \end{aligned}$$

3.3 Solving POMDPs

POMDPs are solved by value iteration as with MDPs. However, it has been already mentioned that a POMDP policy is a mapping between belief states and actions as opposed to a mapping between states and actions as in the MDP case. Therefore, Equation 3.1 that provides the t-step optimal value function for MDPs becomes:

$$V_t^*(b) = \max_{a \in \mathcal{A}} \left[\rho(b, a) + \gamma \sum_{b' \in \mathcal{B}} \tau(b, a, b') V_{t-1}(b') \right], \tag{3.4}$$

where $\mathcal{B}$ is the set of all possible belief states.

As it can be observed in the above equation, the defined transition, $\mathcal{T}(\cdot)$, and reward, $\mathcal{R}(\cdot)$, functions have been replaced by the functions $\tau(\cdot)$ and $\rho(\cdot)$, respectively. This is because the transition and reward functions have to be defined over a belief state, b, instead of a single state, since the true state of the agent is not completely known. Hence, the new functions are defined as:

$$\begin{aligned} \tau(b, a, b') &= p(b'|a, b) \qquad (3.5) \\ &= \sum_{z \in \mathcal{Z}} \sum_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} b(s) \mathcal{O}(z, s', a) \mathcal{T}(s, a, s') \end{aligned}$$

$$\rho(b, a) = \sum_{s \in S} b(s) \mathcal{R}(s, a). \tag{3.6}$$

By the above definitions, the evaluation of the expected reward of a t-step value function is now dependent on the observation perceived as well as the action it performed. In result, the

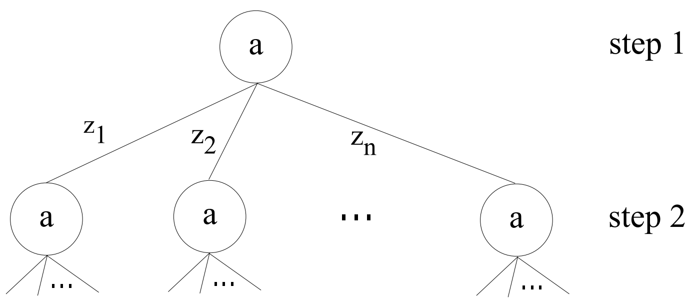


Figure 3.3: An example policy tree of a POMDP.

policy that has to be computed for a POMDP is actually a *policy tree*. An example policy tree is shown in Figure 3.3 for T steps. In this policy tree, the root node represents the first action to be executed and subsequent actions are determined by a subtree that is chosen dependent on the actual observation the robot perceived after executing an action.

Having defined the policy tree, Equation 3.4 is re-expressed by substituting Equations 3.5 and 3.6 as:

$$\begin{aligned} V_t^{\mathbf{p}}(b) &= \sum_{s\in S} b(s)\mathcal{R}(s,a_{\mathbf{p}}) + \gamma \sum_{z\in\mathcal{Z}}\sum_{s'\in\mathcal{S}}\sum_{s\in\mathcal{S}} b(s)\mathcal{O}(z,s',a_{\mathbf{p}})\mathcal{T}(s,a_{\mathbf{p}},s')V_{t-1}^{z_{\mathbf{p}}}(s') & (3.7)\\ &= \sum_{s\in S} b(s)\left[\mathcal{R}(s,a_p) + \gamma \sum_{z\in\mathcal{Z}}\sum_{s'\in\mathcal{S}} \mathcal{O}(z,s',a_{\mathbf{p}})\mathcal{T}(s,a_{\mathbf{p}},s')V_{t-1}^{z_{\mathbf{p}}}(s')\right] & (3.8)\\ &= \sum_{s\in S} b(s)V_t^{\mathbf{p}}(s) & (3.9)\\ &= b\cdot\alpha_t^{\mathbf{p}} & (3.10)\end{aligned}$$

Where $V_t^{\mathbf{p}}(b)$ is the value of executing the policy tree $\mathbf{p}$ of depth t and $a_{\mathbf{p}}$ is the root node action of the policy tree $\mathbf{p}$. $V_{t-1}^{z_{\mathbf{p}}}(s')$ is the value of executing the policy tree of depth $t-1$ that was entered when the observation $z_{\mathbf{p}}$ has been perceived by its parent policy tree $\mathbf{p}$.

In Equation 3.10, the so called α-vectors have been introduced, that is the set of hyperplanes defined by a policy tree $\mathbf{p}$ of depth t as:

$$\alpha_t^{\mathbf{p}} = \langle V_t^{\mathbf{p}}(s_1), ..., V_t^{\mathbf{p}}(s_n)\rangle$$

Table 3.2: POMDP value iteration algorithm.

```
$t = 0$
for all $\alpha$-vectors in $\Gamma_0$
    for all $s \in \mathcal{S}$
        $\alpha_0(s) = 0$
    end
end
while $\max_{b \in \mathcal{B}} |\max_{\alpha_t \in \Gamma_t} \alpha_t \cdot b - \max_{\alpha_{t-1} \in \Gamma_{t-1}} \alpha_{t-1} \cdot b| < \epsilon$
    $t = t + 1$
    for all policy trees $\mathbf{p}$
        for all $a \in \mathcal{A}$
            for all $s \in \mathcal{S}$
                $\alpha_t^{\mathbf{P}}(s) = \mathcal{R}(s,a) + \sum_{z \in \mathcal{Z}} \sum_{s' \in \mathcal{S}} \mathcal{O}(z, s', a_{\mathbf{p}}) \mathcal{T}(s, a_{\mathbf{p}}, s') \alpha_{t-1}^{z_{\mathbf{P}}}(s')$
            end
        end
    end
    $V_t^*(b) = \max_{\mathbf{p} \in \mathbf{P}} b \cdot \alpha_{\mathbf{p}}^t$
end
```

The set of all α-vectors is Γ.

Having defined the α-vectors the optimal value function is now expressed as:

$$V_t^*(b) = \max_{\mathbf{p} \in \mathbf{P}} b \cdot \alpha_{\mathbf{p}}^t$$

The value iteration algorithm for POMDPs is shown in Table 3.2.

Following, an example of computing a 2-step POMDP value function is shown graphically. A POMDP with 2 states, 2 actions and 2 observations will be considered. As in the MDP case we will start by computing the one-step value function and policy tree. There are two one-step policy trees, one for each possible action, with their associated value functions as shown in Figure 3.4. The value function for the one-step policy trees is simply the immediate reward for each action. In this example it is assumed that $R(s_1, a_1) = 0, R(s_1, a_2) = 1$ and

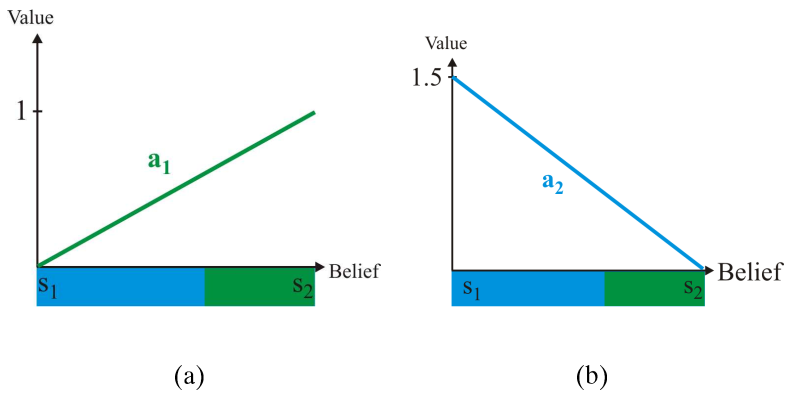


Figure 3.4: The one step POMDP value function for each action.

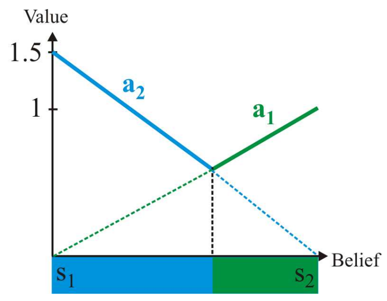


Figure 3.5: The complete one step POMDP value function.

$R(s_2, a_1) = 1.5, R(s_2, a_2) = 0$. Since there are only two states in this example POMDP, the belief is shown as a line ranging from zero to one, where the zero value represents that the robot is certain about occupying state 1 and as the value increases closer to 1 the robot is more probable that it occupies state 2. The complete one-step value function is the combination of the value function of all one-step policy trees and determines which policy tree is assigned to a region of the belief space, as shown in Figure 3.5. Therefore, if the belief is closer to state s_1 then action a_2 has a higher value and is the action to be executed.

The complete 2-step value function will be computed by building all possible 2-step policy trees. There are two possible 2-step policy trees, since there are two possible actions in the POMDP of this example and the root node of a policy tree is an action. There are also two leaf nodes in the 2-step policy trees since there are two possible observations and the consecutive action to be executed is dependent on the observation perceived after executing the initial action. In Figure 3.6 the policy tree for the initial action a_1 is shown.

To compute the value function of the policy tree shown in Figure 3.6, we need to evaluate the immediate reward and the expected reward. The evaluation of the immediate reward the robot will receive for taking action a_1 is straightforward and has been explained when evaluating the 1-step value function. However, when evaluating the expected reward we have to consider all possible observations, and how the robot's belief is transformed when it executes action a_1 and observes each of the two possible observations. The expected reward for taking action a_1 and observing each of the two possible observations is shown in Figure 3.7. The value functions shown in this figure are the result of transforming the complete one-step value function, shown in Figure 3.5, according to the observation and transition probabilities of the POMDP. The procedure for obtaining the two-step value function for executing a_1 as the initial action is shown graphically in Figure 3.8. In this figure, the belief distributions have been colored according to the action with the highest reward for each area of the belief. The bottom belief distribution, the one of the horizon 1 value function, shows with blue color the area where action 2 would be chosen and with green color the area where action 1 would be chosen. If we assume that the robot belief is represented by the vertical line denoted with b, then it can be seen what actions the robot would execute for each observation that it might perceive. Therefore, the initial action would be a_1, and in the case the robot perceives observation z_1 the consecutive action will be a_1, otherwise it will be action a_2. The value function shown is built by incorporating the transformed horizon 1 value functions to the appropriate regions of the belief after adding the immediate reward functions that have been evaluated previously and are shown in Figure 3.4. To obtain the complete two-step value function the same procedure is followed for having as initial action the action a_2.

In the procedure described, each policy tree produces a value function that is linear in the belief space. The complete t-step optimal value function, V_t^*, is the upper surface of all the

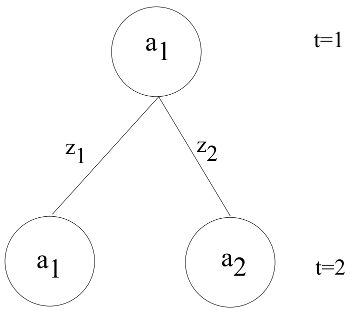


Figure 3.6: The two step policy tree for executing action a_1.

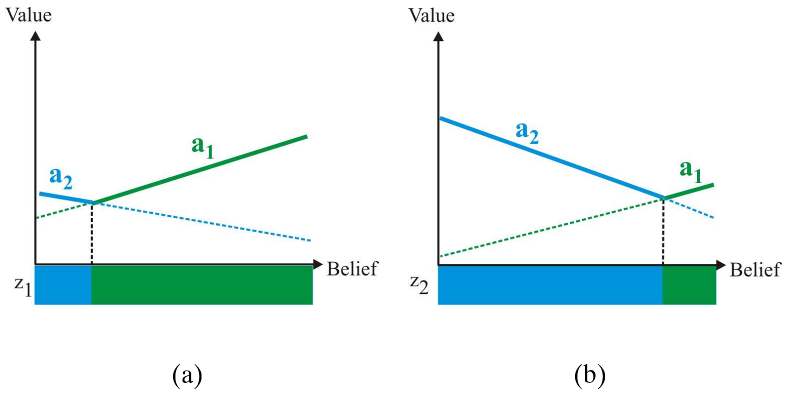


Figure 3.7: The transformed horizon 1 value function for taking action a_1 and perceiving each of the possible observations.

linear functions induced by each policy tree. Hence, the complete t-step optimal value function is piecewise-linear and convex.

3.3.1 Complexity of Solving POMDPs

When solving the POMDP exactly for a single step in time t, the time complexity is

$$\mathbf{O}\left(|\mathcal{S}|^2|\mathcal{A}||\Gamma_{t-1}|^{|\mathcal{Z}|}\right), \tag{3.11}$$

where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{Z}$ are as explained in Section 3.2 the set of states, actions and observations respectively. In Section 3.3 Γ has been defined as the set of all α-vectors.

The size of the set of α-vectors at any time t is equal to

$$|\Gamma_t| = |\mathcal{A}||\Gamma_{t-1}|^{|\mathcal{Z}|}. \tag{3.12}$$

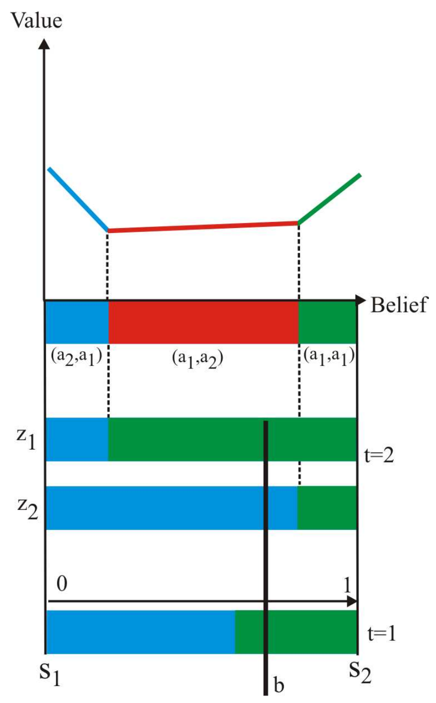


Figure 3.8: The two step value function for executing action a_1 as the initial action.

The doubly-exponential nature of the POMDP solution is apparent by Equations 3.11 and 3.12. Furthermore, it should be noted that the POMDP is solved over the whole belief state $\mathcal{B}$, that has dimensionality $|\mathcal{S} - 1|$ and hence in real-world applications it can become extremely large.

By the above, it is obvious that solving exactly POMDPs is a computationally expensive procedure. The complexity of solving a POMDP at a finite horizon is PSPACE-complete whereas in the infinite horizon case it is undecidable [72]. This computational complexity has lead into many methods used for pruning α-vectors that do not contribute to the value function. These vectors are the ones that are dominated in the whole belief space by other α-vectors. One of the most popular algorithms for pruning is the Witness algorithm [71].

Although pruning algorithms manage to control to some extent the exponential increase of the number of α-vectors, still the size of POMDPs they can handle is some tens of states. Hence, many approximation methods have been proposed in the literature and are detailed in the following section.

3.3.2 Approximation Methods

Some of the most commonly used approximations for solving POMDPs so far are heuristic methods that are based on solving the underlying MDP. These methods offer a great complexity reduction but they use greedy heuristics to compensate the partial observability of the robot's state.

The *most likely state* (MLS) [84] heuristic solves the underlying MDP for the state with the highest assigned probability. Therefore, the value function becomes:

$$V_t^*(s) = \max_{a \in \mathcal{A}} Q(\arg\max_{s \in \mathcal{S}}(b(s)), a)$$

where the Q-function has been defined in Equation 3.2 for solving MDPs and provides the expected reward when being in a state s and executing an action a. Therefore, the MLS heuristic solves the MDP only for the state with the highest probability of being occupied by the robot. However, the full belief is maintained at all times. It is obvious by the above that the MLS heuristic will decide a wrong policy in cases where the uncertainty in the belief is high.

Another MDP-based heuristic is the *voting* heuristic [108]. In the voting heuristic the underlying MDP of all possible states is solved and the action obtained is weighted according to the belief probability distribution. The optimal value function in this case is:

$$V_t^*(s) = \max_{a \in \mathcal{A}} \sum_{s \in \mathcal{S}} b(s) \delta(\pi_{MDP(s)}, a)$$

where

$$\pi_{MDP(s)} = \arg\max_{a \in \mathcal{A}} Q(s, a)$$

and

$$\delta(a_i, a_j) = \begin{cases} 1, & \text{if } a_i = a_j \\ 0, & \text{if } a_i \neq a_j \end{cases}$$

The voting heuristic compensates the partial observability in a better manner than the MLS heuristic but still there are cases depending on the belief distribution that it might fail.

There are numerous other approximation techniques present in the literature that have been reviewed thoroughly in [1, 51, 99]. More recent approximation methods are those based on state-space compression [93], belief compression [99] and point-based value iteration where the POMDP is solved for a sampled set of belief points [51, 91, 92, 110].

All of the approximation methods mentioned above have been applied successfully to problems with at most a few thousand states. As a result these methods have been applied to POMDPs used as high level planners. However, the navigation problem in realistic environments and the use of POMDPs as a unified navigation model, i.e. the problem considered in this thesis, is orders of magnitude larger than these approximation methods can handle.

3.4 Learning POMDPs

Learning POMDPs involves determining the structure of the POMDP, i.e. the probability matrices that are comprised by the transition probabilities, $\mathcal{T}(s, a, s')$, and the observation probabilities, $\mathcal{O}(s', a, z)$.

The most commonly used method for learning POMDPs is the Baum-Welch algorithm that is derived by the theory of Expectation-Maximization (EM) algorithms. EM algorithms attempt to iteratively adjust the model parameters such that the likelihood that the obtained training data were generated by the current model parameters is maximized. Formally, this is expressed as $\max P(M|\lambda)$, where M is the obtained training data and λ is the current model parameters.

The Baum-Welch algorithm [64], that is an adaptation of the EM algorithm, is applied for learning the POMDP probability matrices. The training data M is comprised of action and observation pairs, $M = \langle (a_1, z_1), (a_2, z_2), ..., (a_T, z_T) \rangle$, where T is the length of the execution trace. The model parameters λ are the probability matrices and the initial probability distribution b, $\lambda = \langle \mathcal{T}, \mathcal{O}, b \rangle$.

The Baum-Welch algorithm uses a forward and a backward propagation to calculate the following probability distributions in order to update the model parameters.

alpha values, $\alpha_t(s)$: The probability that the agent occupies state s at time t given the observation-action pairs until that time.

$$\alpha_t(s) = p(s_t = s | z_{1,\ldots,t-1}, a_{1,\ldots,t-1}).$$

beta values, $\beta_t(s)$: The probability of obtaining the observation sequence of the training data from time t to the end of the execution trace, given that the agent occupies state s at time t and executes the action sequence of the training from time t to the end of the execution trace.

$$\beta_t(s) = p(z_{t+1,\ldots,T} | s_t = s, a_{t,\ldots,T}).$$

gamma values, $\gamma_t(s, s')$ and $\gamma_t(s)$: The gamma values are more precise estimates of the same probability distribution as the alpha values since they utilize the whole execution trace.

$$\gamma_t(s, s') = p(s_t = s, s_{t+1} = s' | z_{1,\ldots,T}, a_{1,\ldots,T})$$

$$\gamma_t(s) = p(s_t = s | z_{1,\ldots,T}, a_{1,\ldots,T})$$

scaling factors, $scale_t$: The scaling factors are used to prevent underflowing when the algorithm runs for long training sequences.

$$scale_t = p(z_t | z_{1,\ldots,t-1}, a_{1,\ldots,t-1})$$

Having defined the variables used by the Baum-Welch algorithm, the forward-backward propagation, a dynamic programming approach, that applies Bayes' rule repeatedly to calculate the scaling factor, alpha, beta and gamma values is performed as follows:

1. Initialize the scale factor and alpha values as:

$$scale_1 = \sum_{s \in \mathcal{S}} p(z_1|s)p(s_1 = s)$$

$$\alpha_1(s) = p(z_1|s)p(s_1 = s)/scale_1, \text{ for all } s \in \mathcal{S}$$

2. Determine the scale factor and alpha values by forward propagation as:

$$scale_t = \sum_{s \in \mathcal{S}} \left[p(z_1|s) \sum_{s' \in \mathcal{S}} \left[p(s|s', a_{t-1})\alpha_{t-1}(s) \right] \right], \text{ for } t = 2, ..., T$$

$$\alpha_t(s) = p(z_t|s) \sum_{s' \in \mathcal{S}} \left[p(s|s', a_{t-1})\alpha_{t-1}(s) \right] /scale_t, \text{ for } t = 2, ..., T \text{ and for all } s \in \mathcal{S}$$

3. Initialize the beta values as:

$$\beta_T(s) = 1/scale_T, \text{ for all } s \in \mathcal{S}$$

4. Determine the beta values by backward propagation as:

$$\beta_t(s) = \sum_{s' \in \mathcal{S}} \left[p(s'|s, a_t)p(z_{t+1}|s')\beta_{t+1}(s') \right] /scale_t, \text{ for all } s \in \mathcal{S} \text{ and for } t = T{-}1, ..., 1$$

5. Determine the gamma values as:

$$\gamma_t(s, s') = \alpha_t(s)p(s'|s, a_t)p(z_{t+1}|s')\beta_{t+1}(s'), \text{ for all } s, s' \in \mathcal{S} \text{ and for } t = 1, ..., T - 1$$

$$\gamma_t(s) = scale_t\alpha_t(s)\beta_t(s), \text{ for all } s \in \mathcal{S} \text{ and for } t = 1, ..., T$$

Finally, the transition and observation probabilities are updated as:

$$p(s'|s, a) = \frac{\sum_{t=1,...,T-1|a_t=a} \gamma_t(s, s')}{\sum_{1,...,T-1|a_t=a} \gamma_t(s)}, \text{ for all } s, s' \in \mathcal{S} \text{ and } a \in \mathcal{A}$$

$$p(z|s) = \frac{\sum_{t=1,...,T|z_t=z} \gamma_t(s)}{\sum_{1,...,T} \gamma_t(s)}, \text{ for all } s \in \mathcal{S} \text{ and } z \in \mathcal{Z}$$

3.5 Conclusions

This chapter presented the mathematical background of POMDPs. The methodology used to learn and solve a POMDP has been also presented. This background is required for the next chapter that presents the proposed hierarchical representation of POMDPs utilized for the autonomous navigation problem. Finally, the complexity of solving POMDPs has been illustrated that necessitated the need for the hierarchical POMDP that is proposed in the next chapter.

4

Hierarchical POMDPs

In this chapter, a hierarchical representation of POMDPs for autonomous robot navigation (RN-HPOMDP) that can effectively model large real world environments at a fine resolution is presented. The proposed RN-HPOMDP can be solved in real time. It is utilized as a unified framework for autonomous robot navigation, implying that no other external modules are used to drive the robot. The RN-HPOMDP integrates the modules for localization, planning and local obstacle avoidance; it is solved on-line at each time step and decides the actual actions the robot performs.

Two other HPOMDP approaches are currently present in the literature that employ either state space hierarchy [119], applied as a high level mission planner, or action and state space hierarchy [89], applied for high level robot control and dialogue management. Independently and concurrently with these works, we have come up with a HPOMDP* that applies both state space and action space hierarchy. It is specifically designed for the autonomous robot

*Preliminary versions of the RN-HPOMDP are presented in [33, 34].

navigation problem, hence the term RN-HPOMDP, and offers specific advantages over the two approaches mentioned above.

POMDP solution methods suffer from the “curse of dimensionality” [59] and also the “curse of history” [91]. Applying both state space and action space hierarchy, as in the RN-HPOMDP, both curses can be harnessed. In the following the structure of the RN-HPOMDP is presented along with the methodology used for learning and planning with the RN-HPOMDP in Sections 4.4 and 4.5, respectively. A detailed comparison of our approach and the other two approaches present in the literature can be found in Section 4.7.

The formulation of a flat POMDP for the predictive navigation problem is first presented that will be explained in the following sections of this chapter how it is transformed to a hierarchical POMDP.

4.1 POMDP Formulation for Robot Navigation

In the following we present a formulation of flat POMDPs for autonomous robot navigation in a unified framework. The POMDP decides the actions the robot should perform to reach its goal and also robustly tracks the robot’s location in a probabilistic manner. In the problem considered in this thesis, we are interested in dynamic environments and hence the POMDP also performs obstacle avoidance. All three functionalities are carried out without the intervention of any other external module.

In our implementation the robot perceives the environment by taking horizontal laser scans. In addition, an occupancy grid map (OGM) of the environment obtained at the desired discretization is provided. The OGM is used to determine the set of possible states the robot might occupy. Laser measurements are used to obtain observations. In the following, the elements of a POMDP, $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \mathcal{O} \rangle$, are instantiated for robot navigation as:

set of states, $\mathcal{S}$: Each state in $\mathcal{S}$ corresponds to a discrete entry cell in the environment's occupancy grid map (OGM) and an orientation angle of the robot with respect to a global reference system, i.e. each state s is a triplet (x, y, θ).

set of actions, $\mathcal{A}$: It consists of all possible rotation actions from $0°$ to $360°$ termed as "action angles". The discretization of the robot orientation angles and action angles depends on the number of levels of the POMDP hierarchy (see later Section 4.2).

set of observations, $\mathcal{Z}$: The observation set is the element of the POMDP that assists in the localization of the robot, that is the belief update after an action has been taken. The set of observations is instantiated as the output of the *iterative dual correspondence* (IDC) [75] algorithm for scan matching. At each time step, an observation is obtained by feeding the IDC with the current scan of the robot and a reference scan of the environment in which the robot operates. The IDC also requires an estimate of the robot's position from which the current scan was obtained, which is given as the robot's position before it performed the action. This position is taken to be the most likely state of the robot's belief state. This definition of the observation function enables us to define it independently of the actual environment structure and dependent only on the robot motion model. The belief distribution is not initialized as a uniform distribution but rather based on the assumption that the robot has a good estimate of its initial position. It is reasonable to assume that the robot cannot move too far from its previous state at a single time step. Therefore, the output of the IDC algorithm, that is the dx, dy and $d\theta$ from the estimated location provided, will be within certain limits. The output of the IDC algorithm is discretized and thus the set of observations remains small and manageable.

reward function, $\mathcal{R}$: Since the proposed POMDP is used as a unified framework for robot navigation that will provide the actual actions the robot will perform and also carry out obstacle avoidance for moving objects, the reward function is updated at each time step. The reward function is built and updated at each time step, according to two *reward grid maps* (RGMs): a *static* and a *dynamic* [33]. The RGM is defined as a grid map of the environment in analogy with the OGM. Each of the RGM cells corresponds to a specific area of the environment with the same discretization of the OGM, only that the value associated with each cell in the RGM represents the reward that will be assigned to the robot for ending up in the specific cell. The static RGM is built once by calculating the distance of each cell to the goal position and by incorporating information about cells belonging to static obstacles. The dynamic RGM is responsible for incorporating into the model information about the current state of the environment, i.e. whether there are objects moving within it or other unmapped objects. In our implementation the robot perceives the environment by taking horizontal laser scans. Hence, at each time step the current laser scan is used to detect the location of objects that are not present in the map. The location of all detected objects form the dynamic RGM where the corresponding cell values are zeroed. Superimposing the static and dynamic RGMs provides the reward function that is updated at each time step. It should be noted that the choice of including in the reward function information about moving objects has alleviated the need of modelling moving objects as observations. Modelling the position of moving objects as observations would increase dramatically the size of observations since it would have to be at least equal to the size of the grid of the modelled environment. The detailed procedure for building the RGMs and also the update procedure is given in Section 5.4.

transition and observation functions, $\mathcal{T}$ and $\mathcal{O}$: They are initially defined according to the motion model of the robot and then they are learned as explained in Section 4.4. Since observations have been defined to depend only on the robot motion when an action is executed, the observation function can also be defined according to the motion model. That is, according to the robot motion model the expected displacement of the robot (dx, dy, $d\theta$) for executing each action can evaluated and hence form the initial observation function that is learned following the procedure described in 4.4.

The above instantiation of a POMDP for robot navigation refers to a flat POMDP and in the following sections it will be explained how these elements are embedded into the proposed hierarchical POMDP. The described POMDP will be referred to as the corresponding flat POMDP hereinafter.

4.2 The RN-HPOMDP Structure

The RN-HPOMDP is built through an automated procedure using as input a map of the environment and the desired discretization of the state and action space. The map of the environment can be provided as either a probabilistic grid map obtained at the desired discretization or a CAD map. The grid map provided or obtained by the CAD map is used to determine the state space of the corresponding flat POMDP. The flat POMDP is not required to be built and maintained; it is used only as a virtual structure that is used to built the levels of hierarchy of the RN-HPOMDP.

4.2.1 The corresponding flat POMDP

The state space of the corresponding flat POMDP is composed of all possible states that the robot might occupy represented by (x, y, θ) triplets, where (x, y) are the coordinates of the robot location in the grid map and θ is its orientation. The set of orientation angles is composed of all directions from $0°$ to $360°$ discretized at a desired step of ϕ degrees. The action space is also composed of all rotation actions from $0°$ to $360°$, termed as *action angles*, discretized with the same step size of ϕ degrees as with the orientation angles. The set of observations does not need to be defined in this virtual structure of the flat POMDP since it is not decomposed in the hierarchical structure of the RN-HPOMDP. Furthermore, the transition, observation and reward function are also not defined for the corresponding flat POMDP since they will be defined directly for each POMDP of the RN-HPOMDP.

4.2.2 Determining the number of levels of hierarchy of the RN-HPOMDP

The RN-HPOMDP structure is built by decomposing the corresponding flat POMDP with large state and action space into multiple POMDPs with significantly smaller state and action spaces. Therefore, in all levels other than the bottom level, POMDPs are composed of states and actions that have a coarse discretization and do not represent the actual state the robot occupies or the actual action the robot will perform. Hence they are termed as *abstract states* and *abstract actions* [119].

The process of building the hierarchical structure is performed in a top-down approach. The number of levels of the hierarchical structure is determined by the desired discretization of the action angles or the orientation angles, since their discretization is the same in the RN-HPOMDP structure. The discretization of the orientation and action angles has been chosen to be the same in the RN-HPOMDP structure but this is a designer's choice and is not compromising as it does not affect the performance of the RN-HPOMDP.

Thus, if the desired discretization of the action angles or the orientation angles is ϕ degrees, the number of levels of the RN-HPOMDP structure, L, will be

$$L = \log_2(90^\circ/\phi) + 1.$$

As explained in the following sections, the top-level of the RN-HPOMDP has a discretization of angles of 90° and at each subsequent level the discretization is doubled. Hence, the number of levels of the hierarchical structure is given by the log_2 of the ratio of the top-level discretization and the desired discretization plus one level that is the top-level.

The number of levels of the RN-HPOMDP structure in conjunction with the desired discretization of the state space affects the size of the top-level POMDP and in effect the performance of the RN-HPOMDP in time complexity as it will be explained in the following sections. Therefore, the choice of the number of levels of the RN-HPOMDP structure should be made by considering the desired discretization of the state and action space but also the resulting size of the top-level POMDP.

4.2.3 Construction of the top-level of the RN-HPOMDP

The top level of the hierarchical structure is composed of a single POMDP with very coarse resolution. Hence it can represent the whole environment with a small number of abstract states. The grid resolution of the top level states is equal to $d \times 2^{L-1}$, where d is the desired discretization of the whole RN-HPOMDP structure and L is the number of levels of the structure. The orientation angle of the robot and the action angles are also discretized in a very coarse resolution of $90°$ and thus represent the basic four directions $[0°, 90°, 180°, 270°]$.

The total number of states of the top level POMDP is equal to $|\mathcal{S}^0|/2^{2(L-1)}$, where $|\mathcal{S}^0|$ is the number of states of the corresponding flat POMDP. The number of states of the top level POMDP is reduced once by 2^{L-1} because of the coarser grid resolution and again by 2^{L-1} because of the coarser resolution of the orientation angle, as compared to the corresponding flat POMDP respectively.

To summarize, the top-level is always composed of a single POMDP with predefined discretization of the orientation and action angles at $90°$. The state space size of the top-level POMDP is variable and dependent to the discretization of the corresponding flat POMDP and the number of levels of the hierarchical structure. Hence, the number of levels of the RN-HPOMDP structure, L, should be such that it ensures that the size of the top-level POMDP remains small.

4.2.4 Construction of the intermediate levels of the RN-HPOMDP

Subsequent levels of the RN-HPOMDP are composed of multiple POMDPs, each one representing a small area of the environment and a specific range of orientation angles. The actions of an intermediate level POMDP are a subset of the actions of the corresponding flat POMDP.

In detail, each state of the top level POMDP corresponds to a POMDP at the immediate next level, as we go down the hierarchical structure. A POMDP at an intermediate level l, has states that represent grid locations of the environment at a resolution of $d \times 2^{(L-l)}$, where l is the current intermediate level. Thus, by going down the hierarchical structure the grid resolution of a level's POMDP is twice the resolution of the previous level. Therefore, when a top level state, that corresponds to a specific grid location, is decomposed it will be represented in the immediate next level POMDP by an area of 2×2 cells with double resolution than the top level's resolution.

Orientation angle decomposition

Going down the hierarchical structure, the resolution of the orientation angle is also doubled. Since the resolution of the orientation angle is increased as we go down the hierarchical structure, the whole range of possible orientation angles, $[0°, 360°]$, cannot be represented in every intermediate level POMDP. This would dramatically increase the size of the state space and therefore we choose to have many POMDPs that represent the same grid location but with a different range of orientation angles.

The range of orientation angles that is represented within each intermediate level POMDP is expressed in terms of the orientation angle, θ_p, of the previous level state that is decomposed, and is equal to

$$\left[\theta_p - \frac{90^\circ}{2^{l-2}}, \theta_p + \frac{90^\circ}{2^{l-2}}\right],$$

where l is the current intermediate level. By the above expression of the range of orientation angles, every intermediate level POMDP will always have five distinct orientation angles.

For example, if the state of the top level POMDP, $l = 1$, has orientation angle $\theta_p = 90^\circ$, the range of orientation angles at the next level, $l = 2$, will be equal to $[0^\circ, 180^\circ]$. As mentioned earlier the angle resolution of the top level is always equal to 90° and the next level will have double resolution, i.e. 45°. Therefore, the range of orientation angles $[0^\circ, 180^\circ]$ will be represented by five distinct orientation angles. As shown in Figure 4.1, the grid location represented by the top level state is decomposed into four POMDPs, where each one represents a different range of possible orientation angles. Consequently, the size of the state space for every intermediate level POMDP is constant and equal to 20, since it always has five possible orientation angles and it represents a 2×2 area of grid locations. It is easily deduced by the expression that determines the range of orientation angles for an intermediate level POMDP, that a grid location represented by a state will correspond at the next level l to $2^{2(l-1)}$ POMDPs.

Action angle decomposition

Action angles are decomposed from the top level POMDP to the next intermediate level in the same manner as with the orientation angles. The resolution of the action angles at each level is the same as the resolution of the orientation angles. Hence, it is equal to $90^\circ/2^{l-1}$. As a result, a top level state is also decomposed into multiple POMDPs, each one with a different range of orientation angles but also with a different range of action angles. The range of an action set is equal to

$$\left[a_p - \frac{90^\circ}{2^{l-2}}, a_p + \frac{90^\circ}{2^{l-2}}\right],$$

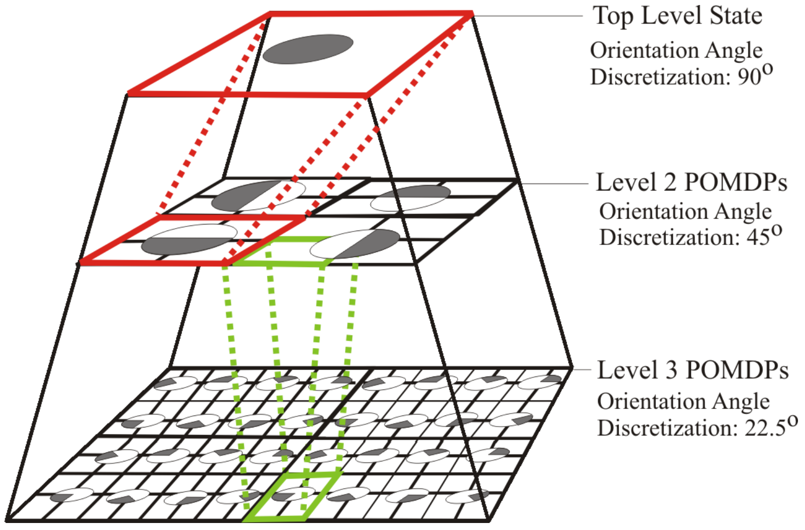


Figure 4.1: State space hierarchy decomposition. The figure depicts the decomposition of a top level state to lower level states. The top level state corresponds to 4 POMDPs at level 2, each one decomposing the location of the top level state into 4 locations, and its orientation in one of the ranges denoted by the shaded region of the circles for each POMDP. This state decomposition continues at lower levels until the desired discretization of the environment has been reached.

where a_p is the previous level action and l is the current intermediate level. The action angles set is also always composed by five distinct actions according to the above expression.

4.2.5 Construction of the bottom-level of the RN-HPOMDP

The procedure described in the previous section is used to built all intermediate levels of the hierarchical structure until the bottom level is reached. Bottom level POMDPs' state and action space is discretized at the desired resolution as a flat POMDP would be discretized. Hence, bottom level POMDPs' states are the actual states the robot occupies and equivalent to the corresponding flat POMDP states. Similarly, bottom level POMDPs' actions are the actual actions the robot executes and equivalent to the actions of the corresponding flat POMDP.

Table 4.1: Properties of the RN-HPOMDP structure with L levels.

	Top Level	**Intermediate Level l**	**Bottom Level**
No of POMDPs	1	$\|\mathcal{A}^{l-1}\| \times \|\mathcal{S}^{l-1}\|$	$\|\mathcal{A}^{L-1}\| \times \|\mathcal{S}^{L-1}\|$
Size of $\mathcal{S}$	$\|\mathcal{S}^0\|/2^{2(L-1)}$	20	$5 \times (2+r)^2$
Range of orientation angles	$[0°, 360°]$	$\left[\theta_p - \frac{90°}{2^{l-1}}, \theta_p + \frac{90°}{2^{l-1}}\right]$	$\left[\theta_p - \frac{90°}{2^{L-1}}, \theta_p + \frac{90°}{2^{L-1}}\right]$
Resolution of orientation angles	$90°$	$90°/2^{l-1}$	$90°/2^{L-1}$
Size of $\mathcal{A}$	4	5	5
Range of action angles	$[0°, 360°]$	$\left[a_p - \frac{90°}{2^{l-2}}, a_p + \frac{90°}{2^{l-2}}\right]$	$\left[a_p - \frac{90°}{2^{L-2}}, a_p + \frac{90°}{2^{L-2}}\right]$
Resolution of action angles	$90°$	$90°/2^{l-1}$	$90°/2^{L-1}$

The bottom level is composed of multiple POMDPs having the same properties as all other intermediate levels' POMDPs, only that the grid location the bottom level POMDPs represent is overlapping by a region r. Overlapping regions are required to be able to solve the bottom level POMDPs for border location states. Table 4.1 summarizes the properties of the RN-HPOMDP structure.

4.3 The Reference POMDP (rPOMDP)

In the previous section, it has been described how state space and action space abstraction is performed in the RN-HPOMDP structure, to enable real-time solution. However, the maintenance of the transition and observation functions of each POMDP in the RN-HPOMDP structure has not been discussed yet.

The corresponding flat POMDP would require to hold a transition matrix of size $(|\mathcal{S}^0|^2 \times |\mathcal{A}^0|)$ and an observation matrix of size $(|\mathcal{S}^0| \times |\mathcal{A}^0| \times |\mathcal{Z}|)$, where $|\mathcal{S}^0|$ and $|\mathcal{A}^0|$ are the sizes of the state space and action space, respectively, of the corresponding flat POMDP. The size of the observation space, $|\mathcal{Z}|$, is the same for the flat POMDP and the RN-HPOMDP since there is no observation space hierarchy.

The RN-HPOMDP structure requires to hold the transition and observation matrices for all the POMDPs at all levels. As can be seen in Table 4.1, the number of POMDPs at each level is large and dependent on the size of action space and state space. Consequently, even thought each POMDP's observation and transition matrix is small, the total memory requirements would be extremely large. The RN-HPOMDP has in total larger memory requirements than the corresponding flat POMDP, although the flat POMDP memory requirements are already very hard to manage for large real-world environments. For this reason, the notion of the *reference* POMDP (rPOMDP) is introduced.

The transition and observation matrices hold probabilities that carry information regarding motion and sensor uncertainty. The rPOMDP will be used to model motion and sensor uncertainty in a significantly smaller fraction of the world that the robot operates and then transfer this information to the RN-HPOMDP that models the whole world. This is accomplished by exploiting the feature that transition and observation probabilities can be defined using only the relative state of the robot instead of its actual state in the whole world.

The transition probability from a state s to a new state s', when the robot has performed an action a, is only dependent on the action a. Therefore when the robot is executing an action a, the transition probability will be the same for any state s when the resulting state s' is defined relatively to the initial state s.

The probability that the robot observes a feature z, when it is in a state s and performs an action a, can also be defined in the same manner as with the transition probabilities, since the set of features $\mathcal{Z}$ is the result of the scan matching algorithm when feeded with a reference laser scan and the actual scan the robot perceived (cf. Section 4.1). Therefore, perceived features are dependent on the actual motion of the robot, i.e. the action a it performed. Consequently observation probabilities can be defined using a relative initial state.

The rPOMDP is used to model the transition and observation probabilities that can occur in a single time-step of a robot movement, i.e. when the robot performs a single action a, using a relative initial state. Therefore, the rPOMDP needs to hold all possible transitions for executing any single action a regarding that the robot is always located in the same invariant position within the world. This has the effect that the rPOMDP state space size is dependent only on the largest possible transition that can occur in a single time-step that is independent from the actual size of the environment within the robot operates. In essence, the rPOMDP exploits the feature that the transition and observation probabilities are defined in Section 4.1 in a such a manner that they are only robot dependent instead of environment dependent.

4.3.1 Construction of the rPOMDP

The rPOMDP is built by defining a very small state space, defined as an $R \times R$ square grid (in our implementation $R = 7$) representing possible locations of the robot and all the orientation angles of the robot that would be assigned in the flat POMDP. The center location of the state space represents the invariant state s_r of the robot. The action and observation spaces are defined in the same manner they would be defined for the corresponding flat PODMP. This rPOMDP requires to hold transition and observation matrices of size $((R \times 2^{2+L})^2 \times |\mathcal{A}|)$ and $((R \times 2^{2+L}) \times |\mathcal{A}| \times |\mathcal{Z}|)$, respectively. The size of the matrices is only dependent on the size of the set of actions and observations and the number of levels of hierarchy, L, since the number of levels defines the discretization of the robot's orientation angle. By the above, it is obvious that no matter how big is the environment that is to be modelled with the RN-HPOMDP the

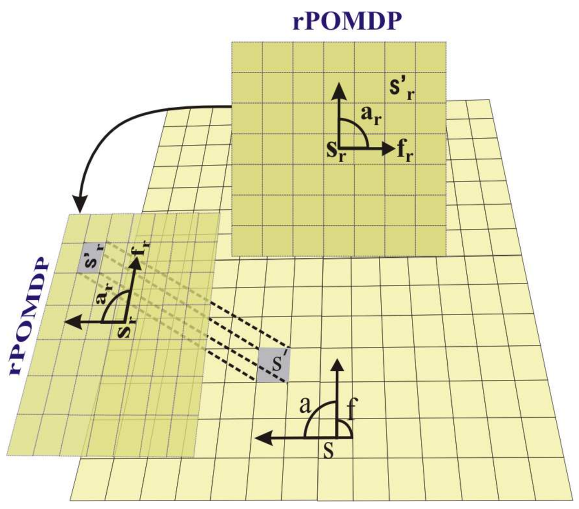


Figure 4.2: Translation and rotation of the rPOMDP transition probabilities matrix.

use of the rPOMDP allows to have reasonably sized matrices, depending on the choice made for R, that are easy to maintain and learn.

Given the rPOMDP, transition and observation probabilities for each POMDP in the RN-HPOMDP hierarchical structure are obtained by translating and rotating the reference transition and observation probability distributions over the current POMDP state space, as shown in Figure 4.2. In this figure it is shown that the rPOMDP transition probabilities are translated and rotated such that the resulting rPOMDP state, s_r', corresponds to the resulting state of the robot, s', for which the transition probability is desired to be determined and the reference action a_r is equivalent to the actual action the robot executed, a. The transfer of probabilities is performed on-line while a POMDP is solved or the robot's belief is updated.

The transition probability for any POMDP of the hierarchical structure, $\mathcal{T}(s, s', a)$, i.e. the probability of the robot ending up in state s' when it initially occupies a state s and executes an action a, is equivalent to the transition probability of the rPOMDP, $\mathcal{T}_r(s_r, s_r', a_r)$. The initial state of the rPOMDP s_r, is the invariant state of the rPOMDP. The invariant state s_r is decomposed as the location and orientation triplet (x_r, y_r, f_r), where the invariant location

(x_r, y_r) is the center location of the grid that represents the rPOMDP state space and the invariant orientation f_r is the orientation angle of 0°. The invariant state has been chosen to be represent by the center location of the grid so that the resulting states can also be represented within the reduced size grid of the rPOMDP. Actually, the size of the grid size of the rPOMDP, R, has to be chosen such that it ensures that the transitions can be represented within its constrained size. In our implementation, where R is equal to 7, it implies that the robot can move in a single time step within a radius of 3 grid cells. The choice of the invariant orientation angle is clearly a designer's choice and it does not affect the performance of the probability transfer between the rPOMDP and any POMDP of the hierarchical structure.

To obtain the desired transition probability, a rotation of the rPOMDP transition probabilities is performed by determining the reference action by

$$a_r = a + f - f_r.$$

Following, a translation of the rPOMDP transition probabilities is performed to obtain the probability of resulting to the reference result state, s'_r, that is determined by the following equation:

$$\begin{bmatrix} x'_r \\ y'_r \\ f'_r \end{bmatrix} = \begin{bmatrix} x_r \\ y_r \\ f_r \end{bmatrix} + \begin{bmatrix} x' - x \\ y' - y \\ f' - f \end{bmatrix},$$

where, the states s, s', s_r and s'_r are decomposed to the location and orientation triplets $(x, y, f), (x', y', f'), (x_r, y_r, f_r)$ and (x'_r, y'_r, f'_r), respectively.

In essence, the rPOMDP makes use of the fact that when a robot executes a rotation action with constant speed it will translate and rotate by exactly the same amount no matter where exactly in world it is.

In the same manner, the observation probability for any POMDP of the hierarchical structure, $\mathcal{O}(s, z, a)$, is equivalent to the observation probability of the rPOMDP, $\mathcal{O}_r(s_r, z_r, a_r)$. The reference observation, z_r', is now determined as:

$$\begin{bmatrix} dx_r \\ dy_r \\ df_r \end{bmatrix} = \begin{bmatrix} d\cos(f_r + a_r) \\ d\sin(f_r + a_r) \\ df \end{bmatrix},$$

where the observations z and z_r are decomposed into (dx, dy, df) and (dx_r, dy_r, df_r), respectively, since observations are defined as the position and angle difference between laser scans, and d is the distance $d = \sqrt{dx^2 + dy^2}$.

It should be stressed out that the capability of transferring the observation probabilities between the rPOMDP and a POMDP of the hierarchical structure is due to way the have been defined to be dependent only on the robot motion.

4.4 RN-HPOMDP Learning

Since a POMDP is a probabilistic model, learning the parameters of this model, i.e. the transition and observation matrices, is crucial to the performance of the POMDP model and specifically to its performance in keeping track of the robot's true position and orientation.

For the RN-HPOMDP structure, learning is performed only for the *reference* POMDP, since the latter transfers its learned parameters to the whole hierarchical structure, as described in Section 4.3.

Learning the rPOMDP parameters is performed by initializing the probability matrices and adjusting their parameters iteratively according to an execution trace, that is composed of action and observation pairs, to maximize the likelihood that the execution trace was obtained by the model. The Baum-Welch [64] algorithm, detailed in Section 3.4, is utilized for this purpose.

Since learning is performed only for the rPOMDP, when collecting data for the execution

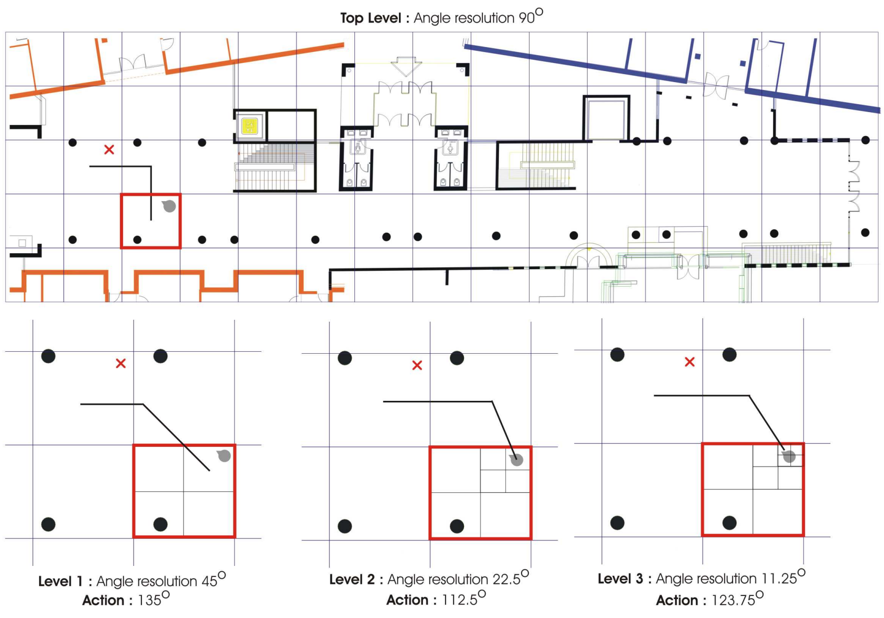

Top Level : Angle resolution 90°
Level 1 : Angle resolution 45°
Action : 135°
Level 2 : Angle resolution 22.5°
Action : 112.5°
Level 3 : Angle resolution 11.25°
Action : 123.75°

Figure 4.3: Planning with the RN-HPOMDP.

trace the observation and action pairs are converted to *reference* observations and actions. Conversion is performed by the inverse procedure described in Section 4.3. Consequently, learning is performed very fast since the rPOMDP has a very small state space. Evaluation results of the learned model are presented in Section 7.2.

4.5 RN-HPOMDP Planning

Solving the RN-HPOMDP to obtain the action the robot should perform, involves solving a POMDP at each level. The intuition of the RN-HPOMDP solution is to obtain at first a coarse path that the robot should follow to reach a goal position, and then refine this path at each subsequent level in the area that the robot's current position lies, as shown in Figure 4.3. The algorithm that implements the above is presented in Table 4.2 and its details are explained in the following.

During the RN-HPOMDP planning procedure the belief distribution of the corresponding flat POMDP is maintained at all times. This distribution will be denoted as the *full belief*.

Before solving any POMDP at any level, the *full belief* is compressed, by the functions `compressTopBelief()` and `compressBelief()`, to obtain the belief distribution of the POMDP to be solved. Belief compression is performed according to the state abstraction present at each level of the RN-HPOMDP structure, i.e. the discretization reduction of each level as compared to the the discretization of the corresponding flat POMDP. Therefore, the belief assigned to an abstract state, a state with coarse discretization at any level of the hierarchical structure other than the bottom level, will correspond to the sum belief of all the corresponding flat POMDP states that the named abstract state has integrated. The function `compressTopBelief()` is used to obtain the belief distribution at the top-level POMDP of the hierarchical structure. The abstract states of the top-level POMDP represent all states of the corresponding flat POMDP. Hence the sum of the belief of all top-level POMDP states is equal to one since the sum of the belief of all states of the corresponding flat POMDP is also equal to one. The function `compressBelief()` is used to compress the *full belief* to the belief of POMDPs at all other levels of the hierarchical structure. The belief of any level's POMDP abstract state is again obtained by summing the belief all the integrated actual states. However, intermediate level POMDPs represent only of a small fraction of the *full belief* and hence the belief distribution obtained for any intermediate level POMDP is normalized to one.

The top level POMDP is solved, by the function `solveTopLevel()`, at an infinite horizon, until the goal state is reached. The top level POMDP produces abstract actions, i.e. actions at a coarse resolution that infer only the general direction the robot should follow and not the actual action it will perform. The abstract action to be executed, a_p, as dictated by the top level POMDP solution, determines which POMDP at the immediate next level of the hierarchical structure will be solved to obtain a new refined abstract action, that has a finer discretization but still it is not the actual action the robot will perform.

The POMDP to be solved at each intermediate level is determined by the function `se-`

Table 4.2: RN-HPOMDP planning

```
while not reached the goal state
    compressTopBelief(top level)
    $a_p$ = solveTopLevel(top level)
    for $l = 2$ to $L$
        $whichPOMDP$ = selectPOMDP($l, a_p$)
        compressBelief($l, whichPOMDP$)
        $a_p$ = solveLevel($l, whichPOMDP$)
    end
    executeAction($a_p$)
    $z$ = getObservation()
    $belief_L$ = updateBelief($whichPOMDP, a_p, z$)
    $full\ belief$ = updateFullBelief($belief_L, whichPOMDP$)
end
```

lectPOMDP(). This function searches a level l for the POMDP among all POMDPs in that level that satisfies the following two criteria:

- The zero moment of the full belief distribution over the area that is defined by the candidate POMDP states is maximum, where b has more mass.
- The set of actions of the candidate POMDP contains an action that has minimum distance from the the previous level solution's action, a_p.

The function selectPOMDP() determines the POMDP to be solved at each level and can also accommodate cases where the belief distribution is not unimodal. As noted in Section 4.1 the belief distribution is initialized as unimodal since it is assumed that the robot has a good estimate of its initial position. However, during execution the belief distribution can become multimodal. In these cases the selectPOMDP() function can select the correct peak of belief distribution to determine the POMDP to be solved due to the second criterion for choosing POMDPs. However, in cases that this is not possible multiple POMDPs can be solved at a level, a POMDP at each peak of the belief distribution. This approach provides an efficient solution when there are multiple peaks in the belief distribution. In the event where no information is available, i.e. the belief has a uniform distribution, that however is not often the case in the application considered in this thesis the RN-HPOMDP will not be able to determine which POMDP it should solve at an intermediate level. Yet, methodologies for overcoming this problem can be further explored as noted in the Future Work Section.

The structure of the RN-HPOMDP, as described in Section 4.2, ensures that when solving an intermediate level POMDP the action obtained from the previous level will be refined to a new action since the action subset range is equal to

$$\left[a_p - \frac{90^\circ}{2^{l-2}}, a_p + \frac{90^\circ}{2^{l-2}}\right].$$

Therefore the solution of an intermediate level POMDP is bounded according to the previous level solution, a_p.

The `solveLevel()` function provides a new abstract action and the described procedure continues until the bottom level is reached where an abstract action will be refined to an actual action, that is the action the robot will perform.

In our current implementation of the RN-HPOMDP, all POMDPs at all levels are solved, within functions `solveTopLevel()` and `solveLevel()`, by value iteration using the Voting heuristic (explained in Section 3.3). However, this is not an inherent feature of the RN-HPOMDP structure, as any other POMDP solution method can be used. Furthermore, the POMDP solution method used can also be different for each level of the hierarchical structure.

When the robot executes the action obtained by the bottom level POMDP solution, an observation, z, is obtained and the belief distribution of this bottom level POMDP is updated by `updateBelief`(), according to the equations provided in Section 3.2.1. Bottom level

POMDPs are composed of actual states and actions, i.e. subsets of states and actions that compose the corresponding flat POMDP. Hence, updating the belief of a bottom level POMDP, $belief_L$, amounts to updating a specific region of the *full belief*. Therefore, the updated belief distribution of the bottom level POMDP that was solved is transferred to the *full belief* by the function `updateFullBelief()`. To update all regions of interest in the *full belief* the procedure of updating the belief of all bottom level POMDPs that the sum of the belief of all its states is non-zero and includes in the action set the action a_p that the robot executed is performed and then the probabilities are transferred to the *full belief*. According to Equation 3.3 that performs the belief update it can be observed that only the observation and transition probabilities for the executed action a_p are used. Furthermore, states with zero belief do not contribute to the belief update according to the same equation. Hence, the hierarchical structure and its bottom level can be used to determine which areas of the full belief space should be updated. This procedure can be though intuitively similar to the procedure used in belief compression [93, 99] methods or point-based methods [91, 110].

4.6 Complexity Analysis

In the complexity analysis that follows, execution times are evaluated for the POMDP solution using exact methods and heuristics. The Voting or MLS heuristic, that have the same complexity, will be assumed as the heuristic used for solving the POMDP. This will assist in the comparison between the RN-HPOMDP and the other hierarchical approaches present in the literature that use the above mentioned heuristics. As already mentioned, in our implementation the Voting heuristic is used to solve POMDPs at all levels.

4.6.1 Approximate solution

The flat POMDP solution - when solved with the MLS or Voting heuristic - has time complexity for a single step, $\mathbf{O}\left(|\mathcal{S}|^2|\mathcal{A}|\right)$, that is the complexity of solving the underlying MDP. This makes the POMDP solution intractable when dealing with real world environments at an acceptable resolution, e.g. having more than 10 million states as in our application.

Obtaining a solution by the RN-HPOMDP can dramatically improve the computation time required. Referring to Table 4.1, where the properties of the RN-HPOMDP structure are detailed, the solution of the top level POMDP requires

$$\mathbf{O}\left(\left(\frac{|\mathcal{S}|}{2^{2(L-1)}}\right)^2 \times 4\right)$$

computational time, where L is the number of levels of the hierarchical structure.

The solution of all intermediate levels POMDPs requires $\mathbf{O}(C_1)$ time, since the size of the state space and action space is constant and predefined. The bottom level POMDP solution is $\mathbf{O}(C_2)$, since the state space and action space is again constant and predefined.

Therefore, the total computational time required to solve the RN-HPOMDP is

$$\mathbf{O}\left(\left(\frac{|\mathcal{S}|}{2^{2(L-1)}}\right)^2\right) + (L-2) \times \mathbf{O}\left(C_1\right) + \mathbf{O}\left(C_2\right),$$

which becomes

$$\mathbf{O}\left(\left(\frac{|\mathcal{S}|}{2^{2(L-1)}}\right)^2\right),$$

that is actually the complexity of the top level POMDP. The top-level POMDP state and action space size can remain small regardless of the size of the whole environment by increasing the number of levels, L, of the hierarchical structure.

4.6.2 Exact solution

When solving the POMDP exactly for a single step in time t, the time complexity is

$$\mathbf{O}\left(|\mathcal{S}|^2|\mathcal{A}||\Gamma_{t-1}|^{|\mathcal{Z}|}\right),$$

where $|\Gamma_{t-1}|$ is the set of α-vectors required to represent the value function at time $t-1$. The size of the set of α-vectors at any time t is equal to

$$|\Gamma_t| = |\mathcal{A}||\Gamma_{t-1}|^{|\mathcal{Z}|}.$$

As explained previously, the time complexity of solving the RN-HPOMDP is equal to the time complexity of solving the top level POMDP. The top level POMDP of our hierarchical structure has reduced state space and is equal to $\left(\frac{|\mathcal{S}|}{2^{2(L-1)}}\right)$, where L is the number of levels. Furthermore, the action space is constant and equal to four. Therefore, the time complexity and size of the RN-HPOMDP when solved exactly is

$$\mathbf{O}\left(\left(\frac{|\mathcal{S}|}{2^{2(L-1)}}\right)^2|\Gamma_{t-1}|^{|\mathcal{Z}|}\right)$$

and

$$|\Gamma_t| = |\Gamma_{t-1}|^{|\mathcal{Z}|},$$

respectively.

Apart from the notable reduction in computation time due to the reduced size of the state and action space, it should be noted that the above mentioned times are for a single time step. The infinite horizon solution of a flat POMDP would require these computations to be repeated for a number N of time steps until the goal point is reached, that is dependent on the number of states of the flat POMDP, $|\mathcal{S}|$. In the RN-HPOMDP case, only the top level POMDP is solved at an infinite horizon, and the number of time steps N' until the goal point is reached, is now dependent on the number of states of the top level POMDP, $(|\mathcal{S}|/2^{2(L-1)})$.

From the above complexity analysis, we may conclude that the proposed approach takes care of the “curse of dimensionality” [59] and also the “curse of history” [91].

4.7 Comparison With Other HPOMDP Structures

In this section we compare the RN-HPOMDP against the other two HPOMDP approaches present in the literature, in terms of time complexity for solving the HPOMDP, state space and action space abstraction methodology, and the application framework of the HPOMDP. Finally, a comparison of the RN-HPOMDP and the most recent approximation methods for solving a flat POMDP is presented.

4.7.1 Comparison with the Theocharous approach

The Theocharous [119] approach uses a topological map of the environment where the state abstraction in high levels of the HPOMDP, has a physical meaning based on the environment. Thus, abstract states are manually defined such that they represent a corridor or a junction. On the other hand, the RN-HPOMDP is built through an automated procedure that requires as input only a probabilistic occupancy grid map or a CAD map of the environment.

The Theocharous HPOMDP has been used as a high-level planner where the POMDP is solved once to obtain the shortest path to the goal position. As a result, the state space resolution is set to $2m^2$ and the action space is discritized at a resolution of 90°. Our approach models the environment at a fine resolution (e.g. $5cm^2$) and the action resolution can be discretized up to 1° based on the number of levels of hierarchy. Finally, the RN-HPOMDP is used as a global planner that is solved at each time step to provide the actual actions the robot will perform without the intervention of any other intermediate modules. The RN-HPOMDP integrates the modules for planning, localization and local obstacle avoidance.

The Theocharous approach, uses the MLS heuristic and has time complexity between

$\mathbf{O}(|\mathcal{S}|^{\frac{2}{d}}N|\mathcal{A}|)^{(see\ \dagger)}$ and $\mathbf{O}(|\mathcal{S}|^2|\mathcal{A}|)^{(see\ \ddagger\ )}$, based on how well the HPOMDP was constructed. The time required to solve the RN-HPOMDP is $\mathbf{O}((|\mathcal{S}|/2^{2(L-1)})^2)$, hence the complexity reduction of our approach is significantly greater and also is not dependent on any quality measure of the hierarchical structure.

4.7.2 Comparison with the Pineau approach

In the Pineau HPOMDP approach [89], actions are grouped into abstract actions called subtasks. Subtasks are defined manually and according to them state abstraction is performed automatically. States that have the same reward value for executing any action that belongs to a predefined subtask are clustered. Observation abstraction is performed by eliminating the observations that have zero probability over all state clusters for that actions belonging to a specific subtask.

Planning with the Pineau HPOMDP involves solving the POMDP defined for each action subtask. All POMDPs are solved using the exact POMDP solution method.

The HPOMDP proposed by Pineau does not have a guaranteed reduction of the action space and state space since it is dependent on the action abstraction that is defined manually. The authors have performed experiments (real and simulated) only for problems of high level behavior control. Hence it is not clear whether their approach of state abstraction could be applied to the problem of autonomous robot navigation in the context that we have defined or, more importantly, if it would perform as efficiently as our approach does, since the latter has a guaranteed reduction of the state space that is equal to $\left(|\mathcal{S}|/2^{2(L-1)}\right)$. On the other hand, the authors in [89] do not state how well their approach performs in terms of state space

$^{\dagger}d$ is the depth of the tree and N is the maximum number of entry states for an abstract state.

‡The size of the action space $|\mathcal{A}|$ was added in the time complexity of the Theocharous approach, so that the comparison with the complexity of a flat POMDP and our approach can be direct, although in their approach $|\mathcal{A}|$ is constant and equal to 4.

abstraction.

The HPOMDP proposed by Pineau has been used in a real world application for high level robot control and dialogue management [89]. This application has been modelled using 576 states, 18 observations and 19 actions categorized into 3 subtasks. The problems encountered with our approach are many orders of magnitude larger (e.g. $|\mathcal{S}| = 18,411,520$, $|\mathcal{A}| = 256$, $|\mathcal{Z}| = 24$) and can be solved in real time.

4.7.3 Approximation methods for solving flat POMDPs

A short discussion on the performance of approximation methods for solving flat POMDPs follows in this section. This discussion will allow us to elaborate further on the performance of the RN-HPODMP and also necessitate further the need of the proposed hierarchical structure, at least when considering the autonomous robot navigation problem.

In [51] a review of approximation methods for solving POMDPs is presented. The complexity of representative methods reviewed can be seen in Table 4.3. Furthermore, one of the most recent methods for approximation is the Point Based Value Iteration (PBVI) [91] method. The time complexity of PBVI is $\mathbf{O}(|\mathcal{S}||\mathcal{A}||\Gamma_{t-1}||\mathcal{Z}||B|)$, where $|B|$ is the size of the finite set of belief points and $|\Gamma|$ remains constant throughout iterations.

To summarize, the time complexity of all approximation methods is in the best case polynomial to the size of the POMDP. All the above mentioned methods have been applied to problems where the POMDP was comprised of a few thousand states. The problem we consider consists of many orders of magnitude larger state space. As a result, the reduction of the state space that the RN-HPOMDP offers and also the reduction of the action space is crucial to its performance. Furthermore, since the proposed hierarchical structure is not restricted to a specific method for solving the underlying POMDPs, a combination of an approximation method for solving a flat POMDP with the proposed hierarchical structure can dramatically

Table 4.3: Complexity of solving a POMDP with the approximation methods reviewed in [51].

Approximation Method	Complexity
MDP	$\mathbf{O}(\lvert\mathcal{A}\rvert\lvert\mathcal{S}\rvert^2)$
QMDP	$\mathbf{O}(\lvert\mathcal{A}\rvert\lvert\mathcal{S}\rvert^2)$
Fast Informed Bound (FIBM)	$\mathbf{O}(\lvert\mathcal{A}\rvert^2\lvert\mathcal{S}\rvert^2\lvert\mathcal{Z}\rvert^2)$
UMDP	$\mathbf{O}(\lvert\mathcal{A}\rvert\lvert\mathcal{S}\rvert^2\lvert\Gamma\rvert_{t-1})$, $\lvert\Gamma_t\rvert = \lvert\Gamma_{t-1}\rvert\lvert\mathcal{A}\rvert$
Grid based interpolation extrapolation [a]	$\mathbf{O}(\lvert G\rvert\lvert\mathcal{A}\rvert\lvert\mathcal{S}\rvert^2\lvert\mathcal{Z}\rvert C_{eval})$

[a] $|G|$ is the size of a finite set of grid points used to update the value updates and C_{eval} is the computational cost of evaluating the interpolation-extrapolation rule for $|G|$ points, where in some cases this cost can be eliminated.

Table 4.4: Computation time required to solve a HPOMDP with the compared approaches.

	POMDP size		CPU time (sec)
Theocharous [119]	$\lvert S\rvert = 575$	$\lvert A\rvert = 4$	2.11 - 5.7
	$\lvert S\rvert = 1385$	$\lvert A\rvert = 4$	5.05 - 26.12
Pineau et. al. [90]	$\lvert S\rvert = 11$	$\lvert A\rvert = 6$	2.84
	$\lvert S\rvert = 20$	$\lvert A\rvert = 30$	77.99

improve its performance.

4.7.4 Computational time comparison

Further to the theoretical comparison presented in the previous section, for indicative comparison purposes we provide the CPU times required to solve the RN-HPOMDP and also the

Table 4.5: Computation time required to solve the RN-HPOMDP with varying grid size and 5 levels.

Grid size	**POMDP size**		**CPU time (sec)**
$5cm \times 5cm$	$\lvert S\rvert = 18,411,520$	$\lvert A\rvert = 64$	18.520
$10cm \times 10cm$	$\lvert S\rvert = 4,602,880$	$\lvert A\rvert = 64$	0.911
$15cm \times 15cm$	$\lvert S\rvert = 2,038,080$	$\lvert A\rvert = 64$	0.426
$20cm \times 20cm$	$\lvert S\rvert = 1,150,720$	$\lvert A\rvert = 64$	0.257
$25cm \times 25cm$	$\lvert S\rvert = 734,976$	$\lvert A\rvert = 64$	0.262
$30cm \times 30cm$	$\lvert S\rvert = 503,808$	$\lvert A\rvert = 64$	0.251

Table 4.6: Computation time required to solve the RN-HPOMDP with varying number of levels and grid size of $10cm \times 10cm$.

No. of Levels	**POMDP size**		**CPU time (sec)**
3	$\lvert S\rvert = 1,150,720$	$\lvert A\rvert = 16$	201.210
4	$\lvert S\rvert = 2,301,440$	$\lvert A\rvert = 32$	16.986
5	$\lvert S\rvert = 4,602,880$	$\lvert A\rvert = 64$	0.911
6	$\lvert S\rvert = 9,205,760$	$\lvert A\rvert = 128$	0.460
7	$\lvert S\rvert = 18,411,520$	$\lvert A\rvert = 256$	0.411

Theocharous and Pineau HPOMDP approaches in Tables 4.4, 4.5 and 4.6. It should be stressed out, that the times referring to the Pineau approach are the ones from their initial version of HPOMDP [90] where there was only action space hierarchy. It should be also noted that the CPU times mentioned are the ones the authors state and have not been obtained using computers of the same power. Another point is that the Theocharous approach is solved using the MLS heuristic and in our approach the POMDPs are solved using the Voting heuristic that has the same computational complexity with the MLS heuristic. However, the Pineau HPOMDP is solved using exact methods. Regardless of the mentioned differences, the superior computational performance of our approach can be easily extracted from the tabulated results since the size of the problem is many orders of magnitude larger.

4.8 Conclusions

This chapter presented the proposed navigation model, the Robot Navigation-Hierarchical POMDP (RN-HPOMDP). The methodologies for building the hierarchical structure, learning and planning with it have been introduced. The notion of the reference POMDP (rPOMDP) has been also established that facilitates compact modelling of all the required POMDP functions. Finally, the complexity of solving the RN-HPOMDP has been evaluated and a comparison with other hierarchical POMDP approaches has been conducted. The following chapters will present how the RN-HPOMDP is utilized for the predictive navigation task proposed in this thesis.

5

Motion Prediction and Tracking

In this chapter the methodology for predicting the future motion of humans and/or other objects is described. Two kinds of prediction are utilized: short-term and long-term prediction. Short term prediction refers to the one-step ahead prediction whereas long-term prediction refers to the prediction of the final destination point of the obstacle's movement. Furthermore, the motion tracker utilized will be described that functions in conjunction with the long-term prediction module. Finally, this chapter concludes by describing how the information obtained by the prediction modules is integrated into the POMDP model used for navigation.

5.1 Short-Term Prediction

The short-term prediction of the future motion, i.e. the one-step ahead prediction, is obtained by a Polynomial Neural Network (PNN). In PNNs each node's transfer function is a polynomial. PNN's can model any function since the Kolmogorov-Gabor theorem states that any function can be represented as a polynomial of the form

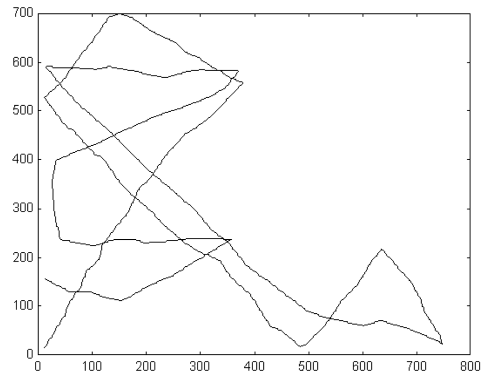


Figure 5.1: The data set used for NN training.

$$f(x) = a_0 + \sum_i a_i x_i + \sum_i \sum_j a_{ij} x_i x_j + \sum_i \sum_j \sum_k a_{ijk} x_i x_j x_k + ...$$

where x_i is the independent variable in the input variable vector $\vec{x}$ and $\vec{a}$ is the coefficient vector. PNNs approximate the Kolmogorov-Gabor polynomial representation of a function by having as a transfer function at each node, a second order polynomial of the form

$$f(x) = a + bx_1 + cx_2 + dx_1^2 + ex_2^2 + fx_1x_2,$$

where x_1 and x_2 are the inputs to the node.

The input to the network, x_1 and x_2, is the moving obstacle's position at times $t-1$ and t. The positions of the moving obstacle are given to the network as a state. The output of the network is the predicted position at time $t+1$. The topology and weights, that is the polynomial coefficients, of the PNN are determined via training with an evolutionary method [32]. The data set used was composed of 4500 samples and is shown in Figure 5.1. It was obtained by arbitrary movement in the environment that the robot operates.

The results obtained from the trained NN are illustrated in Figure 5.2. The first 3000 samples from the data set were used for training and the rest 1500 samples for evaluating the

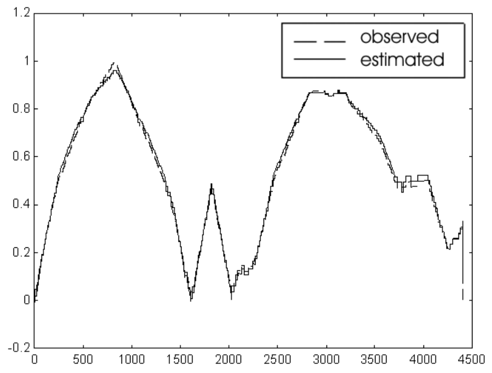


Figure 5.2: The results obtained from the trained NN.

network. It can be seen that the network gives a prediction with a small error for the first 3000 samples but as well for the rest of the samples that the network has not seen before. Therefore, the network obtained generalizes well for unforeseen situations.

5.2 Long-Term Prediction

Prediction methods known so far give satisfactory results for one-step ahead prediction. For the robot navigation task it would be more useful to have many-steps ahead prediction. This would give the robot sufficient time and space to perform the necessary manoeuvres to avoid obstacles and more importantly change its route towards its destination position. It is desirable for the robot to develop a behavior that will prefer routes that are not crowded and thus avoid ever getting stuck. Long-term prediction will enable the robot to execute paths that are "elegant" and "aware" as defined in the Introduction Chapter of this thesis.

It is unlikely that any of the available prediction methods would give satisfactory results for many-steps ahead prediction, given the complexity of the movement behavior. Thus, it is proposed to employ a long-term prediction mechanism. The long-term prediction refers to the prediction of a human's or an obstacle's final destination position. It is plausible to assume that humans mostly do not just move around but instead move purposively with the intention

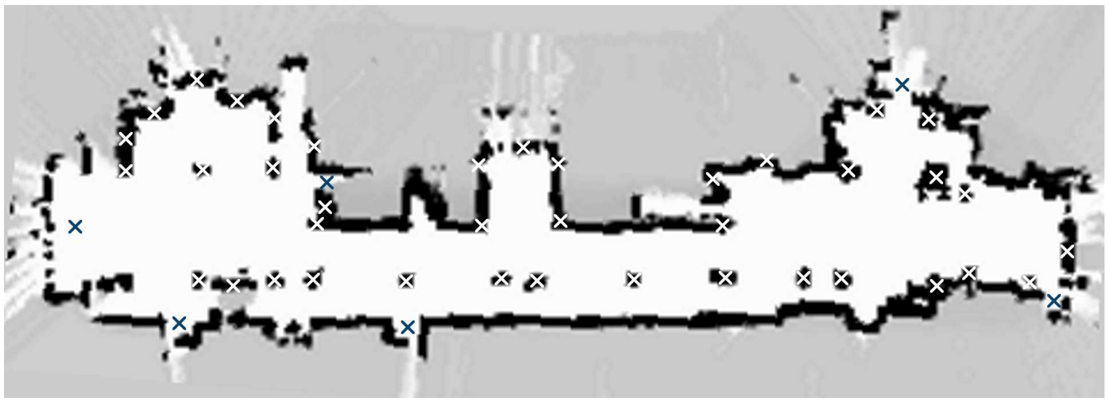


Figure 5.3: The “hot” points defined for the FORTH main entrance hall, marked with “x".

of reaching a specific location. Our approach for performing long-term prediction is based on the definition of the so-called “hot points” (HPs), the points of interest in the environment where people visit often. For example, in an office environment desks, doors and chairs are objects that people have interest in reaching them and could be defined as points of interest. In a museum, the points of interest can be defined as the various exhibits that are present. Moreover, other features of the environment such as the entry points, passages, e.t.c, can be defined as points of interest. Evidently, points of interest convey semantic information about a workspace and hence can only be defined with respect to the particular environment.

The HPs in an environment can be defined either manually or through an automated procedure. In the following, an automated procedure for obtaining a map of HPs is described. The methodology for obtaining the long-term prediction is given for both cases where HPs are defined manually and where they are learned through an automated procedure. The manually defined HPs for the FORTH main entrance hall where the experiments were conducted are shown in Figure 5.3. First, the methodology for obtaining the estimated destination position of a moving object with manually defined HPs is presented and then this approach is extended to be used with a map of HPs.

5.2.1 Estimation of a moving object's destination position

Once the points of interest of an environment are defined, then the long-term prediction refers to the prediction of which HP a moving obstacle is going to approach. At each time step t, the tangent vector of the obstacle's positions at times $t-1$, t and the predicted position at time $t+1$ is taken. This tangent vector essentially determines the global direction of the obstacle's motion trajectory, termed as Global Direction of Obstacle (GDO). This direction is employed to determine which HP a moving obstacle is going to approach. A HP is a candidate final destination point if it lies roughly in the direction of the evaluated tangent vector. In order to find such points, we establish a *field of view*, that is an angular area centered at the GDO. HPs present in the field of view are possible points to be reached, with a probability w_i, according to a probability model. The latter is defined as a gaussian probability distribution with center at the GDO and standard deviation in proportion to the angular extent of the field of view. Thus, points of interest present in the center of the field of view are assigned a high probability, and points of interest present in the periphery are assigned a lower probability.

With this approach, at the beginning of the obstacle's movement a multiple number of points of interest will be present in its field of view but as it continues its movement the number of such points is decreased and finally it usually converges to a single point of interest.

In Figure 5.4 an example of how the procedure for long-term prediction with manually defined HPs is shown. At the beginning of the obstacle's movement the long-term prediction obtained is shown in Figure 5.4(a). It can be observed that at this point there are multiple candidate destination points for the obstacle's movement. However, the destination point that infers that the moving objects is going to walk down the stairs is estimated as the most probable destination point of the obstacle's movement as directed by the GDO, shown as a green vector in this figure. If the whole obstacle motion trajectory, shown in the same figure, is observed it is obvious that the estimated destination point dictated by the long-term prediction methodology is not the correct one. However, the long-term prediction obtained can be utilized partially. Therefore, when a long-term prediction is obtained it is utilized only partially for a short interval that is close to the obstacle's current position. Hence, the long-term prediction when utilized only partially it provides a rather good estimate of the future motion of the obstacle's movement. The partial utilization of the obtained long-term prediction is fully detailed in Section 5.4, where its integration to the navigation model is explained. Additionally, the long-term prediction is updated at each time step, that is a short time interval, and hence bad estimates can be corrected quickly. Consequently, as the object has advanced through its motion trajectory the long-term prediction estimate is closer to the actual destination point as shown in Figure 5.4(b).

The example shown in Figure 5.4 has been chosen such that to demonstrate the weakness of using manually defined HPs and hence necessitate the use of the map of HPs. When the obstacle is close to the end of its motion trajectory, as shown in Figure 5.4(c), there is no HP defined in the direction dictated by the GDO. Instead, there are two points in the periphery of the GDO, and one of them is the actual destination point of the obstacle. However, the actual destination point will be assigned a very low probability since it is located in the periphery of the GDO. The same holds for the other HP located within the field of view. In contrast, there is no prediction with high probability. In this scenario, if there was a map of HPs available there would be defined a whole area of points of interest instead of two unique HPs. Specifically, in the area of the map under discussion there are two sofas where people often go there and sit. When the map of HPs is constructed the whole area covered by the sofas is determined as an interesting point instead of the two unique points in the center of the sofas that have been manually defined. Hence, we would still have obtained a long-term prediction with high probability. Additionally, the probability of each estimated destination point is not determined only according to each location within the field of view but also to the popularity of this point as determined through the learning procedure.

The map of HPs can also take care of the worst case scenario where there are not manually defined points present in the field of view marginally and hence there would be no estimate available.

5.2.2 Map of hot points

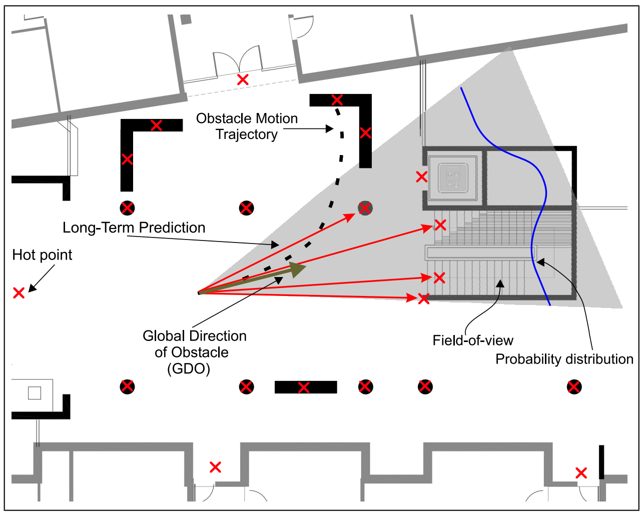
An alternative approach to defining manually the hot points of the environment is to obtain a map of hot points of the environment through an automated procedure.

The map of hot points is a probabilistic map that gives for each point of the environment the probability that this point is a hot point. This map is built off-line by a learning procedure that uses motion traces of humans operating in the environment. At every step of each collected motion trace the probabilities of the map of hot points are updated in a similar manner to this used when building an occupancy grid map of an environment from sensor readings.

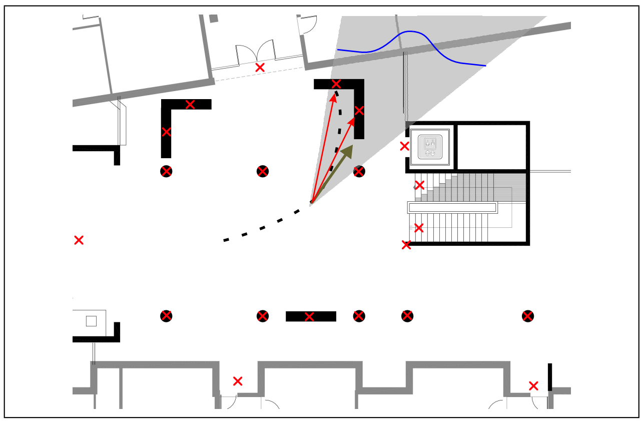
Having obtained the Global Direction of Obstacle (GDO), defined in Section 5.2.1, the *field of view* is determined. The field of view now defines the area of cells that their probabilities of being a hot point is going to be updated. Lines from the obstacle's current position at all possible angles within the field of view are examined whether they intersect with a feature of the environment. This intersection point is considered as a possible hot point (HP). The probability assignment is performed by obeying two rules:

- the smaller the angular distance, α, of the candidate cell and the GDO the higher the probability;
- the smaller the distance, r, of the candidate cell and the obstacle's current position the higher the probability.

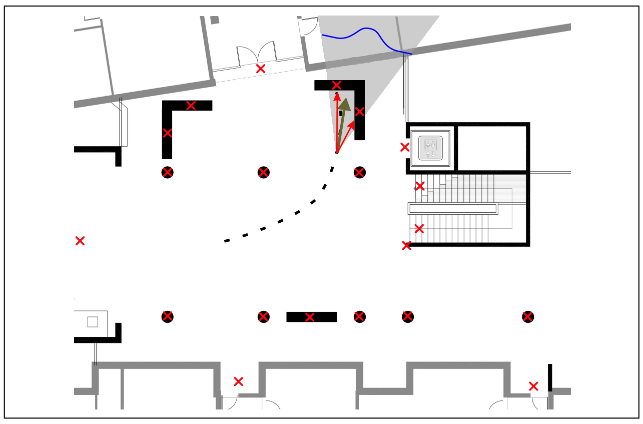
The probability of a cell (i, j) being a hot point, $p(H_{i,j})$, given an obtained GDO at time t, GDO_t, is given by:



(a)



(b)



(c)

Figure 5.4: An example of making long-term prediction for an object's movement.

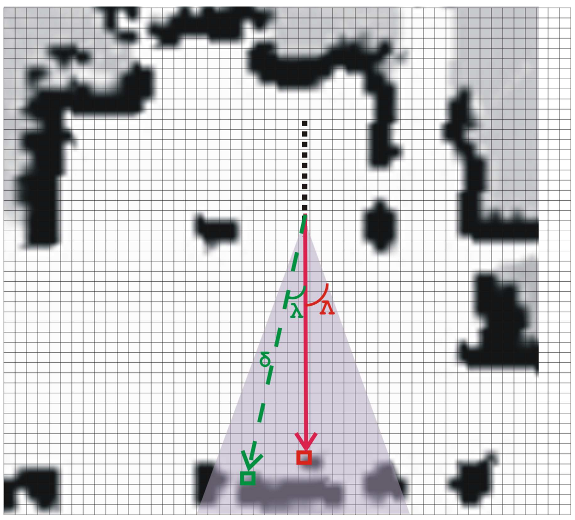


Figure 5.5: The probability assignment for possible hot points is dependent on the angular distance of the considered cell and the GDO and its distance from the obstacle's current position.

$$p(H_{i,j}|GDO_t) = \frac{1}{2}\left(\frac{\Delta - \delta}{\Delta} + \frac{\Lambda - \lambda}{\Lambda}\right).$$

As shown in Figure 5.5, the maximum angular distance from the GDO is Λ and is defined by the size of the field of view. Δ is a constant that determines the maximum distance allowed from a cell to be considered as a hot point.

The update of the probabilities is performed according to Bayes rule as:

$$p(H_{i,j}|GDO_t) = \frac{p(GDO_t|H_{i,j})p(H_{i,j}|GDO_{t-1})}{p(GDO_t|H_{i,j})p(H_{i,j}|GDO_{t-1}) + p(GDO_t|\overline{H}_{i,j})p(\overline{H}_{i,j}|GDO_{t-1})}$$

Having obtained the probability map of hot points long-term prediction is now performed using this probability map by considering all lines within the field of view that intersect with a feature of the environment according to the probability assigned at the map of hot points.

The map of hot points obtained for the FORTH main entrance hall is shown in Figure 5.6. The map was constructed by taking laser measurements at various times of the activity in the

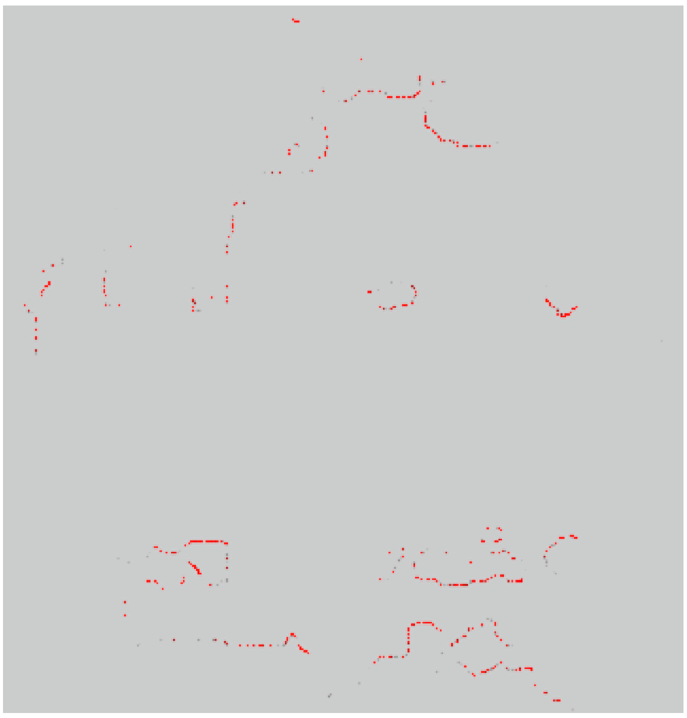


Figure 5.6: The map of “hot” points obtained for the FORTH main entrance hall.

FORTH main entrance hall in order to obtain real motion paths that will reveal the true points of interest in the environment.

5.3 Motion Tracking

The algorithm used for object tracking in this paper is a modification of the commonly used Kalman tracker. In the Kalman tracker, a Kalman filter is used for predicting the position of a previously detected object and hence decide if the object actually moved to its predicted position. In our approach, the Kalman filter is substituted by the short-term and long-term prediction obtained as described in the previous sections. The data association is performed by a nearest neighbor filter that is validated by the long-term prediction module.

Initially, the laser scan obtained is filtered to decide which range measurements belong to moving objects. Filtering is performed by superimposing the current laser scan with the probabilistic grid map of the environment to remove range measurements that correspond to

Table 5.1: The algorithm for motion tracking

```
$new\ objects$ = detectMovingObjects ($current\ laser, previous\ laser, OGM$)
for all $detected\ objects$
    $predicted\ position\ of\ object_i$ = shortTermPrediction($detected\ object_i$)
end
for all $new\ objects$
    if $\min_{detected\ objects} distance(new\ object_i, detected\ object_j) < \epsilon$
        $nearest\ object$ = $\arg\min_{detected\ objects} distance(new\ object_i, detected\ object_j)$
        $trajectory$ = longTermPrediction($nearest\ object$)
        if verify($new\ object_i$, $trajectory$) = true
            match $new\ object_i$ with $nearest\ object$
        else
            $nearest\ object$ = $\arg\min_{detected\ objects-nearest\ object} distance(new\ object_i, detected\ object_j)$
            $trajectory$ = longTermPrediction($nearest\ object$)
            if probabilityBelonging($new\ object_i$, $trajectory$, T) < $\rho$
                match $new\ object_i$ with $nearest\ object$
end
for all unmatched $new\ objects$
    for all $detected\ objects$
        $trajectory$ = longTermPrediction($detected\ object_j$)
        if probabilityBelonging($new\ object_i$, $trajectory$, T) < $\rho$
            match $new\ object_i$ with $detected\ object_j$
    end
end
for all unmatched $new\ objects$
    add to $detected\ objects$
end
```

features of the static map of the environment. Following, the current laser scan is superimposed with previous laser scan to decide if there are range measurements that correspond to moving objects. Range measurements that correspond to moving objects are those that were not present in the previous laser scan and have not been filtered out by superimposing the current laser scan with the probabilistic map of the environment.

The range measurements detected as belonging to moving objects that cover an area within a specific range, are combined to obtain the position of each moving object. At this point, the position of each currently moving object has to be decided if it belongs to the trajectory of a previously detected object or if it is a newly detected object.

Matching the positions of the currently detected moving objects with previously detected objects is performed by utilizing the short-term and long-term prediction. For each previously detected object a short-term and long-term prediction is obtained. The distance of a currently detected moving object position with the short-term prediction of a previously detected object is evaluated. The minimum evaluated distance of all previously detected objects for a specific current position is regarded to belong to this object if it is smaller than a certain threshold.

However, data association with the nearest neighbor filter is verified by the long-term prediction prediction for the motion trajectory of the object indicated to be matched with. If the newly detected object's position belongs to the trajectory indicated by the long-term prediction with a probability higher than a certain threshold then matching is achieved. Otherwise, matching is verified with the immediate next object in distance measures as dictated by the nearest neighbor filter.

After the described matching procedure there might be positions of currently detected moving objects that were not matched to any previously detected objects. These unmatched positions can either belong to newly detected objects or they were falsely not matched with a previously detected object. The latter case is possible when an object has moved for a larger distance than the short-term prediction indicated or when an object has been occluded by other objects for a short time and hence intermediate positions of its trajectory are missing. To deal with these situations we utilize the long-term prediction.

The long-term prediction provides a prediction of which “hot” point an object is approaching. Therefore, the unmatched positions of currently detected moving objects are examined whether they belong to one of the trajectories indicated by the long-term prediction for each previously detected object. The decision whether an unmatched position belongs a trajectory indicated by the long-term prediction is performed in a probabilistic manner. The long-term prediction provides a probability for each trajectory to a “hot point” predicted as a final destination point of an object’s movement. Furthermore, a probability is assigned according to the time difference between an object’s last detected position and the position of a currently detected object, and the estimate of how far in the future the object would reach the position in the trajectory to a “hot” point that was matched with the position of a currently detected object. If the probability is higher than a threshold that has been determined by trial and error, then the position of the currently detected object is added to the previously detected object.

Remaining positions of currently detected objects that were not matched by the short-term or long-term prediction are regarded as new objects.

5.3.1 Tracker results

To demonstrate the effectiveness of the proposed tracker algorithm with the use of long-term prediction we have set up an experiment where two people walk diagonally and cross each others path. The robot was set still to observe only the motion of the two people. The tracker can easily distinguish that there are two persons present, since they are quite far away and they are not occluded. However, near the middle of the two people’s motion where they are too close to each other the robot perceives only one person as shown in Figure 5.7. This is inevitable in almost any tracker since the two persons are too close to each other. When the two persons continue their movement and come apart from each other again it would be desirable for the tracker to be able to correctly associate the two persons with their previously detected paths. A tracker that uses only a nearest neighbor filter would in most cases associates only one person, e.g. the one notated with the blue square, as the one continuing its movement and regard the other person as newly detected motion. Furthermore, in many cases where the same experiment was performed the tracker using only the nearest neighbor filter has associated conversely the two persons, depending on the speed at which they were moving and the neighboring threshold set in the tracker. The use of the long-term prediction overcomes this problem and associates correctly the two person's movement in almost all cases. This is due to the verification step performed and association step based on the long-term prediction.

5.4 Prediction Integration Into The Model

The short-term and long-term prediction are integrated in the global model by including them in the reward function of the POMDP. The reward function is built and updated at each time step, according to two *reward grid maps* (RGMs): a *static* and a *dynamic* [33]. The RGM is defined as a grid map of the environment in analogy with the OGM. Each of the RGM cells corresponds to a specific area of the environment with the same discretization of the Occupancy Grid Map (OGM), only that the value associated with each cell in the RGM represents the reward that will be assigned to the robot for ending up in the specific cell. Thus, it would be desirable that this value gives a description of the state of this square area in the environment as

- how far it is from the goal position,
- whether it is occupied by a static obstacle,

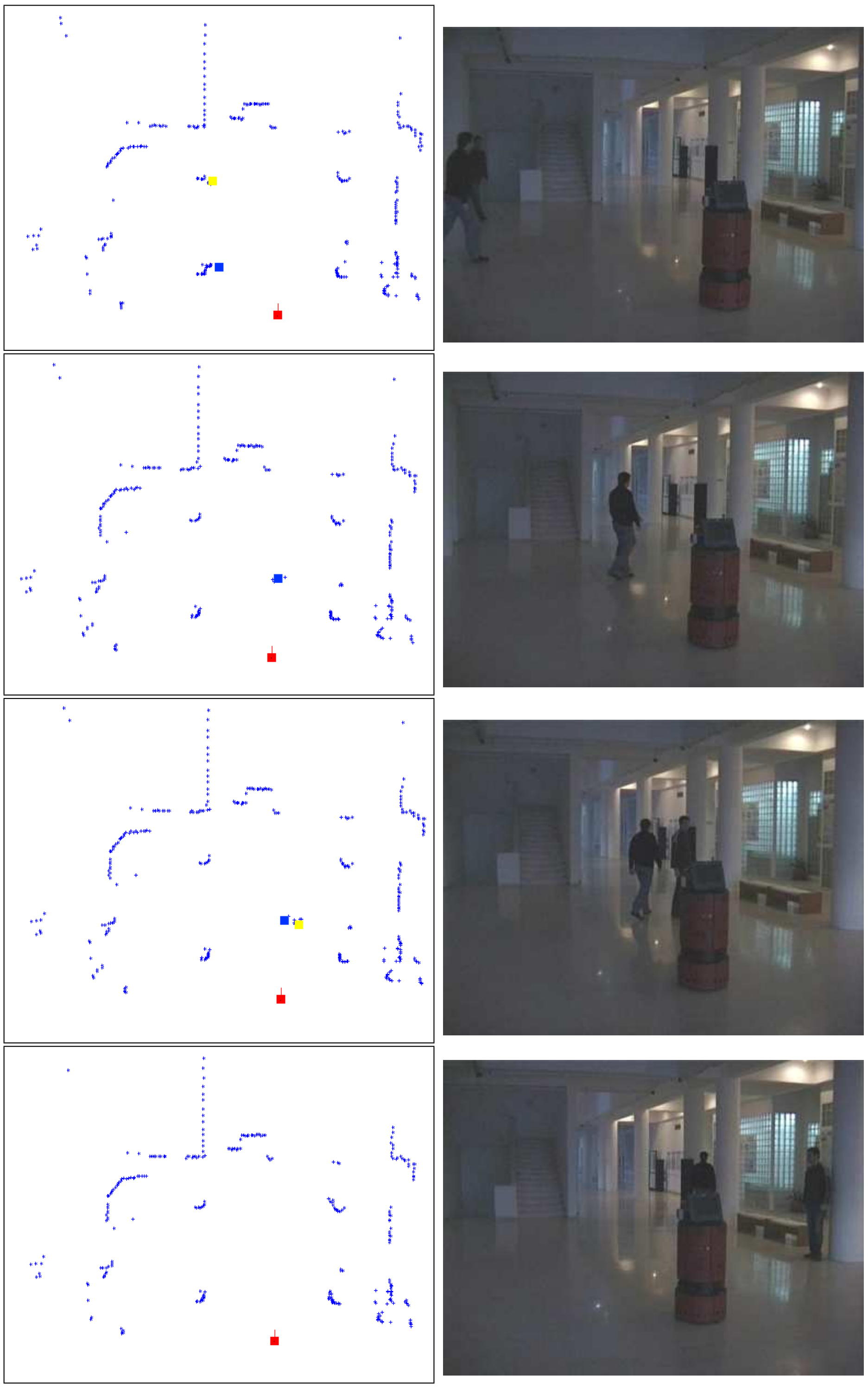


Figure 5.7: Tracking the motion of two persons.

- whether it is occupied by a moving obstacle, i.e. human or other robot,
- whether it will be occupied and how soon by a moving obstacle.

The static RGM is built once by calculating the distance of each cell to the goal position and by incorporating information about cells belonging to static obstacles. The distance of each cell to the goal position is evaluated using the Euclidean or city block distance metric, although other distance metrics can also be used. Hence, it includes the first two sources of information concerning the goal position and static obstacles.

Information provided from the short-term and long-term prediction modules is included in the dynamic RGM. The inclusion of the short-term prediction is trivial and it involves zeroing the reward associated with the grid cell that is the predicted next-time position of the obstacle.

The long-term prediction refers to the prediction of the destination position of the obstacle's movement. Thus, the reward value of the grid cells that are in the trajectory from the obstacle's current position and its predicted destination position is discounted. Hence, the value of a cell, (i, j), in the dynamic grid map, DGM, is given by a function

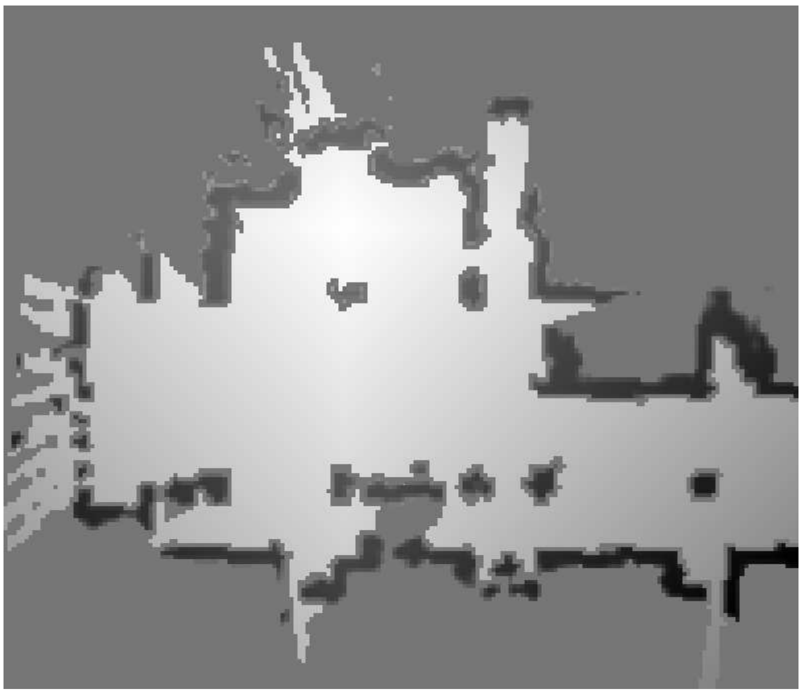
$$DGM(i, j) = w_i \cdot extent^{\gamma},$$

where w_i is the probability that the predicted destination point will be reached, $extent$ is a constant with range $[0, 1]$, that controls how far the robot should stay from obstacles and γ is the factor that determines how much the value of $extent$ should be discounted. The value of γ depends on the estimate of how far in the future this cell will be occupied. For example, if a cell (i, j) is to be occupied shortly in the future, γ will be close to 0, and thus the reward assigned to cell p will be small. On the other hand, if cell p is to be occupied far in the future, γ will be close to 1 and thus the reward assigned to this cell will not be significantly discounted. The γ factor is defined such that the long-term prediction is utilized only partially for predictions that are not too far in the future. Furthermore, the γ factor is dependent on the distance between the robot's current position and the position of predicted object position to be discounted. In this way, the movement of objects and humans that are far from the area that the robot operates currently are not affected significantly.

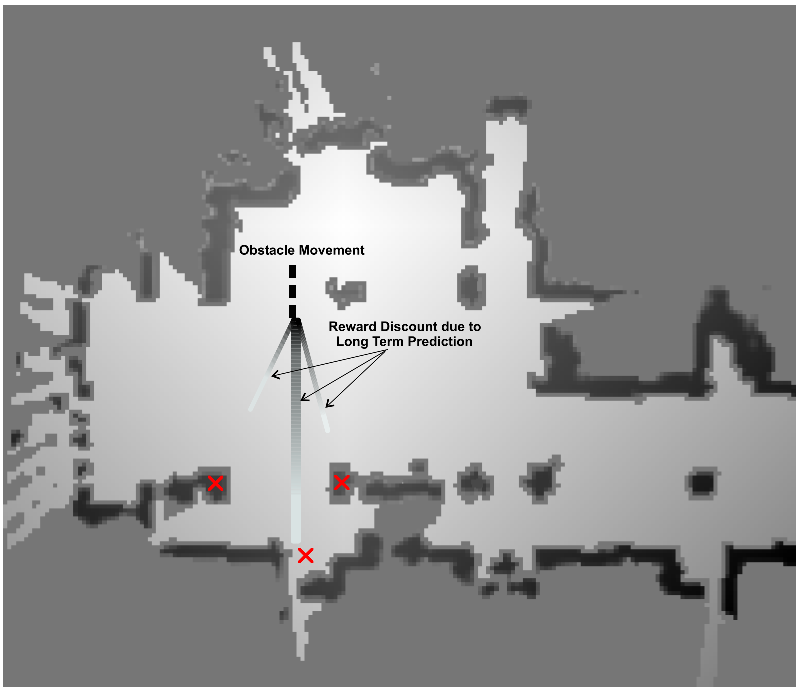
Superimposing the static and dynamic RGMs provides the reward function that is updated at each time step. In Figure 5.8 an example of discounting the reward values for an obtained long-term prediction is shown.

5.5 Conclusions

This chapter presented the methodology used for predicting the future motion of humans and/or other objects by incorporating two kinds of prediction: short-term and long-term prediction. Long-term prediction is obtained with the use of the defined "hot" points of the environment. These points can be defined manually or learned through an automated procedure that constructs a map of "hot" points. The long-term prediction module is also utilized in the proposed motion tracker. Finally, the integration of the obtained prediction to the navigation model has been described.



(a)



(b)

Figure 5.8: (a) The static and (b) dynamic RGM. Reward discount is performed according to the obtained long-term prediction. Long-term predictions for hot points present in the periphery of the field-of-view have low probability, w_i, and thus the reward discount is smaller.

6

POMDP Solution for Controlling the Robot's Speed

In this chapter, we develop a methodology of solving POMDPs that allows the robot to decide if it should increase or decrease its speed to avoid an obstacle more effectively. Without control of its speed, the robot has to make detours or follow a suboptimal path to reach its goal in order to avoid collision with obstacles. In many cases, the robot can avoid making these suboptimal actions if it can either increase its speed to bypass the obstacle or decrease its speed to let the obstacle move away from the robot.

The speed of the robot is chosen not to be included as a characteristic of its state as with its location and orientation. Moreover, speed actions are also not included in the action set. Such a choice would further increase the state space or action space. Instead, the solution of the POMDP is modified to account for the speed decision that the robot has to make.

The robot is allowed to move at three different speeds: $normal$, $fast$ and $slow$.

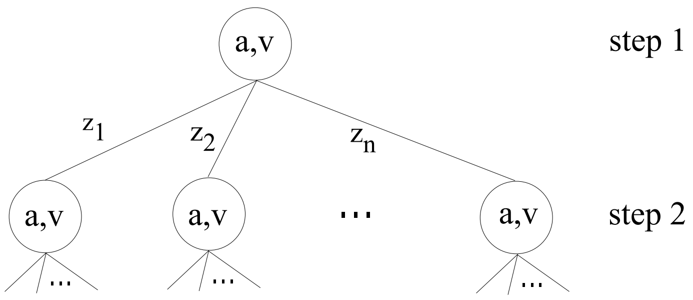


Figure 6.1: An example policy tree of a POMDP with pairs of actions and speeds.

6.1 Exact Solution

As noted in Section 3.3, to solve a POMDP exactly it is required to evaluate the α-vectors by the following equation:

$$V_t^*(b) = \max_{\mathbf{p}\in\mathbf{P}} b \cdot \alpha_{\mathbf{p}}^t$$

that searches all possible policy trees $\mathbf{P}$. Recall that policy trees are defined by having as nodes actions that are connected with each possible observation. To decide the action to be executed as well the speed of the robot, nodes are now defined by pairs of actions and speeds. Thus, an example policy tree is now as the one shown in Figure 6.1.

Since policy tree nodes are composed of action-speed pairs the value function of a policy tree has now to be evaluated by considering the choice of the action as well as the speed. The value function of a policy tree is evaluated by Equation 3.8, that is modified to incorporate the speed decision as

$$V_t^{\mathbf{p}}(b) = \sum_{s\in S} b(s) \left[R_m(s,a,v) + \gamma \sum_{z\in\mathcal{Z}} \sum_{s'\in\mathcal{S}} \mathcal{O}_m(z,s',a,v)\mathcal{T}_m(s,a,v,s')V_{t-1}^{z_{\mathbf{p}}}(s') \right]$$

The above equation dictates that it is required to a have a modified version of the reward, transition and observation functions. However, instead of defining a new POMDP function the

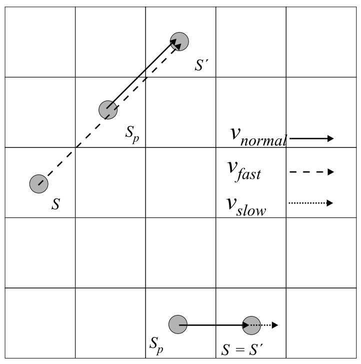


Figure 6.2: Definition of the projected state s_p.

notion of the *projected state* is defined that allows to use the original POMDP functions.

6.2 The Projected State

When the robot is in a state s and performs an action a with a velocity v, other than the $normal$ velocity, it is expected to end up with a certain probability in a state s'. Then, the $projected$ state is defined as the state s_p, where if the robot executes the same action a, from state s_p, with the $normal$ velocity it will end up with the same probability to state s'. Of course, if v is the $normal$ velocity of the robot, then s_p is s.

In Figure 6.2 an example of determining the projected state when the robot moves with the $fast$ and the $slow$ speed is illustrated. For clarity reasons, in this example it is assumed that the $fast$ speed is twice the $normal$ speed and the $slow$ speed is half the $normal$ speed. In the case the robot moves at $fast$ speed it forwards two grid cells since with the $normal$ speed it would forward one grid cell. On the other hand, in the case the robot moves at $slow$ speed it will remain in the same grid cell.

The projected state s_p is determined geometrically by triangulation in the continuous space. The initial state s is transferred to the continuous space regarding that the robot is in the center location of the grid cell represented by state s. Following, the vector of the action angle is constructed in analogy with the vector of the action angle for $normal$ velocity. When the POMDP is built a certain $normal$ velocity is considered along with a certain duration of movement of each action cycle. Therefore, a certain $normal$ speed motion vector is always assumed in the initial transition probabilities. This vector is used to determine the $fast$ speed motion vector or $slow$ speed motion vector since the $fast$ and $slow$ speeds are defined as a fraction of the $normal$ speed. Hence, by triangulation the resulting state s' is determined. Next, the inverse procedure is used to determine the projected state s_p. Knowing the state s' and the $normal$ speed motion vector, triangulation is performed to determine the location of the projected state s_p. Finally, the location of the projected state s_p is transferred from the continuous space to a discrete grid cell.

The determination of the projected state s_p is an approximation according to the motion model of the robot used to determine the transitions between states.

6.3 The Modified POMDP Functions

As mentioned above, the POMDP parameters are not altered to include information about the robot velocity. Therefore, a formulation of the modified reward and transition functions, $\mathcal{R}_m$ and $\mathcal{T}_m$, that relates them to the original $\mathcal{R}$ and $\mathcal{T}$ functions that were built considering the $normal$ velocity of the robot only is required. This is achieved by utilizing the *projected* state of the robot.

Having defined the projected state, the relation of $\mathcal{R}_m$ and $\mathcal{T}_m$ to the original $\mathcal{R}$ and $\mathcal{T}$, respectively, can now be defined. By the definition of the projected state s_p, the relation of $\mathcal{T}_m$ to $\mathcal{T}$ is straightforward, and is written as:

$$\mathcal{T}_m(s, a, v, s') = \mathcal{T}(s_p, a, s').$$

The above equation assumes that the transition probabilities for executing a certain rotation action a at $normal$ speed are preserved when executing the same rotation action at the $fast$ or $slow$ speed. This is a safe approximation since in the context in which the change of speed is used, the robot will move at a speed other than the $normal$ speed for very short intervals only, i.e. only when the robot has to bypass an obstacle with the $fast$ speed or allow it move way by slowing down. Furthermore, the $fast$ and $slow$ speeds are defined as fractions of the $normal$ speed that are rather small and therefore the motion behavior of the robot does not change dramatically.

The definition of $\mathcal{R}_m$ is not as straightforward as for $\mathcal{T}_m$. If $\mathcal{R}_m$ is simply defined as $\mathcal{R}_m(s, a, v) = \mathcal{R}(s_p, a)$, then the robot would always choose to move with the $fast$ speed. This is because the $fast$ speed will always get the robot closer to the goal and thus the reward that it will receive will be bigger. Instead, it is desirable that the robot moves at a different speed from its $normal$ speed only if it has to avoid an obstacle. For that reason change of speed is penalized.

$$\mathcal{R}_m(s, a, v) = \mathcal{R}(s_p, a) - penalty,$$

The penalty factor for change of speed is defined in relation to the reward function to ensure that the robot has the desirable behavior. The reward function is built by calculating the distance of each grid cell to the goal position. The value assigned to each grid cell is the distance to the goal position, inverted and normalized. Therefore, the reward value of neighboring cells will always differ by a certain amount, as it can be seen in Figure 6.3(a). When the static RGM is built, the average expected difference[*] of the reward value between adjacent grid cells for every rotation action a, $Ediff(a)$, can be evaluated.

[*]The term expected does not signify the employment of any estimation procedure for the determination of these values.

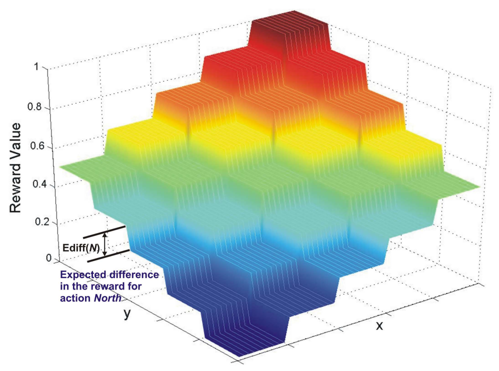
Thus, we define $\mathcal{R}_m$ as:

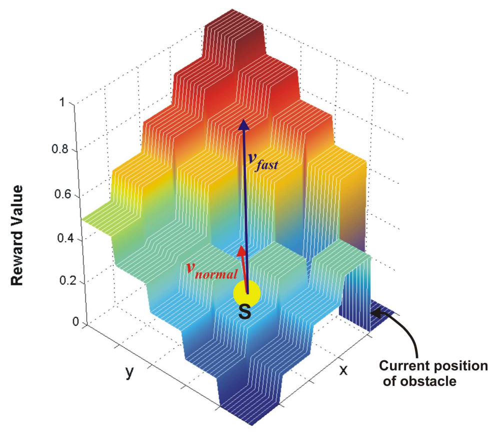
$$\mathcal{R}_m(s, a, v) = \mathcal{R}(s_p, a) - \alpha \cdot \frac{|v - v_n|}{v_n} \cdot Ediff(a),$$

where v_n is the $normal$ velocity of the robot and α is a constant that controls how preferable the velocity changes are. The bigger the value of α, the less preferable the velocity changes are. The $\frac{|v-v_n|}{v_n}$ factor, ensures that when v is the $normal$ velocity, the reward the robot will receive will not be penalized. It also accommodates for the effect of the difference between the $fast$ or $slow$ velocity with the $normal$ velocity on the reward the robot receives by $\mathcal{R}(s_p, a)$. For example, when the $fast$ velocity is double the $normal$ velocity, then we expect that the reward the robot will receive in these two cases will differ by $Ediff(a)$. In the case that the $fast$ velocity is triple the $normal$ velocity, then $\frac{|v-v_n|}{v_n}$ will be equal to 2, as we expect that the reward the robot will receive in these two cases will differ by $2 \times Ediff(a)$.

When there is no obstacle in the route of the robot to its goal, the reward values will be unaltered and dependent only on the distance to the goal. Hence, in the case of the $fast$ velocity, the reward the robot will receive after being penalized, will be the same with the reward the robot will receive for choosing the $normal$ velocity, for α equal to 1. In the case that there is an obstacle moving in the route of the robot to its goal, then the reward values of the cells that are predicted to be occupied by the obstacle in the future will be discounted. Then, the reward the robot will receive for $fast$ velocity will be bigger than the reward for $normal$ velocity even after it is penalized for changing speed.

An example for this case is illustrated in Figure 6.3(b). It is assumed that there is an obstacle moving in the environment. The reward value of the cell that corresponds to the obstacle's current position is set to zero. The reward value of the cells that are in the trajectory from the obstacle's current position to its predicted destination position is discounted. The original reward value of these cells can be seen in Figure 6.3(a). The discount of the reward





(a) (b)

Figure 6.3: (a) The static RGM and (b) An example of the robot choosing to move with the $fast$ velocity.

values is not the same for all cells, since it depends on the estimate of how far in the future each cell will be occupied. The robot is currently at the cell denoted with s. If the robot moves with the $normal$ velocity, it will maximize its reward when executing one of the suboptimal actions to reach the goal since the reward for executing action *North-East* has been discounted due to the long-term prediction. When the reward values for the $fast$ velocity are evaluated, it can be seen that the reward for executing action *North-East* will be the maximum. That is because the robot will end up in a state where its reward has not been discounted due to long-term prediction and even when the expected difference in the reward values for action *North-East* is deducted it will still remain bigger than all other rewards. Hence, the robot will move with the $fast$ velocity and bypass the moving obstacle.

In the case of the $slow$ speed, the reward the robot will receive for executing any action from the projected state will always be smaller than the reward the robot will receive for choosing the $normal$ speed, when there is no obstacle. This reward will be further decreased by the penalty factor. Therefore, for the robot to choose the $slow$ speed, the reward it receives for the $normal$ and $fast$ speed has to be smaller. This will be the case when there is an obstacle very close to the robot and did not have a long-term prediction to be able to avoid it by increasing its speed.

The relation of $\mathcal{O}_m$ to the original observation function $\mathcal{O}$, using the projected state s_p, is straightforward and is written as:

$$\mathcal{O}_m(s,a,v,z) = \mathcal{O}(s_p,a,z).$$

The above definition holds due to the way the observation set has been defined in Section 4.1. An observation is actually the distance the robot travelled when it has executed a certain action a. As a result, the relation of $\mathcal{O}_m$ to the original observation function $\mathcal{O}$ is in analogy with the relation to the definition for the transition function.

6.4 Approximation Methods

The described methodology for controlling the robot's speed can also be applied with any of the approximation methods reviewed in Section 3.3.2 with the use of the modified POMDP functions.

In the case that the MLS heuristic is used the optimal value function is computed as:

$$V_t^*(s) = \max_{a\in\mathcal{A},v\in\mathcal{V}} Q_m(\arg\max_{s\in\mathcal{S}}(b(s)),a,v)$$

where the modified Q-function, Q_m, is now defined as:

$$Q_m^t(s,a,v) = \mathcal{R}_m(s,a,v) + \gamma\sum_{s'\in\mathcal{S}}\mathcal{T}_m(s,a,v,s')V_{t-1}(s').$$

In the case that the $voting$ heuristic the optimal value function is given by:

$$V_t^*(s) = \max_{a \in \mathcal{A}, v \in \mathcal{V}} \sum_{s \in \mathcal{S}} b(s)\delta(\pi_{MDP(s)}, a, v)$$

where

$$\pi_{MDP(s)} = \arg \max_{a \in \mathcal{A}, v \in \mathcal{V}} Q_m(s, a, v)$$

and

$$\delta(a_i, v_i, a_j, v_j) = \begin{cases} 1, & \text{if } a_i = a_j \text{ and } v_i = v_j \\ 0, & \text{if } a_i \neq a_j \text{ or } v_i \neq v_j \end{cases}$$

In the same manner the modified POMDP functions can be applied to other approximation methods present in the literature.

6.5 Conclusions

In this chapter it has been described how a POMDP is solved so that it can decide the action the robot should execute and also its speed of movement. This is accomplished without the need of modelling the robot speed as a characteristic of its state but with the use of the definition of the projected state. Hence, the robot speed is decided without any increase in the POMDP size. Consequently, as it will be demonstrated in the following chapter where the results of the proposed navigation approach are presented, the robot can perform obstacle avoidance not only by performing detours or changing completely the path to the goal but also by increasing or decreasing its speed.

7

Results

This chapter presents experimental results that validate the proposed approach for autonomous robot navigation. Initially the experimental configuration for the real-world environment as well as the simulated is presented. Following, results that validate the learning of the POMDP model are given. Finally, results that demonstrate the behavior of predictive obstacle avoidance are illustrated.

7.1 Experimental Configurations

For testing the performance of the proposed framework, we have performed extensive tests with both real and simulated data. All real data have been assessed on Lefkos, an iRobot B21r robotic platform of our lab, while acting in various indoor areas of FORTH. Simulated tests have been assessed by a home build simulator capable of simulating various robotic sensors with arbitrary error level configurations.

Table 7.1: Lefkos Configuration

Sonar	48 (24 surrounding enclosure, 24 surrounding base)
IR	24 (surrounding enclosure)
Tactiles	56 (24 surrounding enclosure, 32 surrounding base)
Laser	SICK PLS scanner
Vision System	Pan Tilt stereo vision head or panoramic camera
Drive	4-wheel synchronous
Motor	4 high torque 24 VDC servo motors
Translate Speed	90cm/sec
Rotate Speed	167 deg/sec
Translate Resolution	1mm
Rotate Resolution	.35 deg
Steerable Turret	no power needed
Smart Panels	14 (6 surrounding enclosure, 8 surrounding base)
Diameter	52.5cm
Height	106cm
Weight	122.5kg
Floor Clearance	2.54cm
CPUs	2 Pentium based ATX computers
Communications	10 Mbps Wireless radio ethernet

7.1.1 Lefkos

All real experiments presented in this thesis have been assessed on a robotic platform of our laboratory, namely an iRobot-B21r, equipped with a SICK-PLS laser range finder. The range finder is capable of scanning 180 degrees of the environment, with an angle resolution of two measurements per degree and a range measuring accuracy of 5cm. Table 7.1 summarizes Lefkos hardware configuration while Figure 7.1 shows pictures of Lefkos.

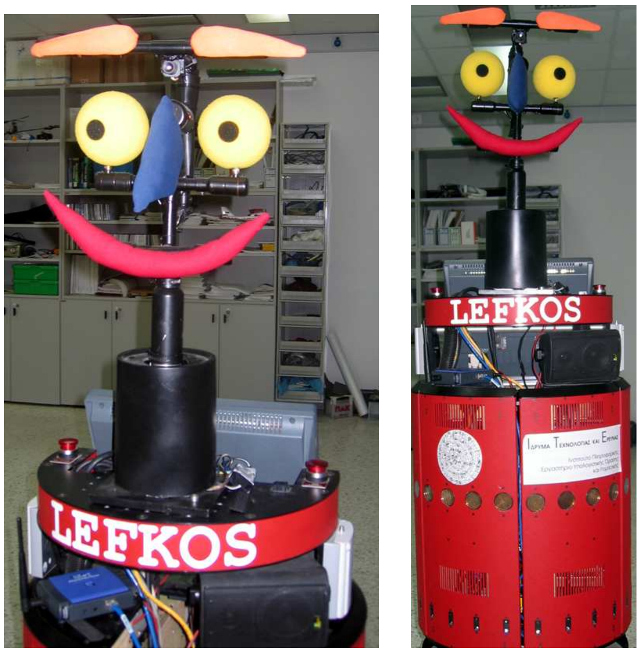


Figure 7.1: Lefkos.

7.1.2 Simulator

For testing the performance of the proposed framework, against arbitrary robot configurations and being able to compare the results with accurate ground truth, a home-built simulator allowing various environment and robotic hardware configurations was utilized. The simulator is capable of performing in arbitrary simulated environments, interpreting and performing all robot motion commands and simulating all robot sensors at various error levels.

7.2 Evaluation of the Learned Model

In order to test the validity of the learning procedure, we have set up an experiment aiming at a quantitative evaluation of the model that results from a learning session in specific environments. Two learning sessions have been performed; a learning session in a simulated environment where the ground truth is available and also one in a real environment. Learning is performed in both cases with the Baum-Welch algorithm detailed in Section 3.4. The environment chosen for both experiments is the FORTH main entrance hall, as shown in Figure 7.3.

In both experiments, execution traces have been collected where the robot goes from a start state to a goal state. The start and goal states were different for each execution trace. The RN-HPOMDP for both experiments was built by discretizing the environment into $5cm^2$ cells with 7 levels of hierarchy, that results to a discretization step of the orientation and action angles of $5.625°$. The model "appropriateness" has been evaluated using the fitness and entropy measures defined in [64] as:

$$fitness = 1/T \times \ln p(o_{1...T}|a_{1...T})$$

$$entropy = 1/(T \ln |S|) \times \sum_{t=1...T} \sum_{s \in S} [\alpha_t(s) \ln(\alpha_t(s))].$$

Fitness is an indicative measure of how well the model explains an execution trace of length T, i.e. how probable is to obtain an observation o when an action a is executed. Entropy is a measure of how certain the robot is about its position. The entropy is evaluated by making use of the alpha values, $\alpha_t(s)$, that express the probability that the agent occupies state s at time t given the observation-action pairs until that time and are utilized by the Baum-Welch algorithm. The Baum-Welch algorithm is repeated for a number of epochs until it converges. The fitness and entropy measures are graphically shown in Figure 7.2 for each training epoch. Ideally, fitness and entropy should converge to zero after a sufficiently large number of training epochs. As expected, convergence to zero is not achieved, as its the case with all learning procedures. Still, after a rather small number of epochs, fitness and entropy converge to low values, indicating the validity of the learned model.

In order to provide additional quantitative results on the model accuracy, the position and orientation accuracy in maintaining the robot's state was measured and is shown in Table 7.2. The peak of the POMDP's belief distribution was used as the model's estimate of the robot's

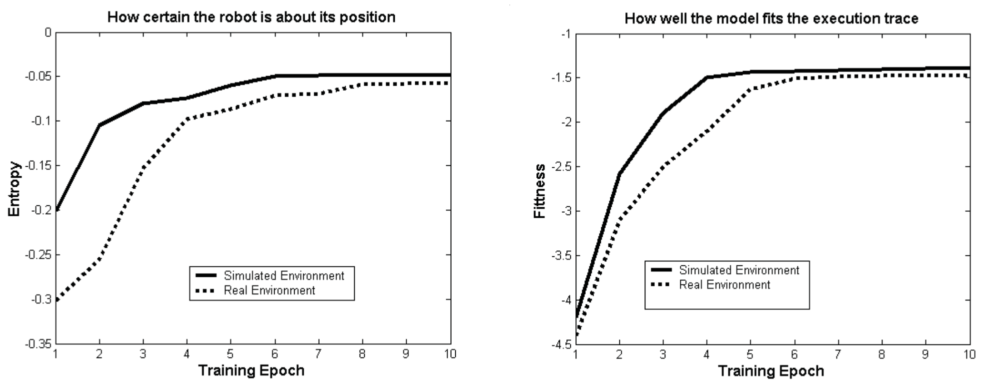


Figure 7.2: Evaluation of the learned RN-HPOMDP model.

current state. As can be observed, the figures indicate increased accuracy of the learned model.

In the simulated environment experiments, where the ground truth is available, the position and orientation errors were measured at each time step during execution between start and goal points.

In the real environment experiments, two distinct robot locations were manually marked on the floor of the FORTH main entrance hall, as shown in Figure 7.3. The robot was driven manually, as accurately as possible, to one of the marked locations and the other marked location was set as the goal position the robot had to reach. The error in the x, y location and orientation between the robot's position after executing the trace obtained by the POMDP model and the marked location it had to reach, was measured manually as accurately as possible.

The mean position and orientation error for both experiments is very close to the discretization of the POMDP, as indicated by the entropy and fitness measures of the learned models. Both experiments, validated that the learned POMDP models were consistent during execution in terms of maintaining the robot's belief and also in reaching the goal position.

Table 7.2: Position and orientation accuracy of the learned model.

Mean Error	**Real Environment**	**Simulated Environment**
$x(m)$	0.053	0.023
$y(m)$	0.061	0.041
$f(deg)$	5.525	5.041

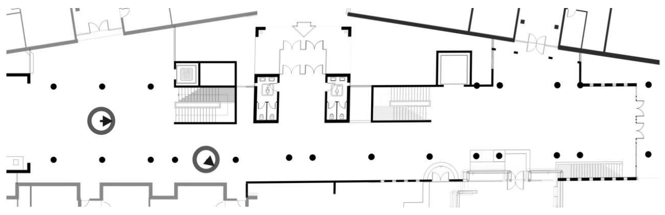


Figure 7.3: The marked locations in the environment where the experimental evaluation of the RN-HPOMDP model was performed.

7.3 Results

In this section a representative set of results of the robot operating in the FORTH main entrance hall is shown. The robot was set to operate for more than 70 hours. The environment was modelled with a RN-HPOMDP of size $|S| = 18,411,520$, $|A| = 256$ and $|Z| = 24$. The RN-HPOMDP was built with 7 levels. Experiments were performed in a dynamic environment where people were moving within it. In all cases the proposed navigation model has shown a robust behavior in reaching the assigned goal points and avoiding humans or other objects. Following, sample paths the robot followed to reach its goal position by demonstrating the four main behaviors it uses to avoid obstacle are presented.

7.3.1 Avoiding obstacles with a detour

In this experiment we demonstrate how the robot avoids two humans moving in the environment in such a manner that they block its route to the goal position. If there were no humans or other objects moving, the robot would follow a straight path to its goal, defined for our experiments as shown in Figures 7.4 and 7.5. In our experiment two humans are moving in the environment. One of them is moving towards the straight path that the robot would follow to reach its goal and the other one is moving in a straight direction vertical to the one the robot would follow. As shown in the figures the robot detects the moving humans and obtains the long-term prediction of their movement and hence decides to make a detour by turning. The decision the robot makes about the detour is long before the robot actually faces the moving humans and where a local obstacle avoider would decide to make a detour.

7.3.2 Avoiding obstacles by following a replanned path

In this experiment we show that the robot can decide to follow a completely different path from the one it would follow in a static environment in order to avoid humans moving. It is obvious from the images shown in Figures 7.6 and 7.7, that the optimal path to reach the goal position if the robot was operating in a static environment it would be to follow an almost straight trajectory. In our experiment, a human was moving to block this static optimal path and the robot decided to follow a completely different path, i.e. follow a trajectory that goes behind the building's column to reach the goal position.

7.3.3 Avoiding obstacles by decreasing the robot's speed

In the experiment shown in Figure 7.8 the robot cannot perceive the movement of both obstacles all the time. The person's movement denoted in the figure with the yellow square is occluded at the beginning and the robot can see it only after it has passed the building's column. However,

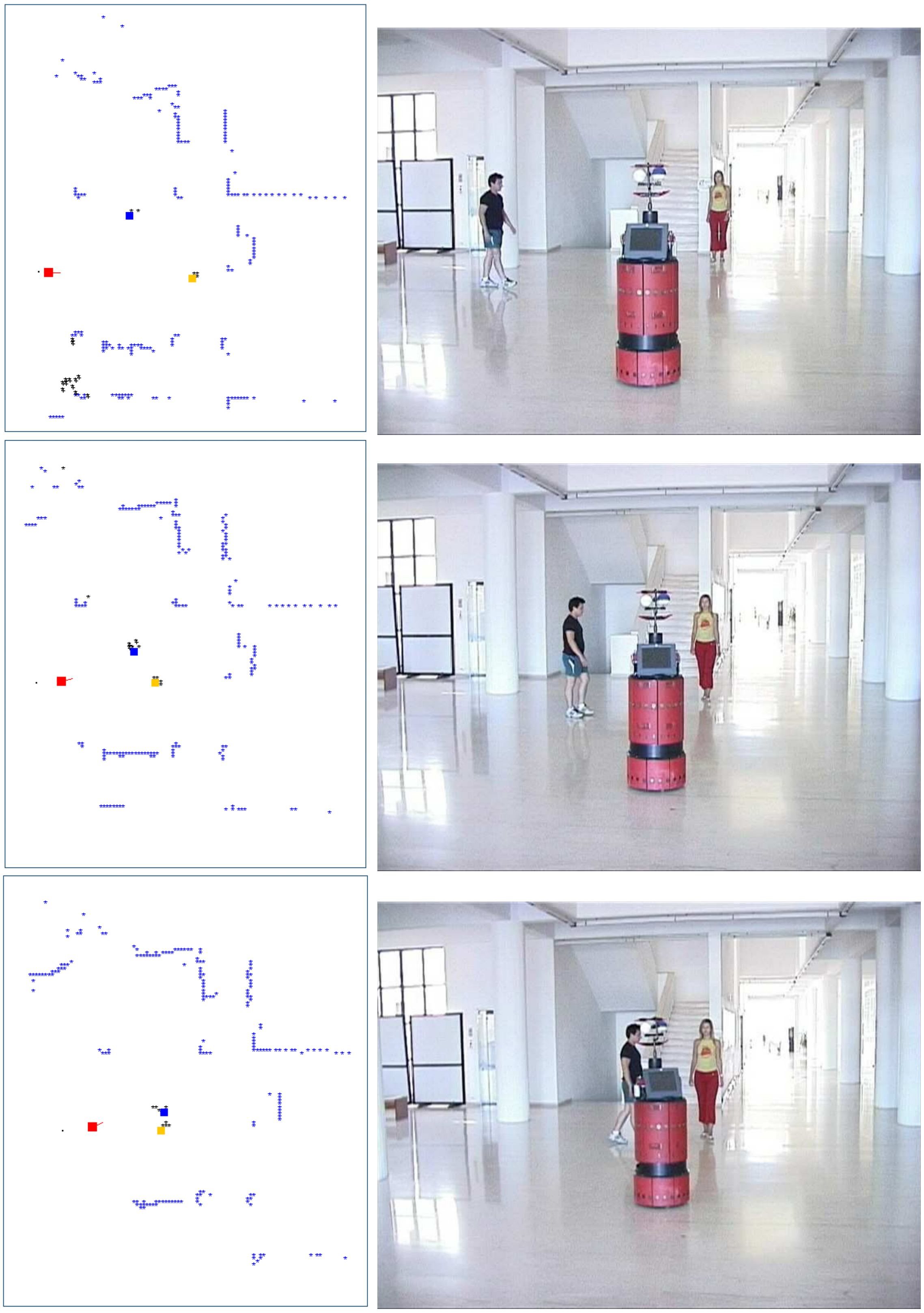


Figure 7.4: Avoiding two moving objects with a detour (I).

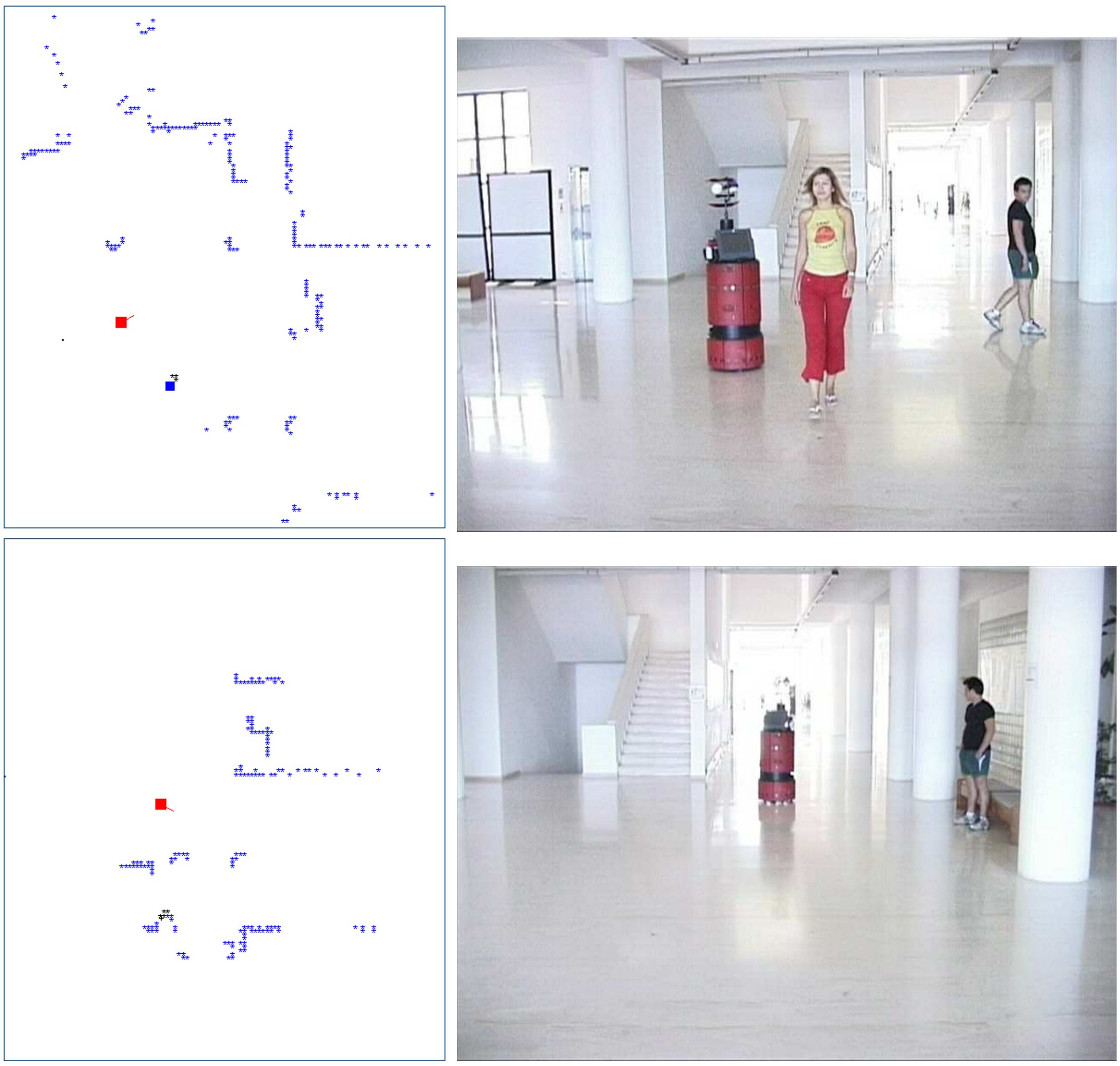


Figure 7.5: Avoiding two moving objects with a detour (II).

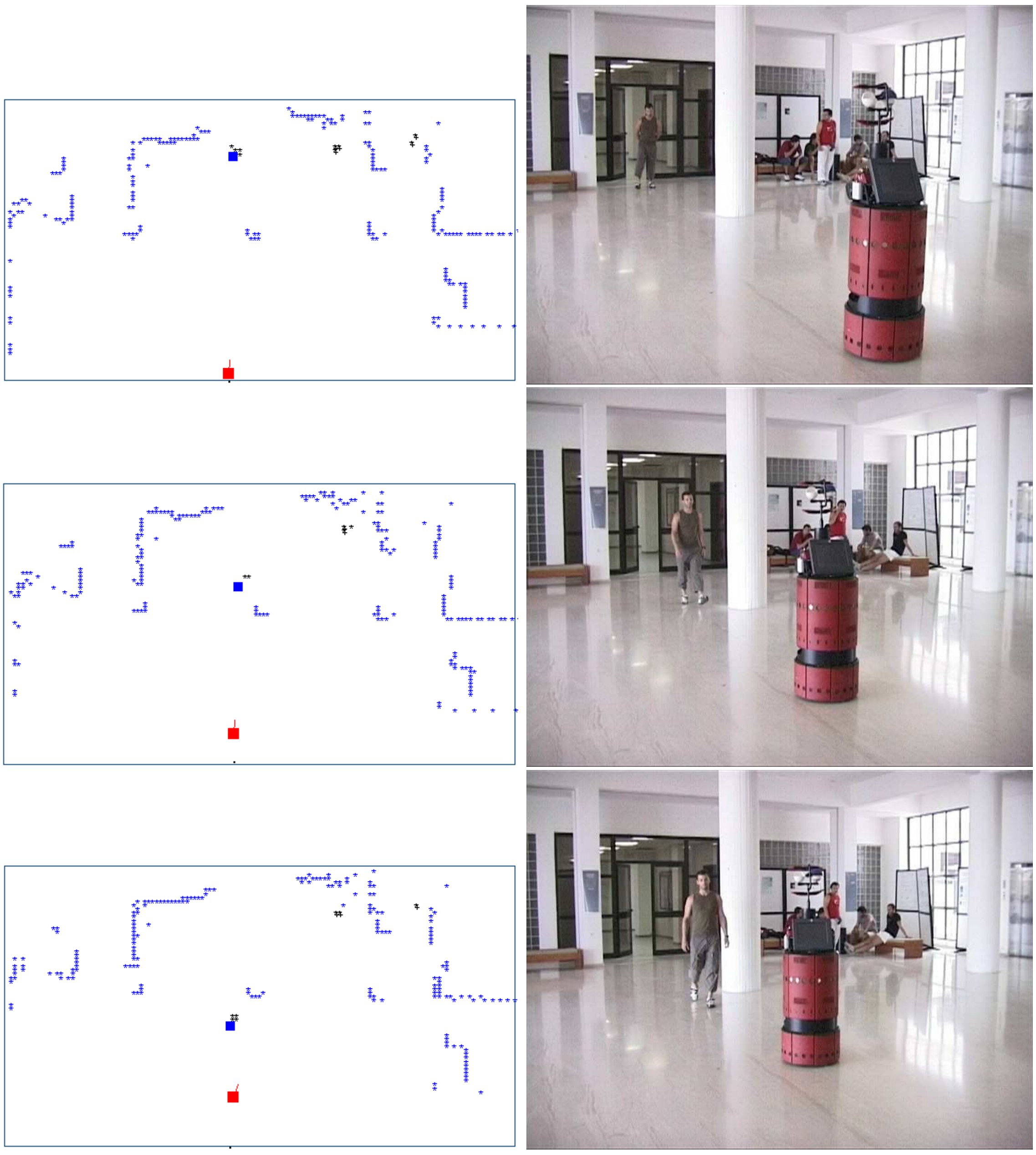


Figure 7.6: Deciding to follow a completely different path (I).

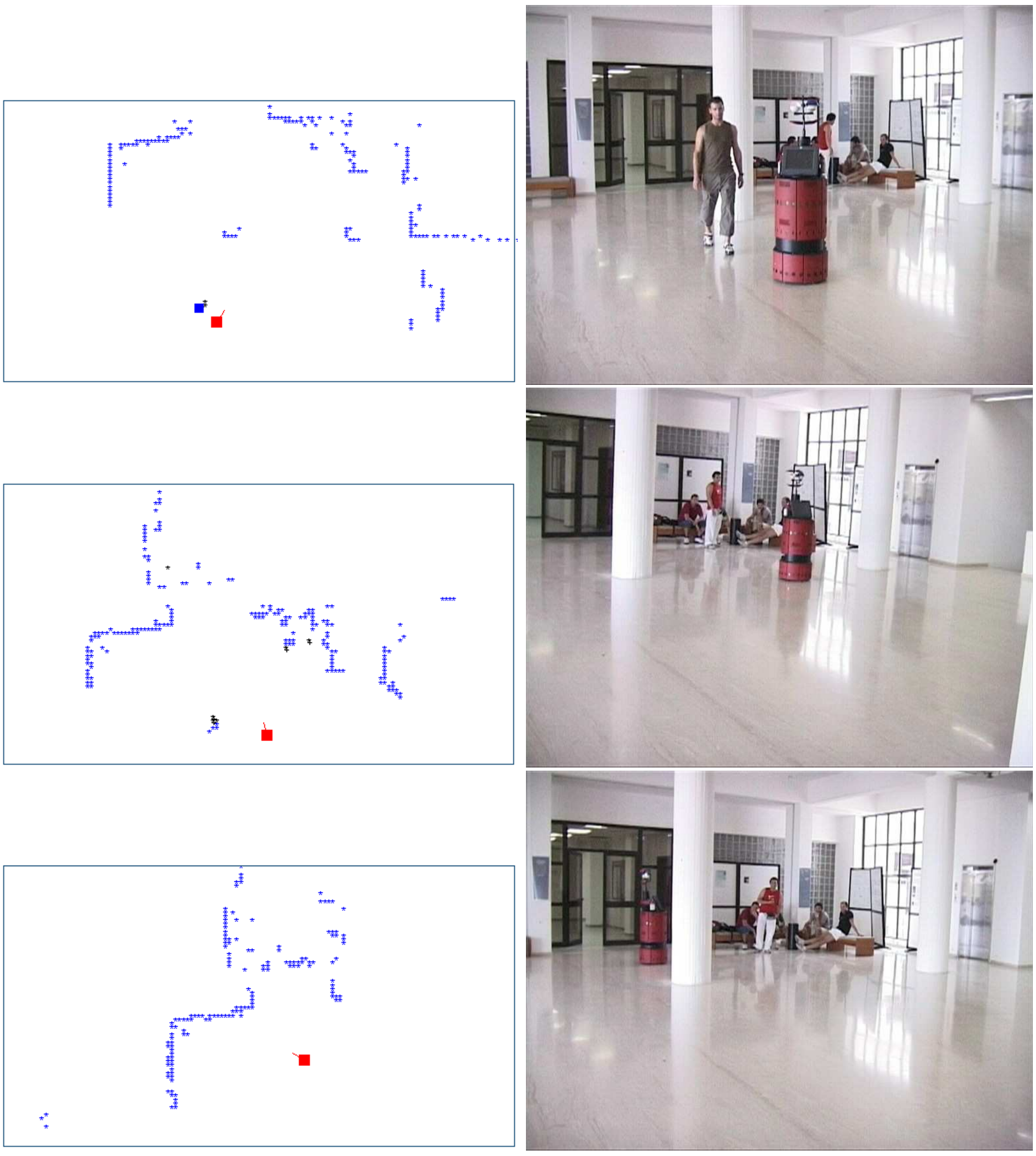


Figure 7.7: Deciding to follow a completely different path (II).

at that point the robot cannot increase its speed to bypass this person or make a detour since there not enough space at that point. Hence, the robot decides to decrease its speed until the person blocking its way to the goal has passed away. After this point the robot reverts to its normal speed and continues its movement until it has reached its goal position.

7.3.4 Avoiding obstacles by increasing the robot's speed

In the experiment shown in Figure 7.9 the robot increases its speed to bypass the human's movement in order to reach its goal position without the need of making a detour. The human's movement is perceived by the robot from the beginning and hence it obtains the long-term prediction early enough to decide to increase its speed to bypass the human's predicted motion trajectory. When the robot has passed the human's motion trajectory its decreases its speed to normal to continue its movement.

7.4 Comparative Results

To further evaluate the appropriateness of the proposed approach a set of comparative experiments have been performed. With these experiments it is aimed to provide quantitative measures of how well the proposed approach performs when it is applied in dynamic environments where the robot's movement is obstructed by humans.

The experiments were performed in the simulated environment of the FORTH main entrance hall. The robot was set to reach various goal points and each goal point was reached in the environment in the case where it is static, i.e. there is no human movement, and in the case where it is dynamic. In the case of operating in a dynamic environment, the same experiment was performed by having from one up to five humans moving within the area the robot has to reach. Furthermore, in the case of having four and five humans moving within the environment, experiments were setup such as that all humans move in the area that the robot

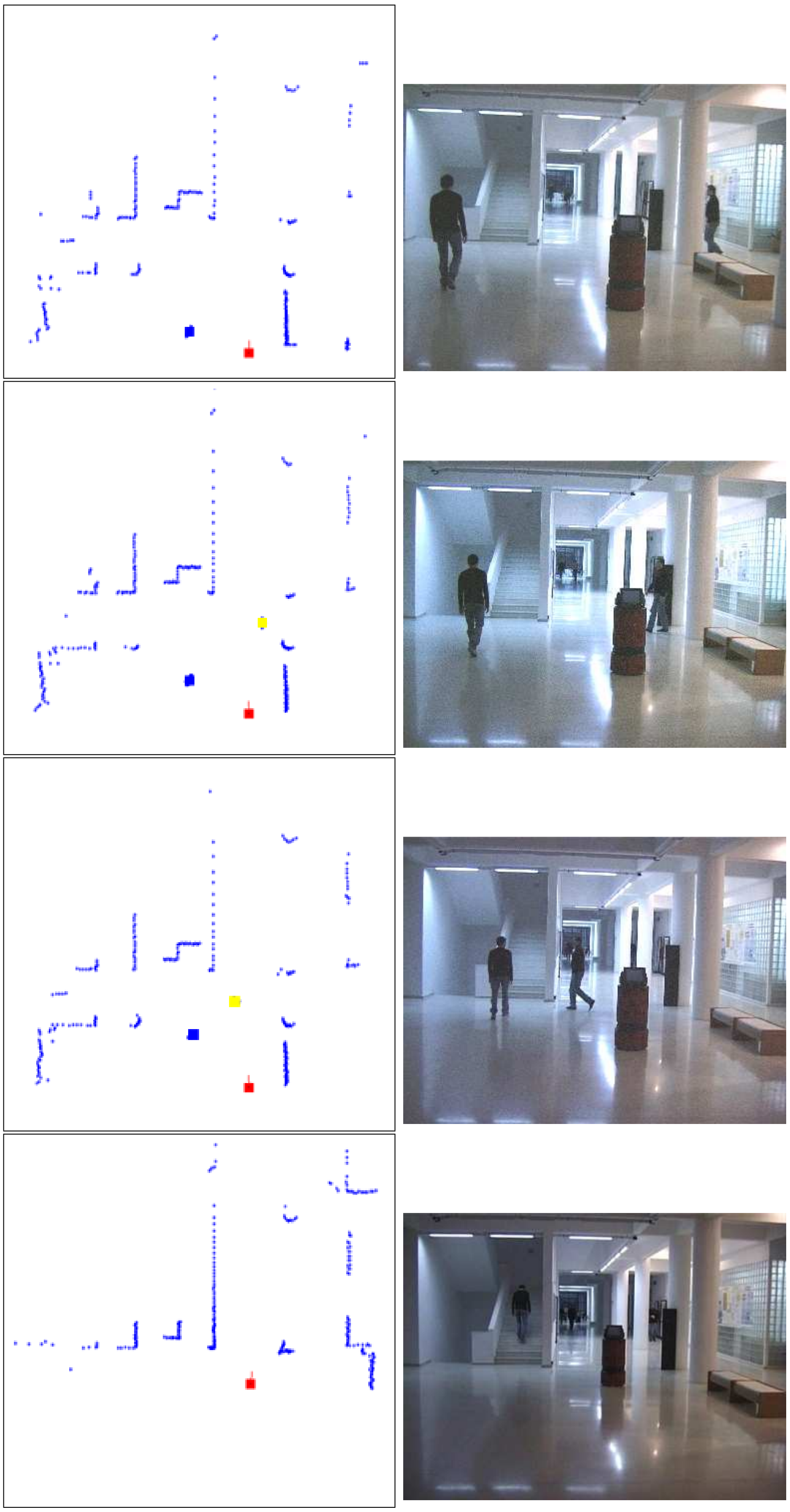


Figure 7.8: Avoiding obstacles by decreasing the robot's speed.

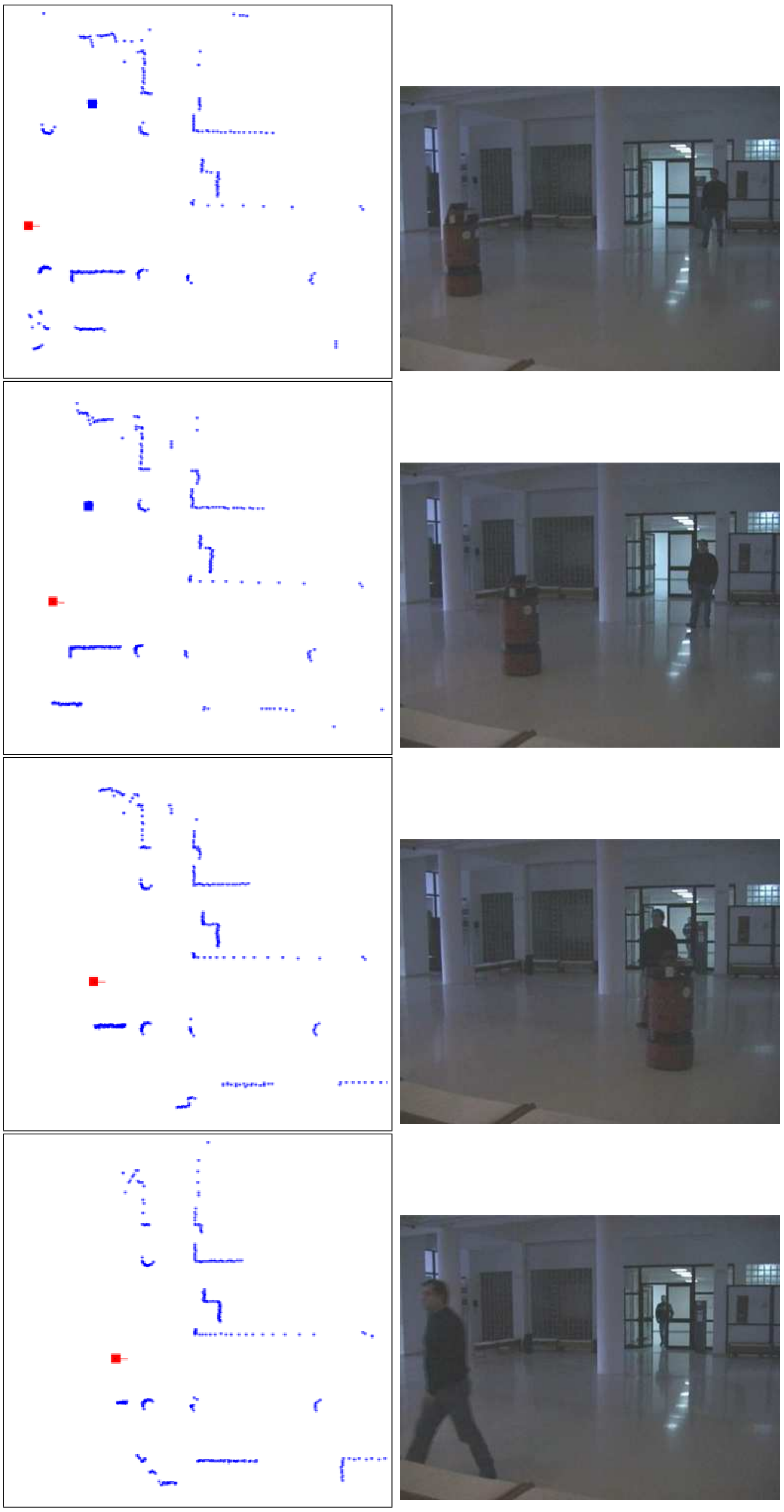


Figure 7.9: Avoiding obstacles by increasing the robot's speed.

approaches to reach the goal point in a static environment but they were also performed with a setup where humans were moving within this area and within the area that the robot would perform manoeuvres to avoid humans.

In Figure 7.10 an example is shown of the human motion areas that are defined for the comparative experiments performed. In this figure a sample configuration for a start and a goal point is shown along with a shaded area that denotes the area within the robot would choose to move to reach the goal point in a static environment. Hence, in the performed experiments humans are set up to move within this area so that the robot would have to employ obstacle avoidance techniques. In the special case of the experiments where there are four or five humans moving in the environment there have been used two setups. In one of them all humans move within the area that the robot will operate and in the other setup humans move also outside this area but still within the area where the goal point lies. The second setup enables us to further evaluate the performance of the proposed approach since alternatives routes to a goal point will be also obstructed by human motion.

In addition, the dynamic environment experiments were performed with and without the use of the prediction module. The experiments were performed using 50 different configurations of start and goal points of the robot. The type of experiments performed for each configuration are summarized in Table 7.3.

The time required to execute the path obtained in the case where the environment is static is taken as the optimal time required to perform each experiment. This time is utilized as a reference time to compare against the time required to reach the goal point in all other types of experiments. Thus, we assume that the desirable behavior of the proposed approach is to be able to reach the goal point in a time that is as close as possible to the time the robot has taken when it operated in a static environment. This measure provides us with an insight of how efficiently and effectively the robot can avoid moving obstacles.

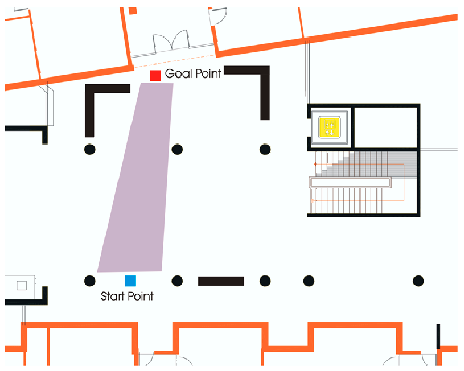


Figure 7.10: An example of how the human motion areas are defined for the comparative experiments performed.

In Table 7.4, the outcome of the performed experiments is presented. In this table, a number that ranges from zero to one has been evaluated for each type of experiment, as defined in Table 7.3, that denotes how close the performance of the proposed approach was to the optimal one,

Table 7.3: Type of experiments performed for each configuration of start and goal point.

Experiment	**No. of humans**	**Prediction Enabled**	**Human Motion Areas**
$static$	0	No	–
$dyn1a$	1	Yes	1
$dyn1b$	1	No	1
$dyn2a$	2	Yes	1
$dyn2b$	2	No	1
$dyn3a$	3	Yes	1
$dyn3b$	3	No	1
$dyn4a$	4	Yes	1
$dyn4b$	4	No	1
$dyn4c$	4	Yes	2
$dyn4d$	4	No	2
$dyn5a$	5	Yes	1
$dyn5b$	5	No	1
$dyn5c$	5	Yes	2
$dyn5d$	5	No	2

i.e. when the robot performed the experiments in a static environment. Since all the experiments were performed in a simulated environment the time taken to complete a motion path is actually the number of time steps that have been executed. The overall performance for each type of experiment has been evaluated as the average of the performance for each of the 50 distinct configurations used, as:

$$C_i = \sum_{k=1}^{50} c_{i,k},$$

where C_i is the overall performance of each type of experiment i as shown in Table 7.4 and c_i is the performance of each type of experiment i when executing a specific configuration k, that is evaluated as:

$$c_{i,k} = \frac{no.\ of\ time\ steps\ in\ static\ environment\ at\ configuration\ k}{no.\ of\ time\ steps\ in\ experiment\ i\ at\ configuration\ k}.$$

As can be observed in Table 7.4, the proposed approached when utilized with motion prediction of the human movement is superior to that when used without prediction. Furthermore, as the number of humans increases the difference in performance of the proposed approach with prediction and without prediction is more apparent. This is due to the fact that when prediction is available the robot can decide to follow a completely different path that is free instead of getting close to humans and making manoeuvres to avoid them, where as the number of humans increases it is more difficult to perform such manoeuvres. This becomes clear when the performance of the proposed approach is observed in the case of the experiments performed with four or five humans and the effect of having them moving in one or two areas. When the proposed approach is utilized with prediction it can be seen that the performance is not affected dramatically since the robot can decide early enough to follow a path that will not get congested. On the other hand, when there is no prediction utilized the robot has to avoid many humans and in many cases it does not have the space to perform appropriate manoeuvres. Consequently, the performance is greatly affected when all humans move within the same area and there in no prediction utilized. The performance in the case of the experiments performed without prediction and with humans moving within two areas is actually equivalent to that of the experiments $dyn2$ and $dyn3$ since the robot does not change its path to the goal.

Finally, overall the performance of the proposed approach with prediction utilized remains good as the number of humans increases and the deviation from the optimal time required when operating in a static environment remains small. As a result, the proposed predictive navigation approach has shown a stable performance even when the number of humans present in the environment increases and is able to produce paths to reach a goal point that are not too far in time measures of the corresponding paths that the robot would execute in a static environment.

Table 7.4: Overall performance of the proposed approach for each type of experiment performed.

Experiment	**No. of humans**	**With Prediction**	**Without Prediction**	**Human Motion Areas**
$dyn1$	1	0.947	0.938	1
$dyn2$	2	0.922	0.873	1
$dyn3$	3	0.859	0.786	1
$dyn4$	4	0.795	0.691	1 or 2
$dyn4$	4	0.823	0.584	1
$dyn4$	4	0.767	0.798	2
$dyn5$	5	0.788	0.609	1 or 2
$dyn5$	5	0.818	0.590	1
$dyn5$	5	0.758	0.628	2

7.5 Conclusions

The results of the proposed predictive navigation approach modelled with the RN-HPOMDP have been presented in this chapter. These results have demonstrated the effectiveness of the proposed approach for navigation in dynamic environments. Its performance has been shown to remain stable when the number of humans and/or dynamic objects present in the environment increases. Furthermore, the predictive navigation approach provides four distinct obstacle avoidance behaviors that are activated by the RN-HPOMDP as required based only on the robot perception and the obtained future motion prediction for humans and/or other objects.

8

Conclusions

To conclude this thesis we will review the work presented, the main contributions of this thesis and we will point out possible directions for future research.

8.1 Summary

In this thesis, we have proposed a new predictive approach to the autonomous robot navigation problem. The proposed approach is based on a Partially Observed Markov Decision Process (POMDP), that is the navigation problem is treated in a probabilistic manner. Furthermore, the POMDP model is utilized as a unified model for navigation that does not require any other external module to perform the tasks of localization, planning and obstacle avoidance.

In this thesis the problem of navigating effectively in crowded environments was considered. The desired robot behavior is to reach its goal points in an optimal manner in quantitative as well as quality measures. Therefore, it is required to reach its goal points in the shortest time possible as far as quantitative measures are concerned but also by following “elegant” and "aware" paths to satisfy the quality measures. The robot executes "elegant" and "aware" paths when they are smooth and do not direct the robot to highly congested areas where it should perform many manoeuvres but also when changes in the environment that might eventually block its route to the goal point can be realized and hence avoided. This kind of behavior is achieved by incorporating into the model future motion prediction. Two kinds of motion prediction are utilized: short-term and long-term prediction. The long-term future motion prediction is the element that allows to plan paths that are "aware" and hence "elegant". An important aspect when planning is how much information is utilized, i.e. whether planning is performed globally or locally. The proposed framework is capable of planning globally and hence take into account all available environment information. Furthermore, global planning is performed continuously and hence the robot can react quickly to changes in the environment. Finally, the robot is allowed to modify its speed, increase it or decrease it, in order to avoid obstacles more effectively.

All navigation tasks are performed with the use of a POMDP. POMDPs are probabilistic sequential decision making models that have been used so far for high-level planning. In this way, the probabilistic manner in which planning was performed was only partially exploited since it is required to use non-probabilistic obstacle avoidance and/or low-level planning methods. The main drawback of POMDPs when used for navigation is their extreme memory and computational requirements. As a result, in this thesis a hierarchical representation of POMDPs specifically designed for robot navigation, the RN-HPOMDP, has been developed that is capable of modelling the environment at a fine resolution and also being solved in real-time on-line. Consequently, uncertainly is considered in all aspects of navigation.

Experimental results have shown the applicability and effectiveness of the proposed approach for indoor navigation tasks where optimal planning and effective obstacle avoidance are required at the same time.

8.2 Contributions

The primary contributions of this thesis are the modelling of the autonomous robot navigation task via HPOMDPs and the future motion prediction of humans and/or other moving objects. Utilization of POMDPs towards the development of a unified probabilistic framework for autonomous robot navigation gave rise to contributions, which are summarized below.

- **RN-HPOMDP.** A novel representation of hierarchical POMDPs is proposed in this thesis specifically designed for the autonomous robot navigation problem. It has been shown that the RN-HPOMDP can effectively model large real-world environments at a fine resolution and it can decide the actions the robot should execute in real-time. The RN-HPOMDP integrates all three aspects of navigation: localization, planning and obstacle avoidance. Finally, it should be stressed out that the RN-HPOMDP performs these three tasks and provides the actual actions the robot executes without the intervention of any other module.

- **Reference POMDP (rPOMDP).** The RN-HPOMDP is modelled in such a manner such that the functions that characterize it are based only on the motion model of the robot. This feature has enabled us to introduce the notion of the Reference POMDP (rPOMDP) that conveys this robot specific information. The rPOMDP is a very small POMDP in size, independently of the actual size of the environment, that is used to maintain all probability matrices of the RN-HPOMDP.

- **Long-Term Prediction.** A novel approach for long-term prediction of the motion trajectory of humans and/or other objects is proposed in this thesis. The long-term prediction is based on the definition of the so called “hot points”, i.e. points of interest in the environment that people tend to visit often. The “hot points” can be either defined manually or be learned through an automated procedure that produces a map of “hot points". The map of "hot points" in essence conveys information regarding the patterns of motion behavior in a specific environment.

- **Motion Tracking.** A new methodology for multiple object motion tracking is also presented in this thesis. The motion tracker utilizes the short-term and long-term prediction obtained for obstacle avoidance to perform the estimation step. The data association step is performed by a nearest neighbor filter that is however verified by the motion trajectory of detected objects obtained by the long-term prediction module. Furthermore, where the nearest neighbor filter fails to match objects due to occlusions the long-term prediction module is again utilized.

- **Control of Robot Speed to Avoid Obstacles.** The robot is allowed to increase or decrease its speed in order to avoid obstacles more effectively. In this manner, unnecessary detours can be avoided. The change of speed is decided by the POMDP used for navigation. The notion of the projected state is introduced that allows the use of the regular POMDP functions without the need of incorporating into the state space or the action space the different speed at which the robot can move.

8.3 Future Work

Throughout this thesis, we have studied the application of POMDPs for the autonomous robot navigation problem along with prediction methods for effective obstacle avoidance. The proposed probabilistic framework consents encouragingly on the applicability of POMDPs for modelling in a unified manner the three navigation tasks of localization, planning and obstacle avoidance. The proposed hierarchical POMDP has been shown appropriate for providing directly the actions the robot should perform without the intervention of any other module and hence exploiting its probabilistic nature in all aspects of navigation. During our study we have gained significant insight into the challenges as well as the opportunities related with such an attempt. We believe that this thesis has contributed to research effort in this topic; still there are interesting aspects that deserve further research endeavors. To conclude this thesis, we will point out some areas appearing to invite productive future work.

- **Reference POMDP (rPOMDP).** The proposed rPOMDP has been utilized to model the robot motion behavior in the POMDP functions. Hence, we were able to transfer the probabilities learned for the rPOMDP to the whole hierarchical structure of POMDPs. Environment specific information is included only in the reward function of the POMDP and this has enabled us to plan with POMDPs in real-time for large environments. However, the rPOMDP has limitations since it defines a small state space in which the motion behavior is modelled. To alleviate this limitation it is proposed to investigate the use of continuous functions that can be learned to model the robot motion behavior and then use these reference function in analogy with the rPOMDP to transfer the required probabilities on-line while planning. An important aspect of this approach is the way in which the observation set of the POMDP is modelled. In our approach, observations do not infer environment specific information as most commonly is the case with POMDPs but instead infer information about the movement of the robot. However, the observation set is limited on the accuracy of the output of the IDC algorithm for scan matching that is used to form the observation set. It is proposed to further investigate the possible approaches in forming the observation set independently of another approach but still inferring the actual robot motion instead of environment information.

- **Hierarchical POMDP (RN-HPOMDP).** The proposed RN-HPOMDP has shown that POMDPs can actually be utilized not only for high-level planning but low-level planning with integrated obstacle avoidance behavior. However, the main deficiency of POMDPs when used for planning in large real-world environments is that they have to maintain a belief distribution of extremely large size. The use of the rPOMDP has provided a method for maintaining the POMDP function and the hierarchical structure has provided a manner of representing the environment at various discretizations and divided into small areas. In essence, planning with the RN-HPOMDP, i.e. solving small POMDPs at various discretizations, is analogous to planning with multiple high-level planners and a low-level planner. Furthermore, when updating the robot's belief about the state it actually occupies with the RN-HPOMDP is actually analogous to updating multiple small regions of the belief and then these regions compose the full belief about the robot's state. A future proposed improvement of the RN-HPOMDP is to built dynamic hierarchical structures based on the current belief state of the robot. In this manner, the hierarchical structure can choose the discretization of each area of the environment based on the importance of its features. Hence, it will capture more effectively the important features of the environment, e.g. the areas of the environment where the robot lies, the goal point is located or where there is activity. Furthermore, the belief update is performed currently in the RN-HPOMDP according to predefined divisions of the environment. With the use of a dynamic hierarchical structure the division of the environment can be dependent on the belief distribution and hence perform the belief update more effectively. This approach can be thought as analogous to the recently proposed point-based POMDP methods, that have gained a great interest in the research community. However, the choice of the belief points in the proposed approach is not random and areas of the belief are chosen instead of unique points. Finally, we should note that although the interest in the research community is mainly focused on developing more efficient approximation methods of POMDPs, it is believed that the use of a hierarchical structure is required to be able to use POMDPs in the problem context of this thesis. This is due to the fact that even the most recently proposed approximation methods cannot be solved in real-time for millions of states, like the proposed hierarchical structure is able.

- **Robot Speed Control.** The modification of the robot's speed of execution has been shown to provide an effective manner for obstacle avoidance. It is proposed to investigate methods that can allow the robot to modify its speed as required to avoid obstacles instead of being able to choose between predefined speeds. With the use of the continuous functions proposed for future investigation this can be achieved without increasing the computational complexity or memory requirements of the POMDP.

- **Future Motion Prediction.** Prediction about the movement of humans and/or obstacles has enabled us to perform obstacle avoidance efficiently and effectively. This was mainly achieved by the long-term prediction module that attempts to predict the final destination point of a human's movement. As noted previously in this thesis, standard prediction methods cannot provide good estimates of the future position of humans. Therefore, learning the motion behavior of humans in a specific environment provides us the means of obtaining long-term predictions. Another method that shares the same intuition is the work presented in [9]. This method however learns specific motion patterns to destination points in an environment. It is believed that learning the destination points humans tend to visit often can be utilized more effectively for future motion prediction since it is not restricted to conform to any specific motion pattern. This approach is proposed to be further extended by learning areas of the environment that tend to get congested and associate them with speed behaviors of humans.

- **Multi-Robot Navigation.** The predictive navigation framework proposed in this thesis is proposed to be extended for multi-robot navigation. The coordination can be achieved with the utilization of Markov games that can handle multiple robots represented as agents modelled with the RN-HPOMDP.

A concluding remark is that autonomous robot navigation can benefit a lot by utilization of POMDPs as confirmed by the increased interest in research on planning under uncertainty. Furthermore, the prediction of human motion can assist in obtaining more effective paths to the destination points a robot has to reach. In this thesis we have introduced a new approach towards this goal and have demonstrated its applicability and effectiveness to demanding navigation tasks. At the same time, the proposed approach provides a basis for future work, pointing to topics that call for research studies.

References

[1] D. Aberdeen. A (revised) survey of approximate methods for solving Partially Observable Markov Decision Processes. Technical report, National ICT Australia, Canberra Austalia, 2003.

[2] J.M. Ahuactzin, E.-G. Talbi, P. Bessiere, and E. Mazer. Using genetic algorithms for robot motion planning. In *Geometric Reasoning for Perception and Action*, pages 84-93, 1991.

[3] K.O. Arras, N. Tomatis, and R. Siegwart. Multisensor on-the-fly localization using laser and vision. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, pages 131-143, 2000.

[4] N. Ayache and O. D. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Transanctions on Robotics and Automation*, 5:804-819, 1989.

[5] H. Baltzakis and P. Trahanias. A hybrid framework for mobile robot localization: Formulation using switching state-space models. *Autonomous Robots*, 15(2):169-191, 2003.

[6] Y. Bar-Shalom, editor. *Multitarget-Multisensor Tracking: Advanced Applications*. Artech House, 1990.

[7] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*. Academic Press, 1988.

[8] M.A. Batalin, G.S. Shukhatme, and M. Hattig. Mobile robot navigation using a sensor network. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, 2003.

[9] M. Bennewitz, W. Burgard, and S. Thrun. Using EM to learn motion behaviors of persons with mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, 2002.

[10] M. Bennewitz, W. Burgard, and G. Cielniak. Utilizing learned motion patterns to robustly track persons. In *Proceedings of the Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, 2003.

[11] M. Betke and L. Gurvits. Mobile robot localization using landmarks. *IEEE Transanctions on Robotics and Automation*, 13:251–263, 1997.

[12] J. Borenstein and Y. Koren. The Vector Field Histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278-288, 1991.

[13] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, 1999.

[14] A. Bruce and G. Gordon. Better motion prediction for people-tracking. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, 2004.

[15] J. F. Canny and M. C. Lin. An opportunistic global path planner. *Algorithmica*, 10(2-4):102-120, 1993.

[16] Anthony R. Cassandra, Leslie Pack Kaelbling, and James A. Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[17] J.A. Castellanos, J.M. Martinez, J. Neira, and J.D. Tardos. Experiments in multisensor mobile robot localization and map building. In *Proceedinds of the IFAC Symposium on Intelligent Autonomous Vehicles*, pages 173-178, 1998.

[18] D. Castro, U. Nunes, and A. Ruano. Obstacle avoidance in local navigation. In *Proceedings of the 10th Mediterranean Conference on Control and Automation (MED2002)*, 2002.

[19] C. C. Chang and K.-T. Song. Dynamic motion planning based on real-time obstacle prediction. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, volume 3, pages 2402-2407, 1996.

[20] Charles C. Chang and Kai-Tai Song. Dynamic motion planning based on real-time obstacle prediction. In *Proceeding of the 1996 IEEE International Conference on Robotics and Automation*, pages 2402-2407, 1996.

[21] C.A. Colios and P.E. Trahanias. A framework for visual landmark identification based on projective and point-permutation invariant vectors. *Robotics and Autonomous Systems Journal*, 35(1):37-51, 2001.

[22] C. I. Connolly. Harmonic functions and collision probabilities. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, pages 3015-3019, 1994.

[23] I. J. Cox. Blanche – an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transanctions on Robotics and Automation*, 7:193-204, 1991.

[24] I.J. Cox. A review of statistical data association techniques for motion correspodence. *International Journal of Computer Vision*, 10(1):53-66, 1993.

[25] A. Curran and K. J. Kyriakopoulos. Sensor-based self-localization for wheeled mobile robots. *Journal of Robotic Systems*, 12(3):163-176, 1995.

[26] B.R. Donald. A search algorithm for motion planning with six degrees of freedom. *Artificial Intelligence*, 31(3):295-353, 1987.

[27] M. Drumheller. Mobile robot localization using sonar. *IEEE Transanctions on Pattern Analysis and Machine Intelligence (PAMI)*, 9:325–332, 1987.

[28] T. Duckett and U. Nehmzow. Mobile robot self-localisation using occupancy histograms and a mixture of gaussian location hypotheses. *Robotics and Autonomous Systems Journal*, 34(2-3): 119-130, 2001.

[29] A. Elganar and K. Gupta. Motion prediction of moving objects based on autoregressive model. *IEEE Transactions on Systems, Man and Cybernetics Part A*, 28(6):803-810, 1998.

[30] W. Feiten, R. Bauer, and G. Lawitzky. Robust obstacle avoidance in unknown and cramped environments. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, pages 2412-2417, 1994.

[31] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal on Robotics Research*, 17(7):711-727, 1998.

[32] A. Foka. Time series prediction using evolving polynomial neural networks. Master's thesis, UMIST - University of Manchester Institute of Science and Technology, 1999.

[33] A. Foka and P. Trahanias. Predictive autonomous robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, 2002.

[34] A. Foka and P. Trahanias. Predictive control of robot velocity to avoid obstacles in dynamic environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, 2003.

[35] A. Foka and P. Trahanias. Predictive autonomous robot navigation. *Autonomous Robots*, under preparation for submission.

[36] A. Foka and P. Trahanias. Real-time hierarchical POMDPs for autonomous robot navigation. *Robotics and Autonomous Systems (RAS)*, in review.

[37] A. Foka and P. Trahanias. Real-time hierarchical POMDPs for autonomous robot navigation. In *IJCAI-05 Workshop: Reasoning with Uncertainty in Robotics (RUR-05)*, 2005.

[38] D. Fox, W. Burgard, and S. Thrun. A hybrid collision avoidance method for mobile robots. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, 1998.

[39] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391-427, 1999.

[40] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23-33, March 1997.

[41] O. Frank, J. Nieto, J. Guivant, and S. Scheding. Multiple target tracking using sequential monte carlo methods and statistical data association. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, 2003.

[42] Kikuo Fujimura. Time-minimum routes in time-dependent networks. *IEEE Transactions on Robotics and Automation*, 11(3):343-351, 1995.

[43] P. C. Gaston and T. Lozano-Perez. Tactile recognition and localization using object models: The case of polyhedra on a plane. *IEEE Transanctions on Pattern Analysis and Machine Intelligence (PAMI)*, 6:257–265, 1984.

[44] D. M. Gavrila. The visual analysis of human movement: a survey. *Computer Vision and Image Understanding*, 37(1):82-98, 1999.

[45] B. Gerkey, R. Vaughan, K. Stoy, A. Howard, G. S. Sukhatme, and M. J. Mataric. Most valuable player: A robot device server for distributed control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, pages 1226-1231, 2001.

[46] W. E. L. Grimson and T. Lozano-Perez. Model-based recognition and localization from sparse range or tactile data. *International Journal of Robotics Research*, 3:3-35, 1984.

[47] J. Guldner and V. I. Utkin. Tracking the gradient of artificial potential fields: sliding mode control for mobile robots. *International Journal of Control*, 63(3):417-432, 1996.

[48] J.-S. Gutmann and D. Fox. An experimental comparison of localization methods continued. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, 2002.

[49] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, pages 736-743, 1998.

[50] J.S. Gutmann. Markov-Kalman localization for mobile robots. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2002.

[51] M. Hauskrecht. Value function approximations for Partially Observable Markov Decision Processes. *Journal of Artificial Intelligence Research*, 13:33-95, 2000.

[52] Milos Hauskrecht. *Planning and Control in Stochastic Domains with Imperfect Information*. PhD thesis, MIT, 1997.

[53] J. Hertzberg and F. Kirchner. Landmark-based autonomous navigation in sewerage pipes. In *Proceedings of the European Workshop on Advanced Mobile Robots*, pages 68-73, 1996.

[54] C.S. Hong, S.M. Chun, J.S. Lee, and K.S. Hong. A vision-guided object tracking and prediction algorithm for soccer robots. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pages 346-351, 1997.

[55] J. Horn and G. Schmidt. Continuous localization of a mobile robot based on 3d-laser-range-data, predicted sensor images, and dead-reckoning. *Robotics and Autonomous Systems*, 14:99-118, 1995.

[56] C. Hue, J.-P. Cadre, and P. Perez. Sequential monte carlo methods for multiple target tracking and data fusion. *IEEE Transactions on Signal Processing*, 50(2):309-325, 2002.

[57] R. A. Jarvis. On distance transform based collision free path planning for robot navigation in known, unknown and time-varying environments. *Advanced Mobile Robots*, pages 3-31, 1994.

[58] B. Jung and G. S. Sukhatme. Tracking targets using multiple robots: The effect of environment occlusion. *Autonomous Robots*, 13(3):191-205, 2002.

[59] Leslie Pack Kaebling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99-134, 1998.

[60] I. Kamon and E. Rivlin. Sensor based motion planning with global proofs. *IEEE Transactions on Robotics and Automation*, 13(6):814-822, 1997.

[61] N. Kehtarnavaz and S. Li. A collision-free navigation scheme in the presence of moving obstacles. In *CVPR'88 (IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Ann Arbor, MI, June 5-9, 1988)*, pages 808-813, Washington, DC., June 1988. Computer Society Press.

[62] Z. Khan, T. Balch, and F. Dellaert. Efficient particle filter-based tracking of multiple interacting targets using an mrf-based motion model. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, 2003.

[63] O. Khatib. Real-time obstacle avoidance for robot manipulator and mobile robots. *International Journal of Robotics Research*, 5(1):90-98, 1986.

[64] Sven Koenig and Reid G. Simmons. Unsupervised learning of probabilistic models for robot navigation. In *Proceedings of the International Conference on Robotics and Automation*, pages 2301-2308, 1996.

[65] K. M. Krishna and P. K. Kalra. Detecting tracking and avoidance of multiple dynamic objects. *Journal of Intelligent Robotic Systems*, 33:371-408, 2002.

[66] E. Kruse, R. Gutsche, and F. M. Wahl. Acquisition of statistical motion patterns in dynamic environments and their application to mobile robot motion planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, volume 2, pages 712-717, 1997.

[67] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[68] A. Lazanas and J.C. Latombe. Landmark-based robot navigation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, page 816–822, 1992.

[69] J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transanctions on Robotics and Automation*, 7:376-382, 1991.

[70] M. Lindstrom and J.-O. Eklundh. Detecting and tracking moving objects from a mobile platform using a laser range scanner. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, 2001.

[71] Michael L. Littman. *Algorithms for Sequential Decision Making*. PhD thesis, Department of Computer Science, Brown University, 1996.

[72] Michael L. Littman, Judy Goldsmith, and Martin Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9:1-36, 1998.

[73] K. Low, W. Leow, and M. Jr. A hybrid mobile robot architecture with integrated planning and control. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS-02)* , 2002.

[74] K. H. Low, W. K. Leow, and M. H. Ang Jr. Enhancing the reactive capabilities of integrated planning and control with cooperative extended Kohonen maps. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, 2003.

[75] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18:249-275, 1998.

[76] V.J. Lumelsky. Incorporating range sensing in the robot navigation function. *IEEE Transactions on Systems, Man and Cybernetics*, 20(5):1058-1069, 1990.

[77] R. C. Luo and T. M. Chen. Target tracking by grey prediction theory and look-ahead fuzzy logic control. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, volume 2, pages 1176-1181, 1999.

[78] R. Madhavan and C. Schlenoff. Moving object prediction for off-road autonomous navigation. In *Proceedings of the SPIE Aerosense Conference*, 2003.

[79] E. Mazer, J. Ahuactzin, G. Talbi, and P. Bessiere. The ariadne's clew algorithm. *Journal of Artificial Intelligence Research (JAIR)*, 9:295-316, 1998.

[80] Jun Miura, Hiroshi Uozumi, and Yoshiaki Shirai. Mobile robot motion planning considering the motion uncertainty of moving obstacles. In *IEEE SMC '99 Conference Proceedings in Systems, Man and Cybernetics*, pages 692-697, 1999.

[81] M. Montemerlo, S. Thrun, and W. Whittaker. Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, 2002.

[82] Yun Seok Nam, Bum Hee Lee, and Moon Sang Kim. View-time based moving obstacle avoidance using stochastic prediction of obstacle motion. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 1081-1086, 1996.

[83] Daniel Nikovski and Illah Nourbakhsh. Learning probabilistic models for decision-theoretic navigation of mobile robots. In *Proc. 17th International Conf. on Machine Learning*, pages 671-678. Morgan Kaufmann, San Francisco, CA, 2000.

[84] I. Nourbakhsh, R. Powers, and S. Birchfield. Dervish an office-navigating robot. *AI Magazine*, 16 (2):53-60, 1995.

[85] Spence Oliver, Mahesh Saptharishi, John Dolan, Ashitey Trebi-Ollennu, and Pradeep Khosla. Multi-robot path planning by predicting structure in a dynamic environment. In *Proceedings of the First IFAC Conference on Mechatronic Systems*, volume II, pages 593-598, September 2000.

[86] J. Gomez Ortega and E. F. Camacho. Mobile robot navigation in a partially structured static environment, using neural predictive control. *Control Engineering Practice*, 4(12):1669-1679, 1996.

[87] L. E. Parker. Cooperative robotics for multi-target observation. *Intelligent Automation and Soft Computing, special issue on Robotics Research at Oak Ridge National Laboratory*, 5(1):5-19, 1999.

[88] V. Perdereau, C. Passi, and M. Drouin. Real-time control of redundant robotic manipulators for mobile obstacle avoidance. *Robotics and Autonomous Systems*, 41:41-59, 2002.

[89] J. Pineau and S. Thrun. An integrated approach to hierarchy and abstraction for POMDPs. Technical Report CMU-RI-TR-02-21, Carnegie Mellon University, 2002.

[90] J. Pineau, N. Roy, and S. Thrun. A hierarchical approach to POMDP planning and execution. *Workshop on Hierarchy and Memory in Reinforcement Learning (ICML)*, 2001.

[91] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2003.

[92] K.-M. Poon. A fast heuristic algorithm for decision-theoretic planning. Master's thesis, The Hong-Kong University of Science and Technology, 2001.

[93] P. Poupart and C. Boutilier. Value-directed compression of POMDPs. In *Neural Information Systems (NIPS)*, 2003.

[94] M. L. Puterman. *Markov Decision Processes - Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[95] D. Read. An algorithm for tracking multiple targets. *IEEE Transactions on Automation and Control*, 24(6):84-90, 1979.

[96] Ioannis M. Rekleitis, Gregory Dudek, and Evangelos E. Milios. On multiagent exploration. In *Proceedings of Vision Interface*, pages 455-461, 1998.

[97] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501-518, 1992.

[98] M. Rosencrantz, G. Gordon, and S. Thrun. Locating moving entities in indoor environments with teams of mobile robots. In *Proceedings of the International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS)*, 2003.

[99] N. Roy. *Finding approximate POMDP solutions through belief compression*. PhD thesis, Robotics Institute, Carnegie Mellon, 2003.

[100] N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation: Robot navigation under uncertainty in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, 1999.

[101] N. Roy, G. Gordon, and S. Thrun. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, 23:1-40, 2004.

[102] A. Saffiotti. The uses of fuzzy logic in autonomous robotics: a catalogue raisonne. *Soft Computing*, 1(4):180-197, 1997.

[103] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, 2001.

[104] D. Schulz, W. Burgard, D. Fox, and A. B. Cremens. People tracking with mobile robots using sample-based joint probabilistic data association filters. *International Journal of Robotics Research (IJRR)*, 22(2):99-116, 2003.

[105] L. Shmoulian and E. Rimon. a^*-dfs: an algorithm for minimizing search effort in sensor-based mobile robot navigation. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, 1998.

[106] R. Simmons. The curvature-velocity method for local obstacle avoidance. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, pages 1080-1087, 1996.

[107] R. Simmons, J. Fernandez, R. Goodwin, S. Koenig, and J. O'Sullivan. Xavier: An autonomous mobile robot on the web. *Robotics and Automation Magazine*, 1999.

[108] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1080-1087, 1995.

[109] K. T. Simsarian, T. J. Olson, and N. Nandhakumar. View-invariant regions and mobile robot self-localization. *IEEE Transanctions on Robotics and Automation*, 12:810-816, 1996.

[110] M.T.J. Spaan and N. Vlassis. A point-based POMDP algorithm for robot planning. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, 2004.

[111] C. Stachniss and W. Burgard. An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, 2002.

[112] A. Stentz. The focused d^* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.

[113] K. Sugihara. Some location problems for robot navigation using a single camera. *Computer Vision, Graphics and Image Processing*, 42:112–129, 1988.

[114] R.S. Sutton. Planning by incremental dynamic programming. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 353-357. Morgan Kaufmann, 1991.

[115] S. Tadokoro, Y. Ishikawa, T. Takebe, and T. Takamori. Stochastic prediction of human motion and control of robots in the service of human. In *Proceedings of the 1993 IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 503-508, 1993.

[116] Satoshi Tadokoro, Masaki Hayashi, and Yasushiro Manabe. On motion planning of mobile robots which coexist and cooperate with human. In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 518-523, 1995.

[117] R. Talluri and J. K. Aggarwal. Mobile robot self-location using modelimage feature correspondence. *IEEE Transanctions on Robotics and Automation*, 12:63-77, 1996.

[118] K. Tanaka. Detecting collision-free paths by observing walking people. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, volume 1, pages 55-60, 2002.

[119] G. Theocharous. *Hierarchical Learning and Planning in Partially Observable Markov Decision Processes*. PhD thesis, Michigan State University, 2002.

[120] S. Thiebaux and P. Lamb. Combining Kalman filtering and Markov localization in network-like environments. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, page 756–766, 2000.

[121] S. Thrun. Bayesian landmark learning for mobile robot localization. *Machine Learning*, 33(1): 41-76, 1998.

[122] S. Thrun, A. Buecken, W. Burgard, D. Fox, T. Froehlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schmidt. Map learning and high-speed navigation in rhino. Technical Report IAI-TR-96-3, University of Bonn, Department of Computer Science, 1996.

[123] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Henning, T. Hofmann, M. Krell, and T. Schmidt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R Murphy, editors, *AI-based Mobile Robots: Case Studies of Successful Robot Systems*. MIT Press, 1998.

[124] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive musuem tour-guide robot minerva. *International Journal of Robotics Research*, 19(11):972-999, 2000.

[125] Sebastian Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93-109, 2000.

[126] N. Tomatis, I. Nourbakhsh, K. Arras, and R. Siegwart. A hybrid approach for robust and precise mobile robot navigation with compact environment modeling. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, 2001.

[127] P.E. Trahanias, S. Velissaris, and S.C. Orphanoudakis. Visual recognition of workspace landmarks for topological navigation. *Autonomous Robots*, 7(2):143-158, 1999.

[128] S.G. Tzafestas, M.P. Tzamtzi, and G.G. Rigatos. Robust motion planning and control of mobile robots for collision avoidance in terrains with moving objects. *Mathematics and Computers in Simulation*, 59:279-292, 2002.

[129] I. Ulrich and J. Borenstein. VFH*: Local obstacle avoidance with look-ahead verification. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, pages 2505-2511, April 2000.

[130] I. Ulrich and J. Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, pages 1572-1577, May 1998.

[131] J. Vanualailai, S. Nakagiri, and J.-H. Ha. Collision avoidance in a two-point system via liapunov's second method. *Mathematics and Computers in Simulation*, 39:125-141, 1995.

[132] D. Vasquez and T. Fraichard. Motion prediction for moving objects: a statistical approach. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, 2004.

[133] J. Vermaak, S. J. Godsill, and P. Perez. Monte carlo filtering for multi-target tracking and data association. *IEEE Transactions on Aerospace and Electronic Systems*,41 (1):309-332, 2005.

[134] L. Wang and W. Tsai. Collision avoidance by a modified least-mean-square-error classification scheme for indoor autonomous land vehicle navigation. *Journal of Robotics Systems*, 8:677-698, 1991.

[135] B. B. Werger and M. J. Mataric. Broadcast of local eligibility for multi-target observation. In *Proceedings of Distributed Autonomous Robotic Systems*, pages 347-356, 2000.

[136] H. Yu and T. Su. Destination driven motion planning via obstacle motion prediction and multi-state path repair. *Journal of Intelligent and Robotics Systems*, 36:149-173, 2003.

[137] J.S. Yu and P.C. Mtiller. An on-line cartesian space obstacle avoidance scheme for robot arms. *Mathematics and Computers in Simulation*, 41:627-637, 1996.

[138] N. H. C. Yung and C. Ye. Avoidance of moving obstacles through behavior fusion and motion prediction. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 3424-3429, 1998.

[139] M. Zefran, J. Desai, and V. Kumar. Continuous motion plans for robotic systems with changing dynamics behavior. In *2nd International Workshop on Algorithmic Foundations of Robotics*, 1996.

[140] J. S. Zelek. Dynamic path planning. In *IEEE Conference on Systems, Man and Cybernetics*, 1995.

[141] Z. Zhang and O. Faugeras. A 3d world model builder with a mobile robot. *International Journal of Robotics Research*, 11(4):269-285, 1992.

[142] Q. Zhu. Hidden Markov Model for dynamic obstacle avoidance of mobile robot navigation. *IEEE Transactions on Robotics and Automation*, 7(3):390-397, June 1991.