Robot Motion Planning for Tracking and Capturing Adversarial, Cooperative and Independent Targets

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Nikhil Karnad

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Adviser:
Professor I. Volkan Isler

September, 2015

Acknowledgements

"You can't cross the sea merely by standing and staring at the water."

– Rabindranath Tagore

My development through graduate school has been as much one of personal growth as a professional one. I owe the engineer I am today to the analytical thinking he instilled in me, and the emphasis he lays on balancing theoretical results with practical relevance. I'm especially grateful for his guidance when I was lost in the sea that is doctoral studies. Professor Volkan Isler's persistence in expecting only the best from me has driven me to become a better researcher, and a better person. Thank you for being an excellent role model, and unassumingly so. It has truly been an honor to be one of your seminal Ph.D. students.

Thanks are due to the University of Minnesota for its facilities, access to books and research journals, and classes on the fine arts that helped me gain a fuller education. I would like to thank the staff at the Department of Computer Science who have been instrumental in helping me through my teaching assistantship, and in enabling my thesis. Access to state-of-the-art robotic technology is not cheap. I am grateful to the National Science Foundation for continuing to fund grants that support our laboratory equipment, and our research.

As a member of the Robotic Sensor Networks group, I have had the privilege of working with a smart group of researchers with diverse backgrounds. Specifically, I'd like to thank Pratap Tokekar, discussions with whom have always helped me get down to the fundamental questions quickly, and to gain new insights. I'm thankful to Patrick Plonski, who, as the then-youngest member of the lab, brought a much-needed fresh perspective to problems I had been toiling away at. I'm thankful to Joshua Vander

Hook, who shared my estimation-theoretic view in the later part of my thesis, and for always willing to grab a burger. Narges Noori has been a sounding board for those especially difficult theoretic problems, which I appreciate a lot. I'd be amiss if I didn't thank the postdoctoral researchers and undergraduate students who helped me reach my goals.

No education is complete without exposure to the real industry. For this, I'd like to thank Dr. Bob Bauer and Dr. Caroline Pantofaru at WillowGarage, who gave me the opportunity to work on cutting-edge remote presence systems during my summer internship. I gained not only invaluable practical experience with real robots, but also a new view on human factors as it relates to robot navigation.

Dr. Onur Tekdas, has been an excellent role model as a graduate researcher, and an even better friend. His work ethic and diligence inspired me to perform better. In many ways, our simultaneous journey through graduate school bonded us as brothers. Du(u)de, thanks for making my time in Minnesota a memorable one.

Dedication

To *Mumz* and *Pappa*, my parents, whose sacrifices made it possible for me to follow my dreams. And to *Bomajju*, my grandfather, whose excellence in math was a hard act to follow.

Abstract

The last ten years have seen robots become smaller, lighter and more maneuverable than ever before, paving the way for their use in the service sector, and even in our homes. A key aspect that elevates a network of robots over a bed of sensors is the ability of individual units to autonomously navigate, forming a sensor-actuator network. These added degrees of freedom demand that we revisit existing motion planning and navigation algorithms and extend them to be able to better contribute to the services they are designed to provide.

In deploying autonomous robots away from laboratory-controlled settings and in to real-world environments, care must be taken because they share their space with other autonomous agents. For instance, a typical home robot has to account for people as well as their pets. Imparting a fair degree of autonomy to robot navigation tasks thus requires reasoning about the mobility of other entities in the path-planning process. This is not a trivial task because such agents can not often be easily characterized or quantified, e.g. human navigation depends on many factors including the perception of hazards, experience from long-term memory, even whim.

In this thesis, we take a deep look at the role of the information available to path-planning algorithms in the context of the mobile target-tracking problem. We start with the problem formulation (Chapter 2), in which robots need to keep track of a single mobile target for as long a duration of time as possible. In the game-theoretic sense, the information available to the target has a different characteristic than the information available to the robots. For example, target behavior varies from adversarial ones in defense applications to cooperative ones in service applications. In the chapters that follow, we present the main contributions of this thesis with this role of information during the path-planning process being the common thread.

Traditional surveillance applications assume the worst-case – the target is an adversary actively trying to escape from the robots. We model this strategic interaction with the theory of pursuit-evasion games, wherein a robot pursues a target, which in turn tries to evade it. We study a mathematical variant of the Lion and Man game in the presence of an obstacle, and show that the final outcome of whether the lion can

track down the man depends, in closed form, on the initial conditions (Chapter 3). We derive optimal player strategies that work regardless of what the other player does, thus providing the worst-case guarantees that such applications demand.

At the opposite end of the spectrum, there exist service applications where the target's trajectory is known to the robots before they need to move. Our motivating example is mobile telepresence, or virtual presence – an emerging technology that enables people to participate in environments they are not physically present in. Specifically, we present a navigation algorithm for a first-of-its-kind person-following robot with telepresence and monitoring applications (Chapter 4). We require that in the presence of another person, the robot should ensure a safe distance by following that person's path without losing track. We present a heuristic planning approach that accounts for both non-holonomic constraints and trajectory smoothness. Further, a control-theoretic implementation is provided.

Targets that navigate autonomously usually do so with their own intentions. While it is uncertain *how* they navigate, their behavior may neither be adversarial, nor completely known to the robots in advance. We present a novel data-driven probabilistic mobility model that can be used by path planners to reason about the uncertainties in decisions made by an individual who is navigating in an indoor environment (Chapter 6). We show that it is possible to preserve long temporal histories over an abstracted representation of the environment, which helps predict future mobility of the target better than previous approaches. We present a multi-robot planning algorithm that builds off of this model. Although our algorithm is designed for long-term planning and offline solution, we are able to execute the robot paths in real-time, and demonstrate extended utility with simulations. We discuss the architecture of a complete system implementation for the telepresence application, including both hardware design and software development (Chapter 8). With an increasing aging population in the U.S., it is our belief that such a system would become relevant in the near future, e.g., to assisted living facilities for purposes of healthcare monitoring.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Modern robots have come a long way from their predecessors in the fifties and sixties, which were primarily designed for use in industrial assembly lines. These envionments were well-structured, resulting in sophisticated control algorithms fine-tuned for pre-programmed tasks, such as picking and placing heavy loads.

With advances in manufacturing technology, the sixties witnessed the dawn of more general-purpose robots capable of both *perception* and *mobility*. A famous example is SRI's Shakey robot (1960-1972) that was able to plan routes and move simple objects around [2]. However, fully autonomous navigation would only become reality four decades later. What started as an unreliable, slow task for the Stanford Cart in 1980 became robust and real-time for its successor, Stanley the autonomous vehicle. In 2005, Stanley would go on to win the DARPA challenge – a remarkable achievement.

Its success would not have been possible without leveraging the simultaneous capabilities of perception, mobility and *planning algorithms*. Today, we have service robots that can vacuum-clean your house while avoiding furniture [3], give you a guided tour of a crowded museum [4], snap well-composed pictures at social events [5], and help the elderly with assistive living [6].

In this thesis, we address the problem of factoring people's navigation decisions directly into the planning process when deciding paths for mobile agents. In contrast to traditional planners that either treat the person as a dynamic obstacle or an adversary, we model him as a passive decision-maker. His preferences are known to the robots, but his actual decision remains uncertain.

Since the underlying cognitive process that governs human wayfinding and navigation is inherently complex, we adopt a data-driven approach by constructing a motion model from trajectory preferences. The probabilistic nature of our motion model serves two main purposes: (i) it helps predict future locations of the person using observation histories of variable lengths, and, (ii) it encodes the uncertainties in the navigation decisions of the person in the form of a probabilistic state machine which can be incorporated into the state space of a multi-robot planner. We utilize this probabilistic motion model in the Partially-Observable Markov Decision Process (POMDP) framework to plan robot paths that anticipate the person's navigational uncertainties *before* he makes his decision.

Our predictive planning approach is not limited to indoor robots. By planning ahead, it facilitates novel services such as acquiring front-views of targets for healthcare and telepresence, and pre-emptive traffic avoidance for rescue vehicles in road networks.

The following is a brief version of the detailed thesis statement presented in Section 1.3.

Brief thesis statement:

> *This thesis is aimed at the design of autonomous planning and navigation algorithms for robots that share the same environment as people. Since people do not select arbitrary paths when they navigate, we argue that by representing them as decision-makers and accounting for their navigation uncertainties in the planning process, we can provide novel services that are more "aware" of people than traditional approaches.*

1.1 Motivating applications

In this section, we examine two major areas were mobile robots have impacted our lives, fundamentally changing the way we interact. First, we present the concept of *telepresence* and discuss the economical and environmental advantages that this technology offers over its counterparts. The second application is healthcare monitoring for the elderly, especially in scenarios where they could pose a health hazard to themselves and others. In both of these settings, we reiterate the importance of predictive planning

algorithms that decide paths for mobile agents in order to provide these services.

1.1.1 Example: Video-enabled collaboration

Telepresence, or virtual presence, is a technology that emulates the experience of an in-person meeting as completely as possible by creating a virtual environment over the Internet where people can collaborate.

Figure 1.1: The Cisco CTS 3000, courtesy of Cisco Systems Inc.

Figure 5.1 shows the Cisco 3000 system designed to accommodate 6 participants per room (12 when two rooms are used) with 1080p video delivered through large screens. Other examples of such systems are the Polycom RPX series, Teliris VirtuaLive, and HP Halo Studio. It is projected that the global telepresence market will hit $ 1 billion in 2016 [7].

These systems offer strong economical benefits to businesses that choose to adopt telepresence over mobilizing their workforce via air, land and sea-based travel [7]. Their benefits are not limited to economic factors – they enable people to interact more frequently over geographical boundaries, which has the potential to change organizational structure from current enterprise hierarchies to more a community-based collaborative setting. Furthermore, the shift from a mobile workforce to virtual meetings has a significant environmental impact, in terms of reducing carbon footprint [8]. As a result, businesses are adopting telepresence as part of their infrastructure. Table 1.1 shows the breadth and depth of IBM's vision of changing infrastructure [9].

Infrastructure	Installed (globally)
IP phones	> 200,000
Audio conferencing	> 2 billion minutes per year
Telepresence or Immersive video rooms	15
High-definition video rooms	100
Desktop video users	5,000
Mobile workforce	40% - 50% of over 200,000 people

Table 1.1: Infrastructure transformation at IBM from 2001-2009 [9]

However, commercial virtual presence systems are not without their limitations. The installation of expensive hardware to support the video production and bandwidth requirements puts these systems out of the reach of everyday users. As shown in Figure 1.1, they are deployed in media-enhanced conference rooms, which forces the participants to visit the room each time, and to be constrained to the small coverage area of the cameras.

In an effort to counteract these limitations on mobility, robotic telepresence systems are being developed. The last few years have seen many products hitting the market, some of which are shown in Figure 1.2.

Figure 1.2: Robotic presence systems. Copyright 2010 The New York Times.

The services offered by these mobile robots require that they be embedded in envionments shared with people. It is important to design robot navigation algorithms that specifically account for the way people navigate. In this thesis, we present a novel approach to uncertainty-based path planning that factors human mobility into the process. We demonstrate the utility of our system in the context of a virtual presence application.

1.1.2 Example: Assisted living

With a rapidly growing aging population and a decline in the number of workers willing to work in healthcare, it expected that the demand for care-related services will increase [10]. Technology can help meet this rising demand, both in-home and in assistive living facilities.

	Alone	With spouse	Other
Men	21%	72%	7%
Women	40%	42%	18%

Table 1.2: Profile of older adults (Source: US Administration on Aging www.aoa.gov)

Aging-in-place is the idea that older people should not have to move from their homes due to age-related deterioration in perceptive, cognitive and motor capabilities. According to the US Administration on Aging, only about 5% of adults beyond the age of 65 live in institutions or nursing homes – a vast majority of them prefer to either live by themselves or with their spouses (see Table 1.2). However, living at home comes with its own risks. Wireless monitoring and other surveillance technologies can help address this concern, but mobile technologies can offer the same services at lesser cost and with more general-purpose applicability. For instance, Alzheimers and dementia patients could pose a threat to themselves as well as others, but their ambulatory patterns are different [11]. Similar to existing surveillance technologies such as [12, 13], healthcare and assitive robots can be designed to recognize these scenarios and respond only when necessary. This can help elderly people feel independent, as well as ease the burden on caregivers.

In addition to healthcare monitoring, such robots also serve a more general-purpose

use. For example, family and friends can log into a robot and remotely accompany a person, which provides them with a richer experience than looking at surveillance footage.

By designing navigation algorithms for home robots to include a higher degree of autonomy, it makes routine tasks easier for the embodied person because he can focus on communicating with a loved one instead of carefully driving the robot while also trying to maintain a conversation.

1.2 Challenges

The paradigm shift from structure and engineered environments to indoor environments such as homes, hospitals and offices, presents significant challenges for robot navigation algorithms. Three major ones are listed below.

Lack of global positioning

The physical structures of indoor envionments interfere with wireless signals, e.g. multi-path effect. Technologies that are traditionally used to triangulate locations in outdoor areas, such as satellite-based global positioning systems (GPS) do not work reliably indoors. Robots navigating in these GPS-denied envionments have to build their own maps, and choose map representations that not only allow for sensor-based localization (usually at centimeter-scale), but also lend themselves to global path planning (usually at corridor scale). There does not exist a single map representation that simultaneously addresses these problems. In this thesis, we use a combination of occupancy grid for map-building and localization, and topological graphs for long-term combinatorial planning.

Uncertainties in navigation decisions

It is difficult to navigate in the presence of people whose decisions are, in general, unknown. This is true not only for pedestrians, i.e. walking behavior, but also on road networks where people make decisions to steer vehicles. Cognitive processes for human navigation are complex and depend on a lot of factors that cannot be easily characterized, e.g. perception of hazards and experience from long-term memory [14].

This relates to the evolutionary viewpoint that humans are intellectually superior to other mammals due to the extensive use of their memory, e.g. learning from mistakes. Although various methods have been suggested to model human wayfinding, e.g. Origin-Destination pairs (O-D), path selection between an O-D pair, and multiple waypoints (many origins/destinations), it is not clear that there exists a single generative mobility model capable of describing the way people navigate. In this thesis, we argue that people do not make arbitrary choices as is the assumption in worst-case analyses. Instead, we take a data-driven approach – paths can be observed, encoded and predicted so long as their inherent long-term dependencies are preserved. Care must be taken not to assume which path the person takes. Rather the uncertainties in his future paths should be folded into the planning process.

Socially-acceptable behavior and interactions

When robots coexist in the same spaces as people, it becomes important for them to navigate in a socially-acceptable manner. What is socially-acceptable is often difficult to characterize with a small set of rules, because it is determined by cultural factors as well as personal factors. For instance, people with experience handling pets are more comfortable letting robots into their personal spaces [15]. In that same paper, the authors address various hypotheses regarding proxemics that are important to consider when designing robots that interact with people. Anthropologist Edward T. Hall describes these spatial factors as the "hidden dimension" that is often taken for granted [16]. It should also be noted that with every new technology comes apprehension. Graceful degradation of navigation strategies in case of unforeseen circumstances can help establish safety and increase the trust that people place in autonomous robots.

In this thesis, we address the first two challenges – those of environment and mobility modelling and designing planning algorithms that explicitly model target mobility. Addressing the third challenge requires the careful consideration of social, cultural and ethical factors and their impact on people – all of which are beyond the scope of this thesis. The interested reader is referred to a recent thesis addressing the formulation of social constraints as navigation objectives [17].

1.3 Research statement

In addressing the applications presented in Section 1.1 and the challenges from Section 1.2, this thesis formulates and solves the target-tracking problem, in which one or more robots, e.g. virtual presence robots, are required to keep track of a mobile target, e.g. a walking person. The following is our research statement, followed by the hypotheses that we seek to address in this thesis. Chapter 2 presents a detailed problem statement along with our contributions.

> *Planning and navigation algorithms for autonomous robots designed to function in the presence of one or more mobile targets need to represent, predict, and reason about the uncertainties in the target's mobility in order to improve tracking performance.*

(A) Motion Model Hypothesis

> The long-term mobility of targets at the scale of path selection can be characterized efficiently and their future locations predicted with good accuracy.

(B) Planning Hypotheses

> The incorporation of our target motion model into uncertainty-based robot planning algorithms yields behavioral robot strategies such as anticipation and guarding by re-use.

(C) System Design Hypothesis

> Robotic applications can be moved out of factories and media-based conference rooms and into peoples' homes by constructing affordable, robust mobile platforms from off-the-shelf components, and complementing them with advanced navigation algorithms that can model human mobility (hypothesis A), planning robot paths (hypothesis B), and executing them in real-time.

Chapter 2

The mobile target tracking problem

As understood in this thesis, the target-tracking problem concerns one or more mobile robots and an independent mobile target that needs to be kept track of by autonomously navigating the robots.

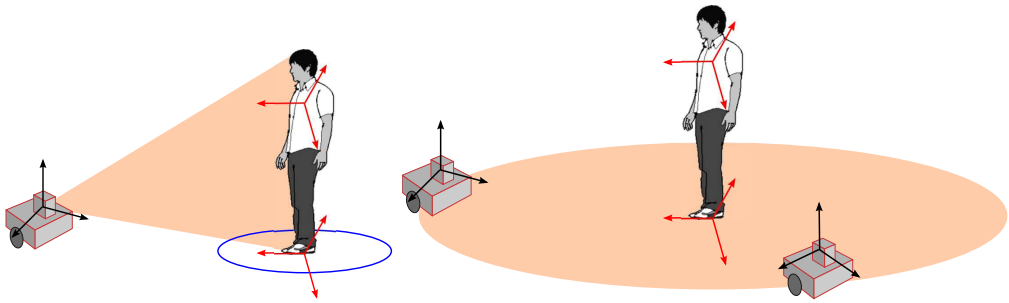


Figure 2.1: Variations of tracking: as an estimation problem (left) and a planning problem given an estimate (right).

We are interested in how controlled robot mobility can be used to obtain long-term tracking performance given an estimate of the target, as opposed to the problem of acquiring such an estimate (see Figure 2.1). There typically exists a constraint that keeps one or more robots "close" to the target according to a predefined property or measure, such as Euclidean distance in the plane, line-of-sight visibility in polygons, or path lengths on a graph.

2.1 Planning objective

In computer vision literature, tracking traditionally refers to the *perception* problem of object-tracking. A single-sensor, such as a camera , attempts to keep visual features of an object within its field-of-view. An early example of an approach designed for for eye-in-hand control of manipulators is available in [18]. Numerous other interesting examples can be found under the umbrella of visual-servoing methods [19]. The distance measure used in these methods is either image-based (IBVS) or position-based (PBVS) [20, 21]. Being a perception problem, tracking is defined as determining control inputs for cameras that maintain image features within desirable thresholds. The primary task of these approaches is that of vision: to simultaneously distinguishing the target from its surroundings while achieving high control performance.

In this thesis, we instead define target-tracking with the primary task of *navigation* and *planning*. This is fundamentally different from the ones mentioned above in at least the following two ways. First, since multiple mobile robots are available, the objective is defined in the joint space of robot actions – it could be desirable for a few of the robots to temporarily lose track of the person in order to achieve better tracking in the future. Second, the geometry of the environment that lies outside the field-of-view of the sensors becomes significant in deciding the paths for the robots to take, e.g. robots could use short-cuts to catch up with the target.

2.2 The scale of target mobility

The scale of the target's mobility is important. For instance, human mobility exhibits different patterns depending on the scale considered. At a fine-grained resolution, a person's trajectory has perturbations, i.e. small deviations in paths due to either perceived hazards or by habit. This is commonly modeled as noise in estimation theory, e.g. the design of filters that estimate the mean and variance of the person's pose based on a sensor measurement model. In this thesis, we employ filters in the sensing phase to smooth out these perturbations and reveal the underlying structure of paths that people take when they navigate on a larger scale, e.g. from an office room to either the elevator or the stairs. We are interested in the uncertainties regarding decisions that a person makes when navigating from point A to point B. For instance, there are multiple

paths a person could take, and it is not clear that people try to optimize for either time or distance – both traditional objectives deemed optimal by planning algorithms.

2.3 Our contributions

From Sections 2.1 and 2.2, it is clear that there are different ways to model the target-tracking problem. In this thesis, we discuss three different approaches that differ based on design decisions regarding: (i) the environment – continuous or discrete, (ii) the scale of mobility – fine-grained or coarse, and (iii) the assumed knowledge of target's strategies – adversarial, fully-known and partially-known. We briefly summarize our approaches in this section and present details in the following three chapters. These contributions were not only the result of publications in the conferences Intelligent Robots and Systems (IROS) and International Conference on Robotics and Automation (ICRA), but also ongoing work aimed at the journals International Journal on Computer Vision (IJCV) and International Journal on Robotics Research (IJRR).

The organization of the chapters in this thesis is summarized in Figure 2.2, with chapter numbers indicated within the circles. There are roughly three main threads in this organization. First is the adversarial target model that assumes the worst-case scenario. Second is the fully cooperative target model, including applications to person-following robots, and the novel view synthesis problem. Last is the mobility model and predictive motion planner. Not shown in this figure is the chapter regarding system implementation, because it stands on its own in that all of the engineering decisions made in the implementation impact all chapters of this thesis.

2.3.1 Adversarial model (Chapter 3)

We start with the worst-case assumption that the target is trying to escape from a robot, while the robot in turn tries to prevent that from happening. This leads to non-cooperative game theory, and more specifically pursuit-evasion games. Resulting strategies from these methods are very useful because they (i) yield guarantees as to which player wins, (ii) provide theoretical bounds for capture time, and (iii) provide worst-case strategies that work no matter what the other player does.

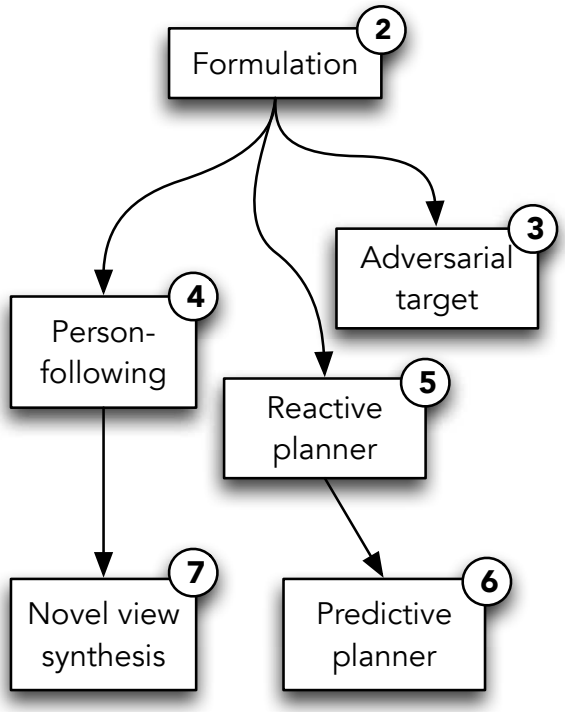


Figure 2.2: Organization of the chapters in this thesis with chapter numbers indicated circles.

We formulate a continuous-space, continuous-time version of a variant of the Lion-and-Man game that models this adversarial scenario in the presence of a single circular obstacle [22]. We present a result in which it is not possible for the robot to always keep track of the person, since the person can loop around the obstacle away from the robot to avoid being seen. However, it is possible for the robot to prevent the person from reaching the obstacle, in which case it can corner the person and thereafter keep him within its sights. We present a characterization of initial locations that completely determine the final outcome of the game.

In the differential game framework, it is not straightforward to incorporate environment geometry such as multiple polygonal obstacles. When a differential model is not imposed on the robot, related research from our group shows that three robots can capture the person in any simply-connected polygonal environment with multiple obstacles [23].

The theoretical bounds from these methods also come at the cost of being difficult to solve due to the strategic interactions between the players: best response actions for the robots are functions of what the target is *capable* of doing and vice-versa. When taken to an extreme, a target that is much quicker than the mobile robots demands a static camera deployment that ends up being expensive. While this is suitable for surveillance tasks and military operations, robots that work alongside people to provide intelligent services do not need to make adversarial assumptions. By characterizing the mobility of the person with a motion model, it is possible to move away from worst-case strategies to results that are not as conservative.

2.3.2 Known target trajectory (Chapters 4 and 7)

On the other extreme from assuming that the target is an adversary is having full knowledge of how the target moves. With this assumption, the robots need only move *after* the target decides its path.

This type of model is common in multi-target tracking [24] and person-following [25]. For instance, an indoor map of a museum could be constructed by a person guiding the robot through the rooms and annotating them with semantic information. In assistive living facilities, it might be in the caregivers' best interest to have robots accompany the elderly either for companionship or healthcare monitoring.

In these applications, the target's trajectory is known to the robots before their paths need to be planned. In the case of discrete space models, e.g. graph representations of the environment, a dynamic programming approach can be used to determine suitable paths for robots [26]. The knowledge of target's trajectory also helps continuous space models because differential constraints such as robot non-holonomy and trajectory smoothness fit nicely into a control theoretic framework. The problem of determining control inputs that guide a differential system along a desired trajectory is known either as trajectory-tracking or regulation. In Chapter 4, we discuss the person-following problem and present an early system prototype in which a person is tracked from a stereo-camera rig mounted on a mobile robotic base. The design and implementation of two existing control theoretic approaches are discussed in the context of that system. We also present a two-state human mobility model that is used to control the responsiveness of the robots to the person [27].

2.3.3 Two-state target mobility model (Chapter 5)

While the known trajectory model works well for reactive robot strategies, it is unsuitable when robot trajectories have to be planned *before* the target moves. The full knowledge of target trajectory ends up being too strong in this case because targets, in general, navigate with independent intention. In Chapter 5, we characterize the target's mobility using a two-state motion model: stationary vs. linear velocity. Our planner determines which mobility state the person is in using an estimator. We derive optimal robot strategies for each target mobility state, and the transitions in between them. We show that a static placement of multiple robots is required to *guard* against the possible directions in which the target can transition from the stationary state to the linear-velocity state. Within the linear-velocity state, we derive optimal multi-robot strategies that maximize the duration of good view acquired by at least one robot. Instead of solving the traditional two-point boundary value problem, we show that a control-input discretization and bounded search method suffices to generate desirable paths.

2.3.4 Uncertainy-based target mobility model (Chapter 6)

By explicitly requiring targets to indicate their decisions to the robots, e.g. people can do so using gestures or cues, such a planner would be interfering with the target's primary intentions. We overcome this undesirable property by modeling the person as a passive decision-making entity whose possible choices, albeit varied and numerous, are known but the actual decision is only observed *after* the robots move.

In Chapter 6, we present a data-driven target motion model that encodes the decision uncertainties inherent in known trajectory preferences. Our model encodes variable-length histories and performs predictions better than existing approaches. Furthermore, the stochastic nature of our motion model characterizes the target's mobility as a probabilistic state machine. We incorporate the states of our motion model into a partially-observable planning framework (POMDP) to yield multi-robot plans that account for target uncertainty. Two desirable outcomes of this process are observed: (i) our planner additionally assumes the worst-case, i.e. no knowledge of person's behavior, when trajectories are observed that were not previously seen, and, (ii) the resulting robot strategies exhibit desirable behaviors such as anticipation, re-use and guarding.

The robot behaviours obtained from our approach could prove to be useful as motion primitives, a future direction aimed at reducing planning complexity.

2.3.5 System implementation (Chapter 8)

We demonstrate the utility of the models and algorithms described in this thesis by presenting a proof-of-concept system implementation for mobile tele-conferencing applications. We discuss a hardware platform design that combines affordable sensors with a robust mobile base. Our platform is designed from off-the-shelf components, making it accessible not just to commercial businesses but more importantly everday users e.g. in their homes. We present the software developed for each component of the system, and propose an overall software architecture that integrates them into a cohesive real-time system. Our code is written primarily in the C++ language, relying on Linux shell scripts for useful utilities. The WillowGarage Robot Operating System (ROS) library was used in three primary ways: (i) to provide a modular implementation transparent to individual component changes, (ii) to prevent re-inventing existing state-of-the art components such as mapping, localization and simulators, and (iii) to contribute our software back to the community for the benefit of future implementations.

Chapter 3

Capturing an adversarial target: Pursuit-evasion games

The problem of tracking a target that behaves adversarially is challenging because the target is trying to *actively* escape from the robot. Typically, the target would change its strategy based on how the robot tries to track it. For instance, in an open room with one target, one robot and a single obstacle, it is very difficult for the robot to reduce its distance from the target. If both have the same speed, the target can always maintain a fixed distance from the robot by looping around the obstacle, in a direction away from the robot's approach. This shows that one robot cannot track down an adversarial target in a room with an obstacle. However, with two robots, one can block the target from looping around the obstacle while the other corners it – i.e. two robots are sufficient to guarantee capture. The subject of pursuit-evasion games addresses these type of strategic problems from a game-theoretic perspective.

In a game of pursuit and evasion, one player (the pursuer) tries to get close to, and possibly capture the other (the evader). The evader, in turn, tries to avoid being captured. Pursuit-evasion games are of fundamental importance to researchers in the field of robotics. Consider the task of surveillance, where a guard pursuer has to chase and capture an intruder evader. Another scenario is search-and-rescue, where a rescue worker has to locate a lost hiker. Since the actions of the hiker are not known a priori, worst-case pursuit and evasion strategies guarantee that the hiker is found no

matter what he does. Problems arising from diverse applications such as collision-avoidance [28], search-and-rescue [29, 30], air-traffic control [31], and surveillance [28] have been modeled as pursuit-evasion games.

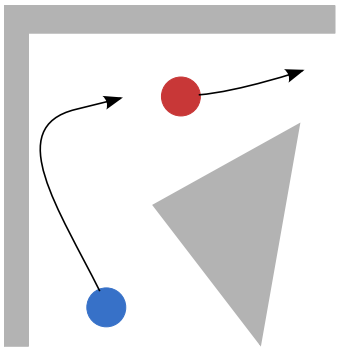


Figure 3.1: An example of a pursuit-evasion game in which a pursuer (blue) is trying to capture an evader (red) in the presence of a polygonal obstacle

3.1 Related work

The objective of a pursuit-evasion game is typically expressed in terms of both players' actions, e.g.

$$\min_{\Gamma_P} \max_{\Gamma_E} d(P, E) \tag{3.1}$$

where Γ_P and Γ_E denote the pursuer's strategy and that of the evader, respectively, and $d(P, E)$ is the distance in between the players. In the target-tracking problem, the target can be thought of as an evader trying to escape the robot, which is pursuing it.

In related work, this objective has also been formulated in terms of line-of-sight visibility [32]. When robots are equipped with omnidirectional sensors and the evader can move arbirarily fast, one can reason about the regions the evader could be in by maintaining information on a simplicial complex [33].

3.2 The Lion and Man game

A classical pursuit-evasion game is the *Lion and Man* game. It was originally posed in 1925 by Rado as follows

A lion and a man in a closed arena have equal maximum speeds. What tactics should the lion employ to be sure of his meal?

The first solution was generally accepted by 1950: the lion moves to the center of the arena and then remains on the radius that passes through the man's position. Since they have the same speed, the lion can remain on the radius and simultaneously move toward the man. Although this strategy works in discrete-time, it was later shown by Besicovitch that exact capture in continuous time takes infinitely long in a bounded arena [34]. However, if the capture distance is set to some $c > 0$, Alonso et al. [35] showed that the lion can capture the man in time $O\left(\frac{r}{s}\log\frac{r}{c}\right)$, where r is the radius of the circular arena and s is the maximum speed of the players. In [36], Sgall studied the discrete time, continuous space variant in a semi-bounded environment: the positive quadrant. He showed that the lion captures the man, if certain invariants are satisfied initially.

Recently, researchers have studied variants of the lion and man game played in environments more complex than a circular disc or the real plane. Isler et al. showed that the lion can capture the man in any simply-connected polygon [37]. Alexander et al. presented a sufficient condition for the greedy strategy to succeed in arbitrary dimensions [38].

The lion and man game in the presence of obstacles remains a challenge. In this paper, we take an important step for solving the lion and man game in an environment with obstacles. We present full characterization of the game in the presence of a single circular obstacle. That is, we present a decision algorithm which determines the winner of the game. We also construct the winner's strategy.

As in the original version of the game, we assume that the players know exact locations of each other at all times and have equal maximum speeds.An important line of research is to study the effect of sensing limitations. Recent progress in this direction includes the study of range-based limitations [39] and bearing-based limitations [40].

Other variants of pursuit-evasion games studied in the robotics community are visibility based pursuit-evasion [41] and maintaining the visibility of an adversarial target [42].

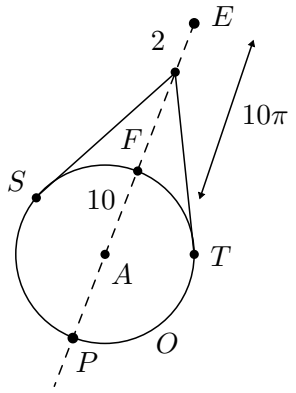


Figure 3.2: A counterexample: The evader wins even though the pursuer is closer to all points on the obstacle.

3.3 Our contributions

We study the lion and man game played in a convex polygonal environment, where both time and space are continuous. There is a single circular obstacle in the environment. The main question we study is: Given initial locations of the players, which player wins the game?

In earlier work [37], researchers have shown that the pursuer can capture the evader in any simply-connected polygon. This intuitively suggests that the evader has to reach the obstacle to win the game. Conversely, the pursuer wins the game if he can separate the obstacle from the evader, and simultaneously make progress toward capture.

Verifying these conditions from an arbitrary initial configuration is difficult. For example, it is easy to see that the evader wins the game if he is closer to the obstacle. However, the condition is not necessary, as shown by the following instance. Consider a circular obstacle O with center A and radius 10 units. (see Fig. 3.2). The initial configuration is such that the pursuer P and the evader E are separated by a relative angle of π radians, measured w.r.t. A. The closest point on the obstacle O to the evader is F, such that $|EF| = 10\pi = |\overset{\frown}{PF}|$. Thus every point on the obstacle is closer to the pursuer than the evader.

If the evader heads straight to F along EF, the pursuer picks one of the directions (clockwise or counterclockwise) to go around the obstacle and reaches F at the same time as the evader, resulting in capture and a pursuer-win. This is true for any point

$F \in \delta O$ if the evader just heads straight to F from E along the shortest path. However, consider the following strategy: the evader heads toward F for 2 units and switches direction toward the tangent point that is farther from the pursuer. For instance, if the pursuer picks the clockwise direction to go around the obstacle, the evader heads to T. The time taken by the pursuer to reach T in the clockwise direction is computed as 44.55 units, whereas the evader reaches T at time 40.126 units. Thus the evader hits the obstacle at T faster than the purser and then continues to loop around the obstacle away from the pursuer. Since the evader avoids capture indefinitely, the evader wins the game.

In this paper, we present a complete characterization to determine the outcome of the pursuit-evasion game for any given initial condition. Our results are organized as follows. In Section 5.3, we formulate the task of the pursuer guarding the obstacle from the evader as a differential game. In Section 3.5.1, we use concepts from optimal control theory to derive optimal control laws. We provide the geometric interpretation of the player strategies in Section 3.5.2. In Section 3.6, we derive necessary and sufficient conditions for each player to win the game. Our main result provides a decision algorithm for the pursuit-evasion problem, presented in Section 3.7. We extend the win-condition to derive an explicit expression for the decision boundary: in Section 3.8, we present a partitioning of the arena into a pursuer-win region and an evader-win region, for a given initial evader location. We conclude in Section 7.5 and suggest directions for future research.

3.4 Problem statement and formulation

An evader E and a pursuer P are playing a game of pursuit and evasion inside a simply-connected convex polygon $\mathcal{P}$, with a single circular obstacle O. We say that the pursuer captures the evader if the geodesic distance between the players goes to zero as time goes to infinity. On the other hand, if the evader can guarantee a non-zero lower-bound on the distance between the players, the outcome of the game is an evader-win.

The game is played in continuous time and continuous space.

Let R be the radius of O (see Fig. 3.3). At any time $t \in [0, \infty[$, the state of the

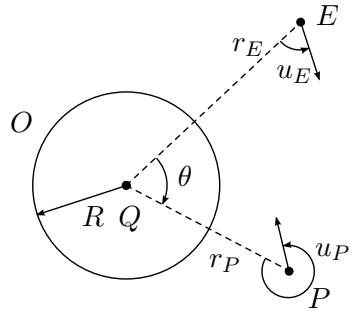


Figure 3.3: The pursuer P guarding the obstacle O from the evader E.

game is defined by three variables:

$$\mathbf{x}(t) = [r_P(t), r_E(t), \theta(t)]^T$$

where $\theta(t)$ is angle between the players E and P, subtended at the center of O. The radial distance of P from the center of O is denoted as $r_P(t)$ and that of E is $r_E(t)$. We drop the dependency on time from the notation and just use r_P, r_E and θ where appropriate.

Both players are modeled as point objects with the same maximum speed, $v = 1$. This is done by scaling all distances w.r.t. the value of v (normalization). The pursuer P (respectively the evader E) can pick a direction relative to his/her radius r_P (respectively r_E) to move along. This is the control input $u_P(t)$ (resp. $u_E(t)$), where $u_P(t), u_E(t) \in S^1 \, \forall t$. We study a game of complete information: both players know each others' locations at all times.

The kinematic equations (KE) for the state are

$$\begin{aligned} \dot{r}_P &= \cos u_P \\ \dot{r}_E &= \cos u_E \\ \dot{\theta} &= \frac{\sin u_E}{r_E} - \frac{\sin u_P}{r_P} \end{aligned} \tag{3.2}$$

The players occupy the part of the polygon outside of the obstacle O. Thus $r_P(t) \geq R, \quad \forall t$ and $r_E(t) \geq R, \quad \forall t$. We restrict the relative angle between the players as $\theta \in [-\pi, \pi]$

In order to win, the evader must guarantee a lower-bound on the distance between the players. This happens when the evader reaches the boundary of the obstacle O

without getting captured. In fact, this is the only way the evader can win the game as we will see in Section 3.6.2. On the other hand, if the pursuer can prevent the evader from reaching O and simultaneously make the distance between them go to zero, the pursuer will win the game. The pursuer can do so, if he is able to make the relative angle θ go to zero before the evader hits O (This statement is formalized and proven as Theorem 1 of Section 3.6.2.). We use these observations to formulate an equivalent game with the following objective.

Suppose the evader E hits the boundary of O at time $T \geq 0$, i.e. $r_E(T) = R$. Then, the value of $\theta(T)$ describes the outcome of the game: if $|\theta(T)| \neq 0$, we know that E reached O before P and thus E wins the game. If not, we will show that the pursuer can align himself with the evader before T and proceed to win the game by playing the Lion's strategy (see Theorem 1). The objective, or value, of the game is thus given by

$$J = |\theta(T)|$$

where

$$T = \min\{t : r_E(t) = R\}$$

We wish to solve the optimal control problem: what should be $u_P^*(t)$ and $u_E^*(t)$ so that E maximizes J and P minimizes it? It is worth noting that we study the game of kind, and seek strategies that are optimal in terms of the outcome of the game.

This problem falls in the context of differential games. Although the solution process is along the lines of the Lady in the Lake problem (see [31], Sec. 8.5, pp. 452–456), our problem is significantly different: if both the lady and the man had equal velocities, the lady would always win the game by swimming along the line joining them, in the direction away from the man. In contrast, the outcome of our game, depends on the initial conditions.

3.5 Optimal player strategies

In this section, we use optimal control theory, in the realm of differential games, to derive the optimal strategies for the pursuer and evader. Further, we present the geometric interpretation of the strategies.

3.5.1 Optimal control solution

Let $\mathbf{x}(t)$ be the state vector, and $\mathbf{u}(t)$ the control input. Optimizing an objective function of the form

$$J(u) = h(\mathbf{x}(T), T) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)dt$$

with a terminal payoff $h(\cdot)$ and an integral payoff $g(\cdot)$ subject to the KE (4.12) is equivalent (Pontryagin's Maximum Principle) to optimizing the Hamiltonian H given by

$$H(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}(t), t) = g(\mathbf{x}(t), \mathbf{u}(t), t) + \mathbf{p}^T(t)[\mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t)]$$

where $\mathbf{p}(t)$ is a vector of Lagrange multipliers, also known as the costate variables. $\mathbf{a}(t)$ is the vector of state transition equations from the KE (4.12).

In our problem, we only have a terminal payoff $h(\mathbf{x}(T), T) = |\theta(T)|$ and no integral payoff. Thus the Hamiltonian for our system is

$$\begin{aligned} H &= p_P(t)\dot{r}_P(t) + p_E(t)\dot{r}_E(t) + p_\theta(t)\dot{\theta}(t) \\ &= p_P(t)\cos u_P(t) + p_E(t)\cos u_E(t) \\ &+ p_\theta(t)\left(\frac{\sin u_E(t)}{r_E(t)} - \frac{\sin u_P(t)}{r_P(t)}\right) \end{aligned}$$

The Isaacs equation is

$$\min_{u_P}\max_{u_E} H = 0 \tag{3.3}$$

We use a standard result from optimal control theory to obtain necessary conditions for u_P^* and u_E^* to optimize the Hamiltonian (see [43], 5.1-17b, pp. 187-188). For all time t the following is true.

$$\begin{aligned} \dot{\mathbf{x}}^*(t) &= \frac{\partial H}{\partial \mathbf{p}} \\ \dot{\mathbf{p}}^*(t) &= -\frac{\partial H}{\partial \mathbf{x}} \\ \mathbf{0} &= \frac{\partial H}{\partial \mathbf{u}} \end{aligned}$$

Since the final state at time T is free, except for $r_E(T) = R$, we have additional boundary conditions (commonly referred to as transversality conditions) given by

$$\frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}^*(T), T) - \mathbf{p}^*(T) = \mathbf{0}$$

Applying these necessary conditions to our problem, we have

$$\begin{aligned}
\dot{p}_P^*(t) &= -\frac{\delta H}{\delta r_P} = -\frac{p_\theta^* \sin u_P^*}{{r_P^*}^2} \\
\dot{p}_E^*(t) &= -\frac{\delta H}{\delta r_E} = \frac{p_\theta^* \sin u_E^*}{{r_E^*}^2} \\
\dot{p}_\theta^*(t) &= -\frac{\delta H}{\delta \theta} = 0 \\
0 &= \frac{\delta H}{\delta u_P} = -p_P^* \sin u_P^* - \frac{p_\theta^*}{r_P^*} \cos u_P^* \\
0 &= \frac{\delta H}{\delta u_E} = -p_E^* \sin u_E^* + \frac{p_\theta^*}{r_E^*} \cos u_E^*
\end{aligned} \tag{3.4}$$

The boundary (transversality) condition is

$$\begin{aligned}
&\frac{\partial h(T)}{\partial \theta} - p_\theta^*(T) = 0 \\
&\Rightarrow p_\theta^*(T) = \frac{\partial \left(|\theta(T)|\right)}{\partial \theta} = \operatorname{sgn}(\theta(T))
\end{aligned} \tag{3.5}$$

From (3.4) and (3.5) we have $\forall t, \quad p_\theta^*(t) = \operatorname{sgn}(\theta(T))$. Solutions for u_P^* and u_E^* that optimize H are

$$\begin{aligned}
(\sin u_P^*, \cos u_P^*) &\parallel \left(-\frac{p_\theta^*}{r_P^*}, p_P^*\right) \\
(\sin u_E^*, \cos u_E^*) &\parallel \left(\frac{p_\theta^*}{r_E^*}, p_E^*\right)
\end{aligned} \tag{3.6}$$

Let the constants of proportionality for the parallel vectors be c_P and c_E. Substitute in the Isaacs equation (3.3) to get $c_P = -c_E$. Now use the evader's boundary condition in (3.6):

$$c_E \sin u_E^* = \frac{p_\theta^*}{r_E^*}$$

At $t = T$, the evader hits the boundary of the obstacle i.e. $r_E(T) = R$. The evader wins the game then on by remaining on the boundary of O and moving in the direction that takes him away from the pursuer i.e. his velocity vector is tangent to O thereafter. In terms of his control variable u_E, we have $u_E(T) = \operatorname{sgn}(\theta(T)) \cdot \frac{\pi}{2}$. Therefore

$$\begin{aligned}
c_E \operatorname{sgn}(\theta(T)) &= \frac{\operatorname{sgn}(\theta(T))}{R} \\
\Rightarrow c_E &= R^{-1}
\end{aligned}$$

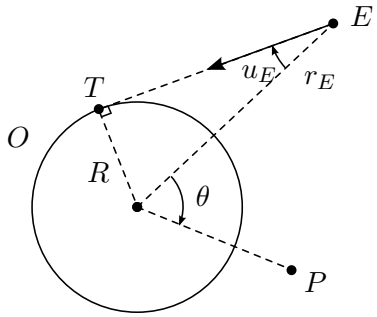


Figure 3.4: E moves away from P along a tangent to O.

Finally, solve for u_P^* and u_E^* by substituting known values into (3.6).

$$\begin{aligned} r_P(t)\sin u_P^*(t) &= -r_E(t)\sin u_E^*(t) \\ &= R\,\mathrm{sgn}(\theta(T)) \end{aligned} \tag{3.7}$$

3.5.2 Geometric interpretation

To understand the physical meaning of (3.7), consider the evader: $r_E(t), u_E(t)$. The argument holds for the pursuer using $r_P(t), u_P(t)$ instead.

$$\sin u_E^*(t) = -\frac{R}{r_E(t)}\,\mathrm{sgn}(\theta(T))$$

Let Q be the center of the obstacle O (see Fig. 3.4). Consider triangle ΔQET, where T is the point on O where the tangent from E touches it. Clearly, Equation (3.7) is satisfied, meaning that the solution for the evader E is to always head toward O along the direction of the tangent from E to O. Since there are two possible tangents for any evader location, he picks the one in accordance with the value of $\mathrm{sgn}(\theta(T))$. Since $\theta \in [-\pi, \pi]$, we know that he will pick the tangent in such a way that his direction of motion makes the value of the relative angle θ *greater*.

The pursuer P, being the minimizing player, moves along the tangent circle in the direction that *reduces* the relative angle between the players.

3.6 Deciding a winner

Given an initial condition for our game, and the optimal strategies derived in Section 3.5, we characterize under which conditions the game ends in a pursuer-win and an evader-win.

First, we use a direct observation to eliminate one case: when the evader is initially closer to the obstacle than the pursuer. If $r_E(0) < r_P(0)$, the evader E reaches the boundary of the obstacle before the pursuer P simply by heading directly to the point on the obstacle closest to E. Thereafter, the evader loops around the obstacle and avoids being captured indefinitely, leading to an evader-win.

Next, consider the case when $r_E(0) \geq r_P(0)$. Let the initial relative angle be $\theta(0) = \theta_0$, say. If $\theta_0 = \pm\pi$, the pursuer can pick either direction to go around O. If not, he picks the smaller of the angles. Once he picks a direction, the pursuer will not switch. This is because by switching directions, the pursuer allows the relative angle θ to increase while the evader is decreasing his distance to the obstacle. The evader would then be in a better configuration i.e. same angular separation as before, but with a lesser distance to the the obstacle.

3.6.1 Time to termination

The time at which the evader hits the obstacle (T_{hit}) is the length of the tangent w.r.t. his initial location, since he moves with unit velocity. By Pythagoras' theorem, we get

$$T_{hit} = \sqrt{r_E(0)^2 - R^2} \tag{3.8}$$

The pursuer wins the game if he can *align* himself with the evader. The time at which this happens is T_{align} such that $\theta(T_{align}) = 0$. If $T_{hit} < T_{align}$, the evader reaches the obstacle before the pursuer prevents him, and thus the evader wins the game. If not, the pursuer can align himself with the evader, after which the pursuer can execute the Lion's strategy and win the game. The Lion's strategy guarantees capture for the pursuer when he stays on the radius of a growing circle, between a fixed center point and the evader's location (see Fig. 3.5). A description of this pursuit strategy in continuous time can be found in [34].

There are two ways to compute T_{align}. First, we can integrate the relative angle between the players (Equation (4.12)) from $\theta(0)$ to 0 as time goes from 0 to T_{align}.

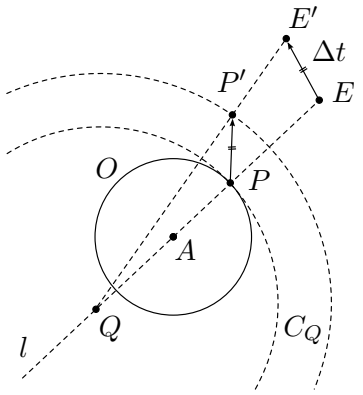


Figure 3.5: An illustration of the Lion's move used by the pursuer P to stay on the evader's radial line w.r.t. Q.

This method is presented in Appendix 3.9. The second method, uses arguments from geometry and worst-case player strategy analysis, comparing the total lengths of the paths that each player takes. We present the second method as follows.

Given the optimal strategies for both players, we know the exact points of tangency of their trajectories w.r.t. the obstacle. This allows us to compute the time taken by the pursuer P to reach the point of tangency of the evader E. Let us call that point as F.

If the pursuer does not align himself with the evader by this time, the evader wins because he hits the obstacle without getting captured. We therefore have an upper bound on the alignment time T_{align}. The time taken by a pursuer traveling at unit velocity to reach the point of tangency F of the evader w.r.t. the obstacle O is the sum of the length of the pursuer's tangent to O, and, the length of the arc from the pursuer's point of tangency to F (see Fig. 3.6).

For the rest of the paper, instead of denoting the time at which $\theta(t)$ becomes 0 as T_{align}, we let T_{align} be the time at which the pursuer reaches the point F. In essence, if P reaches F before E reaches F, we have that the pursuer wins. However, the exact time of alignment lies in the interval $[0, T_{align}]$, provided the evader plays his optimal strategy. In the following section, we show that no matter what the evader does, the game is decided just by comparing the time both players take to reach the point F. Our denotation of T_{align} as the time taken to reach F is therefore justified.

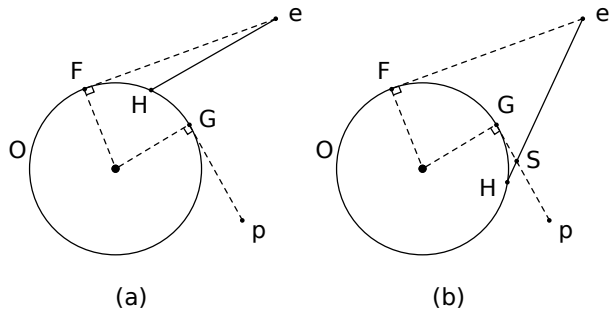


Figure 3.6: Different cases for when the evader hits the obstacle at H: (a) H is on $\overset{\frown}{GF}$, and, (b) EH intersects PG at S.

3.6.2 Pursuer-win condition

In general, the evader may choose not to play the optimal strategy, which affects two aspects of the game: (i) the point on the boundary of the obstacle O at which the evader hits O, and, (ii) the time at which an alignment possibly occurs. In this section, we show that whenever $T_{align} \leq T_{hit}$, the pursuer wins the game irrespective of what the evader does with (i) and (ii).

For the rest of our discussion, we adopt the following notation (see Figure 3.6). Let P be the pursuer's initial location and E that of the evader. Let F be the point of tangency of E w.r.t. O in the direction he heads away from P to increase the relative angle between the players. Let G be the point of tangency, w.r.t. O, of the pursuer along the direction he heads to decrease the relative angle between the players.

As derived earlier, T_{hit} is the length of line segment EF. Further, T_{align} is given by $PG + \overset{\frown}{GF}$.

Lemma 1. *The following two statements S_1 and S_2 are equivalent.*

S_1: $T_{align} \leq T_{hit}$

S_2: *The pursuer can align with the evader in finite time i.e. the relative angle between the players goes to zero in finite time, irrespective of the evader's actual trajectory.*

Proof. $S_2 \Rightarrow S_1$: Let the relative angle between the players go to zero in finite time. The evader cannot be on the boundary of the obstacle unless the pursuer is coincident with him. If not, the evader can always maintain angular separation from the pursuer

by looping around the obstacle along the boundary of O, maintaining a non-zero lower-bound on the relative angle between the players – a contradiction. Thus the pursuer aligns himself with the evader before, or exactly at, the time the evader hits the obstacle. When the evader hits the obstacle at the point F, we have $T_{align} \leq T_{hit}$ by definition.

$S_1 \Rightarrow S_2$: Suppose $T_{align} \leq T_{hit}$. By definition, we have

$$PG + \overset{\frown}{GF} \leq EF \tag{3.9}$$

We have two possible cases: either the evader hits the obstacle at some finite time, or, he does not hit the obstacle ever. In the second case, the pursuer has a higher angular velocity than the evader about the center O and since the evader does not hit the obstacle, the relative angle between them goes to zero (players are aligned) in finite time.

We now focus on the first case: the evader hits the obstacle at some point, call it H. We show that the pursuer is closer to H no matter where H lies on the boundary of O. Thus the pursuer can reach H before the evader no matter where H lies on O. Note that we consider that the evader takes the shortest possible path to reach H, but our argument holds for any evader strategy that takes him to H, because it will take him at least as long as the time along the shortest path to H.

Case 1. H lies on $\overset{\frown}{GF}$ (see Fig. 3.6 (a)). Expand (3.9)

$$\begin{aligned} PG + \overset{\frown}{GF} &\leq EF \\ \Rightarrow PG + \overset{\frown}{GH} + \overset{\frown}{HF} &\leq EF \\ \Rightarrow PG + \overset{\frown}{GH} &\leq EF - \overset{\frown}{HF} \end{aligned}$$

Use triangle inequality in ΔEFH, where FH is a chord of the obstacle O i.e. $FH < \overset{\frown}{FH}$ to get

$$\begin{aligned} PG + \overset{\frown}{GH} &\leq EF - HF \\ \Rightarrow PG + \overset{\frown}{GH} &\leq EH \end{aligned}$$

Thus the pursuer is closer to H than E and thus can align himself with the evader before the evader reaches H.

Case 2. H lies on the boundary of O such that EH intersects PG. Call the point of intersection as S (see Fig. 3.6 (b)). Expand (3.9)

$$\begin{aligned} PG + \overset{\frown}{GF} &\leq EF \\ \Rightarrow PS + SG + \overset{\frown}{GF} &\leq EF \\ \Rightarrow PS &\leq EF - SG - \overset{\frown}{GF} \end{aligned}$$

Use triangle inequality in ΔSGF, where S lies outside the circle O such that FS is a secant of O i.e. $SF < SG + \overset{\frown}{GF}$ to get

$$PS \leq EF - SF$$

Finally, triangle inequality in ΔEFS gives us

$$PS \leq ES$$

Thus the pursuer is closer to S than E and thus can align himself with the evader before the evader reaches S (and thereby H).

Case 3. H lies on the part of the boundary of O beyond F. The shortest path from E to H wraps around F: it is $EF \cdot \overset{\frown}{FH}$. The shortest path from P to H wraps around G and is given by $PG \cdot \overset{\frown}{GF} \cdot \overset{\frown}{FH}$. Since $PG + \overset{\frown}{GF} \leq EF$, adding $\overset{\frown}{FH}$ gives us the required result: P is closer to H than E and thus reaches H before E. Since E hits O at H, we have that the players are aligned.

In all of the cases, we observe that the pursuer is closer to all points on the obstacle than the evader when the condition $T_{align} \leq T_{hit}$ is true. Consequently, he can align himself with the evader in finite time. □

Lemma 1 results in a configuration of the game where the pursuer and evader are radially aligned w.r.t. O such that the pursuer is closer to the center of O than the evader. We show that from this point on, the pursuer wins the game by following the Lion's strategy, adapted to a simply-connected polygon (as explained in [37]). We summarize this result in the following lemma by proving that the initial conditions for the existence of a winning pursuer strategy are satisfied at the time of alignment.

Lemma 2. *When the players are aligned radially w.r.t. O, with the pursuer closer to O than the evader, the pursuer wins the game by following the Lion's strategy.*

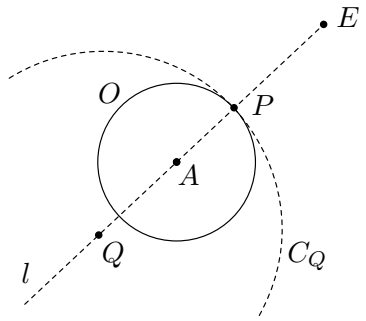


Figure 3.7: Existence of a separating circle when the players are aligned.

Proof. First, we show that there exists a circle that separates the pursuer from the evader, constructed as follows (see Fig. 3.7).

Suppose the pursuer is at P and the evader at E. Let the center of the obstacle O be A. Let l be the line passing through E, P and A. Pick a point Q on l such that a circle C_Q centered at Q passes through P and completely contains O. For example, if Q coincides with P, then O is the same as C_Q. We can pick any other point Q on l farther away from P than A and that will work as well. The other extreme is when Q is at infinity on l, at which point C_Q degenerates to the tangent to O at P. Therefore, such a circle always exists when the players are aligned.

The pursuer follows the Lion's strategy, depicted in Fig. 3.5: he always remains on the radius CE' for an evader move from E to E' (say). This sandwiches the evader between the boundary of the polygonal arena $\mathcal{P}$ and a growing circle C_Q with a fixed center Q. The continuous time Lion's strategy is discussed in [34]. For a detailed analysis of the discrete time version of the Lion's strategy in simply-connected polygons, the interested reader is directed to [37].

The game terminates at the boundary of the polygonal arena where the evader is captured. If the evader plays sub-optimally, the game could terminate before he reaches the boundary as well, but the outcome of the game remains unchanged i.e. the pursuer wins by capturing the evader. □

We combine Lemma 1 and Lemma 2 to state our main result.

Theorem 1. *When $T_{align} \leq T_{hit}$, the pursuer wins the game by first aligning himself with the evader, then executing the Lion's strategy. If not i.e. if $T_{hit} < T_{align}$, the evader*

reaches the obstacle and wins the game thereafter by looping around its boundary and avoiding capture indefinitely.

3.7 Decision algorithm

Let the initial configuration of the game be $G(0) = (r_P(0), r_E(0), \theta(0), \mathcal{P}, O)$, where $r_P(0)$ is the initial radial distance of the pursuer from the center of the obstacle O and $r_E(0)$ that of the evader. $\theta(0)$ is the initial relative angle between the players. $\mathcal{P}$ is a description of the simply-connected polygonal arena that contains the obstacle O and the players in its interior.

The radius of the circular obstacle O is R, a given constant. Let the center of O, denoted by A, be the origin of our coordinate frame. We further set the positive X-axis along the evader's radius, which makes the relative angle $\theta(t)$ the angle subtended by

the pursuer's radius w.r.t. evader's radius for all time t.

Algorithm 1: DecideWinner$(r_P(0), r_E(0), \theta(0), \mathcal{P}, O)$

if $r_E(0) < r_P(0)$ **then** `// E is closer to O than P`
  Output "Evader wins" ;
  return ;
end
$A \leftarrow$ CENTER(O) ;
$E \leftarrow$ CARTESIAN$(r_E(0), 0)$;
$P \leftarrow$ CARTESIAN$(r_P(0), \theta(0))$; `// Polar to Cartesian`
$F \leftarrow$ TANGENTOFEVADER(E, P) ;
$G \leftarrow$ TANGENTOFPURSUER(P, E) ;
`// Evader's hit time to the obstacle`
$T_{hit} \leftarrow |EF| = \sqrt{r_E(0)^2 - R^2}$;
$\theta_{align} = \angle GAF$;
$T_{align} = |PG| + |\overset{\frown}{GF}| = \sqrt{r_P(0)^2 - R^2} + R\theta_{align}$;
if $T_{align} \leq T_{hit}$ **then** `// Theorem 1`
  Output "Pursuer wins" ;
  return ;
else
  Output "Evader wins" ;
  return ;
end

We assume that a feasible description is provided i.e. unexpected conditions, such as the players starting from inside the obstacle, are not checked for. Our decision algorithm is listed as Algorithm 1.

The three subroutines are used in our algorithm are

- CARTESIAN(r, θ)
  Converts from Polar coordinates to Cartesian coordinates.

- TANGENTOFEVADER(E, P)
  Computes the point on intersection of the tangent from the evader's location E to the circular obstacle O, taking into account the value of $-\operatorname{sgn}(\theta)$ to decide which

of the two possible tangents to use.

- TangentOfPursuer(P, E)
  Computes the point on intersection of the tangent from the pursuer's location P to the circular obstacle O, taking into account the value of $\text{sgn}(\theta)$ to decide which of the two possible tangents to use.

3.8 Decision boundary

The winning condition derived in Section 3.6 is a comparison of the length of the evader's tangent to the length of the pursuer's path to the evader's point of tangency. We can use this result to answer a more general question: given the evader's initial location E, a description of the polygon $\mathcal{P}$, and the obstacle O, what is the pursuer-win region? In other words, what is the boundary of the region within which the pursuer starts and wins the game, and outside of which the pursuer is unable to capture the evader?

Given an initial evader location E, the points of tangency from E to O, call them T and S as before, are fixed and can be computed directly (see Fig. 3.8). Since the length of the evader's tangent is known, call it w ($w = T_{hit}$), we can compute the set of all points that are at most a distance of w from T and S. However, we need to account for the direction of traversal of the pursuer. For instance, we retain the part of the region of distance at most w from S that lies on the other side of the line EO when compared to S i.e. if the pursuer were to head toward S from close by, the evader would rather head to T and thus it is necessary to check the distance of the pursuer from T rather than S. The region is symmetric about the line EO (the evader's radius) because the evader can switch tangents when the relative angle between the players hits π.

The equation for the boundary of the region in polar form is (r, θ) where

$$r(\theta) = \left\{R^2 + (w - R(\theta - \alpha))\right\}^{\frac{1}{2}}$$
$$\text{where, } \cos\alpha = \frac{R}{r}$$

Fig. 3.8 was obtained by varying θ in discrete steps and computing the corresponding $r(\theta)$. The resulting region was plotted in Java and the snapshot produced.

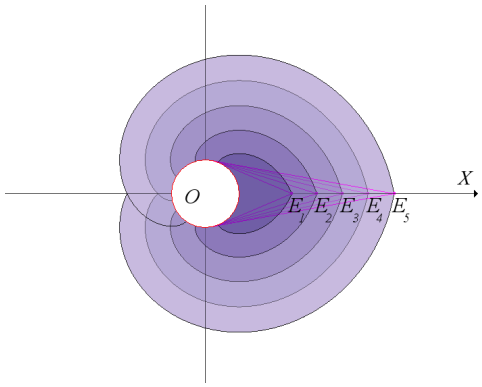


Figure 3.8: For each evader location E_i (moved along positive X away from O), the shaded regions are pursuer-win regions. The evader's tangents to O are also shown.

It can be seen that the polar coordinate equation is of the form

$$a\theta + br^2 + c\cos^{-1}\frac{R}{r} + d = 0$$

where a, b, c, and d are known constants. By substituting the initial radius and angle of the pursuer into the equation, we can check on which side of the boundary the pursuer lies. In that sense, we call this equation the decision boundary. This gives us an alternate method for deciding which player wins the game.

In comparison, our solution approach in Section 3.6 uses lengths of simple curves (tangent and arc of a circle), and an analysis of worst-case strategies to derive a simpler decision formula.

3.9 Alternate method to compute T_{align} by integrating $\theta(t)$

Before presenting the relative angle between the players (θ) as a function of time, we derive the evader's radial distance function $r_E(t)$.

We derive an expression for the evader's trajectory. The pursuer follows the same geometric idea, differing only in the direction he picks to traverse around the obstacle. For simplicity of notation, let $r(t) = r_E(t)\ \forall t$ be the evader's radial distance from the center of O. Assume that $\text{sgn}(\theta(T)) = +1$ w.l.o.g. From (4.12) and (3.7), we have

$$\dot{r} = \cos u_E^* = \sqrt{1 - \sin^2 u_E^*} = \frac{\sqrt{r^2 - R^2}}{r}$$

Rearrange

$$\frac{r\,dr}{\sqrt{r^2 - R^2}} = dt \tag{3.10}$$

Change the variable to w such that

$$\begin{aligned} r &= R\csc w \\ \Rightarrow dr &= -R\csc w \cot w\, dw \end{aligned}$$

Substitute in (3.10)

$$\begin{aligned} -R\csc^2 w\, dw = dt \Rightarrow Rd\left(\cot w\right) = dt \\ \Rightarrow \cot^2 w = \frac{t^2}{R^2} + \frac{2c_P t}{R} + c_P^2 \end{aligned}$$

Switch back to variable r.

$$\begin{aligned} &\frac{r^2}{R^2} - 1 = \frac{t^2}{R^2} + \frac{2c_P t}{R} + c_P^2 \\ &\Rightarrow r^2(t) = t^2 + 2c_P Rt + R^2(1 + c_P^2) \end{aligned} \tag{3.11}$$

Use the initial distance of the evader from the origin, $r(0) = r_E(0)$, to solve for c_P. This gives us the expression for $r(t)$ as a function of the initial evader location and the game time.

$$r(t) = \left(r_E(0)^2 + t^2 - 2t\left(\sqrt{r_E(0)^2 - R^2}\right)\right)^{\frac{1}{2}} \tag{3.12}$$

We know that T_{align} is the time at which the relative angle between the players reaches zero. From (4.12) and (3.7) we express the rate of change of the relative angle between the players as a function of their radial distances from the origin. The closed form solution for the latter was obtained as (3.12).

By definition, the relative angle is the angle the pursuer's radius subtends w.r.t. the evader's radius. When this angle is negative, the evader moves in the counter-clockwise (positive) direction about the obstacle to increase it. Thus the direction chosen by the evader is along $-\operatorname{sgn}(\theta(T))$ and that of the pursuer is along $\operatorname{sgn}(\theta(T))$.

$$\dot{\theta} = -\frac{R\,\mathrm{sgn}(\theta(T))}{r_E^2} - \frac{R\,\mathrm{sgn}(\theta(T))}{r_P^2}$$

Substitute for $r_P(t)$ and $r_E(t)$ from (3.12) and separate the variables.

$$\begin{aligned} -\frac{d\theta}{\mathrm{sgn}(\theta(T))} =& \frac{R\,dt}{\left(t - \sqrt{r_E(0)^2 - R^2}\right)^2 + R^2} \\ &+ \frac{R\,dt}{\left(t - \sqrt{r_P(0)^2 - R^2}\right)^2 + R^2} \end{aligned} \tag{3.13}$$

This equation can be integrated using the standard form

$$\int \frac{dx}{x^2 + a^2} = \frac{1}{a}\tan^{-1}\frac{x}{a}$$

We integrate from 0 to T_{align}, during which time the relative angle θ goes from $\theta(0) = \theta_0$ (say) to $\theta(T_{align}) = 0$. To reduce clutter, let $k_1 = \sqrt{r_P(0)^2 - R^2}$ and $k_2 = \sqrt{r_E(0)^2 - R^2}$.

$$\mathrm{sgn}(\theta(T)) \cdot \theta_0 = \tan^{-1}\left[\frac{t - k_2}{R}\right]_0^{T_{align}} + \tan^{-1}\left[\frac{t - k_1}{R}\right]_0^{T_{align}} \tag{3.14}$$

Substitute the endpoints for t, let $k_3 = \tan(\mathrm{sgn}(\theta(T))\theta_0)$ and use the addition formula for inverse tangent[1] :

$$\tan^{-1} u - \tan^{-1} v = \tan^{-1}\frac{u - v}{1 + uv}$$

to obtain the following quadratic in T_{align}

$$\begin{aligned} &T_{align}^2\left[k_1k_2k_3 - k_3R^2 + k_1R + k_2R\right] \\ +&T_{align}\left[-k_1k_3r_E(0)^2 - k_2k_3r_P(0)^2 - Rr_E(0)^2 - Rr_P(0)^2\right] \\ &+k_3r_P(0)^2r_E(0)^2 = 0 \end{aligned} \tag{3.15}$$

Case 1. Consider the case when $\theta_0 = 0$ or $\theta_0 = \pm\pi$ i.e. $k_3 = 0$. The quadratic simplifies to

$$T_{align}^2\left[k_2R + k_1R\right] + T_{align}\left[-Rr_E(0)^2 - Rr_P(0)^2\right] = 0$$

[1] To use this formula, we further need to verify that $|uv| < 1$

Giving us $T_{align} = 0$, which corresponds to the case when $\theta_0 = 0$ i.e. the players are aligned to begin with. In the other case, divide by $T_{align} \neq 0$

$$T_{align} = \frac{r_P(0)^2 + r_E(0)^2}{k_1 + k_2}$$

Case 2. $\theta_0 = \pm\frac{\pi}{2}$, in which case $k_3 = \pm\infty$. Since $k_3 \neq 0$, divide throughout by k_3 to get a new quadratic in which we let $k_3 \longrightarrow \infty$ i.e. $\frac{1}{k_3} \longrightarrow 0$ to obtain

$$T_{align}^2 \left[k_1 k_2 - R^2\right] + T_{align} \left[-k_1 r_E(0)^2 - k_2 r_P(0)^2\right] + r_P(0)^2 r_E(0)^2 = 0$$

which is a simplified quadratic in T_{align}.

In all other cases, the quadratic given by (3.15) applies. Thus the solution to (3.15) and the special cases listed above together allow us to solve for T_{align} in closed form and compare the value to T_{hit} to decide which player wins the game. The equations are quadratics because the players can pick either direction to travel about the obstacle. The appropriate solution can be picked by ensuring that the pursuer moves along the lesser of the two relative angles between the players.

3.10 Summary and future work

In this chapter, we studied the single-target tracking problem as a non-cooperative game in which the target is trying to escape from the robot, which in turn is trying to get reasonably close to the target. When there is a circular obstacle present in the environment, we are still able to formulate a differential game and solve in closed form for the optimal pursuer and evader strategies. The evader cannot win unless he heads to the obstacle and loops around it. The choice of which direction to take around the obstacle is reflected by a quadratic equation that emerges from our solution.

Two things are important to take away from this chapter. First, it is not simple to incorporate multiple circular obstacles into the differential game framework directly, since the evader now has the ability to choose *either* of the obstacles to hide behind. In fact, the evader could head toward *both* obstacles, in an effort to mask his actual choice from the pursuer until the very end. Although it intuitively seems that the evader has

the advantage in this case, it is not clear why such a strategy is optimal, or better than heading straight to either one of the obstacles instead.

In the following chapter, we look at the other end of the spectrum, where the target's motion is known to the pursuer in advance. When designing a practical system for service applications, the truth is typically in between these two ends of the spectrum. In Chapter 6, we present a model where the target's preferences are known, but its decision remains uncertain.

Chapter 4

Vision-based navigation for person-following

In the previous chapter, we studied the case where the target is trying to break the robot's tracking objective as soon as possible. For instance, the target could use its knowledge of the robot's limited visibility to determine geometrically the *shortest* distance to escape from its sensing range. However, in service applications, this is not typically the case. In fact, since the target is a person trying to obtain a service from the robot, he has an incentive to cooperate with it instead.

In this chapter, we study a variant of the target-tracking problem in which a robot with sensory and kinematic constraints has to follow a person around in an indoor environment with obstacles. We present a motivating application from the industry in an exciting emerging area: telepresence robots for remote meetings and healthcare monitoring. These robots typically have an onboard camera and it is required for them to maintain the person in view as he walks around in an indoor environment. However, it is not possible for robots with non-holonomic constraints to exactly follow the person's path, e.g. if he side-steps. The problem of finding suitable wheel velocities for the robot so that it follows the person's path as closely as possible is formulated as a control-theoretic problem. We discuss two existing approaches to this problem in the context of our application: the landing curve method [44] and the method of dynamic feedback linearization [45]. We evaluate them using both simulations and real experiments – on

a robot platform that we built in our lab, and on two robots from the industry.

4.1 The person-following problem

The idea for person-following emerged from the idea of active vision, which states that the information extracted from visual perception serves an application-dependent purpose.

Systems for tracking people with mobile cameras typically use either face detectors, color histograms, contour-based models or combinations thereof [46, 47, 48]. In [49], researchers explicitly model foreground and background which is quite slow in practice. Additionally, the user is always required to face the camera, which is too restrictive for a telepresence application.

In [50], researchers used image-based servo-control on a robot with one camera to keep the image of the torso of a person large enough. When depth-of-target is required, either laser scanners are combined with monocular cameras [51, 52], or a stereo camera pair is used [47]. In our system, we use a simple stereo rig made from two off-the-shelf web-cameras. The WillowGarage Texas robot uses a combination of laser and vision sensors.

Individual camera platforms in our system require only local information and therefore do not need a map in advance. We also estimate the trajectory of the person in real-time and react to how he moves in an on-line fashion.

Other related systems either use a robot with more degrees of freedom [53], or complex social objectives [25].

4.2 The WillowGarage remote presence system

During Summer 2010, I had the amazing opportunity to intern with WillowGarage Inc., a robotics start-up in Menlo Park, California. During my stay there, I designed and implemented a marker-based person-following module for both the PR2 robot and the Texas (shown in Figure 4.1).

Although roboticists refer to the Texas as a robot, folks at WillowGarage prefer *remote presence system*, because technically it was designed to be purely tele-operated,

Figure 4.1: The WillowGarage Texas remote presence systems

without any autonomy.

Regardless of the name, the Texas system allows a user to log-in remotely from a website, pilot it around, and speak, listen and watch through its sensors. It is part of a new generation of telepresence systems that firms can deploy to help employees collaborate over long distances without the hassle of travel and airfare.

The Texas is not as powerful as the PR2 – it has a single core processor and no tilting laser scanner to avoid tall obstacles. Additionally, it's primary processes are communication based: audio and video through Skype. It is therefore necessary to consider implementations that use as less of a CPU load as possible. Running full-scale indoor localization algorithms on top of existing services would bring this system to a halt.

Interesting challenges were encountered through the usage studies with the Texas:

- When it passes through a wireless hole, the pilot is logged off and the robot is stranded
- It cannot avoid tall obstacles since it does not have a tilting laser scanner
- Often, the pilot of a Texas does not wish to actively navigate. Examples include when in conversation with a walking person, and to follow a person into a

conference room for a meeting.

These challenges motivated the implementation of a person-following software application to be installed on the Texas. This application had two main components: perception and navigation. For perception, we used an explicit marker from the ARToolkit library as a token that a remote person could use to guide the robot from one place to another. This system is fiducial, which gives the position and orientation of the marker from the view of a single camera atop the screen of the Texas. Once the marker pose is stored over time, the trajectory history is fed to a controller that keeps the robot following the same path as the person. We now present two different controllers, in progression of generalization that achieve the trajectory tracking task while taking into account the differential constraint of the wheeled robot base.

4.3 Trajectory tracking controllers

Given a trajectory to follow and a set of kinematic equations for the wheeled base of the robot, a controller has to be designed that can take the robot from its current location to the trajectory and track it. In our system, the following are desirable properties for the controller.

1. The controller should be closed-loop, with feedback, so that errors are minimized in real-time.

2. It should obey the differential drive (unicycle) kinematic model.

3. The resultant robot trajectory should be convergent, i.e. it reaches sufficiently close to the goal in finite time.

4. The robot's trajectory should be differentiable so that transitions between consecutive video frames are smooth.

There are two ways to solve this problem. The first is to construct a solution that has the desired properties by using polynomial curves. We call this the landing curve method. The second is to use a technique from non-linear control systems called dynamic feedback linearization, which can be used to derive in closed form the control laws for the set-point problem and the regulator problem for chained-form systems.

4.3.1 Landing curve method

The Landing curve method was designed and developed with the DARPA urban challenge in mind [44]. The goal was to make a four-wheeled car follow a desired trajectory in GPS coordinate space. This method works by defining three error functions and then formulating control laws that drive all of them to zero, i.e. convergence.

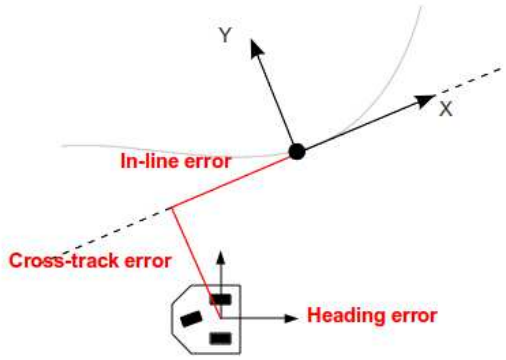


Figure 4.2: Decoupled errors for a trajectory tracking controller: in-line (e_x), cross-track (e_y) and heading (e_θ).

Figure 4.2 shows a sample desired trajectory (gray) and the error functions that determine the controller. The governing equations for error are

$$e_x = (x_t - x_c)\cos\theta_t + (y_t - y_c)\sin\theta_t \tag{4.1}$$

$$e_y = -(x_t - x_c)\sin\theta_t + (y_t - y_c)\cos\theta_t \tag{4.2}$$

$$e_\theta = \theta_t - \theta_c \tag{4.3}$$

where subscript t denotes target (desired) trajectory state and subscript c denotes current (robot) state. e_x is called the in-line tracking error, which denotes how far behind the robot lags the target. e_y is called the cross-track error, which denotes how far off to the side of the target the robot is. It is orthogonal in direction to e_x. e_θ is the heading error of the robot w.r.t. the target.

Once these errors are determined, a polynomial curve can be used to help the robot approach the desired path. However, this polynomial should also satisfy the dynamic

constraints of the vehicle (position, heading and curvature). A cubic polynomial minimally satisfies these constraints. Let (x_p, y_p) be the path generated by the cubic polynomial function $y_p = P(x_p)$.

$$P(x_p)|_{x_p=0} = 0 \tag{4.4}$$

$$\frac{dP(x_p)}{dx_p}|_{x_p=0} = 0 \tag{4.5}$$

$$\frac{d^2P(x_p)}{dx_p^2}|_{x_p=0} = 0 \tag{4.6}$$

$$\Rightarrow P(x_p) = c(x_p)^3 sgn(e_y) \tag{4.7}$$

Once this path is determined, the next step is to derive the heading controller from this path, and then design a bang-bang state velocity controller to drive the in-line error e_x to zero. The resulting equations are as follows. Details about derivations and Lyapunov stability can be found in [44].

$$\theta_p = \theta_t - arctan\left(3c\left(\frac{e_y}{c}\right)^{2/3} sgn(e_y)\right) \tag{4.8}$$

$$v_s = \dot{e}_x + v_b\sqrt{2a_{max}|e_x|} \tag{4.9}$$

$$\frac{dv_b}{dt} = -Av_b + (1 - v_b)\max(0, e_x) - (1 + v_b)\max(0, -e_x) \tag{4.10}$$

$$a_c = \begin{cases} a_{max} & \text{if } v_s/dt > a_{max} \\ -a_{max} & \text{if } v_s/dt < -a_{max} \\ v_s/dt & \text{otherwise} \end{cases} \tag{4.11}$$

Figure 4.3 shows the resulting robot trajectory with the cubic landing curve for two different values of the steepness factor c. Values of $c > 1$ cause the system to overshoot and then converge, whereas values of c closer to 0 have slower convergence rate to the desired trajectory.

4.3.2 Dynamic feedback linearization

If the smoothness condition were removed, we could use in-place rotations and pure-translations to execute the local motion plan. However, the imposition of this constraint makes the controller design non-trivial. The previous Landing curve method works well

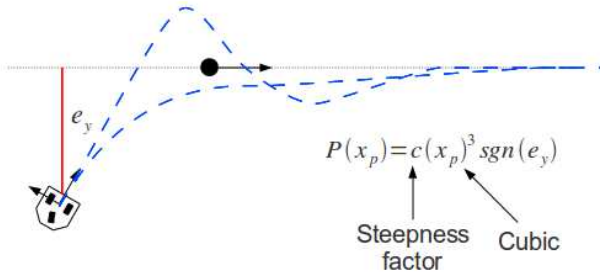


Figure 4.3: Cubic landing curve convergence shown for two different values of the steepness factor c.

in practice, although the steepness factor c determines how quickly the robot catches up with a turn in the desired trajectory. Although that controller has been shown well to work in practice, it uses a single parameter to tune convergence, which indirectly affects the turning radius. In our studies, we instead employ an advanced error-decoupling technique from control systems theory, known as dynamic feedback linearization, which works as follows.

The differential-drive kinematic model of our robot is given by the following equation.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{4.12}$$

Feedback controller design is challenging because (4.12) is non-invertible (insufficient rank). Fortunately, there exists a controller that satisfies our requirements and it can be obtained from (4.12) using *dynamic feedback linearization* [54, 45]. The process is summarized below.

First, we define a new coordinate system where the goal is at the origin and the robot is at $[x, y, \theta]^T$. Next, we add an integrator on linear velocity $v = \xi$, with linear acceleration acceleration $a = \dot{\xi}$. Then, we rewrite the equations in terms of the new controls by differentiating.

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\xi\sin\theta \\ \sin\theta & \xi\cos\theta \end{bmatrix} \begin{bmatrix} a \\ w \end{bmatrix}$$

and

$$\dot{\xi} = u_1 \cos\theta + u_2 \sin\theta \tag{4.13}$$

$$v = \xi \tag{4.14}$$

$$w = \frac{u_2 \cos\theta - u_1 \sin\theta}{\xi} \tag{4.15}$$

These equations define a system known as a the dynamic compensator. It has a potential singularity at $\xi = v = 0$, i.e. when the unicycle is not rolling. It turns out that this singularity occurs because of the structural property of non-holonomic systems. This is taken into account when implementing the control.

Assume that a smooth desired trajectory $(x_d(t), y_d(t))$ is to be tracked. Thanks to the decoupling of errors from the dynamic extension equations shown above, we can directly use an exponentially stabilizing feedback control to drive the errors to zero. The equations for the proportional-derivative (PD) law follow.

$$u_1 = \ddot{x}_d(t) + k_{p1}(x_d(t) - x) + k_{d1}(\dot{x}_d(t) - \dot{x}) \tag{4.16}$$

$$u_2 = \ddot{y}_d(t) + k_{p2}(y_d(t) - y) + k_{d2}(\dot{y}_d(t) - \dot{y}) \tag{4.17}$$

where all of the PD gains k_{pi} and k_{di} are positive. The velocities $\dot{x}$ and $\dot{y}$ can be derived from the state history via time difference, or, odometric information can be used when available. In our controller implementation, we followed the same method as in [45] to pick PD gain values $k_{p1} = 2$, $k_{d1} = 3$, $k_{p2} = 12$ and $k_{d2} = 7$. For the velocities, we used odometric readings from the wheel encoders.

4.4 Vision-based person-following

For the perception part on our robotic platform, we used computer vision algorithms as implemented by Ryan Lloyd using the stereo rig. One module was responsible for person detection using Haar cascades. This is a slow vision step, since it searches over the entire image for a persons face. However, it is run only occasionally – at initialization and when tracking is lost. The other module was a person-tracker using the fast Lucas-Kanade pyramidal scheme to track good features across consecutive frames. Figure 4.4 shows the color model net, which is the central model used to interconnect blobs and

detect people. This model is initialized manually at start-up, but the process can be automated by using background subtraction since the cameras are initially stationary.

Figure 4.4: (a) Tracing a subject for a color model histogram or Gaussian mixture model, and (b) Building a color model net

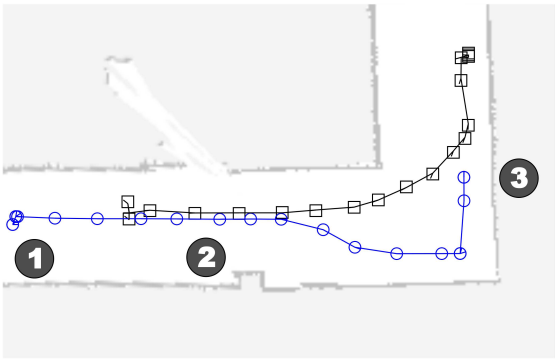
The primary contributions of this effort were: (i) novel adaptations of computer vision algorithms for human detection, tracking and pose estimation for real-time operation in a robust fashion, and (ii) implementation of a smooth trajectory-tracking controller in order to get smooth frame transitions from the camera.

4.4.1 Navigation controller

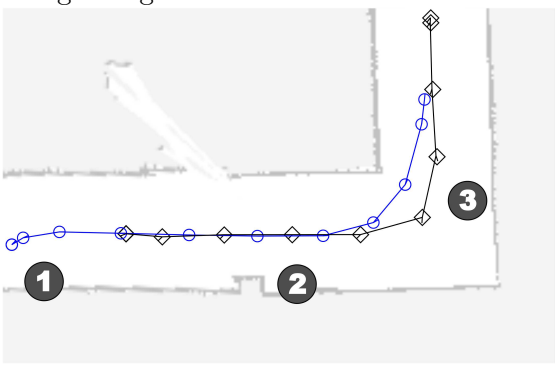
For the person-following task, we stored a fixed-length history of the person's pose as reported by the vision module. Since the controller from Section 4.3.2 requires a smooth trajectory, we processed the history using a smoothing window to ensure that derivatives were well-defined. The video stream captured by the cameras using the local planner reported was not as smooth when goals were given to the robot as if it were connected to the human with a rigid rod. The output is shown in Figure 4.4.1. By using the proportional error control instead, we were able to achieve smoother transitions when the person changed his trajectory. An example of the resulting robot trajectory in simulation is shown in 4.4.1.

4.4.2 Experimental results

We conducted 6 experiments on the second floor of an office building. In each experiment, we used one robot to maintain a distance of 1m to 2m from a walking person.



Simulation using a "rigid rod" to connect the human to the robot



Simulation using a proportional error control

Figure 4.5: Simulation where a robot (circle) maintains a frontal view of a human (square) with (1) good view acquisition, (2) through a corridor, and, (3) around a corner.

Experiments 1, 2 and 3 were conducted at night ($\sim$ 9 pm) under challenging low light conditions. Experiments 4, 5 and 6 were conducted during the day ($\sim$ 11 am) in corridors with areas of bright light (windows).

We observed that the color model was able to adapt to different lighting conditions.

The challenges faced during our experiments stemmed from two main sources: the latency of the wireless channel, and, the tracker locking on to objects in the environment. The plots for robot trajectory (from odometry) and tracked human pose (from vision) are shown below.

Experiment	Duration (min. sec.)	Good human poses (%)
1	5m52s	70.91
2	3m01s	79.41
3	7m58s	60.64
4	1m28s	99.61
5	1m43s	41.78
6	1m21s	99.21

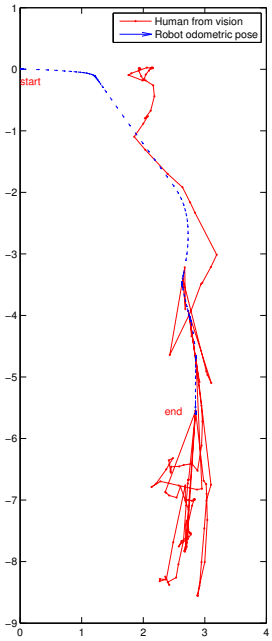
Table 4.1: Results from experiments: 1, 2 and 3 were during the night, and, 4, 5 and 6 were during the day

4.5 Summary

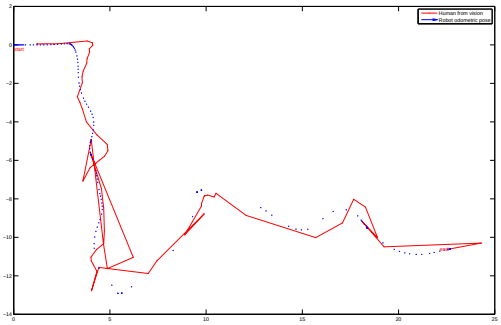
In this chapter, we studied the single target tracking problem in which the target is a person and since the robot follows the person, the trajectory of the target is known to the robot ahead of time.

The study of the person-following problem brought two things to light. First, we demonstrated a real-time, marker-less person detection and tracking module that helped our robots keep visual track of a subject. Second, the careful design of a convergent wheel velocity controller at the lower-level allowed us to integrate perception and navigation on a mobile base with a non-holonomic constraint. From the person-following experiments, we noticed two challenges. The wobble of the stereo cameras caused a significant offset in the pose estimate of the subject. Considering trends in person-tracking literature, we do not believe this will be a problem in the near future. We observed that the rapid variability of wireless signal strength in indoor environments led to slow system throughput in certain experiment runs. An interesting future direction from a sensor networks perspective is to model and incorporate wireless signal strength in the optimization criteria of our system.

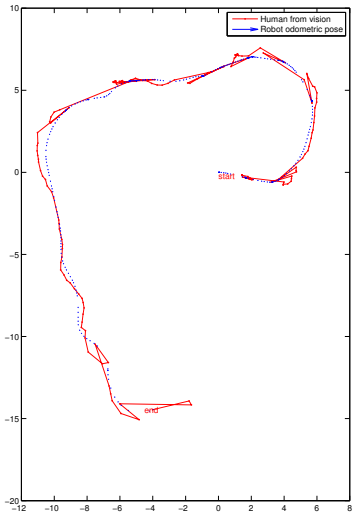
In this chapter, the trajectory of the person was known to the robot ahead of its motion. In the next chapter, we augment the trajectory-based human motion models used so far with contextual information about environmental geometry. This abstract model will enable us to represent and reason about long-term temporal dependencies that are inherent in the navigational decisions made by people along their trajectories.



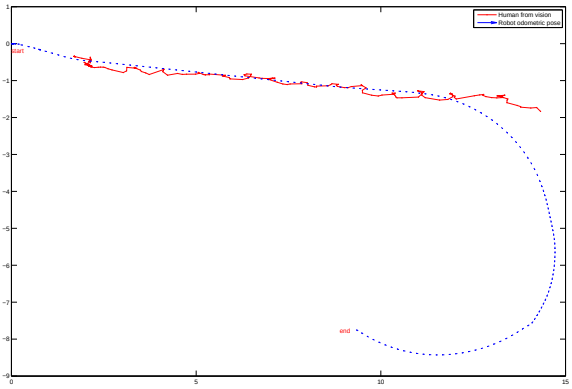
Experiment 1



Experiment 2

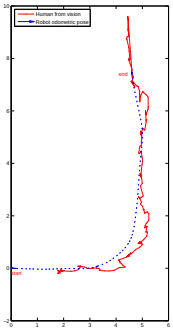


Experiment 3



Experiment 4

Experiment 5



Experiment 6

Chapter 5

A Reactive planner for tracking a mobile target

5.1 Introduction

Telepresence systems which allow people to save time and money by enabling them to participate in meetings and conferences from remote locations are becoming common. For example, commercial telepresence systems are offered by Cisco, HP, Polycom, Avaya, LifeSize, Tandberg, Teliris and others [7]. These systems, though useful, have the following disadvantages. A typical solution from Cisco costs about USD 300,000 (Figure 5.1). Telepresence systems in the industry have their origins in academia. Marvin Minsky is credited with the conceptualization of telepresence [55]. Sabri and Prasada [56] discuss early video-conferencing systems from an implementation perspective. Studies on modern multi-party telepresence systems emerged in the mid-1990s from research in computer graphics and virtual reality. Early systems include CAVE and ImmersaDesk [57] from the University of Illinois at Chicago, and Virtualized Reality from Carnegie Mellon University [58]. Other notable telepresence systems include Virtual Auditorium [59], Virtue [60], Digital Amphitheater [61], and Coliseum [62]. The National Tele-immersion Initiative (NTII) emerged as a collaboration between the University of Pennsylvania, University of North Carolina and Brown University [63, 64, 65]. The quality of 3D reconstruction of the environment and people has also been studied in this context [66]. More recently, the TEEVE multi-stream 3D tele-immersive system

was developed by the University of California at Berkeley together with the University of Illinois at Urbana-Champaign. These researchers studied techniques for image processing and 3D reconstruction, networking (end-to-end adaptations, transmission issues), and streaming data dissemination protocols in a series of papers [67, 68, 69]. All of these systems make significant contributions to the field of telepresence. However, all of them use cameras that are set-up in predetermined rigs, with little or no discussion on user mobility. Recently, industrial research labs have started working on single robot platforms for teleimmersion. Examples include Nokia's Jeppe [70] and HeadThere's Giraffe [71].

Recent advances in robotics have enabled affordable machines with sensors, actuators and network communication capabilities, such as the iRobot® Create® and the WooWee™ Rovio™ (Figure 5.2), to permeate the domestic consumer market.

In this paper, we present a mobile video-conferencing platform using commercial off-the-shelf robots. Such a system has the advantage of freeing the end-user from the aforementioned limitations of existing systems. Applications of our system are not limited to home users and large corporations. Educational institutions can use our system to provide opportunities for distance education and international collaboration. When used as a data collection tool, such a system can help the healthcare industry. For instance, health hazards can be prevented by monitoring the deterioration of ambulatory patterns in patients with dementia [11, 12, 13].

5.2 System overview

Our mobile video-conferencing system consists of multiple iRobot Create robots (differential drive), each carrying an Asus EEE PC connected over a serial interface. These inexpensive netbooks are powerful enough to run a fully-flavored Linux operating system. They also have a built-in embedded 1.3 megapixel camera. The laptops control the robots using the iRobot Open Interface (OI) specification. Communication between the robots and a central workstation uses an ad-hoc wireless network.

A key component of our system is the user's mobility model. In robotics literature, researchers model the human as an adversary and plan worst case strategies that maintain visibility of the user at all times [32, 72]. However, these methods do not directly

Figure 5.1: The Cisco TelePresence 3200 System. Courtesy of Cisco Systems, Inc. Unauthorized use not permitted.

Figure 5.2: Two off-the-shelf robots that are suitable for indoor use.

apply to our case because they are too conservative. In general, under the adversarial model, it is usually not possible to maintain a *good frontal view* of the user. For instance, people can make sudden turns away from cameras. Therefore, we introduce a restricted mobility model, and present algorithms to maximize the time during which the system gets a good view of the user. A formal specification is presented in Section 5.3. Section 5.4 outlines the optimal robot trajectories resulting from our formulation.

Any video-conferencing system has to bring together various components including perception, usability studies, audio and video compression, network protocols and bandwidth. In this paper, we focus on the motion planning aspect of the system. To simplify the sensing component, we use color-based markers (Section 5.7), and leave

vision aspects to future work. Our main contribution is the development of motion planning algorithms that enable video-conferencing systems to be mobile.

5.3 Problem formulation

We start by defining the coordinate frames and our state representation. Throughout this paper, we refer to the subject of the video-conference as the human, or the user. Define a mobile telepresense system with $n + 1$ entities in an environment with no obstacles: the set of n mobile robots $S = \{s_1, s_2, \ldots, s_n\}$ and a human user H.

5.3.1 State space

We eliminate the need for localization with respect to a world frame by defining a relative state space in which the user is always at the origin (0,0). We assume that each robot has an onboard camera, with an optical frame exactly coincident with the body frame. Using on-board cameras, any robot $s_i \in S$ estimates its own distance r_i from the user and its bearing θ_i relative to the user's *direction* of motion (see Figure 5.3). For a discussion on estimation of these parameters and the assumptions of our vision system, see Section 5.7.

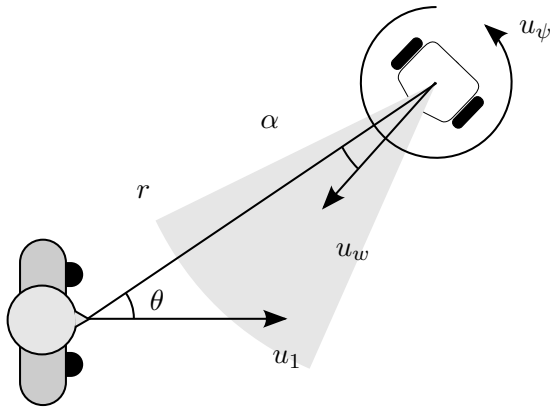


Figure 5.3: The state of a robot in polar coordinates, relative to the human. Control inputs are also shown.

5.3.2 Motion model

Each robot is a differential drive with two controls [73]: translational speed u_w, and rotational speed u_ψ. Let α be the angle between the optical axis each robot w.r.t. the line joining the robot and the man. When $\alpha = 0$, the robot's optical axis is directed toward the user. The straight-line motion of the user is modeled by a forward velocity $u_1 = 1$. Let u_w^{MAX} be any robot's maximum speed (relative to the user's maximum speed). We require that the robots be at least as fast as the user, i.e. $u_w^{MAX} \geq 1$. The following equations govern the state transition for a single robot with variables r, θ, α, u_w, and u_ψ.

$$\dot{r}(t) = -u_1 \cos\theta(t) - R u_w(t) \cos\alpha(t) \tag{5.1}$$

$$\dot{\theta}(t) = \frac{u_1 \sin\theta(t) - R u_w \sin\alpha(t)}{r(t)} \tag{5.2}$$

$$\dot{\alpha}(t) = \frac{R}{L} u_\psi(t) + \dot{\theta}(t) \tag{5.3}$$

In (5.1), (5.2) and (5.3), R is the wheel radius of the robot and L is the length of its axle. Since the robots are identical, these constants are the same for all of them.

5.3.3 Objective

The main objective of our robotic video-conferencing system is to maintain a good frontal view of the user for as long a duration as possible.

Let $\theta_{good} > 0$ be a constant angle measured with respect to the normal vector to the human torso facing front (Figure 5.4). Intuitively, if $|\theta_i| > \theta_{good}$, robot s_i is too far out to the side of the user for it to have a good frontal view. We have a similar constraint for the direction the robot faces, i.e. for α_i. We further want the complete height of the human to be within the viewing frustum of the robot's camera and his image in the camera to not be too small. These constraints can be expressed in terms of constants r_{min} and r_{max}, which define an annulus around the user on the ground plane.

Definition. We define the good view region, henceforth the *desirable set* $\mathcal{D} : [0, \infty[\rightarrow$

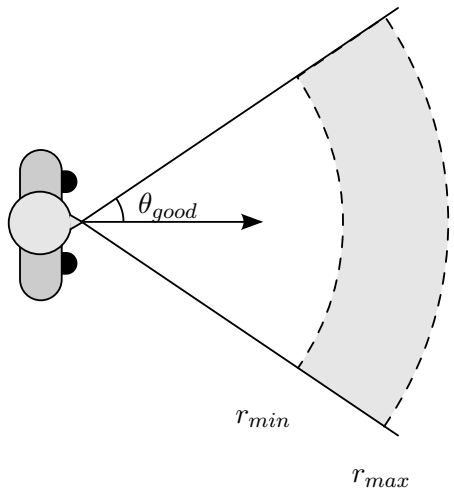


Figure 5.4: The desirable set $\mathcal{D}$ is shown (shaded) in the (r, θ) plane.

$\mathbb{R} \times [-\pi, \pi]^2$ at any time $t \geq 0$ as follows.

$$\begin{aligned}\mathcal{D}(t) = \{(r(t), \theta(t), \alpha(t)) \text{ s.t. } & r_{min} \leq r(t) \leq r_{max}; \\ & \text{and } |\theta(t)| \leq \theta_{good}; \\ & \text{and } |\alpha(t)| \leq \theta_{good}\}\end{aligned}$$

Property 1. *For any given time t, there exists a robot $s_i \in S$ such that its state $q_i(t)$ is desirable, i.e. $q_i(t) = (r_i(t), \theta_i(t), \alpha_i(t)) \in \mathcal{D}(t)$.*

The desired region is always centered at the user and aligned with the normal to his torso. Note that it may not be physically possible for the mobile robots to satisfy Property 1 at all times. For instance, when a lecturer turns to face a whiteboard, there might be boundaries covering the desirable region. Therefore, a reasonable objective for our system is to *maximize* the time during which Property 1 is satisfied.

Problem Statement. Given initial configurations for the robots in S relative to the user, find trajectories for each of them such that the time during which Property 1 is maximized.

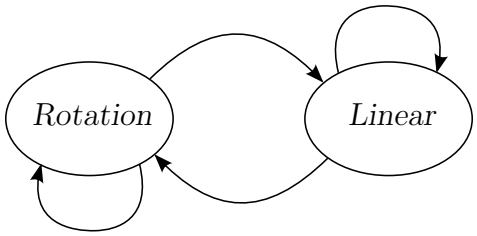


Figure 5.5: State machine for the user's motion pattern.

5.4 Human motion models

In our system, the user wants to broadcast his view using the mobile robots as a service. Therefore, we do not model the user as a strategic player who tries to minimize our objective. We observed that when explaining a topic, or giving someone a tour of our research lab, subjects do not rapidly simultaneously change their position and orientation. We model the user's trajectory as a sequence of the following two motion patterns. Modeling other complex motions, such as those in a dance, is part of future work.

1. **Rotation**
   This pattern models the case when the user makes minor adjustments to his position, but his orientation can change rapidly.

2. **Linear**
   This pattern models the user when he moves from one point to another along a straight line in the direction that his torso faces. For instance, from one part of a large room to another.

The motion model for the user can be thought of as a state machine with one state for each motion pattern (Figure 5.5). Transitions occur when the user either crosses a threshold distance in the rotation state, or, when he comes to a stop at the end of a linear motion pattern.

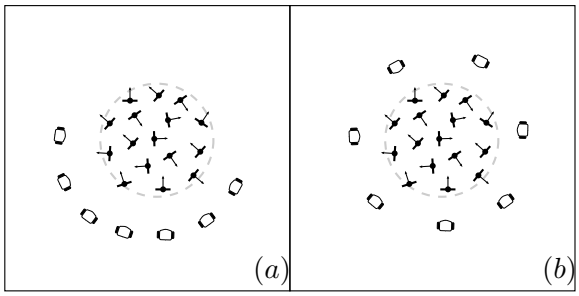


Figure 5.6: Angular separation of the robots for the ROTATION motion pattern of a single user: placement (b) covers more instantaneous changes in user orientation than placement (a).

5.5 Motion planning algorithms

In this section, we derive control strategies for our mobile robots using the assumptions made by the human motion model.

5.5.1 Motion pattern - Rotation

We say that the user is rotating if (s)he does not deviate from a fixed location by a predefined threshold distance, while his torso can assume an arbitrary orientation. This distance is illustrated in Figure 5.6 using a dashed circle. When the user makes minor adjustments to his position and orientation, we would like the robots to remain still. If instead, we allow the robots to move around with the user, the view of each robot can change too rapidly for it to be perceived as comfortable.

Since the robots maintain a circular formation for this user motion pattern, we are looking for an optimal placement of robots around the user. Since we ignore the user's small displacements, we now focus on his orientation. We assume that any orientation in $[-\pi, \pi]$ is *equally likely*, which means that the robots need to cover every possible orientation without bias.

The optimal placement for the robots is to spread themselves around the annulus with equal angular spacing. This can be proved by contradiction. Any optimal strategy that does not assign one robot per $2\theta_{good}$ angle will, by pigeonhole principle, assign more than one robot to a sector, thus leaving at least one sector empty. By re-assigning the extra robot to that sector, we end up with a strategy that accounts for more orientations

than the optimal strategy - a contradiction.

Remark. Within each sector that subtends an angle of $2\theta_{good}$ at the user, a robot can be placed anywhere such that the user lies within its field-of-view. We chose the bisecting angle for ease of implementation and visualization.

Remark. If we do not have a sufficient number of robots, i.e. when $n < \frac{2\pi}{2\theta_{good}}$, then the strategy that assigns not more than one robot to any $\frac{2\pi}{2\theta_{good}}$ different sectors is optimal, since no other strategy can cover a larger set of user orientations.

Therefore, for the ROTATION motion pattern, the robots maintain a circular formation around the user, with an equal angular separation between them w.r.t. the user.

5.5.2 Motion pattern - Linear

The user is in the LINEAR state when he translates beyond the distance threshold of the ROTATION state by moving along a straight line. When this transition occurs, the desirable region $\mathcal{D}$ stops rotating and instead, executes a translational motion parallel to the user's velocity vector (assumed to be the same as the normal to her/his torso).

When the user moves from one point to another in this state, we would like the robots to move with the user to his new location while maximizing the length of time during which Property 1 is satisfied. However, non-holonomic constraints prevent the differential drive robots from executing translational motion in arbitrary directions. Our goal is to find control inputs $u_w(t)$ and $u_\psi(t)$ for each robot $s \in S$, such that the length of time during which Property 1 holds is maximized.

We define two different robot strategies for the linear state, depending on whether the robot initially lies in $\mathcal{D}$ or not. Assume we have a sufficient number of robots in our system. We are guaranteed to have a robot $s_{best} \in \mathcal{D}$ at the time when the user starts his linear motion. In case of ties, e.g. if there are surplus robots, we pick the robot that has the least value of $|\theta|$. We call this the best robot. The goal of the best robot is to obtain the best view possible, i.e. reach the $\theta = 0$ ray as the user moves.

Now that the best robot is assigned the sector bisected by the $\theta = 0$ ray, the role of the remaining $n - 1$ robots is to maintain the circular formation around the user by allocating one robot for each of the remaining sectors. This can be implemented by

using an ordered sequence of predefined robot IDs, and a fixed direction around the sectors, say counterclockwise.

We have defined the boundary conditions for each of the robots. In the following section, we present optimal motion plans.

5.5.3 Breadth-first state-space search

Once the user starts his linear motion pattern, the robots have to start moving as soon as possible to avoid losing view. Therefore, we define a system parameter τ, which is a finite time horizon before which we would like the robots to be at their respective goal regions. Once τ units of time expire, the robots continue with the process of estimating the user's motion pattern and accordingly either maintain a circular formation, or, execute new finite-horizon trajectories.

For a given time interval of length τ, we build a reachability graph [73] $G(V, E)$ for the best robot s_{best}. In this graph, nodes q are discrete robot states and edges correspond to trajectories resulting from discretized controls (move back, move back and turn left, move back and turn right). For a fixed time discretization Δt, we generate $\tau/\Delta t$ levels of G, with a total of $|V| = 3^{\tau/\Delta t}$ nodes. We use heuristics to reduce the number of nodes in this graph, but in general, it is still exponential w.r.t. the time discretization. Non-exponential methods such as those in [74], are part of ongoing work.

An example of a heuristic we use is the relative angle θ. It can be pruned when it exceeds $\pi/2$, because the robot is too far out to the side of the user to return to the best view in time. Note that we also only explore backward robot motion for s_{best} because it faces opposite to the user's velocity vector.

Definition. The value (cost) of a state $q \in V$ of the reachability graph as the length of time up to that state, during which Property 1 was satisfied:

$$\begin{aligned} C(q(t) \equiv \langle r(t), \theta(t), \alpha(t) \rangle) = C(parent(q(t))) \\ + isDesirable(q(t)) * \Delta t \\ - K|\theta(t) - \theta_{sector}| \end{aligned} \tag{5.4}$$

There are three terms in (5.4), the first of which counts time from the beginning of the interval. The second term adds on Δt conditionally and the third one drives the value

of θ to θ_{sector}. The constant K converts angular difference to time units, for instance by dividing by the maximum angular speed of the robot. Define $isDesirable(q(t)) = 1$ if $q(t) \in \mathcal{D}(t)$ and 0 otherwise. $\theta_{sector} = 0$ for the best robot and we distribute the remaining θ_{sector} values (one for each robot) at equally-spaced angles around the circle.

The best trajectory is the one that terminates at a leaf node with the maximum value. The complete trajectory is found by backtracking.

Remark. The function $isDesirable(q)$ can be thought of as an integral cost that adds up over time. However, the discontinuities at the geometric boundaries of the region $\mathcal{D}$ make it difficult to apply traditional optimal control methods, because the partial derivatives of $isDesirable(q)$ w.r.t. the state q do not exist everywhere.

5.6 Simulation results

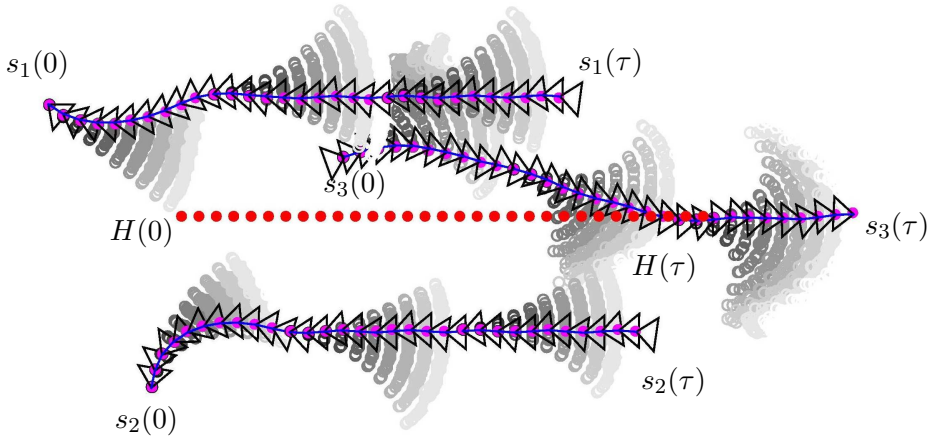


Figure 5.7: This figure is best viewed in color. Time increases from left to right. The user moves along the positive X-axis (H, or solid red). The robots (s, or solid magenta) explore the state space (hollow gray circles). The robots have a restricted field-of-view (triangles).

Figure 5.7 shows the solution we obtained on a sample instance with $n = 3$ robots. The optimal robot trajectories are shown as the user moves along a straight line. The figure uses a color gradient for the reachability graph nodes, in which the colors are darker early in the simulation (near $t = 0$) and get progressively lighter as time progresses. To manage the explosion of the number of states, we use depth-limited state

exploration. The robots start at relative angles 20°, 120° and −100°, and end at relative angles 0.75°, 140.11° and −120.09°. The good view angles were defined as $\theta_{good} = 40°$. For the robot that starts at 20° (s_{best}), all locations on its trajectory have a desirable view. The other two robots do not have a good view throughout this scenario. From this instance, we observe that the best robot initially saturates its turning angle and gradually straightens as it reaches the $\theta = 0°$ ray. We are currently trying to use this approach to possibly arrive at a geometric description of the curves the robot should follow, akin to the concept of T-curves by Bhattacharya et al. In their paper [75], motion trajectories were derived for limited field-of-view robots that navigate while maintaining visibility of a stationary landmark.

In order to view the user from each robot's perspective, we developed a new simulator. The application was written using simple data structures in C++ with OpenGL/GLUT for the visualization and user interface. The projections inherent to OpenGL allowed us to render the same scene from different camera viewpoints as the robots navigated in the environment. The human model used in the simulator is a 3D Studio Mesh from Artist-3D [76]. Figure 8.3 shows four different views, with a top view provided for visualization.

5.7 Vision-based state estimation

Our system uses vision as the primary sensing modality to extract the motion pattern of the human, as well as the state of the robots. Initially, we collected recorded human motion sequences, and extracted human motion parameters using background subtraction. This data was primarily used to validate the motion model.

In the real system, since the robots are moving, background subtraction is difficult. To simplify the complexity of the vision component, we placed a colored marker on the human (see Figure 5.9). We use the Continuously Adaptive Mean SHIFT (CAMSHIFT) algorithm [77, 78] to compute marker parameters. This algorithm uses the color histogram of a predefined marker as the model density distribution, and a back-projected image as the probability distribution of the model. We chose this algorithm because it works robustly even when the camera is in motion. We used the CAMSHIFT implementation provided in the OpenCV library. Once we track the human motion in

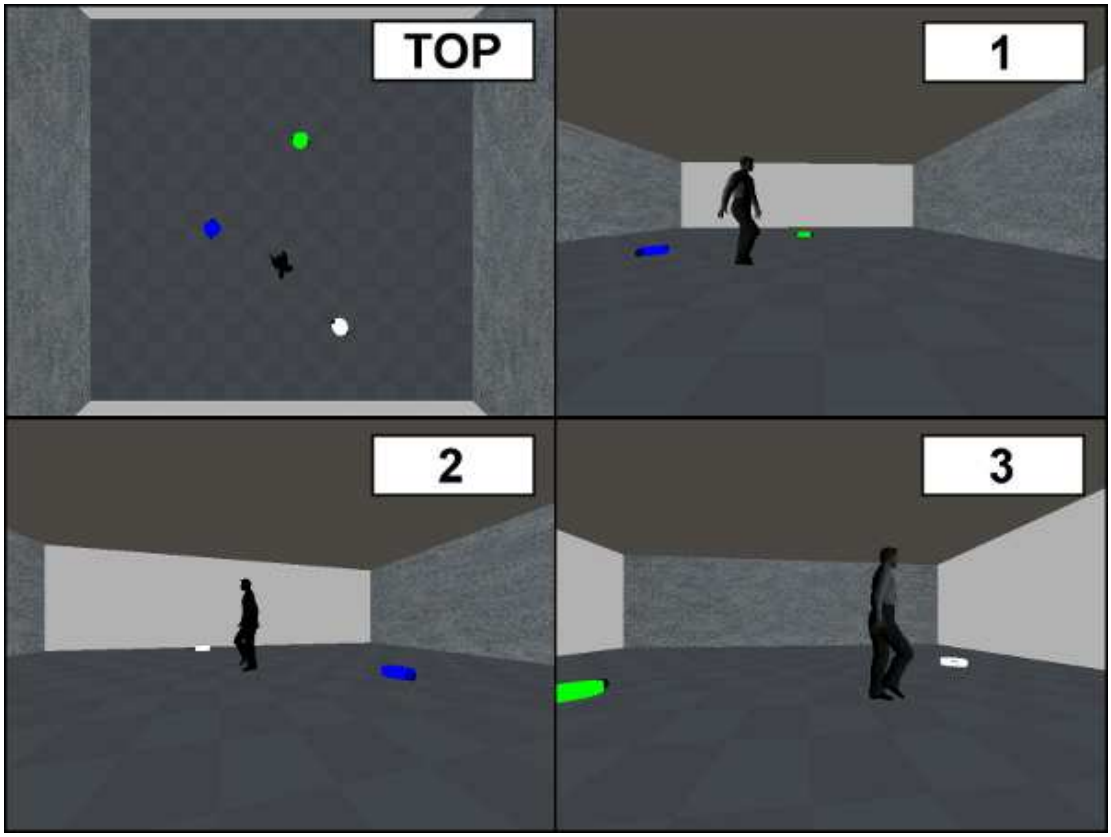


Figure 5.8: A simulator (C++/OpenGL/lib3ds) to render camera views as seen by individual robots 1, 2 and 3, and a top view.

each frame, we obtain the dimensions of the image of the marker along with the centroid of its bounding box. Using the intrinsic camera parameters and the known height of the marker, we obtain the distance of the person from the camera. The angle of the human motion with respect to the camera's optical axis is also determined from the centroid position of the bounding boxes of consecutive frames. Once these parameters are determined, we estimate the state of the human as follows. The first centroid point of the human is initialized as the reference point. If the centroid in successive frames moves by move than a threshold λ_1 with respect to the reference point, we classify the user to be in the Linear state. If not, the user remains in the Rotation state. If for λ_2 frames, the centroid does not move with respect to the reference, we set the state back to Rotation and the reference point is updated. Figure 5.11 shows the output of the

Figure 5.9: Proof-of-concept experiment showing the best robot (left) and a view from that robot at a different time (right).

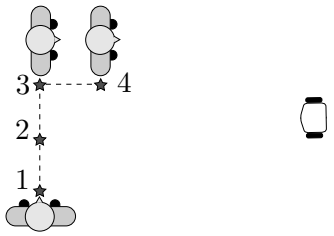


Figure 5.10: Example user trajectory for testing the vision-based state estimation from a stationary robot: Linear from 1 to 2, Rotation back and forth at 2, Linear from 2 to 3, Rotation by 90° at 3, Linear from 3 to 4.

state estimator.

5.8 Summary

In this chapter, we relaxed the requirement that the robots know the trajectory of the person before-hand. This introduces uncertainty in the person's state, requiring the robots to react to the person by actively sensing his mobility. We designed a two-state mobility model for the person and incorporated it into a reative motion-planning algorithm for mutiple robots that need to acquire a good *front view* of the person. We formulated a cost measure for good-view acquisition and solved the problem numerically, yielding to encouraging simulation results. We demonstrated the utility of our system by combining a vision-based tracking technique with our numerical motion plan to

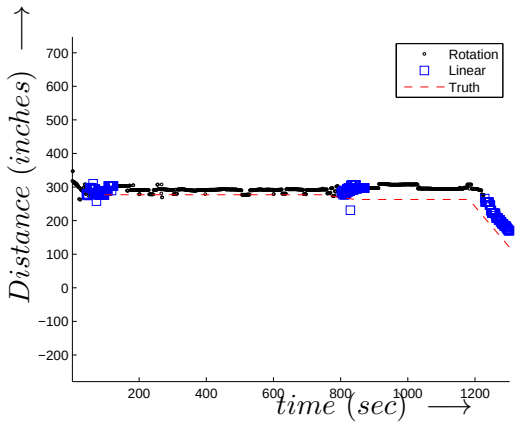


Figure 5.11: Real-time vision-based state estimation on the human trajectory from Figure 5.10.

successfully track and keep a person in good view using our in-house robot platform. However, it should be noted that this chapter did not explicitly model the environment around the person, i.e. it is applicable to open spaces. In the following chapter, we address the problem that it might not be possible for robots to always wait for the person and move reactively - there are situations where predictive planning yields better tracking performance, e.g. when the robots approach a hallway intersection.

Chapter 6

A Predictive planner for tracking an uncertain target

The problem of tracking a target with mobile sensors has widespread applications in robotics. Military operations and surveillance tasks demand worst-case guarantees and treat targets as adversarial, resulting in strategic interactions and pursuit-evasion games [31, 37]. However, there are also important applications in which targets are non-adversarial in nature. For instance, service robots can be used to monitor ambulatory patterns of elderly patients with Alzheimer's disease or dementia, who could pose a threat to themselves or others [12, 13]. Telepresence robots can be used to visit patients or have meetings with people in remote locations that are not easily accessible [64, 62, 7]. In order to achieve a high degree of autonomy in applications where robots coexist with people, navigation and planning algorithms need to account for human mobility patterns. A similar idea has shown favorable results in other applications, ranging from opportunistic wireless routing [79] and smart-home activity recognition [80], to reactive collision-avoidance [81, 82].

6.1 Our contributions

The cognitive process that governs decision-making for human navigation is complex and building a predictive model for it is challenging, e.g. the traditional Markov assumption of a person's current location being dependent only on his previous one does not describe

long trajectories with origin-destination (O-D) pairing [83, 84]. It is not clear that there exist generative models capable of encompassing all of its determining factors. In this paper, we instead build a mobility model from a representative set of previously observed trajectories, similar to data-driven approaches in the machine learning community. The process of collecting such a dataset is out of the scope of this paper because it involves research problems of their own merit, such as sensor placement, estimation theory and data mining. For our purposes, it suffices to know that it is possible to obtain such a dataset, e.g. by instrumenting the environment with sensors, and logging location trace data over time. Given such a dataset, we address the problem of finding a finite motion model that is both useful for predictive path-planning, and, capable of preserving long-term temporal dependencies inherent in human trajectories. To this end, we present a language-based representation of trajectories in which the tokens are vertices of a finite, planar environment graph. In Section 6.5, we introduce a probabilistic motion model based on the trie data-structure constructed from these tokens, which satisfies the requirement of variable-length history-based mobility prediction.

By using a common underlying planar graph representation of the environment for both person trajectories and robot paths, we integrate the human mobility model into a combinatorial planning framework. The probabilistic uncertainties in navigation decisions made by the person demand that the planner be capable of handling uncertainties. For this purpose, we integrate our model with a Partially-Observable Markov Decision Process (POMDP) with a specially-designed state representation, detailed in Section 6.6.

The predictive motion model and the multi-robot anticipatory planner together form the primary contributions of this paper. The feasibility and utility of our robotic system are demonstrated in Section 6.7, where we illustrate its modes of operation, and demonstrate improvements over previous approaches. We start with a literature survey in Section 6.2 and present the overall hardware and software architecture of our system in Section 8.2.

6.2 Related work

In this paper, we are concerned with the interaction of two problems: the *representation* of the uncertainties in navigation decisions, and, the *planning* of robot paths in the presence of these uncertainties. Traditionally, these problems have been dealt with only separately.

Factoring location history into various planning applications has shown encouraging results, e.g., automated museum tours [4], enhanced wireless routing [79], and activity recognition [85]. The process of mining interesting patterns from these histories in an interesting, but separate research problem in itself [86]. Mobility models emerging from these studies encode domain-specific expert knowledge that does not easily generalize, e.g. hierarchical neural networks, and Hidden Markov Models. In this paper, we present a predictive motion model that generalizes to any application where time-stamped location data is available.

In the target-tracking problem, navigation algorithms are designed for a robot to stay close to a target and/or maintain visibility. When the target's trajectory is known in advance, a dynamic programming approach can be applied [32]. Unpredictable targets, on the other hand, pose a more difficult problem, but local predictions have been used to improve tracking performance [87, 88]. Our work builds on these results and makes a fundamental contribution by: (i) incorporating prior knowledge regarding uncertain target trajectories into the planning process, and, (ii) addressing the path uncertainty at a global scale as opposed to local variations.

Sequential decision-making under uncertainty has been traditionally addressed using Partially-observable Markov decision processes (POMDPs) [89]. Various solution methods for POMDPs can be found in [90] and [91]. Existing POMDP methods that make the Markov assumption, i.e. the current state only depends on the previous state, do not model the long-term dependencies inherent in trajectory datasets. Recently, researchers have tried to address this problem by preserving the classic Markov assumption, but instead *biasing* the search for an optimal policy by sampling sequences of actions instead of atomic ones, e.g. by defining a heuristic importance function [92], and by weighting reward-exploitation against information-gain [93]. In contrast, we present a principled way of *constructing* a new belief state that incorporates long-term

dependencies by conditioning on observation histories of varying lengths.

6.3 Problem formulation

We are given a graph G representing a planar environment, a person denoted by h, and n mobile entities (robots) $\vec{r} = \{r_1, r_2, \ldots, r_n\}$. Time is discrete, starting with 0 and increasing in steps of 1. At any time step t, the person is at $h[t] \in V$ and the robots are at $r_i[t] \in V$. They simultaneously move to one of their neighboring vertices. The joint state of the system is denoted by $s = \langle \vec{r}, h \rangle$.

6.3.1 Actions and observations

In the same vein as classic combinatorial planners, we assume that the actions of robots on G are deterministic. In practice, this is achieved by mapping the 2-dimensional output of a reliable localization system, e.g. laser-based SLAM, to the nearest vertex in V. Once the robots take an action, the state of the system is not fully known due to the uncertainty in the person's navigation decisions, i.e. the person's actions are stochastic. Since the full state $\langle \vec{r}, h \rangle$ is partially observable (only $\vec{r}$ is observable), we employ POMDPs to formulate the tracking problem.

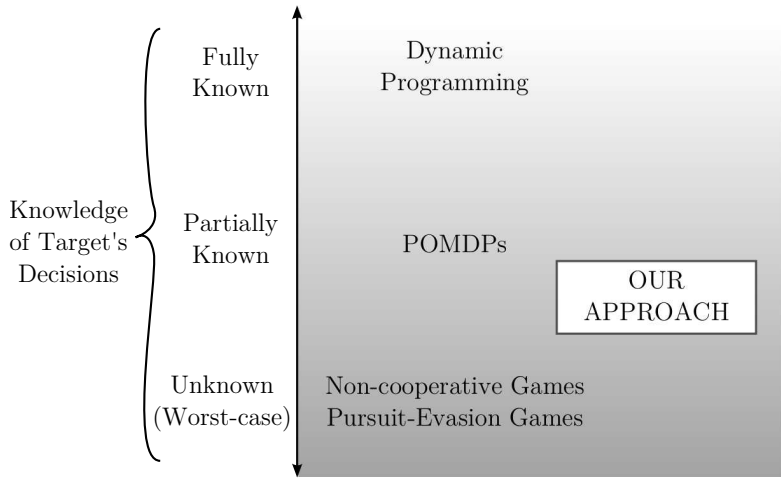


Figure 6.1: Our planning approach measured in terms of the assumed knowledge regarding target behavior

In the context of related problems, this assumption places our work in between the following extremes: (i) knowing the person's path fully in advance – complete information, and, (ii) having no prior information about the person's path. This is illustrated in Figure 6.1.

6.3.2 Preliminaries on POMDPs

Formally, a POMDP is a sextuple (S, A, O, T, Ω, R), where sets S, A and O denote states, actions and observations. Functions T, Ω and $R : S \times A \rightarrow \mathbb{R}$ denote the conditional transition probabilities, observation probabilities and a reward function. A POMDP encodes the uncertainty in state by maintaining a probability distribution over $s \in S$, known as the belief $b(s)$. A policy $\pi(b) \in A$ defines an action to take for every belief state b. The objective is to find a policy π that maximizes the expected reward discounted over time defined as follows.

$$J^{\pi}(b) = \sum_{t=0}^{\infty} \gamma^{\,t} E\left[R(s_t, a_t)|b, \pi\right] \tag{6.1}$$

$$\pi^*(b_0) = \underset{\pi}{\operatorname{argmax}} \;\; J^{\pi}(b_0)$$

6.3.3 Visibility-based optimization

Our objective is to maximize the time during which at least one robot acquires a good front view of the person, as shown in Figure 6.2. When this objective cannot be met, e.g. when the person is facing a notice board, at least one robot should still keep track of the person so that a better view might be acquired in the future.

In previous work [27], we studied the continuous space variant of this objective using a reachability-based state-space search. The challenge was to satisfy curve smoothness in addition to the non-holonomic constraints of the underlying differential drive model. The resulting contributions are suitable for use with Local Planning Methods (LPMs) (see Chapter 14 [32]).

In this paper we are interested in plans over larger horizons than typically handled by local planners. To this end, we define instead the following POMDP reward function

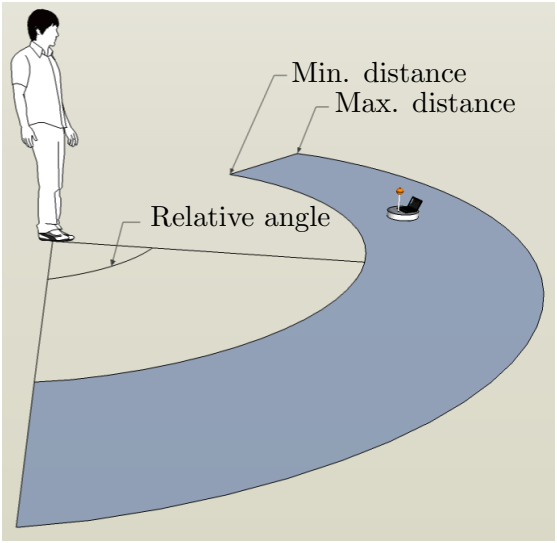


Figure 6.2: Geometric good-view objective based on acquiring a front view of the person relative to his torso

based on our graph-based discrete environment representation.

$$R(\vec{r}[t], h[t]) = \begin{cases} 1 & \text{if } \exists r_i \in \vec{r} \text{ s.t. } h[t] = r_i[t] \\ 0 & \text{otherwise.} \end{cases} \tag{6.2}$$

The existential operator in (6.2) prevents multiple robots from clustering around the person. As the simulations in Sec. 6.8 show, this has the desirable effect that they are able to simultaneously guard multiple intersections.

6.4 Modeling the Environment

Researchers from cognitive science have determined the fourth floor of our office building to be fairly complex for navigation and wayfinding [94]. Like most other buildings, the only map available for this environment is a custom-drawn floor plan, as shown in Figure 6.3. It is used mainly for accessing building infrastructure such as stairs, elevators and network closets (shaded regions in Figure 6.3). They were not designed for robot planning and it is neither straightforward nor advisable to use them directly with state-of-the-art path planners because their information: (i) is uncertain for localization via laser-based scan matching, and (ii) could be outdated due to architectural changes.

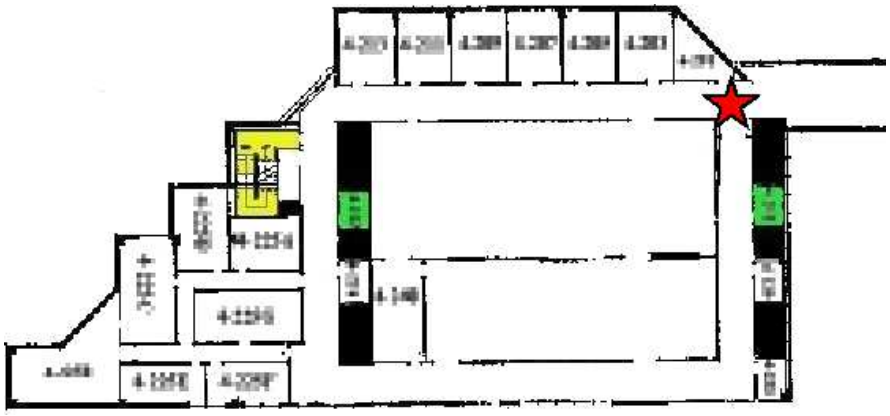


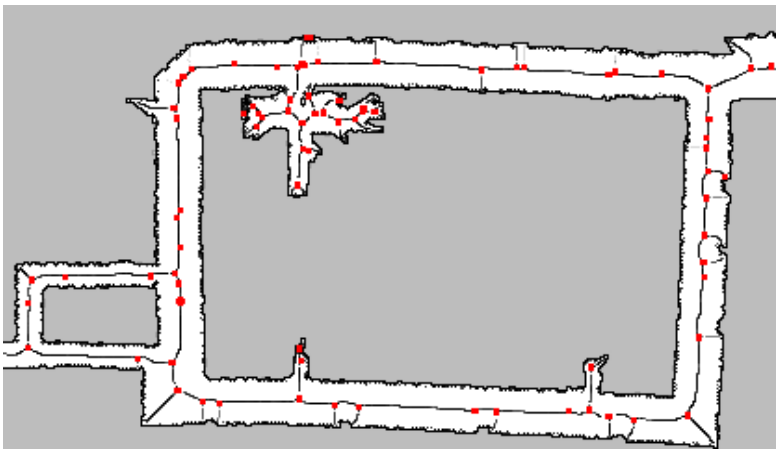
Figure 6.3: A typical floor plan image, shown here for a part of the fourth floor of our office building with a loop and a branch (marked with a star).

6.4.1 Map representation

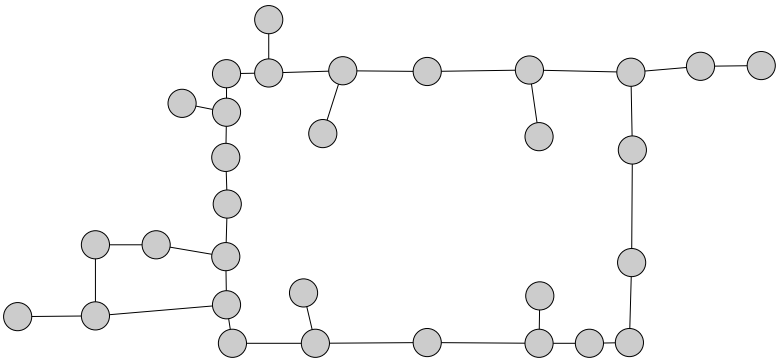
In this section, we describe three different approaches to building graph-based maps that are usable by planning algorithms. For all three approaches, we first constructed a map of the environment using the laser scanner on board the robot via SLAM using the `ros_gmapping` package. This ensures that the most up-to-date map of features is obtained, and further, it can directly be used to localize the robots with Monte-Carlo localization using the `amcl` package.

- **MapOG.** Occupancy grid graph.
- **MapDT.** Medial axis graph approximation using image-based distance transform.
- **MapMC.** Driving a robot in the metric map in simulation, and manually introducing graph vertices and their connections.

The main trade-off in this process is the number of graph vertices generated versus the preservation of topological properties. A very descriptive map can be constructed using a lot of nodes, but this increases the planner complexity. On the other hand, a topologically minimalistic representation is sufficient for combinatorial planning tasks, but difficult to construct without manual intervention. This can be observed from the results in Table 6.1.



MapDT: Distance transform



MapMC: Manually processed

Figure 6.4: Dicrete environment representations

6.4.2 Trajectory representation

In the previous section, we constructed a graph representation $G = (V, E)$ of the environment from a laser-based metric map. Mobility traces of people obtained from continuous sensors such as GPS, mobile phones, PDAs, and laser scanners contain time-stamped 2D data of the form (t, x, y). Much work in the A.I. and Machine Learning communities has been on unsupervised learning of interesting classes of patterns from continuous trajectory data, e.g. the inference problem in [86]. Instead, we assume that our trajectories are of the form (t, v) where $v \in V$. Typical datasets, such as the example shown in Figure 6.8.3, can be transferred to G by mapping any point (x, y) to its nearest neighbor

Environment	Number of vertices		
	MapOG	MapDT	MapMC
LabLoop	$\sim$ 12000	131	31
Floor4Full	$\sim$ 32000	181	50

Table 6.1: Robot action space as a result of map discretization

$v \in V$.

Sampling interval, dt	Token-based trajectory representation
2	ddbeec ...
1	ddddbaeeec ...
0.5	ddddddddbbaaeeeeeecc ...

Table 6.2: Coarsening of a sample trajectory $\{[0, d], [4, b], [5, a], [6, e], [9, c] \ldots\}$ by fixed-interval sampling

Any continuous person trajectory of the form:

$$H = \begin{bmatrix} t_0 & t_1 & \ldots & t_i & \ldots \\ x_0 & x_1 & \ldots & x_i & \ldots \\ y_0 & y_1 & \ldots & y_i & \ldots \end{bmatrix},$$

is then discretized from continuous coordinates into vertices on the graph representation $G(V, E)$ of the environment via nearest-neighbor mapping, resulting in a trajectory of the following form:

$$H_G = \begin{bmatrix} v_0 & v_1 & \ldots & v_i & \ldots \end{bmatrix}$$

Since the time discretization is fixed, a trajectory of the form $\{[0, a], [3, c], [5, b], \ldots\}$ would become $[aaaccb \ldots]$ if the sampling interval were fixed to $dt = 1$. Along with the other examples shown in Table 6.2, this demonstrates that our discrete representation captures both the *sequence* of vertices visited by a trajectory, as well as the *duration* of time spent at each vertex location.

6.5 Predictive motion model

In this section, we are interested in finding a suitable representation for the person's state uncertainty as his navigation process evolves over time. Commonly used stochastic constructs that model this process include Markov chains, Markov decision processes, and Markov random fields. However, as the following section demonstrates, the simple Markov assumption shared by these methods does not account for long-term temporal histories. In Section 6.5.2, we present our motion model that solves this problem by encoding variable-length histories.

6.5.1 Higher-order temporal effects

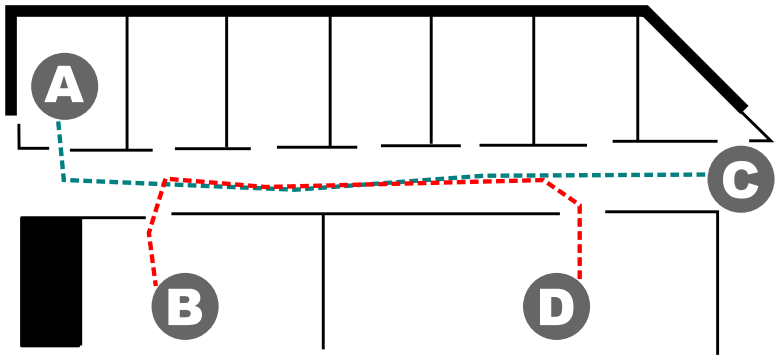


Figure 6.5: Two sample trajectories from the 'Student' dataset. The common portion between the two trajectories thwarts the prediction accuracy of fixed-order Markov models.

Consider the two sample trajectories shown in Figure 6.5, with time progressing from left to right. These trajectories exhibit what is known as origin-destination (O-D) pairing, i.e. the initial location of the person completely determines the final destination. However, a fixed-order Markov assumption, e.g. $h[t]$ depends only on $h[t-1]$, fails to capture this idea. This becomes a significant issue in the corresponding planning problem, because a correct prediction requires only one robot to track the person, whereas an uncertain estimate requires two robots to guard both of the branches that lead to C and D.

Since typical environments contain a large number of such decision points, it becomes all the more important to encode the person's complete location history into the model.

This is not trivial, since the addition of a sequence of locations $h[0], h[1], \ldots, h[t]$ to the state increases its dimensionality without bound. In the following section, we present our motion model that facilitates a compact representation of this history into the state space.

6.5.2 A History-based motion model

In the previous section, we motivated by example that it is important to incorporate the person's complete trajectory history $h[0], \ldots, h[t]$ in the state previously defined by the following representation.

$$s[t] = \langle \vec{r}[t], h[t] \rangle \tag{6.3}$$

The main idea explained in this section is to replace $h[t]$ in (6.3) by a new variable $m[t] \in \mathbb{T}$, where $\mathbb{T}$ is the underlying representation of our motion model.

Observe that any person trajectory of the form $h[0], \ldots, h[t]$ is a sequence of vertices on the graph G. Since V is finite, and time is discrete, trajectories are strings over the finite alphabet $\Sigma = V$. This perspective allows us to use variants of well-studied symbol prediction algorithms to predict person trajectories.

A typical predictor learns conditional probabilities of symbols in various contexts from a predefined text corpus, and uses it to predict new symbols. In our case, the analogue of a corpus is a finite, representative set of person trajectories τ along with frequencies $\upsilon(\tau)$. These frequencies are meant to encode navigation route preferences, e.g. students emerging at the lab prefer to take the exit more often than go to another lab. We make the assumption that such a dataset is available to us, similar to *data-driven* approaches common in the Machine Learning community [86]. The efficient collection of such a dataset involves both sensor placement, and estimation theory, both of which are separate research problems out of the scope of this paper. Our dataset is shown in Figure 6.8.3 and explained in Section 6.8.

Given a trajectory preference dataset $\mathbb{D}$, we construct a predictive motion model that represents the underlying generative process that produced $\mathbb{D}$. For this purpose, we use the trie data structure. Tries are routinely used on embedded devices to predict text via partial string matching, e.g. a dictionary, because they provide a compact

representation of a large number of strings. Searching for a new trajectory z in the trie amounts to a string prefix match operation that takes at most $|z|$ time.

Definition. A trie is a rooted digital search tree for storing strings, in which there is one node for every common prefix of a string. Any path from the root node to a leaf node represents a complete string.

In this paper, we use a variant of the classic trie, known as the enhanced trie [80]. The procedure for constructing an enhanced trie from a trajectory dataset is listed in Algorithm 6.6. The variable history length property that the enhanced trie shares with variable-length Markov models (e.g., [95]) enables it to adapt to higher-order histories in trajectory datasets.

Algorithm 1: TrieConstruct

Input : Environment graph G, trajectory dataset with frequencies $\mathbb{D} = \{(z_i, \upsilon_i)\}_{i=1}^{D}$.
Output: A trie $\mathbb{T}$.

for *each trajectory-frequency pair* (z_i, υ_i) **do**
1 `TrieAdd(`$\mathbb{T}$, z_i, υ_i`);`
2 **foreach** *prefix* `pref(`z_i`)` **do**
3 **foreach** *suffix* z_s = `suff(pref(`z_i`))` **do**
4 `TrieAdd(`$\mathbb{T}$, z_s, υ_i`);`
end
end
end

Figure 6.6: Algorithm for construction our motion model trie

In general, the number of different types of trajectory histories could be exponential in their length, but it is expected that this sort of variety is not exhibited by human behavior. Our data structure is designed to capture such patterns in a compact representation.

We now replace the previously defined vertex-based person state h with a new state

m that compactly encodes histories of trajectories.

$$s[t] = \langle \vec{r}[t], h[t] \rangle \qquad \longrightarrow \qquad s[t] = \langle \vec{r}[t], m \rangle \tag{6.4}$$

6.5.3 Trajectory prediction via blending

A trie $\mathbb{T}$ built from a dataset $\mathbb{D}$ encodes uncertainties in the navigational decisions made by the person in the form of conditional symbol probabilities that can be computed as follows.

For some observed trajectory history $\vec{z} = z[1], z[2], \ldots, z[t]$, the prediction of $z[t+1]$ could conditionally depend on any of the prefixes of $\vec{z}$, of length $0, 1, 2, \ldots, t$. The following recursive formula defines a blend of predictions from these variable-order histories. For ease of notation, let $Pr[i:j]$ denote $Pr(z[t+1] \mid z[i], \ldots, z[j])$. We have,

$$\begin{aligned} Pr(z[t+1]) = \quad & Pr[1:t] \\ & + \Big\{ (1 - Pr[1:t]) * Pr[2:t] \Big\}. \end{aligned} \tag{6.5}$$

In (6.5), the term $Pr[1:t]$ denotes a successful match of the longest prefix. The second term factors out the symbol $z[1]$ from the history by calculating the probability of no match at order-t prefix and a match at the lower order-$(t-1)$ prefix. This inductively proceeds until the order-0 prefix $P[t:t]$, after which the values are zero.

We complete the blending procedure (6.5) by defining the term $Pr[1:t]$ for a successful match $z[1], \ldots, z[t]$ in $\mathbb{T}$, over child nodes Ω of $z[t]$, as follows.

$$Pr\Big(z[t+1] = \Omega \mid z[1], \ldots, z[t]\Big) = \frac{\upsilon(z[1:t] \cdot \Omega)}{\upsilon(z[1:t])} \tag{6.6}$$

The formulæ (6.5) and (6.6) together encode the higher-order temporal dependencies we sought to represent in Section 6.5.1.

6.5.4 Predictor comparison

With the help of an example, we illustrate the utility of our variable-length model over standard fixed-order Markov models. Consider the graph shown in Figure 6.7 with the following trajectory dataset. Vertex A is a start point for one trajectory that ends at K, i.e. $\tau_A = ABCDEFGHIJK$. Vertices $Z, Y, X, W, V, U, T, S, R$ are start

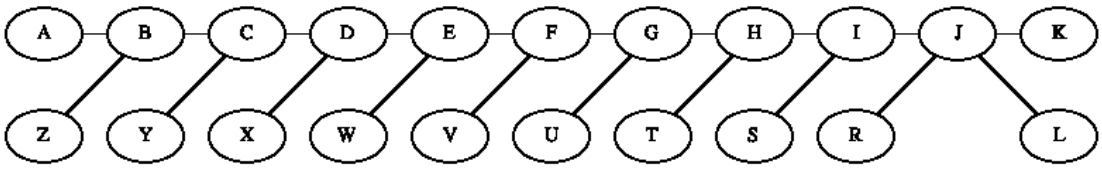


Figure 6.7: Sample trajectories sharing various common edges that demonstrate the importance of history length

points for 9 other trajectories that share common parts with τ_A, but all of them end at L instead of K. Suppose the actual path of a person on this graph is $\tau_A[1:10] = ABCDEFGHIJ$. We require a motion model to predict the next location, as either K or L.

We build order-k Markov models based on these trajectories and use each of them to compute the conditional probability of the next state, i.e. $P(z[11] = K)$. The results are shown in Figure 6.8. As expected, the 9 trajectories other than τ_A have the effect of overshadowing it when the length of the history is insufficient to capture the high-order temporal dependency of K on the initial location A. In this example, although the perfect predictor is an order-10 Markov model, in general there could exist datasets where fixing the history length k of the predictor has a detrimental effect on its prediction accuracy. Our motion model does not have this problem because it is designed to store variable-length histories.

6.6 Our Multi-robot planning algorithms

In general, the person trajectory dataset τ may not contain all edges of the environment graph G. Let G_T be the subgraph of G induced by the person trajectories in the dataset τ. It is possible for a robot to select a path from G that has edges not present in G_T in order to avoid areas of high uncertainty. One such example is illustrated in Figure 6.9. More formally, because $V(G_T) \subseteq V(G)$ and $E(G_T) \subseteq E(G)$, the problem of selecting paths for robots on G is *not* the same as assigning trajectories from T to each robot. This is important because there could be cases where it is beneficial for a robot to use a "shortcut" path from G that is not in T so that the overall path collects higher reward. The generality of our POMDP-based formulation inherently allows for this possibility

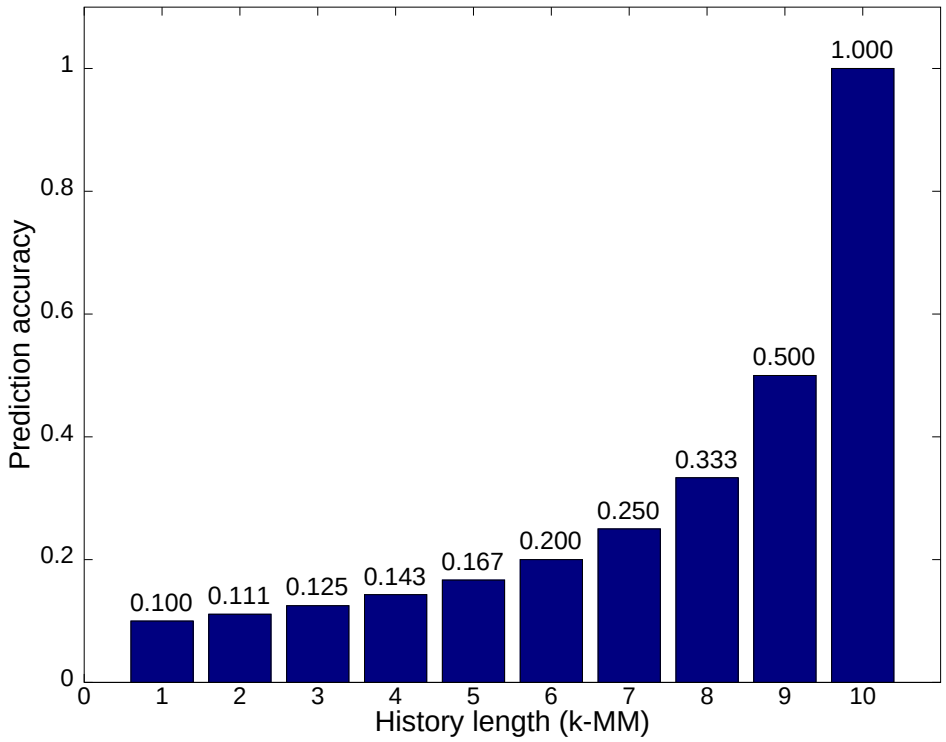


Figure 6.8: Accuracy of order-k Markov model predictors on the example trajectories from Figure 6.7

because robot actions are defined by their vertex neighborhoods from the *entire* graph G.

A POMDP solution is a policy that determines robot actions for *every* belief state. However, the true belief state is not known until an observation is made in real-time, as the person moves. In this section we present two algorithms: one that computes multi-robot plans offline, and one that executes policies based on online observations from the person tracker.

6.6.1 OfflineMRAP: Multi-robot anticipatory planner

Algorithm 6.10 lists the formulation and solution of the anticipatory planner that is done offline with the help of a POMDP solver. Multi-robot policies Γ^* resulting

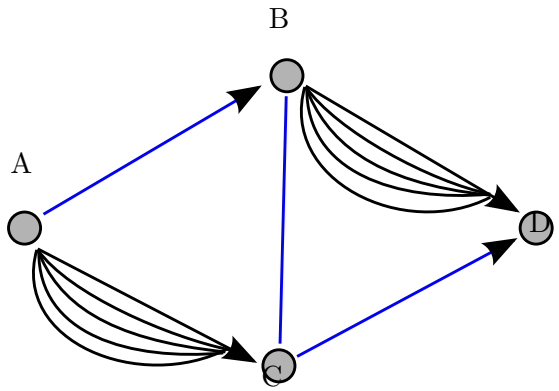


Figure 6.9: Example: $T = \{ABD, ACD\}$, with multiple paths that dilute the probabilities between AC and BD. A robot could take the "shortcut" path outside G_T to avoid uncertain regions (shown in blue), thereby collecting a better total expected reward.

from this phase are saved at the decision center for future look-up. The procedures `TrieConstruct` and `TrieSearch` were described earlier, as part of the motion model in Section 6.5. The procedure `Solve` employs the SARSOP POMDP solver. SARSOP uses successive approximations based on belief-state reachability [88, 92]. It uses a factored belief representation and alpha-vector policy representation, which we interface with during the policy execution phase.

6.6.2 OnlineMRPE: Multi-robot policy executor

Algorithm 6.11 lists the sequence of steps for the online policy execution phase. The procedures called from this algorithm as subroutines are as follows.

- `RandomNeighbor`

  As a design goal, we incorporate a fallback policy into our planner. If an observation history is encountered during policy execution that was not previously observed in the dataset $\mathbb{D}$, the predictor will not be able to find a match. In this scenario, the planner picks a neighboring vertex for the robot to move to, uniformly at random. We chose the uniform selection scheme because this pertains to the system having *no prior information* as to where the person could navigate. Thus every outgoing edge is equally likely.

- `PersonTracker`

  On board each robot, the person tracker is responsible for detection, tracking and filtering of the person's 2-D location in the plane. It also communicates this estimate to the decision center, where multiple observations are fused (the `Observe` subroutine).

- `LocalPlanner`

  The local planning module is a combination of a sequential waypoint-based planner that issues graph vertices as navigation goals, and a lower-level planner that connects initial pose to goal pose. The lower-level planner was tested with two approaches: The default `move_base` planner from ROS [96] that uses the Dynamic Window Algorithm (DWA) for collision-avoidance [97], and a smooth convergent planner that we implemented based on differential equations for wheeled mobile robots (WMR) [54, 45]. Other alternatives could also be easily plugged-in, e.g. the landing-curve method [44] used in Chapter 4.

- `UpdateBelief`

  Since the trie is a tree by definition, any node m uniquely determines a path from the root node of the trie to the vertex represented by m that corresponds to a trajectory prefix $z[1:t]$ from the dataset $\mathbb{D}$. Thus the node m can be used in (6.6) to compute the belief state over future locations based on *complete* observation history.

6.7 Implementation details

6.7.1 Trie structure

Figure 6.12 shows the structure of an enhanced trie for the STUDENT scenario. Suppose that the environment has $|V|$ vertices, and there are M trajectories with an upper bound on the maximum trajectory length H. If the maximum vertex degree is D, then the theoretical number of trajectories the target could take is, in the worst case, exponential: $M * (D+1)^{H-1}$. We use $H-1$ and not H in the exponent because the initial location is known, and, $D+1$ in place of D because the target could stay at the same vertex (self-loops).

6.7.2 Planner frequency.

Although this whole process can be repeated with high frequency ($\sim$ 60Hz), the discrete graph representation only changes the person's state when he moves a sufficient distance. In our experiments, we found that a planner rate of $\sim$ 5Hz suffices.

6.7.3 Tracking filter parameters.

(fig. from tracking tests) In our system, deviations up to 10cm from the planner paths are acceptable. This is due to a combination of two factors – the field of view of our sensor is about 60°, and we do not require robots to follow exact paths, so long as they keep track of the person. This parameter is used to determine the covariance parameters of our filter. We find that it removes jitters from tracker artifacts and performs well in practice.

6.8 Simulations

In this section, we apply our motion model and planning algorithm to the multi-robot target-tracking problem defined in Sec. 6.3.3 on three sets of simulation runs. The first is an intuitive 1-robot example on a toy graph that demonstrates the effect of the discount factor γ on the quality of the robot policy obtained. The second demonstrates the limitation of fixed-order Markov models with regard to location-prediction accuracy. The third set of simulations involves two robots and is based on a real-world STUDENT trajectory dataset acquired in our office building. Unless otherwise specified, we evaluated the policies obtained in Monte-Carlo fashion by sampling person trajectories, executing the multi-robot strategies, and computing statistics for expected reward over multiple trials.

6.8.1 Tracking performance of predictors

Monte-carlo to sample person's trajectory and execute: (i) our planner, and (ii) O(1)-Markov planner, comparison shown in Table 6.3.

Table 6.3: Comparison of predictive motion models

Motion model	Prediction accuracy	Exp. tracking time
O(1)-Markov	25%	26.19%
O(2)-Markov	33%	35.09%
O(3)-Markov	50%	53.36%
O(4)-Markov	100%	84.41%
Our model	100%	84.39%

6.8.2 Effect of discount factor

Although the person's state transitions in our POMDP are restricted to the trajectories encoded in $\mathbb{T}$, the robots are free to move from vertex-to-vertex. The value of γ therefore affects the robot policies obtained as solutions to the discounted POMDP. In our case, setting $\gamma = 1$ ensures that the total reward of a policy reflects the total tracking time, but this is not possible for infinite horizons due to convergence requirements. Intuitively, as $\gamma \longrightarrow 0$, the Bellman equations converge faster, but future rewards are discounted more. The reduced time to solve the POMDP is compensated by the resulting robot strategies behaving in a greedy fashion. We demonstrate this behavior using one robot for the environment and trajectories shown in Fig. 6.8.2. As γ is increased, the robot sacrifices initial certainty for greater rewards in the future. In this example, there are only two classes of robot strategies which are shown in Fig. 6.8.2. We used a binary parameter search to determine that the value at which the robot switches its strategy class is $\gamma = 0.5196$. For 10^6 trials and $t_f = 7$, we report the following.

$$\text{Tracking time was } \begin{cases} 3.0001 \pm 1.5804; & \gamma < 0.5196 \\ 4.1238 \pm 2.1468; & \gamma \geq 0.5196 \end{cases}$$

This simple demonstration suggests that time could be saved on larger problem instances by carefully selecting a value of γ. For the remaining simulations, we selected $\gamma = 0.6$. We are currently investigating methods to select γ based on factors such as the length of the histories stored in $\mathbb{T}$.

6.8.3 The Student Trajectory Dataset

Our robot is an iRobot Create fitted with a Hokuyo laser scanner. We obtained a generalized voronoi diagram (GVD) approximation for our office building (Fig. 6.8.3) using the occupancy grid from `ROS::gmapping` [96], followed by CGAL skeletonization [98], and manual processing to remove excess graph nodes. The resulting graph G had 31 vertices.

We designed a Java™ applet to interactively collect trajectory preference data from students that frequent our environment. Users were instructed to click on sequences of vertices and the resulting trajectories were re-sampled using a fixed time discretization. A total of 100 trajectories of various lengths were recorded (see Fig. 6.8.3). The robots were stopped after $t_f = 32$ and their policies evaluated for the total reward.

The trie constructed from this dataset used 780 nodes to represent the complete histories. Since any node in the trie could be part of the POMDP state (recall $s = \langle \vec{r}, m \rangle$), the person's belief also contains the same number of entries. The joint space for two robots had $31^2 = 961$ states. On this instance, the SARSOP solver took 53.9 sec to converge.

Using the average tracking time as a measure, we compare our planner to the hypothetical optimal planner OPT that is permitted to use as many robots as necessary to keep the person in track for all times, i.e. $OPT = 32$.

In the student trajectory dataset shown in Fig. 6.8.3, two robots are insufficient to guard all possible decision points, and the relative frequencies of routes become important. This reflects practical systems where the number of robots is less than the number of navigation decisions points typically found in trajectory datasets.

The optimal two-robot strategy obtained using Algorithm 6.10 is shown in Figure 6.14. Over 100 simulation trials, our strategy has an average tracking time of 19.76 units, with a 95% confidence interval of $[17.85, 21.66]$. Looking at it from the other perspective, we observed that on average, 65% of randomly sampled person trajectories were tracked by the robots during at least 75% of their lengths.

Observations. Consider the subgraph $G(\Gamma^*)$ shown in Fig. 6.15 induced by the optimal two-robot policy Γ^* shown in Fig. 6.14. Dashed lines denote paths with multiple nodes, and solid lines denote edges between neighboring vertices. The kink denotes that path

K–H is longer than N–H. We make some interesting observations based on the structure of the optimal two-robot policy that could help scale the problem to larger instances. First, the two robots in Γ^* never cross paths when they arrive at a decision point *at the same time*. This is true because crossed robot paths can be replaced by uncrossed ones that collect exactly the same reward. However, paths that cross each other at different time instances can still exist.

Second, our approach handles trajectories of different lengths as follows. The path N–H is shorter than path K–H. Our solution reflects this temporal aspect by making one of the robots wait at vertex H. Third, we used an existential operator in (6.2) to define our reward function. The consequence of this formulation is that it leads to policies in which different robots *simultaneously* guard different branch points. For instance, while one robot covers path MN–H, the other robot moves along along MK–H, thus protecting the branch at vertex M. Although one of these robots will not track the person when he actually picks a path to traverse, our formulation ensures that it is *re-used* later in time to guard the branch at B.

These observations suggest that there might be structural properties that can be used to solve the robot motion strategies in path-space instead of vertex-space, an idea that is currently under investigation.

6.9 Conclusions

In this chapter, we studied the problem of representing the uncertainties in global path behaviors of people, with applications to motion prediction for service robots. Existing mobility prediction models make the simplifying Markov assumption on the person's location, which does not account for higher-order temporal dependencies inherent in trajectory datasets. Although the number of trajectories, in general, could be exponential in the number of underlying nodes in a discrete representation, we expect that human behavior does not exhibit this kind of variety. We capture the underlying structure of human motion using a compact trie data structure capable of storing histories of variable lengths. We incorporate the stochasticity in our motion model into a POMDP-based multi-robot planning formulation so that robots can reason about how the person moves in order to better keep track of him.

We demonstrated the utility of our motion model and our planning algorithm in simulation, using a variant of the visibility-based multi-robot target-tracking problem from our previous work [27]. We constructed a map from a real-world office environment, collected data from students that frequently traversed its hallways, and demonstrated using simulations a two-robot strategy that maximizes the expected total time during which the person is tracked by at least one robot. We observed interesting structural properties that have the potential to reduce the complexity of the multi-robot planning problem on larger instances. We use a lossless representation (trie) to store the complete trajectory history, which could become impractical for large datasets. As part of ongoing work, we are investigating trajectory subset selection heuristics, and approximate trajectory representations.

In this chapter, we discussed the target-tracking problem in which a good front view of an uncertain target needs to be acquired by a small team of robots. We explicitly modeled the environment using an abstract graph representation, constructed a novel uncertainty-based mobility model of the person, and folded it into a motion planning algorithm that helped our robots acquire and maintain a good track of the person.

In the following chapter, we throw new light on an old problem that has found relevance in the context of this thesis: novel view synthesis for tele-immersion systems.

Algorithm 2: Multi-robot anticipatory planner (OfflineMRAP)

Input : Environment graph G, person trajectory dataset $\mathbb{D}$ with frequencies $\{v_i\}_{i=1}^{D}$, a set of robots $\vec{r} = \{r_j\}_{j=1}^{n}$, a planning horizon t_f, discount factor γ, initial locations of the robots and the person $(\vec{r}[0], h[0])$.

Output: Multi-robot strategy Γ^* that maximizes the expected total reward.

▷ *Build motion model*

1 $\mathbb{T} \leftarrow$ `TrieConstruct(`$\mathbb{D}$`)`; ▷ *Algorithm 1*

▷ *Formulate POMDP*

2 $m[0] \leftarrow$ `TrieSearch(`$\mathbb{T}$, $h[0]$`)`;

3 $b_0 \leftarrow \langle\, \vec{r}[0]\ ,\ m[0]\ \rangle$;

4 **foreach** *Location vertex* $v \in V$ **do**

5 **foreach** *Neighboring vertex* u *of* v **do**

6 **foreach** *Robot* r_i **do**

7 Add u–v to the action space of r_i with transition probability 1; ▷ *deterministic*

end

end

end

8 **foreach** *Node* z *in the subtree* $\mathbb{T}(h[0])$ **do**

9 **foreach** *Child node* Ω *of* z **do**

10 Add z–Ω to the action space of h with transition probability $Pr(\Omega \mid z)$; ▷ *stochastic*

end

end

11 Define reward $R(\vec{r}, h)$ as in equation (2), and set it to zero for robot collisions $(r_i == r_j)$;

▷ *Compute optimal robot strategies*

12 $\Gamma^* \leftarrow$ `Solve(`$\mathbb{P}$, b_0, t_f, γ`)`;

13 Save Γ^* for policy execution.

Figure 6.10: Algorithm for uncertainty-based planning

Algorithm 3: Multi-robot policy executor (OnlineMRPE)

Input : Environment graph G, Optimal multi-robot policies Γ^*, a set of robots $\vec{r} = \{r_j\}_{j=1}^{n}$, initial locations of the robots and the person $(\vec{r}[0], h[0])$.

Output: Executes the plan and updates the system state.

▷ *Initialize person state*

1 $m[0] \leftarrow$ `TrieSearch`$(\mathbb{T},\ h[0])$;

▷ *Initialize full belief state*

2 $b[0] \leftarrow \langle\ \vec{r}[0]\ ,\ m[0]\ \rangle$;

3 $i \leftarrow 0$;

4 **repeat**

  ▷ *Look up saved policy* Γ^* *for robot goals*

5 $\vec{g}[i+1] \leftarrow \Gamma^*(b[i])$;

6 **foreach** *Robot* r *in* $\vec{r}$ **do**

    ▷ *Fallback policy : uniform prediction*

7   **if** $b_r[i]$ *was not found* **then**

8     $g[i+1] \leftarrow$ `RandomNeighbor`$(r[i])$;

    end

  end

9 **foreach** *Robot* r *in* $\vec{r}$ **do**

    ▷ *Send goal to local planner*

10    $r[i+1] \leftarrow$ `LocalPlanner`$(r[i],\ g[i+1])$;

    ▷ *Make a local observation*

11    $h_r[i+1] \leftarrow$ `PersonTracker()` ;

  end

  ▷ *Combine observations*

12 $h[i+1] \leftarrow$ `Observe`$(h_{r_1}, h_{r_2}, \ldots, h_{r_n})$;

  ▷ *Belief update*

13 $b[i+1] \leftarrow$ `UpdateBelief`$(b[i],\ h[i+1])$;

14 $i \leftarrow i+1$;

until *end of service* ;

Figure 6.11: Algorithm for policy execution

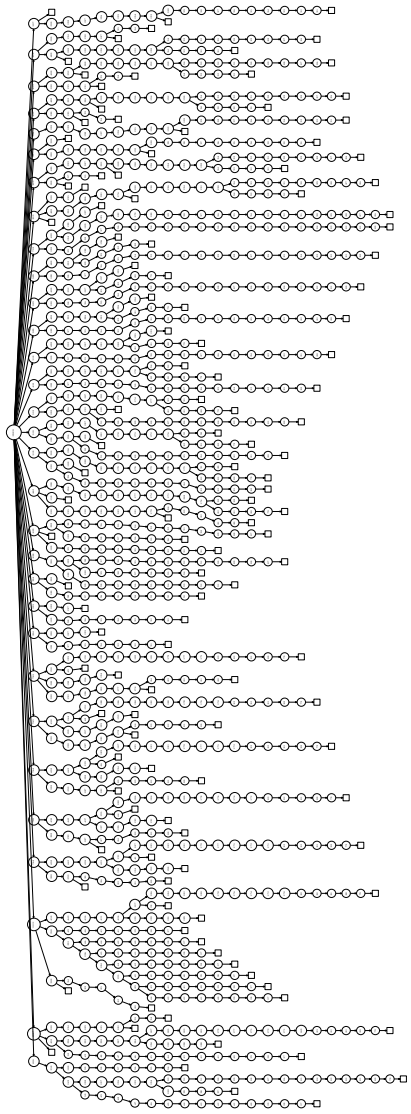


Figure 6.12: Trie for the 'Student' dataset that encodes 19031 timestamped locations on a graph with 31 vertices using 1068 nodes. For max-degree 4 and a history length of 25, this is far from the worst-case upper bound of $31 * (4 + 1)^{24}$.

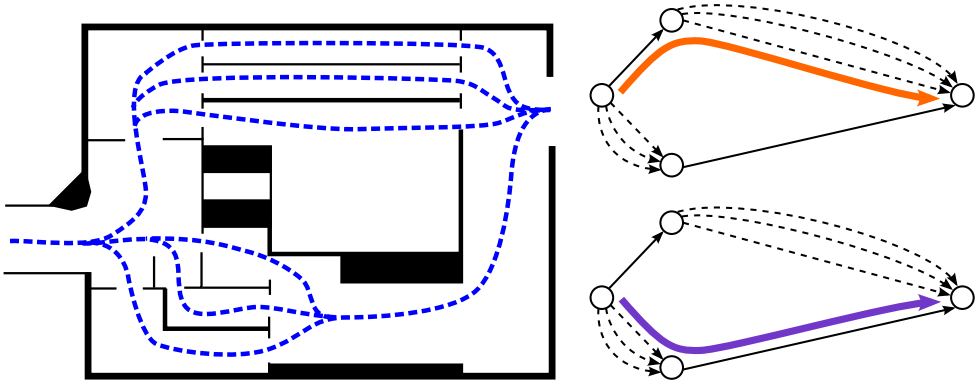


Figure 6.13: Simulation I: (a) A person's trajectories in an indoor environment with branching, and, (b) the effect of discounting on the quality of the corresponding solution: $\gamma = 0.1$ gives a greedy strategy (top), whereas $\gamma = 0.9$ gives a strategy with better average tracking time (bottom).

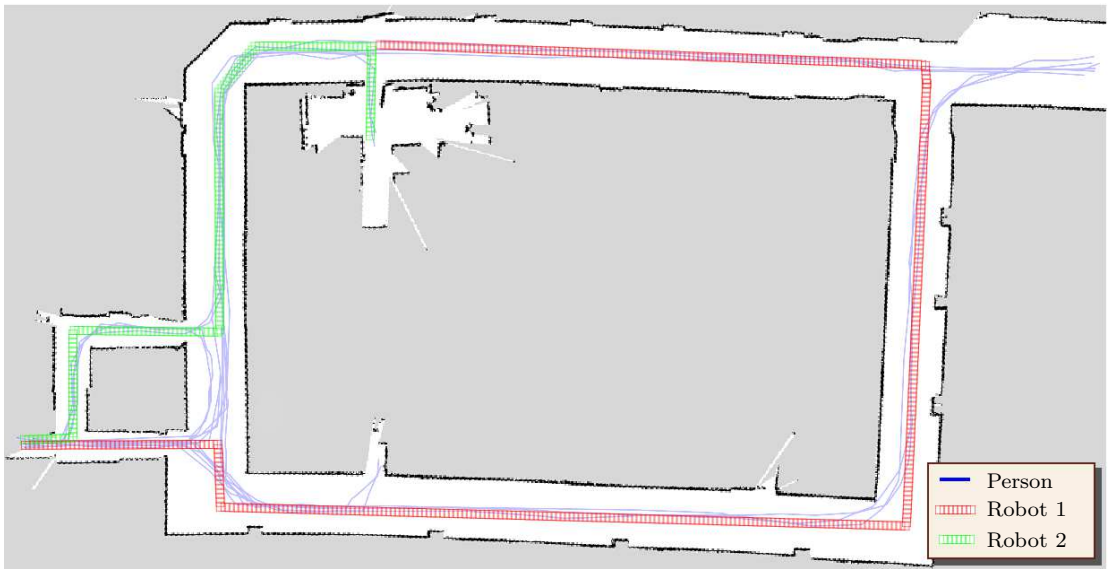


Figure 6.14: The optimal two-robot strategy for the STUDENT trajectory dataset that collects maximum expected reward for an initial belief centered at vertex v_9 from Fig. 6.8.3.

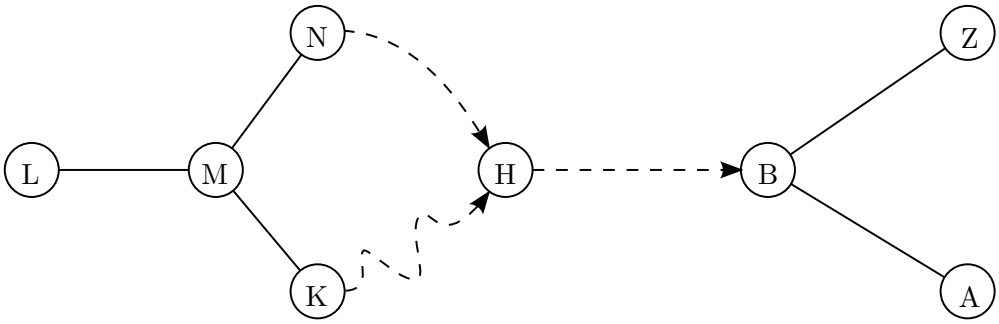


Figure 6.15: The subgraph $G(\Gamma^*)$ induced by the optimal two-robot policy shown in Fig. 6.14. The kink denotes that path K–H is longer than N–H.

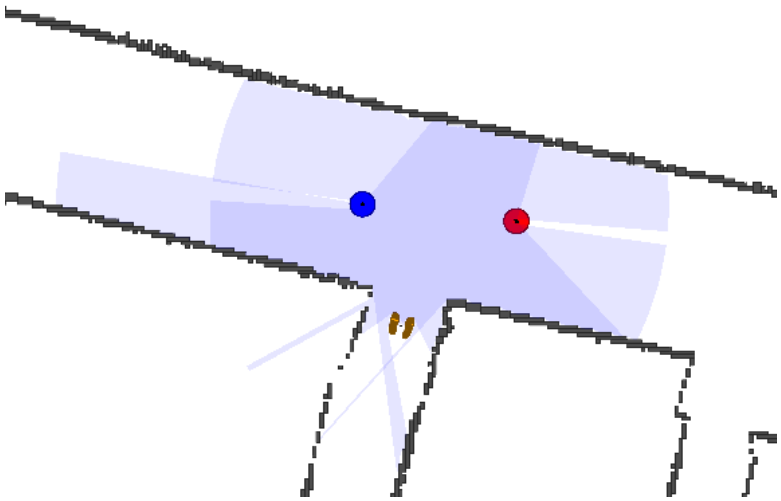


Figure 6.16: Simulation snapshot: Robots anticipate the person's future locations and move before he does

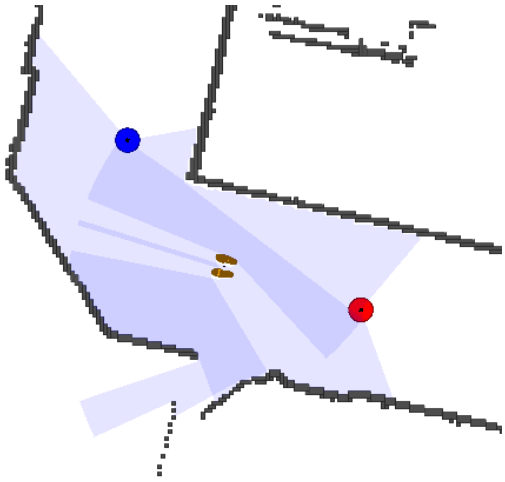


Figure 6.17: Simulation snapshot: Red robot follows the person while the blue robot leads

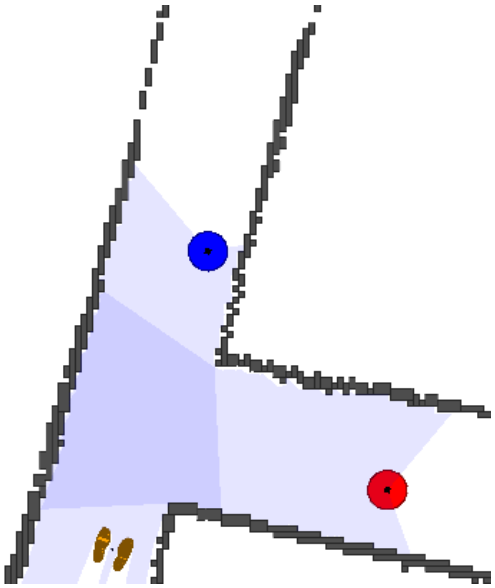


Figure 6.18: Simulation snapshot: Two robots guard a branch as the person approaches an intersection

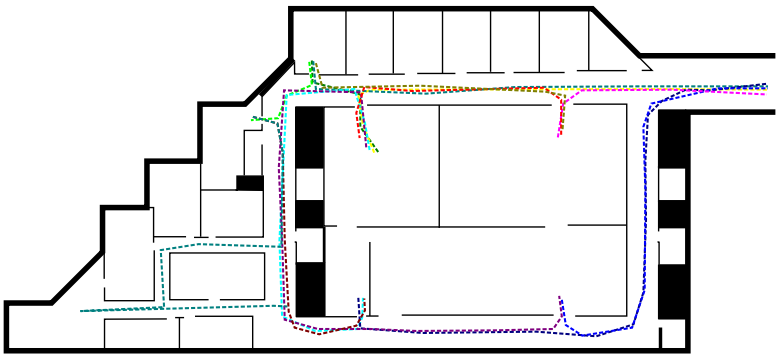


Figure 6.19: The STUDENT trajectory dataset: Traces of paths (colored, dashed) frequented by students

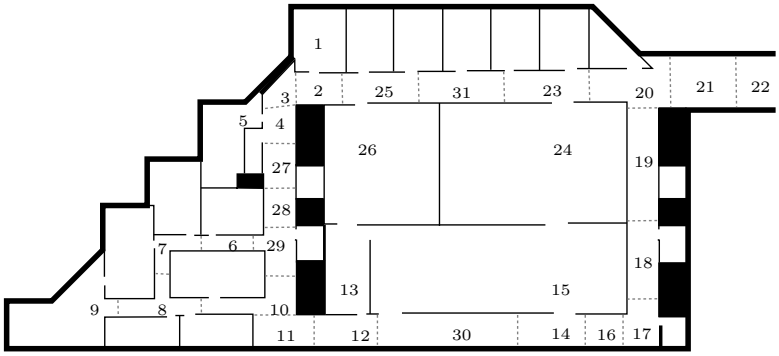


Figure 6.20: The STUDENT trajectory dataset: Labeled map denoting regions where a minimal graph representation could place its vertices

Chapter 7

Camera Placement for Novel View Synthesis

Recent technological advances have made real-time 3-D vision sensors [99, 100, 101], and, compact wheeled robots [102] suitable for everyday use in domestic environments. We envision that cameras with controllable mobile platforms will become increasingly common in the near future, with applications to telepresence and remote meetings, healthcare monitoring, interactive television, and, enhanced distance learning. Due to the diverse nature of these applications, it is important to address the challenges arising from both task-specific vision objectives, and, camera mobility planning, independently.

Early studies in the computer vision community dealing with non-stationary cameras focused on utilizing mobility to actively survey objects in a scene [103, 104]. These methods share a tight coupling between task-specific vision objectives (e.g., obtaining information about occluded parts of an object, and, keeping track of key features in an image), and, sensor planning (resp., next-view planning, and, visual servoing). Although they often lead to closed-form control laws, these approaches are difficult to generalize – altering the underlying vision objective triggers a redesign of the sensor planning algorithm.

In this paper, we propose planar geometry as an interface to abstract vision objectives from camera planning. We present a geometric measure that can be tuned with minimal effort to reflect the trade-off between the coverage of and object by different

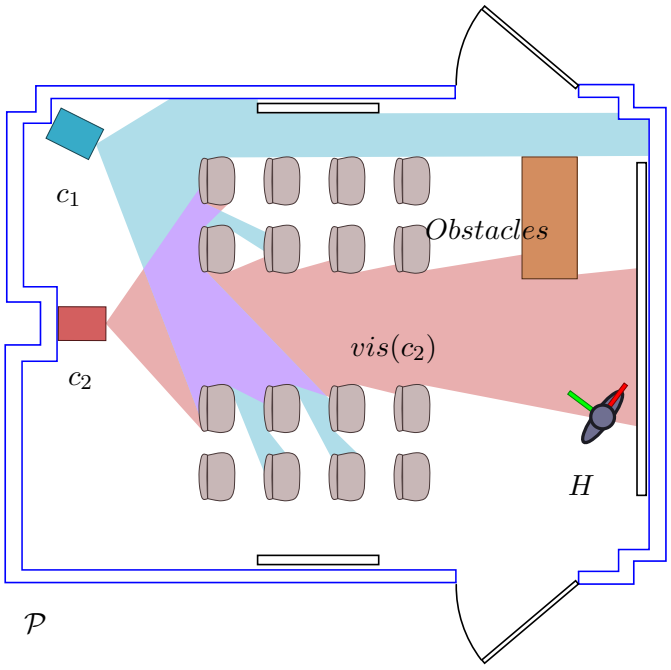


Figure 7.1: A structured planar representation of a theater class, with cameras (c_1, c_2), obstacles, and a performer H. Shaded areas denote visibility polygons within the camera field-of-view.

cameras, and the overlap of object features across them.

To ground our work in an application, we present a case study of a distance learning experiment in which remote students can interactively request for novel views of an educator. Figure 7.1 shows a conventional deployment of static cameras in a studio-style theater class. Instead, a performance in this scenario recorded by moving cameras could later be interactively viewed by a distance learning student from novel viewpoints that he/she deems interesting. This application is especially relevant to the U.S. public school system, which saw a recent growth in the adoption of videoconferencing in the classroom, totaling 31% of all schools as of 2009 [105].

In the following section, we discuss various techniques used to address the novel view synthesis problem. We present our geometric measure and compare it to image-based quality metrics via experiments in Section 7.3. We ground our abstract objective function to a distance learning application in Section 7.4, and show optimal camera

placements in response to a moving person in an indoor environment.

7.1 The novel view synthesis task

The novel view synthesis problem involves generating a view of a scene from a viewpoint where cameras are not physically present. The idea of view synthesis was originally inspired by low-bandwidth video communication – it was more efficient to capture and transmit a few reference views of a scene, rather than transmit all of them.

7.1.1 Image-based view synthesis

Early research in the computer vision and computer graphics communities on synthesizing novel views from reference views emerged from Image-Based Rendering (IBR) techniques such as mosaicking, interpolation, and warping [106]. When a novel view is interpolated from two or more reference images, e.g. [107, 108], only views lying in between the reference images are physically valid for synthesis. Stereo-vision has also been used for synthesis of novel views [109, 110].

In general, since image-based view-synthesis methods do not assume any knowledge of 3-D scene geometry, feature correspondences across multiple views are required. When sensors are mobile, this requirement imposes a constraint on the relative geometry of any two sensors used for view-synthesis. As illustrated in Figure 7.2 (left), different views of the scene need to have sufficient overlap.

7.1.2 Model-based view synthesis

On the other end of the spectrum are methods that assume knowledge about explicit 3-D geometry of objects in the scene. In this paper, we derive motivation from applications for indoor environments, where people are the objects of interest. Recent work in this direction has focused on markerless human pose extraction [111], and, free-viewpoint video [112].

Model-based view synthesis methods are not without their assumptions. They require the acquisition of a good 3-D model of the object (person), and in each frame, the image from each camera needs to be registered to the model. The registration process makes an implicit assumption regarding a sufficient part of the person being

visible in the scene. This is evident from the large static camera rigs used in both [112] and [111]. When there are a few mobile sensors available instead, their mobility can be exploited to obtain information regarding occluded parts of the person by dispersing them throughout the scene.

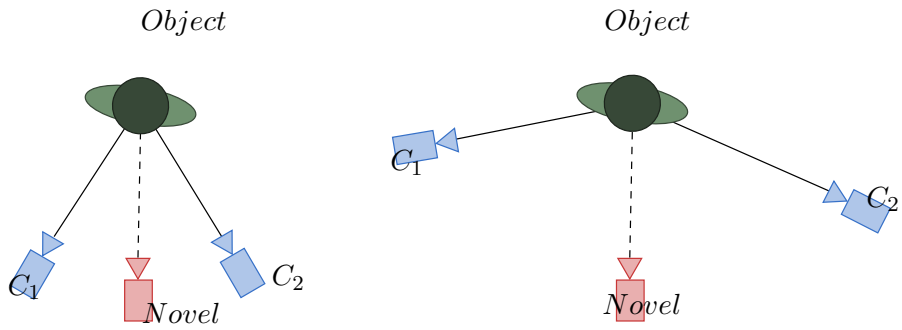


Figure 7.2: Image-based view synthesis methods require sufficient overlap of the object across camera views (left), whereas model-based view synthesis methods require coverage of the object to improve registration (right).

Model-based synthesis methods use sensors to *cover* unseen parts of the object (Figure 7.2, right), a requirement that is tangential to image-based approaches where sensor views need sufficient *overlap* (Figure 7.2, left). Although the choice of whether to use image-based synthesis, or, 3-D model-based synthesis depends on the exact application, in this paper we propose that the trade-off between coverage and overlap constraints of both methods be encapsulated into the sensor planning process using planar geometry as an abstraction.

7.2 Preliminaries

Before we present our geometric objective, we define the following preliminaries that rely on planar geometry.

Environment representation. We assume that a floorplan of the environment is known, although in practice, the map can be acquired on-line using a laser scanner. We represent it by a 2-D polygon $\mathcal{P}$, with polygonal holes for obstacles. Figure 7.1 depicts a sample environment.

Sensor pose. There is a set $\mathcal{C}$ of controllable cameras within $\mathcal{P}$. We assume that each camera can localize itself to a common world frame by matching sensory input to the known map of $\mathcal{P}$. Each camera is represented by its oriented pose in the ground plane, $c_i = (x_1, x_2, \theta)$, and a set of camera parameters $\gamma_i = (g_1, g_2, \ldots)$. We make this distinction explicit to indicate the number of degrees of freedom in our system — γ_i consists of parameters that are constant[1] , whereas c_i is controllable. However, no generality is lost by combining them into a single vector, e.g., if pan-tilt-zoom (PTZ) cameras are made available, one could move some of the parameters from γ_i to c_i.

Field-of-View and Visibility. Given a camera pose $c_i = (x_1, x_2, \theta)$, the field-of-view $FOV(c_i)$ is defined as the set of all points bounded by the viewing frustum of camera c_i projected onto the interior of $\mathcal{P}$. Similarly, the visibility polygon of c_i defines the set of all points in $\mathcal{P}$ that are visible from c_i.

$$Vis(c_i) = \{p \in \mathcal{P} \mid c_i p \text{ lies completely inside } \mathcal{P}\}$$

For limited field-of-view cameras, we restrict the visibility polygon to its field-of-view.

$$V(c_i) = Vis(c_i) \cap FOV(c_i) \tag{7.1}$$

A part-based object model. In Section 7.1, we motivated the separation between overlap-based objectives, and coverage-based ones. We use a planar

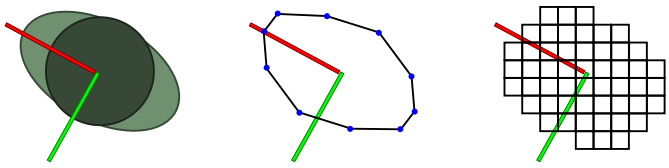


Figure 7.3: Planar views of different models of a person anchored at a common reference frame: ellipsoids (left), points and segments (center), and volumetric grid (right).

point-based model of the person (Figure 7.3, center) as a geometric medium to count the contribution of each camera c_i toward the novel view. In case line segments are used, they can be re-sampled to yield a point set. We define the person model to be $\mathcal{M} = \{s_1, s_2, \ldots, s_m\}$, where m is the number of parts in the model.

[1] Typically, γ_i includes the one-time camera calibration parameters.

With these preliminaries in place, we proceed to define set operations over the parts of $\mathcal{M}$, which allows us to combine relevant information regarding the object across multiple cameras.

7.3 A geometric objective for view synthesis

We use a set theoretic formulation to abstract geometric operations performed on the plane as follows. Any subset M of model parts can be represented by an m-dimensional binary vector $[b_1, \ldots, b_m]$ in which the 1's correspond to parts of model $\mathcal{M}$ that are included in M.

The model $\mathcal{M}$ can be aligned to an object's pose in the scene via a 2-D translation and rotation. Denote this transformation that encodes the object's pose as $h(\cdot)$. We define the parts of $\mathcal{M}$ visible from any camera c_i using (7.1) as follows.

$$parts(c_i, h) = [b_1, b_2, \ldots, b_m] \tag{7.2}$$
$$\text{where, } b_j = \begin{cases} 1 & ; h(s_j) \in V(c_i) \\ 0 & ; \text{ otherwise.} \end{cases}$$

For any subset $C \subseteq \mathcal{C}$ of cameras placed in $\mathcal{P}$, we can compute the parts of $\mathcal{M}$ seen from at least one camera, and, the parts of $\mathcal{M}$ seen from multiple cameras using logical-or and logical-and operations.

$$\bigcup_{c_i \in \mathcal{C}} parts(c_i, h); \qquad \bigcap_{c_i \in C} parts(c_i, h)$$

The number of parts of $\mathcal{M}$ seen from at least one camera can be counted using the

Inclusion-exclusion principle from measure theory (*cf.* [113]).

$$\sum_b \left(\bigcup_{c_i \in \mathcal{C}} parts(c_i, h) \right) =$$
$$\sum_i \sum_b parts(c_i, h) - \sum_{i<j} \sum_b \left(parts(c_i, h) \cap parts(c_j, h) \right)$$
$$+ \ldots + (-1)^{n+1} \sum_b \left(\bigcap_{c_i \in \mathcal{C}} parts(c_i, h) \right) \tag{7.3}$$

where, $\sum_b$ denotes sum over the binary vector.

Although counting the number of parts visible from sensors is a reasonable objective for certain applications, in general one could assign different scores to different parts, depending on the quality metric used in the underlying vision task. We define a generalization of (7.3) to a weighted scoring measure as follows.

$$score \left(\bigcup_{c_i \in \mathcal{C}} parts(c_i, h) \right) =$$
$$\sum_i score \left(parts(c_i, h) \right)$$
$$- \beta \sum_{i<j} score \left(parts(c_i, h) \cap parts(c_j, h) \right)$$
$$+ \ldots + (-1)^{n+1} \beta \cdot score \left(\bigcap_{c_i \in \mathcal{C}} parts(c_i, h) \right) \tag{7.4}$$

where, $\beta \in [0, 1]$.

The parameter β is a constant that scales each of the intersection terms relative to the union terms. To understand the intuition behind β, it helps to look at the extreme cases. When $\beta = 0$, the cameras are treated independently, with all of their scores summed. When $\beta = 1$, each part is scored only once, even if it covered by multiple cameras – this is a suitable value for maximizing the coverage of various parts of the object by different cameras. For applications such as stereo-vision and image-based view synthesis that require the relative baseline between cameras to be small, i.e. maximizing their overlap, a value of $\beta < 1$ should be selected so that parts visible in multiple cameras are scored higher than those visible from just one.

Comparison with image-based quality metrics

Equation 7.4 defines a geometric measure over different camera placements in the scene, using the concepts of planar visibility, coverage, and, overlap. In this section, we demonstrate via experiments that a camera placement $\mathcal{C}^*$ that optimizes our objective (7.4) also improves the quality of the underlying view synthesis task, suggesting that a simpler, more geometrically intuitive objective can be used in place of complicated image view quality metrics.

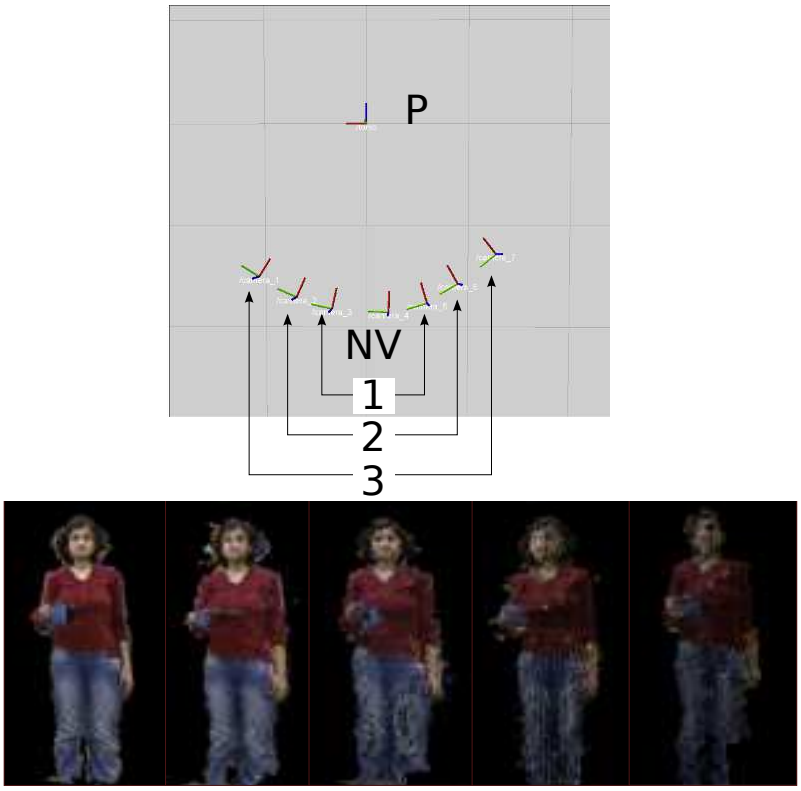


Figure 7.4: Experimental setup to compare synthesized views of a person P with ground truth NV – each number denotes the pair of camera locations corresponding to the same trial.

We obtained ground truth images of a novel view of a person, c_{novel}, by physically placing a camera at its location and acquiring an image I^*, as shown in Figure 7.4.

Simultaneously, we synthesized the same novel view of the person from two cameras c_1 and c_2 placed elsewhere, and compared the synthesized view I_{synth} to I^* over multiple trials.

By using the Microsoft Kinect [99] as our sensor, we had access to real-time 3-D data of the scene. As a result, we implemented an approach similar to 3-D view synthesis methods [112, 111]. We registered 3-D points from the two views to each other using a variant of the iterative closest-point algorithm. Subsequently, we projected the aligned 3-D points to the novel camera image plane via a pinhole camera model. We found that the point cloud registration performance improved drastically as the overlap between the cameras was increased – an expected observation, since registration requires a sufficient number of common points ("inliers"). We incorporated this observation into our quality function simply by setting $\beta = 0.5$ in (7.4), which favors overlap over coverage.

Since there does not exist a canonical measure for image similarity, in this paper we consider two statistical metrics. (i) MSSIM - a mean structural similarity measure commonly used in video encoding based on block-matching, and, (ii) KL - the Kullback-Liebler divergence between pixel intensity distributions. While MSSIM and KL are in no way conclusive of image similarity, we believe that they complement each other – MSSIM captures the local structure of an image, whereas, KL captures the global intensities of its pixels. Figure 7.5 compares our objective to these quality metrics for each trial.

- $Q_{0.5}$: our geometric objective (7.4) with $score(\cdot)$ defined as a normalized counting function, and $\beta = 0.5$,
- $Q_{mssim} = MSSIM(I^*, I_{synth})$, and,
- $Q_{kl} = KL(I^*, I_{synth})$.

The synthesized views corresponding to this experiment are shown in Figure 7.4 (bottom). In all of the trials, the structure of the person in the novel view image was preserved due to the alignment phase in our view synthesis technique. This is echoed by the MSSIM values in Figure 7.5 being consistently good (above 90%). Upon closer inspection, the images on the right side of Figure 7.4 (bottom) exhibit various artifacts, e.g. holes and noisy pixels, due to the oblique projection angles. This undesirable feature is captured by the decreasing values of the KL measure in Figure 7.5.

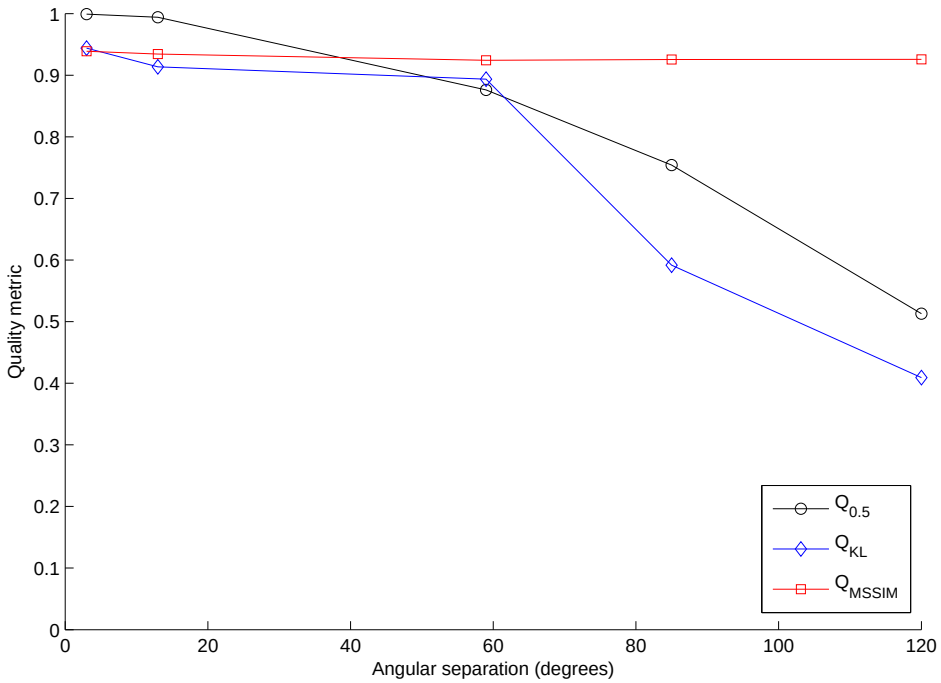


Figure 7.5: A comparison of view synthesis quality metrics for the experiment shown in Figure 7.4. Metrics based on both geometry ($Q_{0.5}$), and, image statistics (Q_{mssim} and Q_{kl}) are shown.

Our geometric objective function $Q_{0.5}$ follows the same trend as Q_{kl}, demonstrating that the $score(\cdot)$ function, and β, can be chosen in an application-specific way to reflect the performance of complicated image-based view quality metrics.

7.4 Dynamic camera placement

To ground our work in an application, we derive motivation from the distance learning scenario introduced in this chapter. In traditional media-enhanced classrooms, a static camera is used to record the entire discourse, with either frequent switching between views, or montages. While this design caters to the basic requirement of archiving the content of the lecture, it is not suitable for generalization to interactive settings, such

as video-walkthroughs, theater performances, and, art studio classes, where interesting viewpoints may change over time. The object of interest in these scenarios

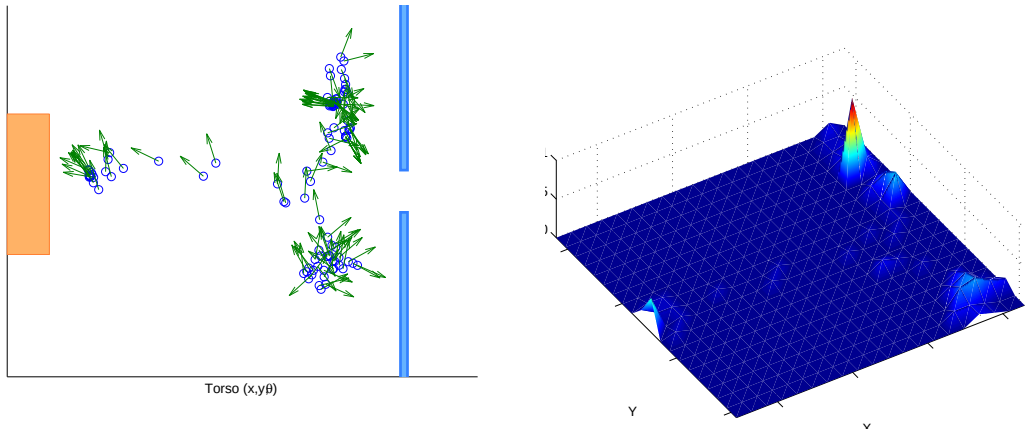


Figure 7.6: Planar pose of an educator's torso in a classroom (left), and, his density distribution in the same space (right).

is a person (an educator, an actor, or an artist). His mobility alters the best placement of cameras in the scene, rendering art-gallery-type solutions and its variants [114] unsuitable. Figure 7.6 shows the locations and orientations of an educator's torso in a demonstration-based class with multiple teaching media – two whiteboards, and, an instrument table.

The notion of time. On the one hand, we can model the sensor planning problem as a responsive system that changes the camera placement rapidly to incorporate the person's mobility. But this is undesirable because resulting videos would be continuously moving, and/or jerky. On the other hand, we can mimic a surveillance system by assuming that the person's density distribution is static over all time, e.g. as in [115]. However, this results in a static placement of sensors which in general can be suboptimal over any finite sub-interval of time. In this case study, we assume that the person's density is static over a fixed time horizon, but changes across consecutive

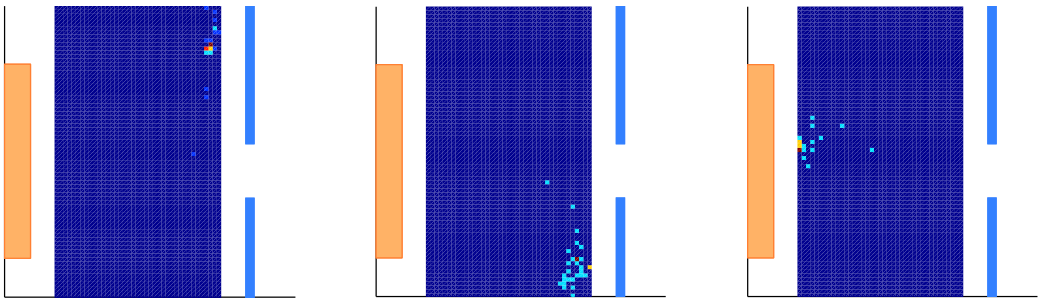


Figure 7.7: The person's mobility as a density distribution spread over three subintervals.

intervals. This is illustrated in Figure 7.7, in which the person's distribution from Figure 7.6 (right) is subdivided into three horizons, with end times $t = T_1, t = T_2$, and $t = T_3$.

A placement to maximize expected quality. Over a fixed time horizon, the static density of the person can be discretized into q cells $[h_1, h_2, ..., h_q]$, each with probability $[p_1, p_2, ..., p_q]$. A camera placement is selected that maximizes our geometric synthesis measure $Q_{0.5}$ in expectation over the distribution $\{h_j, p_j\}$.

$$\arg\max_{c_i} \left\{ \sum_j p_j \cdot score \left(\bigcup_{c_i \in C} parts(c_i, h_j) \right) \right\} \tag{7.5}$$

Solutions for optimal two-sensor placements in this case study are shown in Figure 7.9. The circles represent the person's locations, and the rectangle in the center is an obstacle (the instrument table). The person's locations are assumed stationary over three time horizons $[0, T_1]$ (whiteboard 1), $(T_1, T_2]$ (whiteboard 2), and, $(T_2, T_3]$ (demonstration at the instrument table). The first row shows the solution for static camera placement obtained after averaging the person density over the entire time interval $[0, T_3]$, when cameras are restricted to the far wall of the room (left), and, when they are unrestricted (right). In both cases, the novel view was selected to match views acquired in traditional media-enhanced classrooms. In the unrestricted case, observe that one of the cameras moves closer to the person during $[0, T_2]$ while the other accounts for $(T_2, T_3]$. While this might be optimal in a surveillance scenario, it is

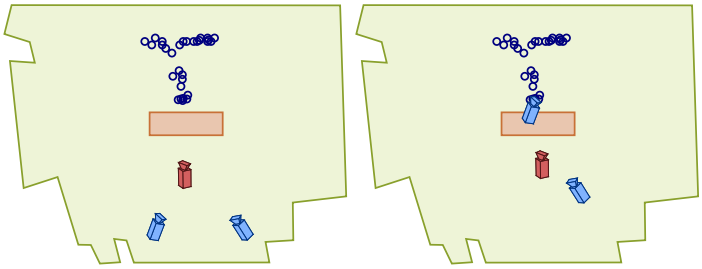


Figure 7.8: Optimal sensor placement (blue cameras) for a static person density distribution. Novel viewpoints are shown as red cameras.

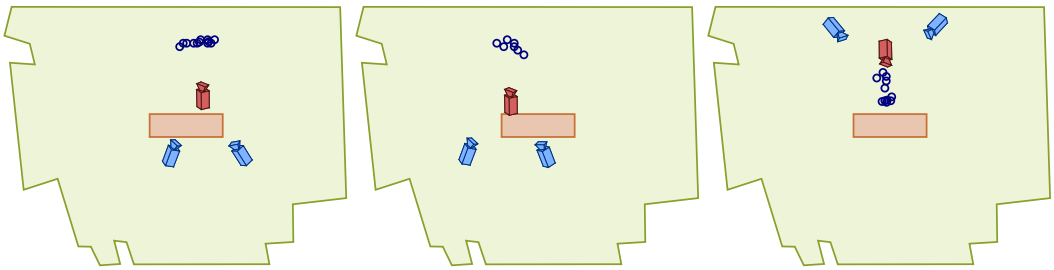


Figure 7.9: Optimal sensor placement (blue cameras) for a fine time resolution which shows the person's mobility (bottom row). Novel viewpoints are shown as red cameras.

undesirable here because it interferes with the person.

In the bottom row of Figure 7.9, camera placements are shown when the complete duration is divided into three parts as mentioned previously. This formulation allows for more interactivity because the novel view can be changed during each sub-interval, e.g. during $(T_2, T_3]$, a novel view of the educator is requested as he turns away from the table.

7.5 Conclusions

In this paper, we derive motivation from video-based distance learning applications, and, interactive video walkthroughs. In both of these scenarios, novel views of a recorded scene are requested by a remote user. Existing view synthesis techniques make implicit

assumptions regarding both the relative 3-D geometry of the cameras, and, the amount of information regarding an object in the scene viewed by multiple cameras. Image-based synthesis techniques in the literature require feature correspondences of an object across different views. In contrast, 3-D model-based techniques benefit from observing the object from mutually distinct viewpoints to improve registration.

In this paper, we use planar geometry and a set-theory-based discrete object model $\mathcal{M}$ as a medium to encapsulate both of these assumptions. We present a geometric quality measure that assigns a value to any given placement of camera sensors in the plane. We generalize this measure using a task-dependent $score(\cdot)$ function, and a parameter β that determines the impact of camera overlap and coverage on the task quality.

We demonstrate via experiments that a camera placement obtained by optimizing our geometric measure also improves the quality of synthesized views of a person. We propose that our geometric objective function be used in place of complicated image-based quality metrics when deciding where to place cameras in a scene.

We ground our work in a case study of a distance-learning application in which cameras are dynamically placed in response to a moving person. We model the person's mobility as a density distribution that is constant over a fixed time horizon, but changes across consecutive intervals. The division of time has the added advantage that the requested novel view can be interactively changed in each horizon. In our application, this has the potential to create a more engaging learning experience, since students can observe the scene from viewpoints that they deem interesting.

The abstraction defined in our geometric measure allows us to change the task-specific vision objectives independently of the sensor planning and placement process. As part of immediate future work, we will investigate more efficient ways to solve the sensor planning problem as defined in (7.5). More specifically, if we define $\mathcal{M}$ to be the set of all locations in the plane instead of just the person, (7.4) is reminiscent of the set-cover problem. It permits provable approximation algorithms in the event that the $score(\cdot)$ function satisfies certain properties. This will help scale our approach to large networks of cameras.

Chapter 8

System design and implementation

In this chapter, we address some of the design decisions that went into our robotic system in terms of both hardware and software design. We also bring to light the implementation challenges we met and the steps we took to address them. It isn't meant to be a software development document or an API specification of any sort, but more an account of our experience with building, integrating and using a complex system.

8.1 Scope

This chapter provides a guide not only to the complete system we designed for indoor target-tracking applications, but also the software and utilities we developed along the way to make the results in this thesis possible. It describes:

- Hardware chosen: Sensors, Robots and Computing.
- Software developed for our reactive planner from Chapters 4 and 5
- Software developed for our anticipatory planning algorithm from Chapter 6
- Software developed for on-line plan execution and robot control
- Software developed for simulations and evaluation of robot plans

- Software developed for view synthesis in the context of Chapter 7

8.2 Overall architecture for mobile tele-conferencing

The system architecture for our proposed multi-robot target-tracking system is shown in Figure 8.1.

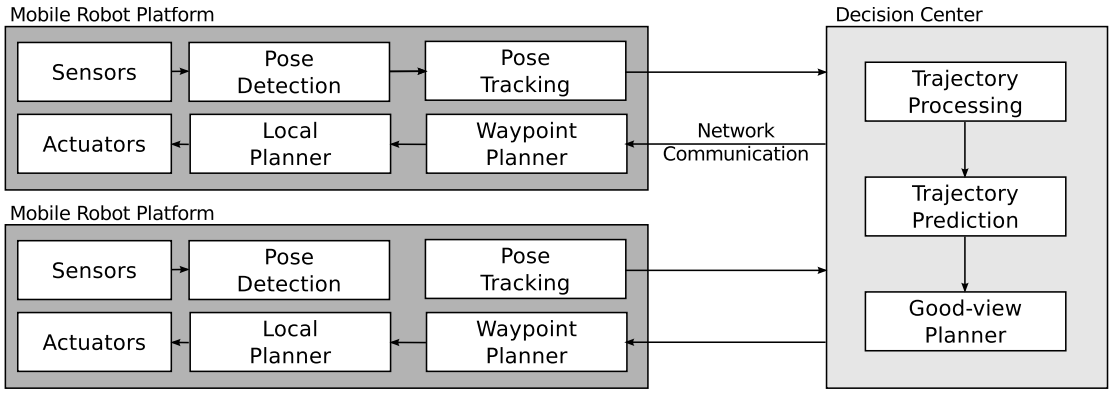


Figure 8.1: Architecture of our autonomous target-tracking system with multiple robots (dark-shaded boxes) and a predictive planner (light-shaded box). The decision center is a separate computer with a dual connection: ad-hoc wifi (robots) and the Internet (remote user).

The process starts with a sensing phase in which the sensors on board each robot are used in two primary ways – to track a person (Kinect sensor) and, to localize the robots in the environment (odometry and laser). It is assumed that a map of the environment is available for localization. In our system, we build this map *a priori* using a Simultaneous Localization and Mapping (SLAM) implementation from the WillowGarage Robot Operating System (ROS) `gmapping` library. As the person walks, his frame of reference is tracked in real-time (∼25Hz) on board each robot. A linear-velocity Kalman filter is used to smooth small motions. The resulting mean trajectory estimate $\vec{h}(t) = [x, y]$ is communicated to a decision center over a wireless channel. There, it is converted into a discrete representation based on an environment graph G. Details are presented in Section 6.4.

Next is the planning phase at the decision center. Based on the observation history

of the person, a prediction is made regarding his locations at time T into the future – the planning horizon. The uncertainty in the person's future locations is encapsulated into a probabilistic belief state $b \in B$. Robot policies of the form $\Gamma_j : V \times B \longrightarrow V$ are then queried to determine robot navigation goals. In practice, optimal multi-robot joint policies Γ are computed offline based on datasets and stored for efficient look-up. An overview of our planner objective was presented in Section 6.3.

The navigation goals are then communicated over the same wireless channel back to each robot for the actuation phase. The local planner determines the control inputs based on a kinematic model and drives the robot to its goal while avoiding obstacles. A brief discussion on suitable local planners was presented in Section 6.7.

The sense-plan-actuate cycle is repeated for the full duration of the service. Multiple robots are necessary because there are non-zero probabilities associated with person trajectories that he is less likely to take. In the event that an anticipatory plan does not match the actual navigation decision of the person during policy execution, the other robots are used to both keep track of the person, and to guard future navigation decisions.

8.2.1 Hardware overview

The primary goal in selecting system hardware was to keep the overall price as low as possible while not compromising on the system goals of sensing, actuation and processing. To this end, we chose off-the-shelf components and assembled the mobile platform shown in Figure 8.2.

Our mobile platform uses the iRobot Create as a wheeled base [102]. We found that robot to be lightweight, affordable and easy to use via the serial interface. An Asus EEE PC (netbook) was the source of computing power, mounted over the base on a custom polymer shelf designed by my colleague Ahmet Onur Tekdaş. The model was P900, with an Intel Celeron M processor (900MHz) and 1GB RAM. We found these netbooks to work well with the vision-based person tracking implementation from Chapter 4 because we used a simple pair of Logitech webcameras as a stereo rig. However, we found that running the various software components to support the Microsoft Kinect sensor caused complete CPU usage, consequently draining its battery life much sooner. The camera sensor was mounted atop a tripod stand which was also attached to the base. In our

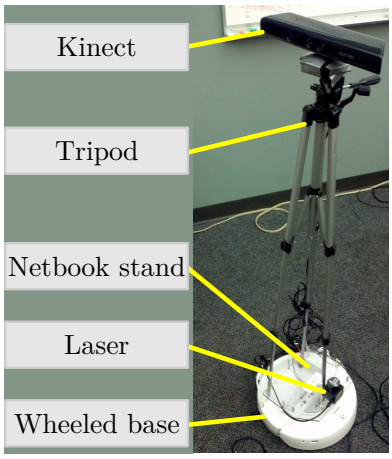


Figure 8.2: Our mobile robotic platform consisting of an iRobot Create mobile base with a mounting stand for a netbook, a Hokuyo URG laser scanner, and a Kinect sensor mounted on a tripod.

experiments, a camera height of about 4.5 feet worked well in practice, neither giving too low a view of the person, nor too high a view.

Hardware component	Approximate Cost (USD)
Tripod stand	20
Materials	50
Kinect sensor	150
iRobot Create	130
Netbook	450
Hokuyo URG laser	1200
Total	2000

Table 8.1: Cost of our custom-built mobile platform

Table 8.1 shows the approximate cost of purchasing components to build our system, totalling to USD 2000 per mobile platform. This is at least an order of magnitude lesser than commercial robots, making it accessible to home users. It should be noted that in indoor environments that are GPS-denied, we need a reliable localization system which in our implementation is laser-based. By using an alternate localization scheme, such as the Kinect for 3D SLAM, it is possible to bring down the cost at the expense of

designing and implementing a more complicated algorithm.

8.2.2 Software overview

All of the software on our system was written in the C++ language and relies on the WillowGarage ROS library for inter-process communication and network transparency via TCP/IP sockets over a wireless ad-hoc network.

The following software modules are labeled with superscripts denoting where they are executed: R – online on each robot, C – online at the decision center, O – offline.

- `person_tracker`^{R}
  From Kinect to history of torso poses; Smooth effects from small local movements
- `person_trajectory_processor`^{R,C}
  Graph-based coarsening from (t,x,y) to (t,v); From (t,v) to splines allows interpolation for use with local planner
- `trie_global_planner`^{C}
  Our predictive planner that performs POMDP policy execution for a cached offline optimal multi-robot policy
- `move_base_wmr`^{R}, `move_base_smooth`^{R}
  Two approaches to local planning: converts graph-based plans to control inputs
- `map_utils`^{O}
  Conversion between image-based maps and graph-based maps; Interface with Stage simulator for manual map-building; Interface with CGAL for straight-skeleton diagram; Interface with OpenCV for fast-marching-based medial axis graph
- `pomdp_utils`^{O}
  Conversion of dataset input into a suitable file format for the SARSOP POMDP solver; Conversion of PolicyX solution format back into graph-based plans for policy execution.

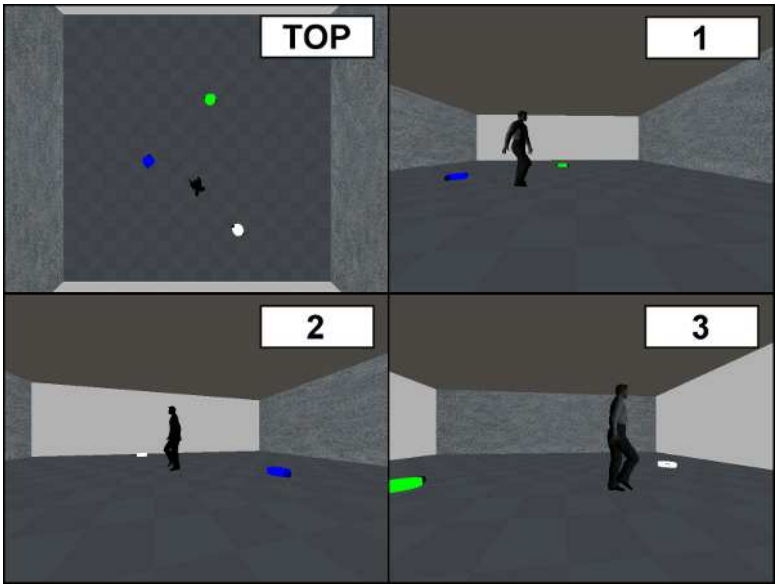


Figure 8.3: A simulator (C++/OpenGL/lib3ds) to render camera views as seen by individual robots 1, 2 and 3, and a top view.

8.3 Simulator: Good-view visualization

In Chapter 5, we described a reactive planner in which the robot that acquires a good view of the person could initially lose view due to smoothness and non-holonomic constraints. In order to visualize how the camera view from the best view robot looks, we developed a 3D simulator using C++ and OpenGL. The "businessman" 3D mesh and model were obtained from the Artist3D model database [76] in 3D Studio Max 3DS format. A 3DS loading routine was used to load, place and render a model of the person in the scene. By shifting the viewpoint to the robot location and rendering the scene, we were able to simulate the functioning of a camera, albeit without noise.

8.4 User interface: Trajectory data collection

In order to collect trajectory preferences from users, we designed a Java-based user interface primarily driven by mouse input, as shown in Figure 8.4.

The interface presents the user with the floorplan of the desired area. The floorplan image was partitioned into sub-polygons by a Voronoi labelling scheme in which the

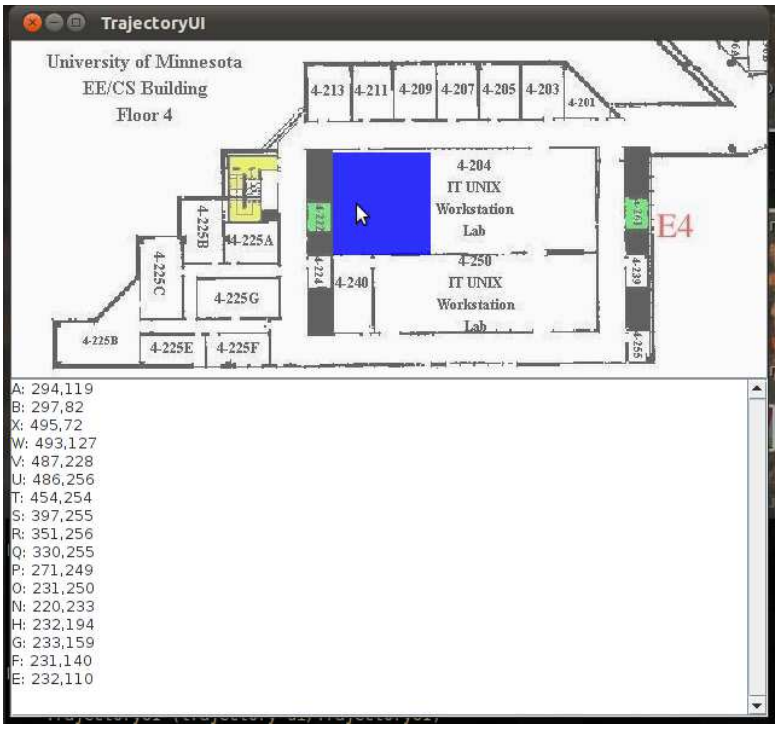


Figure 8.4: Java-based user interface to collect user trajectories indicated by mouse clicks

Voronoi centers correspond to the medial axis graph vertices. By hovering the mouse over different parts of the interface, we highlight the corresponding Voronoi cell (see blue filled box under mouse cursor in Figure 8.4). Clicking on a highlighted cell records the clicked location as visited. Consecutive clicks define a sequence of locations in the image that directly translate to (x, y) coordinates in the global reference frame of the floorplan. Thus trajectories can be specified just by a sequence of clicks. Timing information can be appended to each location in order to represent a time-stamped trajectory. These trajectories were used as the basis for the dataset in Section 6.8.3.

8.5 Code Sample: Graph utility for Voronoi labelling and visualization

```
1 #include <ros/ros.h>
```

```
#include <visualization_msgs/Marker.h>

// CGAL for Delaunay+Voronoi (nearest neighbor queries)
#include <CGAL/basic.h>

// standard includes
#include <iostream>
#include <fstream>
#include <cassert>
#include <string>
#include <sstream>

// includes for defining the Voronoi diagram adaptor
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/Delaunay_triangulation_2.h>
#include <CGAL/Voronoi_diagram_2.h>
#include <CGAL/Delaunay_triangulation_adaptation_traits_2.h>
#include <CGAL/Delaunay_triangulation_adaptation_policies_2.h>
#include <CGAL/Triangulation_vertex_base_with_info_2.h>
#include <CGAL/IO/Color.h>

#include <teleimmersion_motion_planning/VoronoiQuery.h>
#include <teleimmersion_motion_planning/graph_defines.h>

#include "Board.h"

namespace voronoi_node {
  const std::string NAME = "voronoi_node";

  struct VertexInfoCustom {
    VertexInfoCustom() {}
    int vertex_id;
  };

  // typedefs for defining the adaptor
  typedef CGAL::Exact_predicates_inexact_constructions_kernel
                        K;
  typedef CGAL::Triangulation_face_base_2<K>
        Fb;
```

```
39    typedef CGAL::Triangulation_vertex_base_with_info_2<VertexInfoCustom,K>
              Vb;
      typedef CGAL::Triangulation_data_structure_2<Vb,Fb>
             Tds;
41    typedef CGAL::Delaunay_triangulation_2<K,Tds>
                                       DT;
      typedef CGAL::Delaunay_triangulation_adaptation_traits_2<DT>
                         AT;
43
      // typedef for the result type of the point location
45    typedef DT::Vertex_handle       Vertex_handle;
      typedef DT::Point           Point;
47    typedef DT::Edge_iterator       Edge_iterator;
      typedef DT::Segment           Segment;
49
      // ROS-related
51    std::string param_srvtopic, param_graph_file;
      ros::ServiceServer srv_closest_node;
53    ros::Publisher pub_markers;

55    // BGL-related
      teleimmersion::graph_t G;
57    teleimmersion::vertex_label_accessor_t G_labels;
      teleimmersion::vertex_x_accessor_t G_x;
59    teleimmersion::vertex_y_accessor_t G_y;
      teleimmersion::vertex_pos_accessor_t G_pos;
61
      // CGAL-related
63    DT T;
      bool show_triangulation;
65    void load_graph_file(std::string fname);
      void spin(ros::NodeHandle &nh);
67    void cleanup();
      bool callback_closest_node(teleimmersion_motion_planning::VoronoiQuery
          ::Request &request, teleimmersion_motion_planning::VoronoiQuery::
          Response &response);
69    bool saved;
      void draw_triangulation();
71  };
```

```
73 bool voronoi_node::callback_closest_node(teleimmersion_motion_planning::
       VoronoiQuery::Request &request, teleimmersion_motion_planning::
       VoronoiQuery::Response &response) {
     Point q(request.x, request.y);
75   Vertex_handle vh = T.nearest_vertex(q);
     response.vertex_label = G_labels[vh->info().vertex_id];
77   response.vertex_x = vh->point().x();
     response.vertex_y = vh->point().y();
79   return true;
   }
81
   void voronoi_node::load_graph_file(std::string fname) {
83   std::ifstream gin( fname.c_str() );
     if(!gin.is_open()) {
85     ROS_FATAL_STREAM("Unable to open required graph file [" << fname << "
           ]. Please provide correct full path.");
     }
87   boost::dynamic_properties dp(boost::ignore_other_properties);
     dp.property("node_id", boost::get(boost::vertex_name, G));
89   dp.property("label", boost::get(boost::vertex_label, G));
     dp.property("x", boost::get(boost::vertex_x, G));
91   dp.property("y", boost::get(boost::vertex_y, G));
     dp.property("pos", boost::get(boost::vertex_pos, G));
93   if(!boost::read_graphviz(gin, G, dp, "node_id")) {
       ROS_FATAL_STREAM("Unable to parse graph file [" << fname << "].
           Please ensure it is in the corrent format (graphviz dot) and has
           properties {label, pos, x, y} defined.");
95   }
     G_labels = boost::get(boost::vertex_label,G);
97   G_x = boost::get(boost::vertex_x,G);
     G_y = boost::get(boost::vertex_y,G);
99   G_pos = boost::get(boost::vertex_pos,G);
     gin.close();
101  // add points to voronoi diagram
     for(size_t i=0; i<boost::num_vertices(G); ++i) {
103    Point pt(G_x[i], G_y[i]);
       Vertex_handle vh = T.insert(pt);
105    vh->info().vertex_id = i;
```

```
  }
  assert( T.is_valid() );
  ROS_INFO_STREAM("Loaded " << boost::num_vertices(G) << " vertices from
      file [" << fname << "].");
  saved = false;
}

void voronoi_node::draw_triangulation() {
  using namespace LibBoard;
  visualization_msgs::Marker line_list;
  line_list.header.frame_id = "/map";
  line_list.header.stamp = ros::Time::now();
  line_list.ns = "VoronoiDiagram";
  line_list.action = visualization_msgs::Marker::ADD;
  line_list.pose.orientation.w = 1.0;
  line_list.id = 0;
  line_list.type = visualization_msgs::Marker::LINE_LIST;
  // line width is taken only from the x-scale
  line_list.scale.x = 0.1;
  line_list.color.r = 0.8;
  line_list.color.g = 0.8;
  line_list.color.b = 0.2;
  line_list.color.a = 1.0;
  // make a copy of this marker style
  visualization_msgs::Marker line_list_T = line_list;
  line_list_T.color.r = 0.2;
  line_list_T.color.g = 0.2;
  line_list_T.color.b = 0.8;
  // make it different from the previous one for rviz
  line_list_T.ns = "DelaunayTriangulation";
  line_list_T.id = 1;
  //SVG
  Board svg;
  if(!saved)
  {
    svg.setLineWidth(1.0);
  }
  Edge_iterator eit = T.finite_edges_begin(), end = T.finite_edges_end();
  for(;eit != end; ++eit) {
```

```
    // Delaunay segments
145 {
      const Segment &s = T.segment(eit);
147   geometry_msgs::Point p1, p2;
      p1.x = s.source().x();
149   p1.y = s.source().y();
      p2.x = s.target().x();
151   p2.y = s.target().y();
      line_list_T.points.push_back(p1);
153   line_list_T.points.push_back(p2);
      if(!saved)
155   {
        svg.setPenColor(Color::Blue);
157     svg.drawLine(p1.x,p1.y,p2.x,p2.y);
      }
159 }
    // Voronoi segments (from dual)
161 try {
      const Segment &s = CGAL::object_cast<Segment>( T.dual(eit) );
163   geometry_msgs::Point p1, p2;
      p1.x = s.source().x();
165   p1.y = s.source().y();
      p2.x = s.target().x();
167   p2.y = s.target().y();
      line_list.points.push_back(p1);
169   line_list.points.push_back(p2);
      if(!saved)
171   {
        svg.setPenColor(Color::Red);
173     svg.drawLine(p1.x,p1.y,p2.x,p2.y);
      }
175 } catch (CGAL::Bad_object_cast ex) {
      // ignore the Voronoi duals that are not segments
177 }
  }
179 pub_markers.publish(line_list_T);
  pub_markers.publish(line_list);
181 if(!saved)
  {
```

```
183     svg.scale(1.0);
        svg.saveSVG("/tmp/voronoi.svg");
185     saved = true;
      }
187 }

189 void voronoi_node::cleanup() {
    }
191
    void voronoi_node::spin(ros::NodeHandle &nh) {
193   ros::NodeHandle ns("~");
      ns.param<std::string>("srvtopic",param_srvtopic,"get_closest_graph_node
          ");
195   ns.param<std::string>("graph_file",param_graph_file,"undefined.dot");
      voronoi_node::load_graph_file(param_graph_file);
197   srv_closest_node = nh.advertiseService(param_srvtopic, voronoi_node::
          callback_closest_node);
      show_triangulation = true;
199   pub_markers = nh.advertise<visualization_msgs::Marker>("
          visualization_marker", 1);
      ros::Rate r(10);
201   while( nh.ok() ) {
        draw_triangulation();
203     ros::spinOnce();
        r.sleep();
205   }
      cleanup();
207 }

209 int main(int argc, char *argv[]) {
      ros::init(argc, argv, voronoi_node::NAME.c_str(), ros::init_options::
          AnonymousName);
211   ros::NodeHandle nh;
      voronoi_node::spin(nh);
213   return 0;
    }
```

8.6 Code availability

At this time, our software repository is hosted internally by the Robotics Sensor Networks research lab at the University of Minnesota. For access to the code, please contact the author of this thesis.

Chapter 9

Conclusions and Future work

The symbiosis between recent advances in robot manufacturing technologies and algorithm development has burgeoned the use of small, lightweight, robust, yet affordable robots in environments where they were previously absent. Figure 9.1 shows a few examples of robots used for a variety of indoor tasks such as vacuum-cleaning, home surveillance, and mobile video-conferencing.

Figure 9.1: Examples of robots designed for home use: (clockwise from top left) the iRobot Roomba, the Acroname Garcia compared to a coffee mug, the WooWee Rovio and the Nissan BR32C.

Semi-autonomous robots have excellent advantages in that they extend what people can do, but that is also a disadvantage – precious time is spent by a remote operator who has to control the robots. Taking the step from semi-automonomy toward full autonomy requires that robots be able to move about in the same spaces as people – an onus that falls on navigation and planning algorithms.

In the case of virtual presence, or remote presence, this autonomy is particularly useful, since remote users rather focus on either conversations or diagnoses, in the cases of virtual meetings and remote healthcare, respectively. It would be especially impractical to require that the remote user converse with someone while also keeping their eye on how to steer their robot.

In this thesis, we studied the target-tracking problem, in which one or more robots need to keep track of moving target in a complex environment. The applications listed above are variants of this problem differing in how the environment and target are modeled, and what tracking objectives are most suitable.

Modeling the behaviors of targets moving in an indoor environment is not an easy task. For instance, human navigation is a complex cognitive process, relying on factors that are not easy to formalize, such as perception of hazards, familiarity from experience and long-term memory.

9.1 Summary of contributions

In this thesis we discussed three different ways of modeling target mobility – adversarial, fully known and uncertainty-based. The adversarial target model requires the careful use of game theoretic techniques due to the strategic interactions between the players. For the case where the target's trajectory is fully known to the in advance, we solve the problem of determining suitable paths for multiple robots that follow the target's path as closely as possible, while paying attention to kinematic and smoothness constraints. In typical service applications, people navigate with intentions independent of any robots that might exist in the environment. The target is neither adversarial nor does it advertise its trajectories before taking them. We presented a data-driven method that incorporates the uncertainty in the navigation decisions of the target into the path planning process through a probabilistic motion model. In each case, we derived optimal

robot motion plans that were able to keep track of the target.

When considered as a whole, our studies address various aspects of a complete multi-robot remote presence system – from reasoning about what the robots know about the target, down to how a predetermined path can be executed by the robot's real-time control system. The details of our conclusions are as follows.

9.1.1 Adversarial target model

In the worst-case, the target is trying to escape from a robot, while the robot in turn tries to prevent that from happening. We modeled this scenario using non-cooperative game theory [28, 31], more specifically pursuit-evasion games. Resulting strategies from these methods have the following advantages.

(i) A guarantee as to which player wins,

(ii) Theoretical bounds for capture time, and

(iii) Worst-case strategies that work *no matter what* the other player does.

The traditional Lion-and-Man game is played in an obstacle-free circular arena with one lion tracking down a man [34]. In the differential game framework, it is not straightforward to incorporate environment geometry such as multiple polygonal obstacles. When a differential model is not imposed, related research from our group shows that three robots can capture the person in any simply-connected polygonal environment with multiple obstacles [23].

As a first step toward modeling more complex environments in the differential game framework, we formulated a continuous-space, continuous-time variant of the Lion-and-Man game in the presence of a single circular obstacle [22]. We presented a result in which it is not possible for the robot (lion) to always keep track of the person (man). However, it might be possible for the robot to prevent the person from reaching the obstacle in the first place. When this happens, the robot can keep the man within its sights and thereafter track him down. It turned out that the decision as to which player wins the game is completely determined by the initial locations of the robot and the person with respect to the obstacle. We presented a closed-form characterization of initial locations that completely determine the final outcome of the game. Our approach

was constructive in the sense that it yields a partitioning of the state space into pursuer-win and evader-win regions.

The theoretical bounds from these methods came at the cost of being difficult to solve due to a min-max objective criterion: best response actions for the robots are functions of what the target is capable of doing and vice-versa. When taken to an extreme, a target that is much quicker than the mobile robots could better be tracked by a static camera deployment that ends up being less expensive, in terms of energy, distance moved, and control effort. While this approach is suitable for surveillance tasks and military operations, robots that work alongside people to provide useful services do not need to make adversarial assumptions. By characterizing the mobility of the person with a motion model, we show that we can move away from worst-case strategies and present results that are not as conservative.

9.1.2 Known target trajectory

On the other extreme from assuming that the target is an adversary is having full knowledge of how the target moves. With this assumption, the robots need only move *after* the target decides its path.

This type of model is common in multi-target tracking [24] and person-following [25] problems. For instance, an indoor map of a museum could be constructed by a person guiding the robot through the rooms and annotating them with semantic information. In assistive living facilities, it might be in the caregivers' best interest to have robots accompany the elderly either for companionship or healthcare monitoring.

In these applications, the target's trajectory is known to the robots before their paths need to be planned. In the case of discrete space models, e.g. graph representations of the environment, a dynamic programming approach was used to determine suitable paths for robots [26]. The knowledge of target's trajectory also helps continuous space models because differential constraints such as robot non-holonomy and trajectory smoothness fit nicely into a control theoretic framework (Chapter 4). The optimal control problem of determining wheel velocities that guide a differential system along a desired trajectory is known as trajectory-tracking (different from the similarly-titled problem in this thesis) or regulation because this method was originally designed for

maintaining the temperature of containers in a chemical processing plant along a desired profile. In Chapter 4, we discussed the person-following problem and presented an early system prototype in which a person is tracked from a stereo-camera rig mounted on a mobile robotic base. We discussed the design and implementation of two existing control theoretic approaches in the context of our system – one that is intuitive and uses cubic curves for convergence, and one steeped in control theory that uses the concept of dynamic feedback linearization to reparametrize the state space. Our system was then enhanced to react to the movement of the person by incorporating a two-state human mobility model that consists of a stationary state and a linear state 5. In the stationary state, we showed that the best response strategy of the robots is a static configuration in which they are distributed uniformly around the target to guard against future directions of motion. In the linear state, we assumed that the person follows a linear-velocity model and accordingly derived multi-robot trajectories via a heuristic state-space search method based on the projected future location of the person as determined by the model. We demonstrated that our two-state mobility model can be used to control the responsiveness of the robots to the person [27].

9.1.3 An Uncertainy-based target mobility model

While the known trajectory model works well for reactive robot strategies, it is unsuitable when robot trajectories have to be planned *before* the target moves. The full knowledge of target trajectory ends up being too strong in this case because targets, in general, navigate with independent intention. We modeled the person as a passive decision-making entity whose possible choices, albeit varied and numerous, are known but his actual decision is only observed *after* the robots move. This type of planning can be thought of as being pre-emptive in the sense that the robots need to predict where the person moves by looking ahead in time, and planning paths to address the uncertainty in his future locations.

In Chapter 6, we presented a data-driven target motion model that encodes the decision uncertainties inherent in known trajectory preferences. Our model encodes variable-length histories and performs predictions better than existing approaches. Furthermore, the stochastic nature of our motion model characterizes the target's mobility as a probabilistic state machine. We incorporate the states of our motion model into a

partially-observable planning framework (POMDP) to yield multi-robot plans that account for target uncertainty. Two desirable outcomes of this process were observed: (ii) our planner assumes the worst-case, i.e. no knowledge of person's behavior, when trajectories are observed that were not previously seen, and, (i) the resulting robot strategies exhibit desirable behaviors such as anticipation, re-use and guarding. The robot behaviours obtained from our approach could prove to be useful as motion primitives, a future direction aimed at reducing planning complexity.

9.1.4 System design and software implementation

We demonstrate the utility of the models and algorithms described in this thesis by presenting a proof-of-concept system implementation for mobile tele-conferencing applications (Chapter 8). We discuss a hardware platform design that combines affordable sensors with a robust mobile base. Our platform is designed from off-the-shelf components, making it accessible not just to commercial businesses but more importantly everday users e.g. in their homes. We present the software developed for each component of the system, and propose an overall software architecture that integrates them into a cohesive real-time system. Our code is written primarily in the C++ language, relying on Linux shell scripts for useful utilities. The WillowGarage Robot Operating System (ROS) library was used in three primary ways: (i) to provide a modular implementation transparent to individual component changes, (ii) to prevent re-inventing existing state-of-the art components such as mapping, localization and simulators, and (iii) to contribute our software back to the community for the benefit of future implementations.

9.2 Future directions

In this section, we present future research directions based on the problems addressed in this thesis.

9.2.1 Reducing planner complexity

The anticipatory planner presented in Chapter 6 uses the general POMDP framework to solve the tracking problem. This approach has the advantage that complex observation

models that are closer to real-world sensor footprints than a discrete tracking objective can be incorporated simply by re-defining the observation function. One future direction could be to use a geometric, visibility-based observation function instead. However, this is expected to come at the cost of increasing the state space, because topological environment information would not be sufficient to express that level of detail.

From the resulting robot strategies in Chapter 6, we observed that the robots execute certain behaviors such as anticipation, guarding and re-use (see Figure 9.2). However, the following robot was manually introduced into the system in order to never lose track of the person in the event that the path that an anticipatory robot took turned out to be wrong. By modeling observability more explicitly, we believe that we can automate this process, leading to interesting robot behaviors that can be used as motion primitives to simplify the robot action space.

9.2.2 Scalability

Figure 9.3 shows a large pedestrian dataset collected by researchers at NCSU [1]. Such a dataset for pedestrians is typically obtained by tracking GPS locations of handheld mobile devices over long periods of time. Road networks differ from the indoor environments we dealt with in this thesis in that they have an inherent graph structure with edges for roads and vertices for intersections. However, the scale of this mobility is of the order of 50-100m and trajectory datasets span lots of megabytes of storage.

Building a complete history-based motion model from these datasets by directly using the enhanced trie from Chapter 6 is infeasible. Instead, we could build an incremental trie by only adding more information if the predictive power of the model improves by more than a lower-bounded threshold. Such a measure of improvement needs to be carefully defined because the model represents a probability distribution over trajectories – either a Mahalanobis distance or Kullback-Liebler divergence could be used. This method would result in an approximation of the temporal dependencies from the dataset, since not all vertices would be preserved – only the most informative paths would remain.

Even after approximating the input for the target's mobility state, it would take a long time to solve the resulting POMDP due to the sheer size of the state space, and limitations of state-of-the-art solvers. One step in this direction could be to use a

sampling-based approach to approximate the belief space of the POMDP. For instance, a Monte-Carlo method could be used to bias the search for a solution to the POMDP, but care must be taken to determine the right order in which to explore the space. An interesting concept that could help with this direction is the method of POMCPs [116].

9.2.3 Open problems

Bringing together a complete robotic system with theoretical results and a full system implementation is not an easy task. It requires not only the integration of a large number of components to work with each other simultaneously, but also addressing issues that arise from each component.

In this paper, we presented remote tele-conferencing applications from a visibility-based planning perspective. However, the provision of such a service would not be possible without reliable network infrastructure and routing algorithms that are application-aware. For instance, since we use controlled mobility of robots to provide a service, we can opportunistically route video and audio packets via suitably selected access points. However, existing network algorithms are not designed to handle highly mobile nodes with seamless data transfer. We leave this part of the system design as an open problem.

Our Robotic Sensor Networks (RSN) lab addresses various other problems that exploit controlled robot mobility, such as environmental monitoring. In these applications, a centralized approach might not be ideal since bottleneck sensors would die sooner than others. It would be interesting to explore alternate topologies for our system, e.g. if we did not allow robots to communicate at all times, but instead solved a distributed tracking problem. However, using a purely local objective would take away from the desirable global properties such as robots sharing the tracking cost over time. Another interesting direction would be a surveillance or monitoring task that allows for some level of delay-tolerance, e.g. a robot that is meant to inspect an environment for moving targets and deliver its findings to a base station at a later time. We could use the adversarial approach from Chapter 3 as a starting point, and extend it to address more complex sensing and environmental models.

When relying on datasets to observe phenomenon, it becomes important how we extract useful information from data. Machine learning techniques such as data mining have proven useful in finding interesting patterns from location-based data, e.g.

unsupervised trajectory clustering.

Finally, not all objective functions and environments are known before-hand. When the environment map is unknown, there is a delicate trade-off between exploring the unknown environment for better paths, and exploiting known map information – a common problem in the machine learning community. For our applications, we used a known map because we were concerned with indoor applications of the scale of 10-50m, for which data can be collected by mapping algorithms once, so that the best possible service can be provided repeatedly.

One aspect of a robotic system that navigates autonomously in the presence of people is the social and cultural impact of people interacting with the robots. In our system, we can run usability studies targeted at specific demographics, since we expect that younger people and people used to living with pets are more accepting of robot technologies than elderly people or those without exposure to pets [10]. With a growth in robots for home, office and hospital use, it would be interesting to see how people react to autonomous robots as part of their daily lives, and how these robots in-turn could affect social norms.

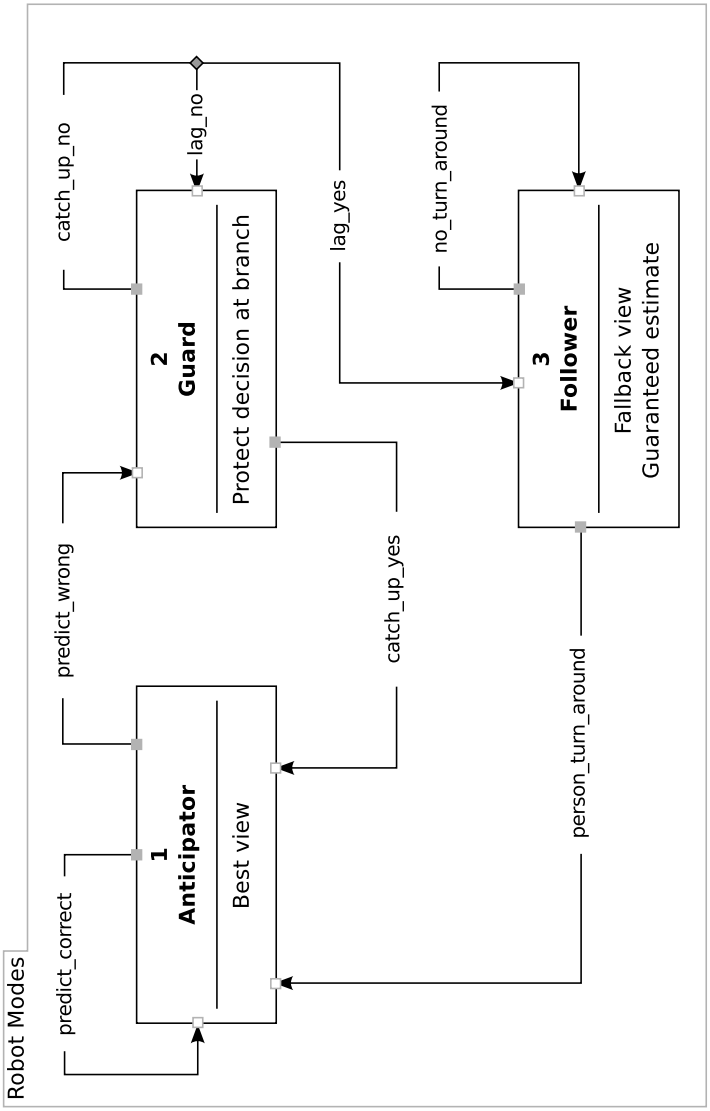


Figure 9.2: Modes of operation for our robot platforms.

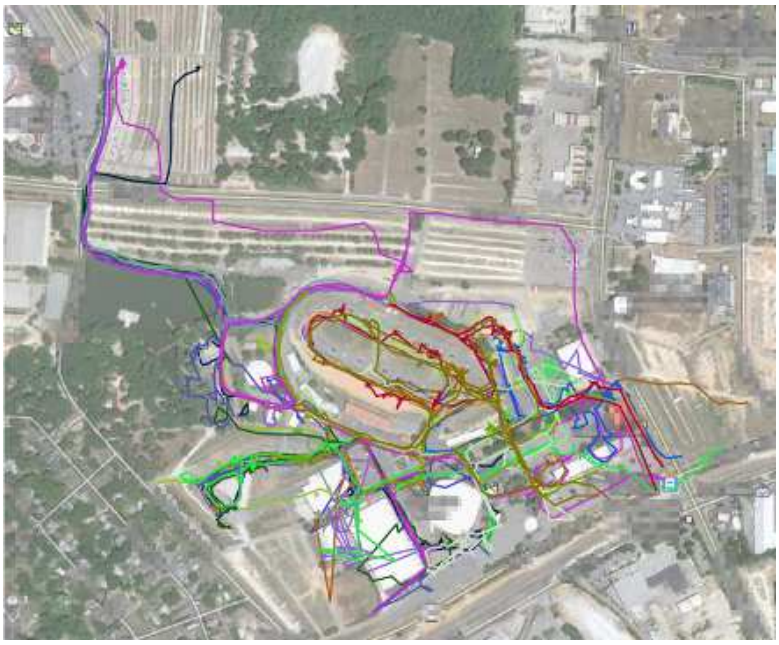


Figure 9.3: Example of a large pedestrian dataset “Statefair” [1]

References

[1] Injong Rhee, Minsu Shin, Seongik Hong, Kyunghan Lee, and Song Chong. On the levy walk nature of human mobility. In *Proc. IEEE INFOCOM 2008*, Phoenix, AZ, April 2008.

[2] N. J. Nilsson. Shakey the robot. Technical report, DTIC Document, 1984.

[3] J. Forlizzi and C. DiSalvo. Service robots in the domestic environment: a study of the roomba vacuum in the home. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 258–265. ACM, 2006.

[4] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *Proceedings of the National Conference on Artificial Intelligence*, pages 11–18. John Wiley & Sons Ltd, 1998.

[5] Z. Byers, M. Dixon, K. Goodier, C. M. Grimm, and W. D. Smart. An autonomous robot photographer. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2636–2641. IEEE, 2003.

[6] B. Graf, M. Hans, and R. D. Schraft. Care-O-bot IIdevelopment of a next generation robotic home assistant. *Autonomous robots*, 16(2):193–205, 2004.

[7] Lucy Hipperson and David Molony. Telepresence: beyond Cisco. Technical Report, Ovum Ltd, February 2008. http://store.ovum.com/Product.asp?pid=38759&etr=infaus.

[8] Verdantix. Carbon Disclosure Project Study 2010: The Telepresence Revolution. Online report retrieved from www.cdproject.net/CDPResults/Telepresence-Revolution-2010.pdf.

[9] IBM Global Technology Services. Transforming your voice, video and collaboration infrastructure. Thought Leadership White Paper, IBM Corporation, September 2010.

[10] A. D. Fisk, W. A. Rogers, N. Charness, and J. Sharit. *Designing for older adults: Principles and creative human factors approaches*, volume 2. CRC Press, 2009.

[11] Richard A. Hussian and Debbie C. Brown. Use of Two-Dimensional Grid Patterns to Limit Hazardous Ambulation in Demented Patients. *J Gerontol*, 42(5):558–560, 1987, http://geronj.oxfordjournals.org/cgi/reprint/42/5/558.pdf.

[12] C. C. Lin, M. J. Chiu, C. C. Hsiao, R. G. Lee, and Y. S. Tsai. Wireless health care service system for elderly with dementia. *IEEE Transactions on Information Technology in Biomedicine*, 10(4):696–704, 2006.

[13] Chieko Greiner, Kiyoko Makimoto, Mizue Suzuki, Miyae Yamakawa, and Nobuyuki Ashida. Feasibility Study of the Integrated Circuit Tag Monitoring System for Dementia Residents in Japan. *American Journal of Alzheimer's Disease and Other Dementias*, 22(2):129–136, 2007, http://aja.sagepub.com/cgi/reprint/22/2/129.pdf.

[14] R. G. Golledge. *Wayfinding behavior: Cognitive mapping and other spatial processes*. Johns Hopkins University Press, 1999.

[15] L. Takayama and C. Pantofaru. Influences on proxemic behaviors in human-robot interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2009.*, pages 5495–5502. IEEE, 2009.

[16] E. T. Hall. The Hidden Dimension, 1990.

[17] Rachel Kirby. *Social Robot Navigation*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2010.

[18] N. P. Papanikolopoulos, P. K. Khosla, and T. Kanade. Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision. *Robotics and Automation, IEEE Transactions on*, 9(1):14–35, 1993.

[19] Peter I. Corke. Visual control of robot manipulators – A review. In *Visual Servoing*, pages 1–31. World Scientific, 1994.

[20] F. Chaumette and S. Hutchinson. Visual servo control. I. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.

[21] F. Chaumette, S. Hutchinson, et al. Visual servo control, part II: Advanced approaches. *IEEE Robotics and Automation Magazine*, 14(1):109–118, 2007.

[22] N. Karnad and V. Isler. Lion and man game in the presence of a circular obstacle. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2009.*, pages 5045–5050. IEEE, 2009.

[23] D. Bhadauria and V. Isler. Capturing an evader in a polygonal environment with obstacles. In *22nd International Joint Conference on Artificial Intelligence*, 2011.

[24] Y. Bar-Shalom and X. R. Li. *Estimation with Applications to Tracking and Navigation.* John Wiley & Sons, Inc., 2001.

[25] R. Gockley, J. Forlizzi, and R. Simmons. Natural person-following behavior for social robots. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 17–24. ACM, 2007.

[26] O. Tekdas, W. Yang, and V. Isler. Robotic routers: Algorithms and implementation. *The International Journal of Robotics Research*, 29(1):110–126, 2010.

[27] N. Karnad and V. Isler. A multi-robot system for unconfined video-conferencing. In *The 2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 356–361. IEEE, 2010.

[28] R. Isaacs. *Differential Games.* Wiley, New York, NY, 1965.

[29] Joseph H. Discenza and Lawrence D. Stone. Optimal survivor search with multiple states. *Operations Research*, 29(2):309–323, March-April 1981.

[30] Maria Prandini, João Pedro Hespanha, and George J. Pappas. Greedy control for hybrid pursuit games. In *ECC01*, pages 2621–2626, September 2001.

[31] T. Basar and G. J. Olsder. *Dynamic Noncooperative Game Theory*. Academic Press, London, revised 2nd edition, January 1995.

[32] LaValle, S.M. and H. H. Gonzalez-Banos and C. Becker and J.-C. Latombe. Motion strategies for maintaining visibility of a moving target. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 1997.*, volume 1, pages 731–736 vol.1. IEEE, April 1997.

[33] S. M. LaValle, D. Lin, L. J. Guibas, J. C. Latombe, and R. Motwani. Finding an unpredictable target in a workspace with obstacles. In *Proceedings., 1997 IEEE International Conference on Robotics and Automation (ICRA), 1997.*, volume 1, pages 737–742. IEEE, 1997.

[34] J. E. Littlewood. *Littlewood's Miscellany*, chapter Lion and Man, pages 114–117. Cambridge University Press, Cambridge, England, 1986.

[35] L. Alonso, A. S. Goldstein, and E. M. Reingold. "Lion and man": upper and lower bounds. *ORSA J. Comput.*, 4(4):447–452, 1992.

[36] Jiří Sgall. Solution of david gale's lion and man problem. *Theor. Comput. Sci.*, 259(1-2):663–670, 2001.

[37] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion in a polygonal environment. *IEEE Transactions on Robotics*, 5(21):864–875, 2005.

[38] Stephanie Alexander, Richard Bishop, and Robert Ghrist. Pursuit and evasion in non-convex domains of arbitrary dimensions. In Gaurav S. Sukhatme, Stefan Schaal, Wolfram Burgard, and Dieter Fox, editors, *Robotics: Science and Systems*. The MIT Press, 2006.

[39] S. D. Bopardikar, F. Bullo, and J. P. Hespanha. Sensing limitations in the Lion and Man problem. In *ACC*, pages 5958–5963, New York, July 2007.

[40] N. Karnad and V. Isler. Bearing-Only Pursuit. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 2665–2670, May 2008.

[41] Leonidas J. Guibas, Jean claude Latombe, Steven M. Lavalle, David Lin, and Rajeev Motwani. Visibility-based pursuit-evasion in a polygonal environment. In *International Journal of Computational Geometry and Applications*, pages 17–30. Springer-Verlag, 1997.

[42] Sourabh Bhattacharya and Seth Hutchinson. On the existence of nash equilibrium for a visibility based pursuit evasion game. In *Workshop on Algorithmic Foundations of Robotics*, 2008.

[43] Donald E. Kirk. *Optimal control theory; an introduction*. Prentice-Hall, Englewood Cliffs, N.J., 1970.

[44] M. Wise and J. Hsu. Application and analysis of a robust trajectory tracking controller for under-characterized autonomous vehicles. In *Control Applications, 2008. CCA 2008. IEEE International Conference on*, pages 274–280. IEEE, 2008.

[45] G. Oriolo, A. De Luca, and M. Vendittelli. WMR control via dynamic feedback linearization: design, implementation, and experimental validation. *IEEE Transactions on Control Systems Technology*, 10(6):835–852, 2002.

[46] C. Schlegel, J. Illmann, H. Jaberg, M. Schuster, and R. Worz. Vision based person tracking with a mobile robot. In *British Machine Vision Conference*, pages 418–427. Citeseer, 1998.

[47] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection. *International Journal of Computer Vision*, 37(2):175–185, 2000.

[48] H. Kwon, Y. Yoon, J. B. Park, and A. C. Kak. Person tracking with a mobile robot using two uncalibrated independently moving cameras. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2877–2883. IEEE, 2006.

[49] Z. Chen and S. T. Birchfield. Person following with a mobile robot using binocular feature-based tracking. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 815–820. IEEE, 2007.

[50] H. Sidenbladh, D. Kragic, and HI Christensen. A person following behaviour for a mobile robot. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 1, pages 670–675. IEEE, 2002.

[51] M. Kleinehagenbrock, S. Lang, J. Fritsch, F. Lomker, GA Fink, and G. Sagerer. Person tracking with a mobile robot based on multi-modal anchoring. In *Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on*, pages 423–429. IEEE, 2002.

[52] M. Kobilarov, G. Sukhatme, J. Hyams, and P. Batavia. People tracking and following with mobile robot using an omnidirectional camera and a laser. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 557–562. IEEE, 2006.

[53] T. Yoshimi, M. Nishiyama, T. Sonoura, H. Nakamoto, S. Tokura, H. Sato, F. Ozaki, N. Matsuhira, and H. Mizoguchi. Development of a person following robot with vision based target detection. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5286–5291. IEEE, 2007.

[54] A. De Luca, G. Oriolo, and M. Vendittelli. Control of Wheeled Mobile Robots: An Experimental Overview. *RAMSETE: articulated and mobile robotics for services and technologies*, page 181, 2001.

[55] M. Minsky. Telepresence. *Omni*, 2(9):45–52, 1980.

[56] S. Sabri and B. Prasada. Video conferencing systems. *Proceedings of the IEEE*, 73(4):671–688, 1985.

[57] J. Leigh, T. A. DeFanti, A. Johnson, M. D. Brown, and D. J. Sandin. Global tele-immersion: Better than being there. In *ICAT*, volume 97, pages 10–17, 1997.

[58] T. Kanade, PJ Narayanan, and PW Rander. Virtualized reality: Concepts and early results. In *Proceedings IEEE Workshop on Representation of Visual Scenes, 1995.(In Conjuction with ICCV'95)*, pages 69–76, 1995.

[59] Milton Chen. Design of a virtual auditorium. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, pages 19–28, New York, NY, USA, 2001. ACM.

[60] O. Schreer, N. Brandenburg, S. Askar, and E. Trucco. A virtual 3D video-conferencing system providing semi-immersive telepresence: A real-time solution in hardware and software. In *Proc. Intl. Conf. eWork and eBusiness*, pages 184–190, 2001.

[61] Ladan Gharai, Colin Perkins, Ron Riley, and Allison Mankin. Large scale video conferencing: A digital amphitheater. In *Proc. 8th International Conference on Distributed Multimedia Systems*, 2002.

[62] H. H. Baker, D. Tanguay, I. Sobel, D. Gelb, M. E. Goss, W. B. Culbertson, and T. Malzbender. The coliseum immersive teleconferencing system. In *Proc. International Workshop on Immersive Telepresence*, volume 6, 2002.

[63] J. Lanier. Virtually there. *Scientific American*, pages 66–75, April 2001.

[64] Herman Towles, Wei chao Chen, Ruigang Yang, Sang-Uok Kum, Henry Fuchs, Nikhil Kelshikar, Jane Mulligan, Kostas Daniilidis, Loring Holden, Bob Zeleznik, Amela Sadagic, and Jaron Lanier. 3D tele-collaboration over internet2. In *In: International Workshop on Immersive Telepresence, Juan Les Pins*, 2002.

[65] David E. Ott and Ketan Mayer-Patel. Coordinated multi-streaming for 3D tele-immersion. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 596–603, New York, NY, USA, 2004. ACM.

[66] J. Mulligan, V. Isler, and K. Daniilidis. Trinocular stereo: A new algorithm and its evaluation. *International Journal of Computer Vision*, 47:51–61, 2002.

[67] Sang-Hack Jung and Ruzena Bajcsy. A framework for constructing real-time immersive environments for training physical activities. *Journal of Multimedia*, 1(7):9–17, 2006.

[68] Zhenyu Yang, Bin Yu, Klara Nahrstedt, and Ruzena Bajcsy. A multi-stream adaptation framework for bandwidth management in 3D tele-immersion. In Brian Neil Levine and Mark Claypool, editors, *NOSSDAV*, page 14. ACM, 2006.

[69] Wanmin Wu, Zhenyu Yang, Dongyun Jin, and Klara Nahrstedt. Implementing a Distributed 3D Tele-immersive System. *IEEE International Symposium on Multimedia (ISM)*, 2008.

[70] Nokia Jeppe - Domestic videoconferencing, 2008. Nokia Research Labs, Helsinki, Finland. http://research.nokia.com/research/teams/extended_home/jeppe.html.

[71] Headthere giraffe - A telepresence robot. http://www.headthere.com/technical.html.

[72] T. Muppirala, S. Hutchinson, and R. Murrieta-Cid. Optimal motion strategies based on critical events to maintain visibility of a moving target. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3826–3831, 2005.

[73] Steven Michael LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[74] S. M. Lavalle and P. Konkimalla. Algorithms for computing numerical optimal feedback motion strategies. *The International Journal of Robotics Research*, 20(9):729, 2001.

[75] S. Bhattacharya, R. Murrieta-Cid, and S. Hutchinson. Optimal paths for landmark-based navigation by differential-drive vehicles with field-of-view constraints. *IEEE Transactions on Robotics*, 23(1):47–59, 2007.

[76] Free 3D models by artist-3d.com. Digital art and 3d media exchange directory on the Internet. http://artist-3d.com/free_3d_models/dnm/model_disp.php?uid=567.

[77] G. R. Bradski et al. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, 2(2):12–21, 1998.

[78] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1197–1203 vol.2, 1999.

[79] T. Liu, P. Bahl, and I. Chlamtac. Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks. *IEEE Journal on Selected Areas in Communications*, 16(6):922–936, 1998.

[80] A. Bhattacharya and S. K. Das. LeZi-update: An information-theoretic approach to track mobile users in PCS networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 1–12. ACM, 1999.

[81] Christian Hofner and Gnther Schmidt. Path planning and guidance techniques for an autonomous mobile cleaning robot. *Robotics and Autonomous Systems*, 14(2-3):199–212, 1995. Research on Autonomous Mobile Robots.

[82] N. Roy, G. Baltus, D. Fox, F. Gemperle, J. Goetz, T. Hirsch, D. Margaritis, M. Montemerlo, J. Pineau, J. Schulte, et al. Towards personal service robots for the elderly. In *Workshop on Interactive Robots and Entertainment (WIRE 2000)*, volume 25, page 184, 2000.

[83] R. Golledge. Path selection and route preference in human navigation: A progress report. *Spatial Information Theory: A Theoretical Basis for GIS*, pages 207–222, 1995.

[84] R. G. Golledge. Human wayfinding and cognitive maps. *Wayfinding behavior: Cognitive mapping and other spatial processes*, pages 26–31, 1999.

[85] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5-6):311–331, 2007.

[86] J. Joseph, F. Doshi-Velez, and N. Roy. A bayesian nonparametric approach to modeling mobility patterns. *Proceedings of the AAAI'10*, pages 1587–1593, 2010.

[87] H. H. González-Banos, C. Y. Lee, and J. C. Latombe. Real-time combinatorial tracking of a target moving unpredictably among obstacles. In *Proceedings. ICRA'02. IEEE International Conference on Robotics and Automation, 2002.*, volume 2, pages 1683–1690. IEEE, 2002.

[88] D. Hsu, W. S. Lee, and N. Rong. A point-based POMDP planner for target tracking. In *Proc. of ICRA 2008. IEEE International Conference on Robotics and Automation, 2008.*, pages 2644–2650. IEEE, 2008.

[89] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[90] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *International joint conference on artificial intelligence*, volume 18, pages 1025–1032, 2003.

[91] H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proc. Robotics: Science and Systems*, 2008.

[92] H. Kurniawati, Y. Du, D. Hsu, and W. S. Lee. Motion planning under uncertainty for robotic tasks with long time horizons. *The International Journal of Robotics Research*, 30(3):308, 2011.

[93] R. He, E. Brunskill, and N. Roy. Efficient planning under uncertainty with macro-actions. *Journal of Artificial Intelligence Research*, 40(1):523–570, 2011.

[94] A. A. Kalia, G. E. Legge, R. Roy, and A. Ogale. Assessment of Indoor Route-finding Technology for People Who Are Visually Impaired. *Journal of Visual Impairment & Blindness*, 104(3):135–147, 2010.

[95] Aphrodite Galata, Neil Johnson, and David Hogg. Learning Variable-Length Markov Models of Behavior. *Computer Vision and Image Understanding*, 81(3):398–413, March 2001.

[96] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[97] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT Press, 2001.

[98] Fernando Cacciola. 2D straight skeleton and polygon offsetting. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.8 edition, 2011. Retreived online from http://www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html #Pkg:StraightSkeleton2.

[99] Microsoft Corp. Redmond WA. Kinect for Xbox 360.

[100] M. Siddiqui and G. Medioni. Human pose estimation from a single view point, real-time range sensor. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2010*, pages 1–8. IEEE, 2010.

[101] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real time motion capture using a single time-of-flight camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010*, pages 755–762. IEEE, 2010.

[102] iRobot Create. Programmable robot. Retrieved Nov 2011 from `http://www.irobot.com/create/`.

[103] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 1(4):333–356, 1988.

[104] A. Blake and A. Yuille. *Active vision*. MIT press, 1993.

[105] Alan D. Greenberg. The 2009 Update: Taking the Wraps off Videoconferencing in the U.S. Classroom. In *White paper*. Wainhouse Research, April 2009.

[106] H. Y. Shum, S. C. Chan, and S. B. Kang. *Image-based rendering*. Springer-Verlag Inc., New York, 2007.

[107] S. M. Seitz and C. R. Dyer. Physically-valid view synthesis by image interpolation. In *Proceedings of the IEEE Workshop on Representation of Visual Scenes, 1995.(In Conjuction with ICCV'95)*, pages 18–25. IEEE, 1995.

[108] S. Avidan and A. Shashua. Novel view synthesis in tensor space. In *Proceedings of the 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1034–1040. IEEE, 1997.

[109] D. Scharstein. *View synthesis using stereo vision*. Springer-Verlag, 1999.

[110] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics (TOG)*, 23(3):600–608, 2004.

[111] G. K. M. Cheung, S. Baker, J. Hodgins, and T. Kanade. Markerless human motion transfer. In *Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*, pages 373–378. IEEE Computer Society, 2004.

[112] J. Carranza, C. Theobalt, M. A. Magnor, and H. P. Seidel. Free-viewpoint video of human actors. In *ACM SIGGRAPH 2003*, pages 569–577. ACM, 2003.

[113] Hiroshi and Narushima. Principle of inclusion-exclusion on partially ordered sets. *Discrete Mathematics*, 42(2-3):243–250, 1982.

[114] J. O'Rourke. *Art gallery theorems and algorithms*, volume 57. Oxford University Press Oxford, 1987.

[115] A. Mittal and L. S. Davis. A general method for sensor planning in multi-sensor systems: Extension to random occlusion. *International Journal of Computer Vision*, 76(1):31–52, 2008.

[116] D. Silver and J. Veness. Monte-carlo planning in large POMDPs. *Advances in Neural Information Processing Systems (NIPS)*, 2010.