Motion Strategies for Visibility based Target Tracking in Unknown Environments

TIRTHANKAR BANDYOPADHYAY

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2009

SUMMARY

Target tracking is an interesting problem and has important applications in security and surveillance systems, personal robotics, computer graphics, and many other domains. The focus of this thesis is on computing motion strategies to keep a moving target in view in a dynamic and unknown environment using visual sensors. The problem of motion planning is complicated by the mobility and visual obstructions from the obstacles in the environment. Without using a-priori information about the target and the environment, this thesis proposes an online tracking algorithm which plans its motion strategy using local information from on-board sensors. In order to track intelligently, the tracker has to choose an action which lowers the danger of losing the target in the future while maintaining it under view in the current step. This thesis proposes a measure called *relative vantage* which combines the risk of losing the target in the current time to the risk of losing the target in the future. A local greedy tracking algorithm called *vantage tracker* is proposed which chooses actions to minimize this risk measure.

Implementing a robust robotic tracker requires dealing with *sensing limitations* such as maximum range, field-of-view limits, *motion limitations* such as maximum speed bound, non-holonomic constraints and *operational limitations* such as obstacle avoidance, stealth, *etc.* This thesis proposes a general tracking framework that incorporates these limitations into the problem of online target tracking. A real robotic tracker was setup using a simple laser range finder and a differential drive robot base and the hardware limitations were addressed in the tracking framework as planning constraints. Such a tracker was able to successfully follow a person in a crowded environment. A stealth constraint was formulated where the tracker has to maintain sight of the target while trying to avoid being detected. Incorporating this stealth constraint into the tracking problem, a stealth tracking algorithm was developed and analyzed for various environments in simulation.

In a 3-D environment, the visibility relationships become complex easily. Moreover, the additional dimension available to the target makes the tracking problem more difficult. A 3-D vantage tracker was developed by generalizing the approach pertaining to the 2-D tracker. Such a tracker generates intelligent tracking actions by exploiting the additional dimension. As an example a robotic helicopter generates a vertical motion to avoid occlusion of the target due to the buildings in an urban scenario when it can improve its visibility by doing so. Such a behavior was generated based only on the locally sensed geometric parameters and no *a priori* knowledge of the layout or the model of the obstacles in the environment was used.

Extensive simulation and hardware results show consistently the improvement in tracking performance of the vantage tracker based tracking framework both in 2-D and in 3-D as compared to previous approaches such as visual servo and those based on increasing the shortest distance to escape for the target.

ACKNOWLEDGMENTS

A doctoral research is rarely the outcome of a single person's effort. Nor is it just the technical component that ensures the successful journey to the doctoral degree. This is an unfairly short acknowledgement of everyone who made this thesis a success.

I dedicate this thesis to my parents for their love, support and efforts to provide for my education. Their enormous personal sacrifices to give me a educational environment cannot be captured in words.

I am fortunate to have found such inspirational advisors, Prof. Marcelo H. Ang, Jr. and Prof. David Hsu, without whose guidance and support I would not be here today. Their exemplary research standards have inspired me to strive constantly to improve myself as a researcher. I am in-debt to them for having faith in me and my work when even I was not so sure.

I am grateful to Prof. Cezary Zieliński for hosting me in WUT, Poland and for his guidance during my stay there. The hardware implementation would not have been possible without the help and training from his students Marek and Piotrek. I am also in-debt to Prof. Franz Hover for his understanding and easing off my obligations in SMART during the incredibly stressful period of thesis submission.

My friends and colleagues in the Control lab and the SoC lab deserve special thanks. Foremost Yuanping for long discussions and invaluable input about the visibility decomposition ideas that generated the core idea of this thesis. Niak Wu, Mana, Gim Hee, James for creating a vibrant atmosphere in the lab by their discussions and sharing of ideas both technical and otherwise that helped motivate, influence and sustain this work. I am especially thankful to Tomek for his involvement in long technical discussions and personal support both in Singapore and Poland. Tomek, Sylwia, Emil and Ewa made the trip to Poland an extremely memorable one.

I thank my seniors Kevin and Bryan in helping and guiding me during the early days of my PhD and show my appreciation to the support from the technicians and laboratory officers of the Control lab.

Last but not least, I would like to give a special note of appreciation to my lovely wife, Byas for her perpetual understanding, support and companionship. In the face of seemingly unending deadlines she mysteriously manages to love me.

TABLE OF CONTENTS

Page

Chapters::

Appendices:

LIST OF FIGURES

LIST OF TABLES

Table **Page**

SYMBOLS

$\mathcal{F}$ Free space: Portion of the Euclidian workspace that is free from obstacles.

R Tracker.

$\mathbf{x}$ Tracker position.

$\mathbf{v}$ Tracker velocity.

V Maximum velocity bound on the tracker.

T Target.

$\mathbf{x}'$ Target position.

$\mathbf{v}'$ Target velocity.

V' Maximum velocity bound on the target.

$\mathcal{V}$ Local visibility of the robot (both 2-D and 3-D).

$\partial\mathcal{V}$ Boundary of the robot's visibility.

$\mathcal{V}'$ Visibility polygon of the target inside the robot's local visibility (both 2-D and 3-D).

$\mathcal{G}$ Escape Gap in $\mathcal{V}$: Portion of $\partial\mathcal{V}$ through which the target can escape. This takes the form of escape edge in 2-D, and escape surface in 3-D.

$\mathcal{B}$ Obstacle boundary. $\mathcal{B}$ and $\mathcal{G}$ constitute $\partial\mathcal{V}$.

$\mathcal{O}_v$ Occlusion Vertex (both 2-D and 3-D).

$\mathcal{O}_e$ Occlusion Edge (both 2-D and 3-D).

$\mathcal{O}_p$ Occlusion Plane in 3-D.

$\mathcal{B}_e$ Obstacle Edge that generates the occlusion plane $\mathcal{O}_p$ in 3-D.

$\mathcal{R}$ Set of all positions reachable by the tracker in one time step.

$\mathcal{L}$ Set of all positions reachable by the tracker that satisfies all the tracking requirements.

Φ A measure of the risk of losing the target from the tracker's visibility.

$\mathbf{v}^\star$ The computed tracker velocity which minimizes the risk function for the next step.

$\mathcal{D}$ Danger zone for a particular $\mathcal{G}$, a region where the target can reach $\mathcal{G}$ faster than the robot.

$\partial\mathcal{D}$ The boundary of $\mathcal{D}$ within $\mathcal{V}$.

G Escape gap zone : In order to assign probability of escape to those $\mathcal{G}$ which do not subtend an angle at the target.

$\mathcal{G}'$ Escape Gap in $\mathcal{V}'$: Portion of the boundary of $\mathcal{V}'$ through which the tracker can hide from the target. The stealth constraint is formulated to stay close to $\mathcal{G}'$.

$\mathcal{S}$ Lookout regions from where the tracker can keep the target in view and exit quickly if required from $\mathcal{V}'$.

L The distance from $\mathcal{G}'$ from which the tracker can escape $\mathcal{V}'$ in one time step.

$\mathcal{D}_N$ Portion of $\mathcal{D}$ in 3-D that is closest to $\mathcal{O}_p$.

$\mathcal{D}_L$ Portion of $\mathcal{D}$ in 3-D that is closest to $\mathcal{O}_e$.

$\mathcal{D}_R$ Portion of $\mathcal{D}$ in 3-D that is closest to $\mathcal{B}_e$.

$\mathcal{D}_V$ Portion of $\mathcal{D}$ in 3-D that is closest to $\mathcal{O}_v$.

CHAPTER 1

INTRODUCTION

This thesis presents motion strategies for a mobile sensor to continuously keep a moving target in view. Tracking the target is an important task for autonomous robots and has many applications. In security and surveillance systems, tracking strategies enable mobile sensors to potentially monitor moving targets continuously in crowded environments. In law enforcement or military operations, reliable information from aerial systems have improved the effectiveness of operations on the ground in urban environments. Smart tracking strategies will be required to automatically generate unobstructed views in the presence of tall buildings and foliage in such operations. In computer graphics, keeping a specific object or activity unoccluded is important for automated viewpoint generation. In home care settings, a tracking robot can follow elderly people around, giving companionship, monitoring their vital signs, and alert caregivers in case of emergencies. Robotic porters can help people carry belongings by tracking and following them to their desired locations.

Tracking a target (an object or human) reliably in a dynamic environment is more than just blindly following. A specific example that illustrates this point is that of an automated personal shopping assistant following a person in a shopping mall or keeping an eye on young kids while their parents are shopping. The shopping mall is an example of a highly dynamic environment with people walking around and creating obstruction and occlusion for the tracking robot. The standard problem of motion planning [1] now has to take motion and visibility constraints into account. While the layout of the environment might be available in some cases, exact maps useful for localizing the robot are hardly provided. Moreover, the target can be completely unpredictable in moving from one shop to another. Following and keeping the target in view in such scenarios require intelligent positioning amid dynamic obstacles making it a significantly challenging task.

The focus of this Ph.D research is to generate motion strategies to keep a target in view in unknown and dynamic environments. Although non-adversarial, the target's motion is rarely known completely. Without a-priori knowledge, the robot has only the local information about the environment and target motion provided by the on-board sensors. This local information is used to compute a motion plan that keeps the target in the tracker's view while planning to avoid future occlusions. The problem becomes becomes more severe especially in a dynamic environment, where such a motion plan has to be adapted to the changing situations quickly. Moreover, hardware limitations in sensing, mobility and operational requirements have to be satisfied while planning the robot's motion. This thesis introduces a *fast local online* algorithm to maximize the duration for keeping the target in view in an unknown and dynamic environment. A general tracking framework is presented that integrates various sensing, mobility, and planning limitations into the primary task of keeping the target in view.

1.1 Scope of the thesis

Target tracking is a complex task involving many aspects of sensing, planning and execution. Mobile target tracking can be broken down into two major sub-tasks : *Target Detection* and *Target Following*.

Target detection refers to identification and localization of the target in the environment. Target identification deals with extracting the target signatures from the raw sensory data. Such a step becomes crucial in the presence of noisy data to reduce the probability of false positives and false negatives. Although the target has been identified, its state may not be known accurately due to noisy data. Many applications require the target's location to be known very precisely, e.g. in missile interception just knowing that a missile is present is not sufficient. Its location, heading, speed must all be computed very precisely to intercept it. Depending on the available sensor modality and its corresponding error characteristics, the target localization problem can become quite daunting. In this thesis, there is only one target and a robust target detection and localization module is assumed. Also the target is visible and identified at the start. In the absence of such an assumption, any target search algorithm can be utilized to locate the target.

While a target can be detected and monitored by a network of sensors, a single mobile sensor can effectively do the job. With a mobile sensor the tracker can follow the target to keep it in view even when it is moving away. Target following problem refers to planning the tracker's motion such that the target is kept within the tracker's view. Target detection and target following are complementary problems. For a moving target, the detection module provides the target's information to the target following module. While the target following module generates motions strategies to ensure that the target is within the sensor's range for the detection module to locate and monitor the target in the next step. A smart target following algorithm can help simplify and improve the target detection and monitoring performance. This thesis focuses on the task of planning the motion strategies both for 2-D and 3-D to reliably follow the moving target and keep it in view.

The *environment* plays a crucial role in the tracking performance. Objects in the environment can limit the tracker's visibility and mobility. If the map is given, the tracker can perform offline computations for optimal actions at different locations. Depending on the complexity of the environment a single tracker may not be guaranteed to track an evasive target. In many cases the target's motion or the environment might not be known a-priori making the problem harder. One way to address the unknown environment is to build a map online and keep optimizing the tracker's actions with respect to this partial map. However, this aids in tracking only when the environment is bounded and the target visits the same locations often. For a large environment, this approach runs into the problem of space and computational limitations, especially on an embedded system with limited memory. Moreover, dynamic environments cannot be handled in this manner.

This thesis approaches the problem of dynamic unknown environments by building a simple polygonal local map of the environment. An objective function called *risk* is formulated. This risk encodes the danger of losing sight of the target from the local visibility in this environmental model. A tracking motion which minimizes this risk gives the local optimal action at each time step. At each step, the environmental model is recomputed along with the risk function and the optimal action. Dynamic environments can be handled in this quasi static manner as long as the sensing cycle runs at a much higher rate than the rate of changes in the environment.

Apart from environmental obstructions the tracking robot's own physical limitations on sensing (sensor range, field of view (FoV), sensor noise, *etc.*) and mobility (non-holonomic constraints, maximum speed, *etc.*) can lower the tracking performance. In addition, there might exist operational limitations of safe navigation and obstacle avoidance. For instance, in a human environment, the human must be given higher preference and losing a target is acceptable in light of colliding with another human. Such constraints need to be included in the motion planning of the trackers.

This thesis presents a generalized tracking framework based on a local greedy optimization in which these limitations can be formulated as tracking constraints. Planning under such an integrated framework generates suitable motion paths to keep the target in view under unknown and dynamic environments.

1.2 Main Results

A list of the main results of the thesis are highlighted below:

A general tracking framework is proposed for tracking a target in an unknown and dynamic environment both in 2-D and 3-D, using only local information from its on-board visibility sensors. An online algorithm is presented in which a suitably chosen risk function is optimized to maximize the time for which the target is visible amid visual and mobility limitations. In general additional mission requirements like stealth and localization could also be imposed on the tracking problem. Implementing the algorithm on a tracking robot in real world which would require dealing with hardware limitations in sensing (limited range, FoV), in mobility (bounded speed, non-holonomic constraints) and operational restrictions (keeping a minimum distance from people walking around). The framework handles all these by formulating these limitations into planning constraints. Such a framework is utilized in implementing the tracking algorithm on a real tracking robot to show the effectiveness of the approach. The tracking robot was able to successfully follow a person in a crowded school cafeteria using our constrained local planning approach.

Relative vantage based tracking approach This thesis introduces the concept of relative vantage in target tracking. In the absence of a map of the environment and a target whose motion is unknown, the most popular tracking strategy is to move towards the target [2] or maximize the shortest distance of the target's escape (SDE) from the tracker's visibility, [3]. But these approaches do not capture the essence of the tracking problem completely. This thesis provides a more principled approach to identify the main components of the tracking problem: the target position, its velocity or heading and the tracker's position w.r.t. the target in the tracker's visibility. Prior work does not consider the relative positioning of the target and the tracker in tracking.

In the proposed approach, the local environment and the relative position and velocity of the tracker and the target is analyzed, and the tracker is continuously positioned towards a strategic location that reduces the chances of losing the target from its view, both in the immediate step and in the future. Experiments show improvement in tracking performance of the proposed vantage tracker as compared to the previous methods such as simple visual servo or those that maximize SDE. In the absence of any obstacles, standard visual servo tracking can be seen as a special case of the proposed tracking strategy.

2-D Vantage Tracker Based on the concept of relative vantage, a *fast, online, local greedy* 2-D vantage tracker is developed. The risk of losing the target is developed into an analytical function based on the line of sight visibility model and a simplified linear motion model. At each step a local plan is generated to minimize the risk of losing the target given the local information knowledge and relative position. A greedy step is taken along the plan generated and the whole thing is recomputed. By approximating the risk measure into a simple analytic form, we are able to run the tracking algorithm at a high frequency. Re-planning at a high rate helps the tracker treat the dynamic environment as a quasi-static environment and is robust to moving obstacles and occlusions.

As no a-priori information is assumed, only locally sensed information is used for tracking. All computations and decisions are made w.r.t. the local environment as sensed by the tracker's sensors. This helps avoid the difficulty of robot localization, making it flexible enough to perform in quite complex and unbounded environment without incurring additional computation cost or planning errors. Local re-planning at a high rate bounds the errors in sensing, motion and planning, and the errors do not accumulate.

This local planning approach bypasses the complexity of global planning approaches, while providing more intelligent tracking behaviors than purely reactive approaches.

3-D Vantage Tracker A 3-D vantage tracker is developed by formulating the concept of relative vantage in a 3-D environment with 3-D visibility model. The additional dimension available to both the target and the tracker, increases the complexity of the problem. A similar approach of local greedy planning keeps the tracking tractable even in complex unknown environments. Results in simulation show interesting behaviors where the tracker exploits the vertical dimension to improve the tracking performance. To the best of our knowledge, such an online tracking algorithm in 3-D for unknown environment and unknown target is among the first to be proposed.

Stealth Tracker In keeping with the general tracking framework discussed above, a tracking algorithm is developed in 2-D by formulating the stealth objective for visibility based sensors. For a line of sight visibility model, visual tracking and stealth are opposing criteria. The opposing requirements are satisfied by the proposed stealth algorithm. A novel stealth tracking algorithm handles these opposing requirements by restricting the motion of the tracker to the target's visibility limits. Simulation results show successful stealth behavior of such a tracker in various environments.

1.3 Thesis Outline

The thesis layout is as follows. Chapter 2 covers the prior work done in target tracking problem.

Chapter 3 introduces the tracking approach that is general to both 2-D and 3-D environments. Specific formulation of the tracking objective function is developed for 2-D. Qualitative and quantitative results are shown in simulation. Also control experiments in simulation are presented to compare the tracker with existing trackers. Hardware implementation and results are shown to demonstrate the performance of the algorithm in the real world.

Chapter 4 introduces the stealth tracker which follows the target while trying to stay out of sight of the target.

Chapter 5 extends the target following problem to 3-D. A formulation of tracking objective function is developed for 3-D environments. The additional dimension introduces additional considerations into the target following problem. This chapter tries to address these concerns and extends the implementation of vantage tracker into 3-D. Simulation results and performance are addressed.

Finally, we conclude the thesis and discuss future work in chapter 6.

CHAPTER 2

LITERATURE REVIEW

The target tracking problem consists of two complementary sub problems *Target detection* and *Target following*. Target detection deals with identifying and localizing the target from a set of noisy sensor data; while target following deals with planning motion strategies for keeping a moving target in view. While the target detection problem has received significant attention in the research community, the target following is becoming more popular in light of intelligent personal robotics.

The target detection problem has been studied extensively in sensor fusion, signal processing and computer vision communities where the term target tracking is synonymous with target detection. Classic examples in signal processing community refer to the application in radar based object detection [4, 5, 6] especially using Kalman filters. Such approaches have been used for detecting people [7]. Particle filters have been applied for detecting a single target in [8, 9] and [10]. Probabilistic data association (PDA) filter has been used in cluttered environments [11, 9]. For handling multiple targets, data association algorithms such as joint probability data association filters (JPDAF), multiple hypothesis tracking (MHT) is popular. JPDAF was used in [12, 9]. Recently, sample based JPDAF has been used in indoor environments to detect people [13]. In [14] the application of MHT for target tracking is shown, while an efficient implementation was proposed in [15, 16, 17]. The problem of detecting within a stream of images has also been extensively addressed by the computer vision community [18, 19, 20].

Classically the work in target detection deals with open spaces, *e.g.* detection of missiles or aircrafts [4, 5, 6]. However, occlusion plays an important role in visibility based tracking and has been addressed in recent years. The effect of occlusions in target detection has been addressed explicitly both for vision [21, 22, 23], laser sensors [7, 17, 24] and a combination of both [25]. The authors in [22] propose a robust target detection scheme in presence of occlusions, where the occlusions are detected using infrared and a target template is searched in the scene by removing the pixels corresponding to the occluding object. In [7, 26, 23, 17, 24, 27] human legs are tracked in spite of occlusions by mobile objects and other humans using bayesian inference. In all the above mentioned work, occlusions are handled in a passive manner to improve the detection of the target. Our approach involves actively avoiding the states where occlusions might hinder detecting the target.

The focus of the thesis is on motion strategies for target following and a simplified target detection approach is adopted for implementing the robot tracker. The sensor data is segmented and a simple nearest neighbor cluster matching is performed to keep track of the target that is initialized at the start. An algorithm running at a high rate exploits the temporal continuity to successfully detect the target. Since a new target position is computed for each time step without maintaining an elaborate history of its motion, errors in target detection and target localization do not accumulate. Motion planning that actively avoids occlusions and improves the target's view at each step enables us to get away with such a simple detection scheme.

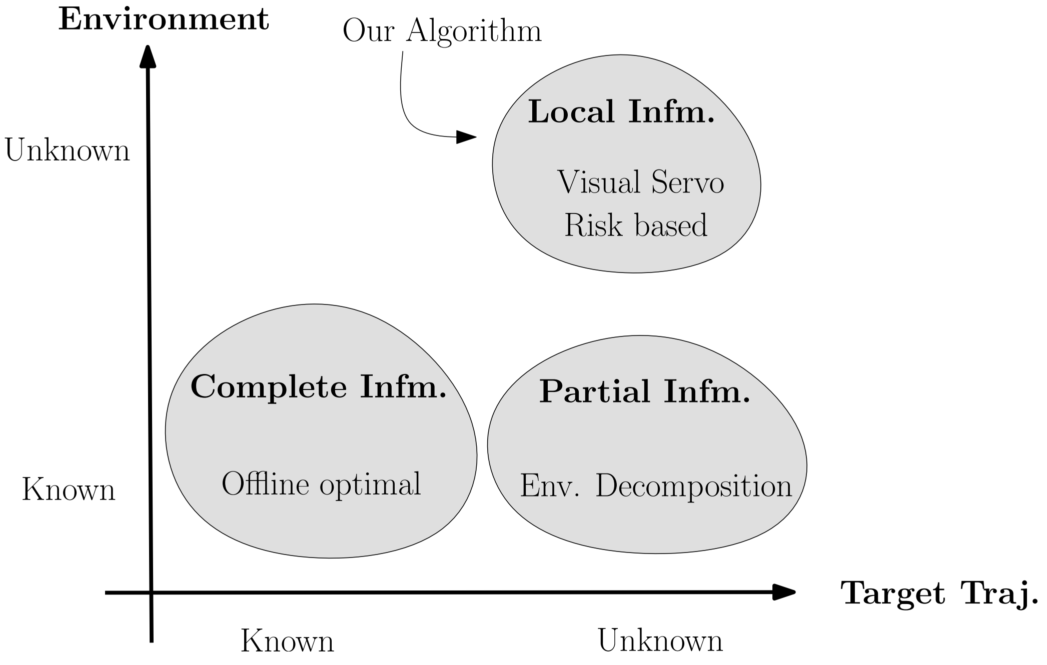


Figure 2.1. Depending on the information available about the target and the environment, the tracking approaches differ. This thesis focuses on tracking an unknown target in an unknown environment.

2.1 Motion Strategies in target tracking

The type of motion strategies used for target tracking depends on the amount of information about the environment and the target available to the tracker. A simple layout of various approaches is shown with respect to the information available to the tracker in Figure 2.1. For example in a completely known environment with known target trajectories, the tracker has the liberty to precompute the motion decisions offline. This allows for the use of computationally extensive approaches to finalize motion strategies with some notion of optimality, *e.g.*, in terms of distance traveled by the tracker, the number of steps for which the target is visible throughout its trajectory, among other criteria. Such sanitized environments are usually restricted to industrial robotics. The availability of the map of the environment is more common than the complete knowledge of the target trajectory. In such partial a-priori knowledge scenarios, some amount of subcomputation could still be done on environmental features, *e.g.*, layout topology (multiple pathways available for navigation) or spatial expansiveness (classifying regions as corridors, rooms) *etc.*. Clearly, such an analysis of the environment would help the tracker compute motion strategies that are globally efficient. Our focus in this thesis is motion planning for an unknown and dynamic environment, where there is no a-priori information and our tracker has to utilize local information. We review well known approaches for tracking with complete and partial a-priori knowledge about the environment and the target motion for completeness.

Complete Information If both the environment and the target trajectory are completely known, optimal tracking strategies can be computed by dynamic programming [28] or by piecing together certain canonical curves [29], though usually at a high computational cost. An offline approach is suitable for such scenarios where the focus is in generating *optimal* paths. In [28], the tracking states are discretized and for a valid set of trajectories, validity ensured by the target's visibility; dynamic programming is used to minimize a loss function that represents a combination of the motion costs and a penalty when the target is not visible. Geometrical computations are applied in [29] to compute a trajectory for the tracker as a combination of straight line and *leaning* curves when the line of sight between the target and the tracker pivots around an obstacle vertex. From a family of trajectories an optimal path is chosen for the tracker that maximizes the time for which the target is visible or minimizes the time for capturing the target depending whether the the target is going to be lost in the future. The strong assumption of the known target's trajectory in addition to the complete tracking environment restricts the application of such approaches to controlled environments.

Partial Information While the environmental information is more readily available, the assumption about the target motion is quite limiting in most circumstances. With the knowledge of the environment, however, the tracker can preprocess to determine regions critical to target tracking. The framework of the pursuit-evasion problem proposed in [30, 31] closely resembles the tracking problem. The objective is to search for an unpredictable target in a given environment using single or multiple trackers. Pursuit evasion has been studied in graphical environments [30], polygonal environments [32, 33, 34], curved environments [35] and also in higher dimensions [36, 37, 38]. Pursuit evasion has also been addressed with constraints in visibility [39, 40] and mobility [41].

While the above techniques used in pursuit evasion focuses on finding or capturing the target, analysing critical visibility events from the known environment could help in keeping the target in view once it has been found. One can preprocess the environment by decomposing it into cells separated by critical curves, [42, 43, 44]. The objective is to apply a cell decomposition of the configuration space and the workspace to compute *escapable cells.* Once such regions are defined, the motion strategies can be precomputed and a guarantee on tracking made. The decomposition helps to identify the best tracker action as well as to decide the feasibility of tracking [45, 46]. In such scenarios, the problem of target tracking has been analyzed at a fixed distance between the pursuer and evader [43], while in [44] a target tracking problem is analyzed with delay in sensing. In [47, 48], a shortest distance of escape (SDE) from the tracker's visibility is minimized for an unpredictable target, both for single and multiple trackers. The problem of keeping a point of interest in view by a limited field of view visual sensor has been addressed in [49, 50, 51] using a robot with non-holonomic constraints. A region based cellular decomposition is proposed in [52] for tracking multiple targets using multiple robots. Depending on the number of targets and the available robots a coarse deployment strategy is applied to the robots. At the individual level, the robots try to move towards the centroid of visible targets to maximize target surveillance. The choice of optimal motion direction can be done either in a deterministic manner [43, 28, 3], or by using randomized sampling strategies [48, 47, 53, 54]. A local visibility based pursuit evasion in a graph using randomized strategy is shown in [55]. For a partially known target motion models in a known environment, the target searching and target following problem can be integrated amid uncertain sensing and positioning information as a partially observable Markov decision process (POMDP) [56], which can then be solved to generate tracker actions.

A-priori information about the environment and the target helps in precomputing critical tracking scenarios. The tracker can execute smarter strategies exploiting the layout information to improve the tracking performance and to regain the target once it is lost from sight. In general such information is not always readily available. Moreover, too much dependence on a-priori information can be detrimental when such an information is outdated or faulty. Furthermore, before the tracker is able to utilize the information about the environment, it has to perform self localization. Localization itself presents a difficult problem in a dynamic environment.

Local Information Lack of information can be addressed in two ways. Firstly, to collect and build a global model the environment while tracking and compute motion strategies based on this global (although incomplete) map. Secondly, to plan based on the local information collected at each step.

The former approach is utilized in [57, 58] for pursuit evasion in unknown environments, where a two step approach is proposed, *exploration*: that involves mapping out the environment first using critical visibility events, and *envisioning*: where this mapped environment is searched in the information space encoded by the visibility events. This approach fails for unbounded environments, where complete mapping is not possible. For such situations the tracker has to rely on locally sensed information, the second approach as mentioned before.

One of the popular approaches for local reactive tracker is to combine vision and control in following the target, [59, 60, 2, 61, 62]. This has been referred to as *Visual Servoing*. The focus is to move closer to the target in order to improve the target's surveillance. In an unknown and unstructured environment, bayesian robot programming is proposed in [63], where significant information compression is possible by de-coupling motion and sensor processing. A Koala robot mounted with Pan/Tilt mechanism was shown to work successfully using a set of motion behaviors. Priors are defined based on the intuition that the robot should avoid obstacles when close to objects and track when far from the target. Motion to a desired goal position using visual servo is shown in [64] that uses a single camera using motion to disambiguate depth. [60] uses the optical flow to compute the displacements and a discrete steady state Kalman Filter to generate the motion control. The success in robust control is attributed to the computation of the states w.r.t. the local coordinates. Vision based tracking has been attempted for various kinds of targets, for example jellyfish in a marine environment [65] and people in office environment [66, 67]. Laser based approaches in people following have become popular in recent years. [27] models a person's gait by tracking both the legs in a simplified lab environment. Following the person of interest has been coupled with obstacle avoidance in a crowd in [68], while [25] combines both laser and camera for tracking robustly in outdoor and unstructured environments. Following a vehicle in forested roads has been shown in [69].

While visual servo based tracking algorithms are simple and easy to implement, it does not explicitly encode any information about the environment. Due to this it fails to react to impending occlusions and motion obstructions. We compare our algorithm with a simplified version of the servo tracker that minimizes its distance to the target.

In an ideal situation, the tracker should be aware of the current scene and perform intelligent tracking, by staying away from oncoming people and not block a door or passageway [70]. However, as mentioned earlier, such a tracker requires prior knowledge of the environment. Our goal is to generate such intelligent behavior using only local information. We adopt the approach of active sensing [71], that plans the motion strategy to compensate for and avoid occlusions. Notable work in this category is [3, 72] which builds a map based on its local visibility. The target can escape through the boundary of this visibility. Based on this map, escape paths that the target might take to escape from the tracker's visibility are computed. A data structure called escape path trees (EPT) is proposed that contains the shortest distance to escape (SDE) of the target to each escapable boundary of the tracker's visibility. An objective function, *risk*, which carries the intuition of the risk of target's escape from the tracker's visibility is formulated based on local parameters of the EPT. The tracker chooses its actions to minimize this risk. The escape paths encode the information of the target's position w.r.t. the current scene, and hence the tracker is able to keep the target in view better than visual servo controllers. This work however, does not consider the relative position of the target and the tracker, which plays a crucial part in determining the risk. We follow a similar risk based approach in target tracking, but propose an improved risk function that includes the relative positioning and show that this improves the tracking performance. We have implemented a version of the above mentioned tracker [3] and shall provide comparisons with our algorithm in chapter 3.

Many applications require additional objectives to be fulfilled while tracking, *e.g.* maintaining stealth [73, 74, 75, 76], improving localization [77], mapping the environment [26], human posture recognition [78] *etc.* We present a general tracking framework that integrates hardware limitations and these mission constraints into the problem of keeping the target in view.

2.2 3-D Tracking

While a lot of attention has been given to tracking problem in 2-D, there has been little work on the 3-D tracking problem. One reason is that the 3-D visibility relationships are significantly more complex than their 2-D counterparts. Although there are data structures for maintaining visibility relationships globally, *e.g.*, aspect graphs [79] and visibility complexes [80], processing all the critical visibility events efficiently in a 3-D environment is a difficult task. The work of Lazebnik tries to characterize and process these visibility events for a visibility-based pursuit-evasion problem [36]. It decomposes the space into conservative cells using a strategy similar to that proposed in [32]. In principle, it is possible to develop a tracking algorithm based on such a global visibility analysis, but to the best of our knowledge, such an algorithm has not yet been developed.

The existing algorithms on 3-D tracking and navigation mostly rely on visual servo control [81, 82]. Visual 3-D target tracking has been applied to underwater [65] and ground targets [83, 84] as well as aerial vehicles. The focus in aerial tracking has been in the development of control strategies to address flight limitations of aerial vehicles while trying to maintain a predesigned distance to a ground target [85, 86, 87]. Aerial tracking of target aircrafts have been addressed using camera [88] and radars [89]. Feature tracking and visual servo based navigation schemes in urban area have been explored in [81], while tracking and landing on a moving vehicle has been demonstrated successfully in [90]. While these approaches are able to control a team of unmanned aerial and ground vehicles for target tracking in a probabilistic game framework [82], they fail take into account the effect of visual occlusion by obstacles.

We follow a similar tracking approach to the 2-D tracking. A local map is computed based on the local 3-D visibility. The target's escape through the occlusions in this visibility is addressed and a motion plan is generated that minimizes the possibility of the target's escape. Interesting behaviors like increasing the altitude are automatically generated by exploiting the additional dimension while tracking. This is an improvement to approaches like [87], where such vertical behaviors are programmed and triggered based on preset criterion. Such pre-programming is not possible in general environments.

For both 2-D and 3-D unknown and dynamic environments, our tracker plans its motion based on local information to avoid visual occlusions while keep the target in view. In planning under a local map and using a one step greedy approach, our tracker can perform better than visual servo based approaches that do not account for environmental occlusions. The local focus of the algorithm makes it tractable in a complex and dynamic environment where complete planning may not be feasible.

CHAPTER 3

MOTION STRATEGIES: 2-D

Target tracking in an unknown and dynamic environment is challenging. In order to keep the target in view, the tracker must plan its motion using only local information. Objects in the environment create visibility and mobility constraints during tracking. Additional requirements like localization, mapping, stealth *etc.*, might be imposed on top of the basic tracking behavior making the problem even more difficult. In this chapter, we present a general framework for target tracking in unknown and dynamic environments among such conditions. An objective function, *risk*, is developed for both 2-D and 3-D environments, that encodes the danger of losing the target from sight. A tracking algorithm is proposed that minimizes it in the local context satisfying the physical and mission constraints. Such an algorithm is formalized in 2-D and a real tracking robot is built that successfully follows a person in a crowded school cafeteria. The 3-D algorithm developed on the same principles is addressed in Chapter 5.

3.1 Problem Formulation

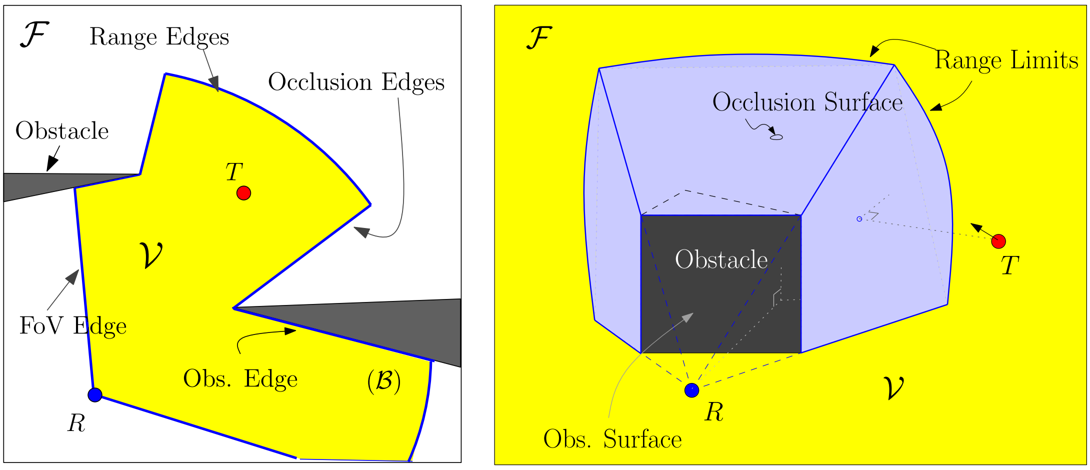
Let us introduce the approach using a simple abstract Euclidean world $\mathcal{W}$ containing rigid obstacles. The tracker (R) and the target (T) are assumed point objects in $\mathcal{W}$. Irrespective of the context, either 2-D ($\mathcal{W} = \Re^2$) or 3-D tracking ($\mathcal{W} = \Re^3$), the concepts developed are the same. The tracker does not have any information about the environment ($\mathcal{W}$) or about the motion of the target apart from what it can sense locally. The tracker has to keep the target in view using visibility based sensors.

3.1.1 Visibility Model

We use the standard straight-line of sight visibility model for the tracker's sensor. Let $\mathcal{F}$ denote the subset of $\mathcal{W}$ not occupied by obstacles. The target is *visible* to the tracker if the line of sight between them is free of obstacles, and the distance between them is less than D_{max}, the maximum sensor range. The *visibility set* $\mathcal{V}(\mathbf{x})$ of the tracker at position $\mathbf{x}$ consists of all the points $\mathbf{q}$ in $\mathcal{F}$, such that the interior of the line segment from $\mathbf{x}$ to $\mathbf{q}$ ($\overline{\mathbf{xq}}$), does not intersect with the boundary of the environment.

$$\mathcal{V}(\mathbf{x}) = \{\mathbf{q} \in \mathcal{F} \mid \overline{\mathbf{xq}} \subset \mathcal{F} \text{ and } dist(\mathbf{x}, \mathbf{q}) \leq D_{max}\},$$

where $dist(\mathbf{x}, \mathbf{q})$ denotes the distance between $\mathbf{x}$ and $\mathbf{q}$. If the sensor has a minimum range D_{min}, we can impose the additional constraint $dist(\mathbf{x}, \mathbf{q}) \geq D_{min}$. Also most of the visibility sensors have field of view (FoV) limitations, e.g a 2-D sicklms-200 laser range finder has a FoV of $[-90, 90]$ in the horizontal plane, while a 3-D Velodyne HDL-64E range sensor has FoV of $[-26, 2]$ in the vertical plane. Such a limitation can be modeled by adding an additional constraint of the visibility to be bounded by $[\theta_{min}, \theta_{max}]$ for each FoV limit. In the following, the visibility set is always taken with respect to the current tracker position, so we omit the argument $\mathbf{x}$. On real robot trackers, the visibility region is obtained by processing sensor data. In simulation, the visibility region can be computed with a rotational plane sweep algorithm [91]. Either way, after processing, the visibility region $\mathcal{V}$ is represented as a generalized-polygon.



(a) Local visibility for 2-D

(b) Visibility occlusion for a single polyhedra in 3-D

Figure 3.1. The visibility models for line of sight in 2-D, 3-D polygonal environment.

Environment representation There is no explicit representation or map of the environment maintained by the tracker. The boundary of $\mathcal{V}$, $\partial\mathcal{V}$, essentially encodes the locally sensed environment. $\partial\mathcal{V}(\mathbf{x})$ constitutes of points on the obstacles boundaries, $\mathcal{B}(\mathbf{x})$, and points in free space boundaries. $\mathcal{B}(\mathbf{x})$ can be extracted from the sensor readings not on the sensing limits.

$$\mathcal{B}(\mathbf{x}) = \{\mathbf{q} \in \partial\mathcal{V} \mid dist(\mathbf{x}, \mathbf{q}) \neq D_{max}\}$$

As before, we drop $\mathbf{x}$, with the understanding that the parameters are defined for the tracker position.

Escape Gaps Any portion of $\partial\mathcal{V}$ lying in $\mathcal{F}$ can be used by the target to exit the tracker's visibility. Due to this they play a central role in planning the tracking strategy. We define them as *Escape Gaps*. Escape gaps $(\mathcal{G})$, are of three types,

1. *Occlusion gaps*: generated when the line of sight is obstructed by an obstacle,

2. *FoV gaps* : gaps due to the field of view limits of the sensor,

3. *Range gaps* : gaps due to the range limitation on the sensor.

The tracker's visibility along with escape gaps, is shown in Figure 3.1 for 2-D and 3-D. In 2-D, escape gaps are curvilinear edges, while in 3-D they take the form of curvilinear surfaces.

The tracker's motion is modeled with a simple discrete-time transition equation. Let $\mathbf{x}(t)$ denote the position of the tracker at time t. If it chooses a velocity $\mathbf{v}(t)$ at time t, its new position $\mathbf{x}(t+1)$ after a fixed time interval Δt is given by

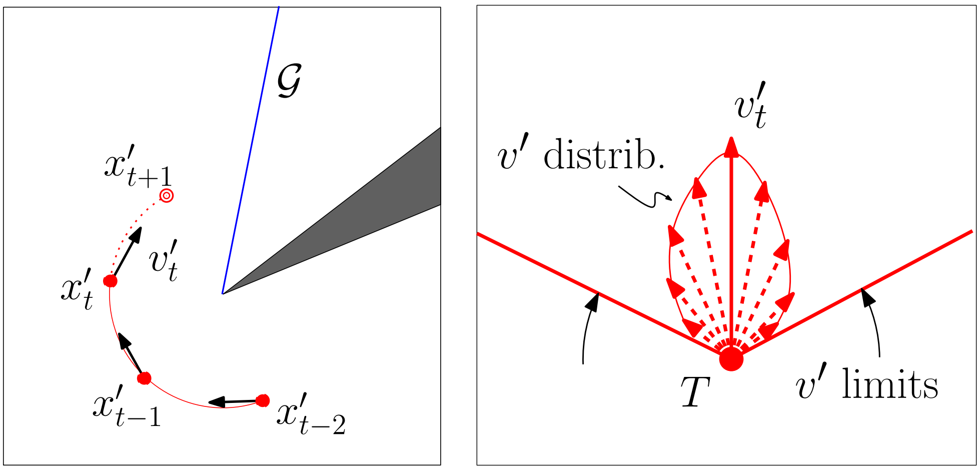
$$\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{v}(t)\Delta t$$

Here, we implicitly assume that sensing occurs every Δt time and that $\mathbf{v}(t)$ is constant during this time. This discrete model is effective as long as Δt remains small. As we will see, our tracking strategy is very efficient and runs at the rate of 10Hz, sufficient for keeping Δt small in many common tasks. We define a region that can be reached by the target from its current position in Δt as a *Reachable Region* ($\mathcal{R}$).

In the following discussion, for the sake of simplicity the tracker is assumed to have a velocity bound V, but has no other kinematic or dynamic constraints. Hence, in one time step, it can reach anywhere inside a N-spherical region, with center $\mathbf{x}(t)$ and radius $V\Delta t$, unless it is obstructed by obstacles.

3.1.2 Motion Model: Target

The target's motion is unknown, but has velocity bound V'. We are interested in cases when $V \geq V'$. Otherwise, the target can easily escape by running straight ahead with maximum velocity and the tracking problem is uninteresting. Predicting



(a) Most likely estimate based on a short history

(b) Velocity distribution about the most likely estimate

Figure 3.2. Predicting a target's next step.

the target's next step, or equivalently its current velocity, can be done in many ways. While complex algorithms like hidden Markov models (HMM) [92] or artificial neural networks (ANN) [93, 94] might predict the motion patterns more accurately, they usually require more information about the target and the environment or have to maintain a significant history.

We choose a simple model based on the target's current heading. Biasing the target's motion on its heading is based on the intuition that although the target moves in an unknown fashion, its heading gives a fair idea of the target's next step. The target's heading is extrapolated based on target's current and previous positions. For example, the two previous positions can be used together with the current position to fit a quadratic trajectory and estimate the heading. This crude model is surprisingly robust in practice for small Δt.

An estimate of the target's heading is represented by a Gaussian distribution $\mathcal{N}(mean, var)$, and its current heading mean bounded with a span (var) of probabilistic heading. In general the span can be 360deg.

$$\begin{aligned} \mathbf{x}'(t+1) &= \mathbf{x}'(t) + w(t)\Delta t \\ w(t) &\sim \mathcal{N}(\mathbf{v}'(t), \sigma^2) \end{aligned}$$

where σ gives the measure of confidence in estimating the target's heading. In the absence of a-priori knowledge, σ can be set arbitrarily high and with a better target model or by learning the motion model [95], σ can be reduced subsequently.

Although we predict the motion of the target from its heading, in the absence of heading information, other approaches like modeling the target motion by Brownian motion models could also be applied in general. In fact, during our implementation of following a person (discussed later in this chapter), a simple laser scan is unable to disambiguate the target's heading and we model the target's next step by a circular bound about its current position.

3.1.3 Problem Statement

The tracking problem can then be described formally as : *For the current tracker position,* $\mathbf{x}$ (t) *and target position* $\mathbf{x}'$ (t)*, find an action* $\mathbf{v}$ (t)*, such that* $\mathbf{x}(t+1) = \mathbf{v}(t)\Delta t \in \mathcal{R}$ *and* $\mathbf{x}'(t+1) \in \mathcal{V}(\mathbf{x}(t+1))$ *for as long as possible.*

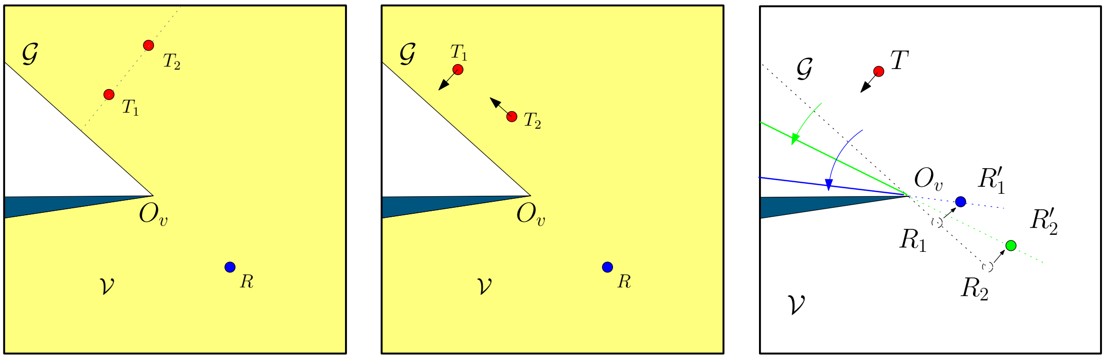
3.2 Overview of Tracking Approach

For unknown and dynamic environments, we propose a general tracking framework that provides a mechanism to integrate various sensing, mobility and operational limitations while trying to keep the target in view for as long as possible.

In a dynamic and unknown environment, a fast online tracking algorithm is required that can quickly adapt to the changing environment. Due to a lack of prior information, such an algorithm has to utilize local information from the on-board sensors in order to plan its motion. Objects in the environment can obstruct the line of sight to create visibility occlusions. Sensor limitations, like field of view (FoV), further reduce the region visible to the tracker and increases the chances of the target's moving out of the tracker's view. While the goal of the tracker is to keep the target in its visibility, $\mathcal{V}$, for as long as possible, the boundary of the visibility, $\partial\mathcal{V}$, has a number of *escape gaps*, $\mathcal{G}$, through which the target may escape. We propose a scalar function $Risk(\Phi)$, that tries to capture the danger of losing the target from $\mathcal{V}$ through these escape gaps. Φ is formulated based on the geometric parameters extracted from its local visibility, *e.g.* the tracker and target position, their relative velocities and distance from the escape gaps. Φ is an aggregated measure due to all the individual escape gaps. Since $\mathcal{V}$ and hence the escape gaps, depend on the tracker's current position, the tracker can choose its actions $\mathbf{v}(t)$, for each time step, such that the escape gaps are moved away from the target in an intelligent manner and Φ is minimized. Let $\mathbf{v}^\star$ be the optimal action that minimizes Φ,

$$\mathbf{v}^\star = \arg\min_{\mathbf{v}} [\Phi]$$

The choice of $\mathbf{v}^\star$, however, has to satisfy the tracker motion model, *i.e.* non-holonomic limitations, maximum velocity bound *etc.*. Some velocities that lead to collision with obstacles are also not feasible. In addition, there might be operational restrictions on the positions that the tracker may be allowed to take, *e.g.* the tracker might have to maintain a minimum distance from all humans etc. Each of these limitations on the tracker's mobility can be represented by individual constraints,



(a) Target position (b) Target velocity (c) Robot position.

Figure 3.3. The factors affecting the risk of losing the target from local visibility. In (c) $\mathcal{V}$ is not shaded for clarity

$\mathcal{C}_i$. These constraints can be combined with the tracker's reachability region $\mathcal{R}$ in position space. We define a *feasible region*, $\mathcal{L}$ (x),

$$\mathcal{L}(x) = \mathcal{R} - \bigcap_i \mathcal{C}_i$$

Our approach of target tracking among unknown and dynamic environments is thus cast as a local greedy optimization problem of choosing an action $\mathbf{v}^\star$, that minimizes Φ while constraining the tracker in $\mathcal{L}$ for the time step Δt,

$$\mathbf{v}^\star = \arg\min_{\mathbf{v}} [\Phi] \qquad s.t. \quad \mathbf{v}^\star \Delta t \in \mathcal{L} \tag{3.1}$$

3.3 Tracking Risk

Factors affecting the risk In the following we try to understand what factors influence the notion of the risk of losing the target. For simplicity and clarity in depiction we choose an example from 2-D. In general a similar case can be made in

3-D. In the following discussion we give an intuition using 2-D examples while the discussion and concepts are developed independently of the dimensionality.

Let us take a look at a simple case of a single escape gap, $\mathcal{G}$ in Figure 3.3. The target, T, tries to escape the tracker visibility, $\mathcal{V}$, by escaping through the gap, $\mathcal{G}$. Clearly the risk of losing the target from $\mathcal{V}$, depends on the target's distance to $\mathcal{G}$. The closer the target is to the gap, the easier it is for the target to escape. In (Figure 3.3a) due to its proximity to $\mathcal{G}$, T_1 can be perceived to pose a higher risk to the tracker than T_2. Additionally, the target's heading also influences the notion of risk of losing the target. In (Figure 3.3b), both T_1 and T_2 are at the same distance from $\mathcal{G}$. However, the consideration of the their relative velocity towards $\mathcal{G}$ puts T_1 at a higher risk than T_2. The risk also depends on the current position of the tracker. In Figure 3.3c, R_1 being closer to the obstacle, is able to swing $\mathcal{G}$ away from the target more effectively than R_2. This means that at R_1 the tracker is at a lower risk of losing the target than at R_2 because it can manipulate the escape gap away from the target in a more effective manner.

Current risk vs Future risk The tracker has to plan its actions (in this context its motion) such that it is able to minimize this risk. As seen in Figure 3.3, swinging $\mathcal{G}$ counterclockwise away from the target can minimize the risk due to the target's position and its relative velocity to $\mathcal{G}$. Such a swinging action decreases the immediate danger of losing the target, which we refer to as *current risk*. However, we notice that if the tracker were to move from R_2 towards R_1, it would be able to swing better in the future and even eliminate $\mathcal{G}$ by reaching $\mathcal{O}_v$, thereby lowering the risk of losing the target in the future, *future risk*, much more effectively. However, the current risk remains unchanged or might even increase if the target moves closer to $\mathcal{G}$ while the tracker is moving from R_2 to R_1. This poses the tracker with the dilemma of choosing actions to balance decreasing the current risk vs the future risk. Given the limited sensing and unknown dynamic environment, such a choice is not trivial.

Relative Vantage We propose the concept of *relative vantage* with respect to a given gap, to encode this dilemma of current vs future into a single objective. *Relative vantage refers to the tracker having a strategically better position compared to the target.* Instead of worrying about the short-long term dilemma, the tracker only has to consider keeping itself in a position that has a better relative vantage. As long as there is a guarantee that attaining such a relative vantage will ensure tracking success in the present and the future, the tracker's primary objective should be to plan for attaining such a vantage position. This idea follows from the simple intuition that the risk of losing the target is low, both in the short term and long term, if the tracker has a strategically superior position.

Let us take a simple example to illustrate the concept of relative vantage. The target and the tracker are in a convex room with a single exit as shown in Figure 3.4. Let us also assume that the tracker's visibility covers the entire room. The target is trying to escape through the exit in the room and the tracker is trying to prevent it from escaping by closing the door. Assuming the same maximum velocities of the target and the tracker, we see that the tracker at R being closer than T_1 can reach the exit faster. In this case there is at least one action, that of moving straight towards the exit, which will guarantee preventing T_1's exit. In this case we say that R has a better strategic position w.r.t. T_1. Clearly, this is not true for T_2. The position of

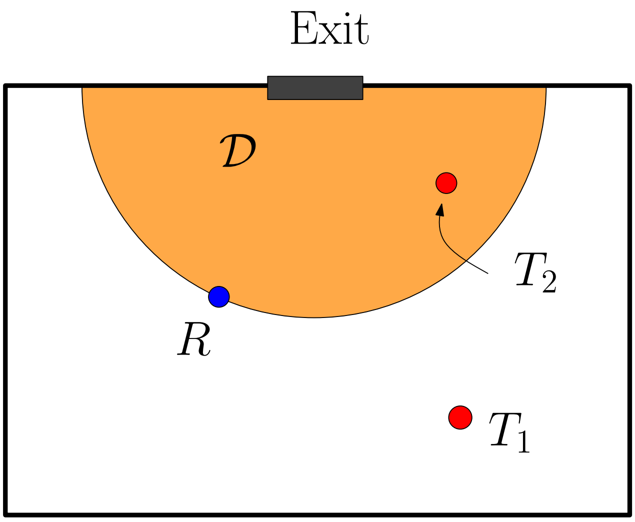


Figure 3.4. Relative vantage: The shaded region is $\mathcal{D}$. The tracker R has a relative vantage over T_1, and not w.r.t. T_2

the tracker R w.r.t. the exit partitions the space into dangerous and safe regions in light of the target's ability to escape.

Danger zone In the context of local target tracking, the escape gaps represent the exit. The tracker can prevent the target's escape by reaching and eliminating $\mathcal{G}$. This gives a similar partition around each gap into regions of safety and danger. We say that all positions of the tracker, from where the tracker can eliminate $\mathcal{G}$ before the target can escape through it, are strategically superior positions and hence have relative vantage over the target.

Such a partition of $\mathcal{V}$ is shown in Figure 3.5a. The green shaded area depicts the region where the tracker does not have a relative vantage over the target. We call this region as *danger zone*, $\mathcal{D}$. As long as the target is outside the region $\mathcal{D}$, the target is guaranteed not to be able to escape. The proof is shown in Section 3.4.1.

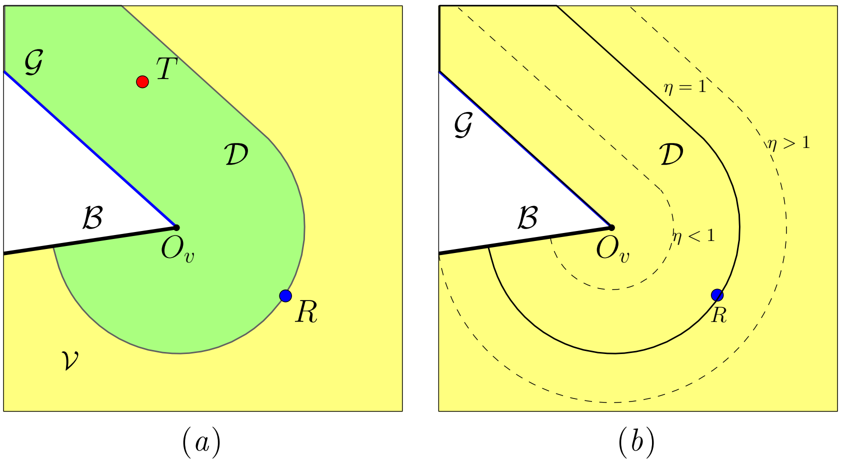


Figure 3.5. Danger zone, $\mathcal{D}$ defined for an occlusion edge $\mathcal{G}$, $(\eta = 1)$. The target is inside $\mathcal{D}$ and so the tracker does not have a relative vantage to the target.

Mathematically,

$$\mathcal{D} = \{\mathbf{q} : \mathbf{q} \in \mathcal{V} \bigwedge time(\mathbf{q}, \mathcal{G}) < time(R, \mathcal{G})\} \tag{3.2}$$

where $time(\mathbf{q}, \mathcal{G})$ denotes the time taken by a target at position $\mathbf{q}$ to reach $\mathcal{G}$, and $time(R, \mathcal{G})$ for the tracker to reach $\mathcal{G}$. If both the target and the tracker take the shortest path to $\mathcal{G}$, the condition becomes,

$$\begin{aligned} time(\mathbf{q}, \mathcal{G}) &< time(\mathbf{x}, \mathcal{G}) \\ dist(\mathbf{q}, \mathcal{G}) &< \eta\, dist(R, \mathcal{G}) \end{aligned}$$

where η is the ratio of the maximum velocity of the target to the tracker, $\eta = V'/V$. We draw $\mathcal{D}$ for various η in Figure 3.5b. In this Chapter and later, we use $\eta = 1$ without loss of generality.

The target must first enter $\mathcal{D}$ before it can escape through $\mathcal{G}$. As long as the tracker is able to manipulate $\mathcal{D}$ such that the target remains outside, the target cannot reach $\mathcal{G}$ and the tracking is guaranteed. The objective of keeping the target outside $\mathcal{D}$ resolves the dilemma of minimizing the current risk vs the future risk. The basis of our tracking approach is to always try to keep the target outside $\mathcal{D}$ and the time taken by the tracker to manipulate $\mathcal{D}$ to bring the target outside gives a measure of the tracking risk (φ).

Vantage time The primary risk factors, *i.e.* the target position, the target heading and the tracker position, can all be incorporated into the term called *vantage time* ($t_{r.v}$). When the target is inside $\mathcal{D}$, $t_{r.v}$ gives the measure of time to move the target to the boundary of $\mathcal{D}$.

Mathematically,

$$Dist(T, \mathcal{D}) \quad = \quad \int_0^{t_{r.v}} V_{eff}(T, \mathcal{D})dt \tag{3.3}$$

where $Dist(T, \mathcal{D})$ gives the distance of the target from the boundary of $\mathcal{D}$. The distance is positive when the target is inside and negative when outside. Negative values of φ are not interesting since the target is guaranteed not to escape when it is outside. $V_{eff}(T, \mathcal{D})$ denotes the relative velocity with which $\partial\mathcal{D}$ approaches the target inside $\mathcal{D}$. The velocity of $\partial\mathcal{D}$ can be computed from the velocity of a representative point on the $\partial\mathcal{D}$. Computing $t_{r.v}$ from Equation 3.3 in its exact form requires the knowledge of the target motion. This is not available for most targets. We approximate this equation by Equation 3.4, extrapolating the current velocity of the target for Δt:

$$t_{r.v} \approx \frac{Dist(T, \mathcal{D})}{V_{eff}(T, \mathcal{D})} = \varphi \tag{3.4}$$

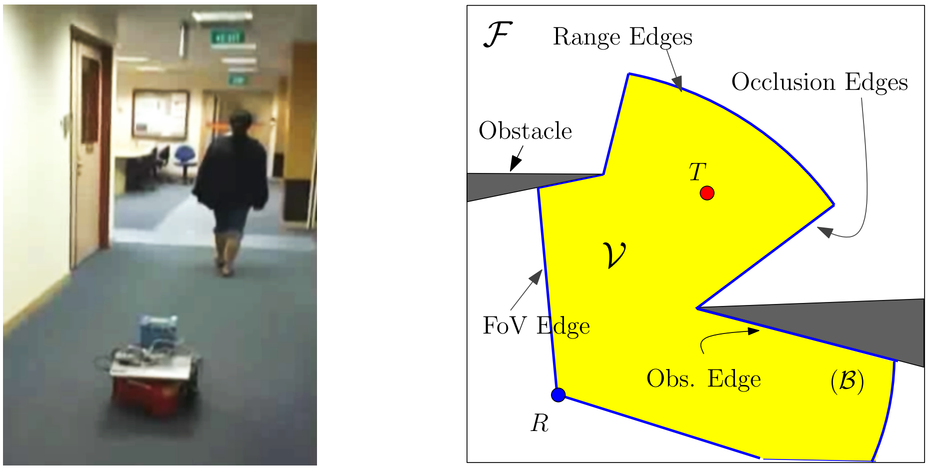
Such an approximation does not introduce a large error as the algorithm recomputes this value at a high rate by keeping Δt small. We use the approximated vantage time, $t_{r.v}$, as the measure of risk (φ) of losing the target. The risk is positive when the target is inside $\mathcal{D}$ and negative outside. Negative risk just denotes that the target is safely in view.

Escape time Range gaps and FoV gaps are generated due to the sensor's limitations and remain fixed relative to the tracker's frame. Hence the tracker's motion affect these gap. In such a situation, the definition of $t_{r.v}$ is redundant. We instead use the *escape time*, t_{esc}, which is defined as the measure of time taken for the target to escape given the current position, velocity of the tracker and the target. Similar to $t_{r.v}$, t_{esc} can be defined as,

$$\begin{aligned} Dist(T, \mathcal{G}) &= \int_0^{t_{esc}} V_{eff}(T, \mathcal{G}) dt \\ t_{esc} &\approx \frac{Dist(T, \mathcal{G})}{V_{eff}(T, \mathcal{G})} \end{aligned} \tag{3.5}$$

For range and FoV gaps, t_{esc} is used as a measure of risk. However, t_{esc} can be defined for occlusion gaps too.

Vantage time vs Escape time $t_{r.v}$ gives a measure of how much time is required by the tracker to attain relative vantage and t_{esc} gives a measure of how much time the tracker has before the target escapes. Note that t_{esc} gives a measure of the current risk alone where as $t_{r.v}$ gives a combined short-term vs long-term estimate of the risk. In critical situations where the target is too close to the edge and losing it is unavoidable, the tracker should maximize the remaining time for viewing the target by maximizing t_{esc}. This is achieved by optimizing $t_{r.v}$ conservatively for occlusion



(a) Tracking scenario for 2-D (b) Geometrical abstraction for 2-D tracking

Figure 3.6. A 2-D tracking scenario.

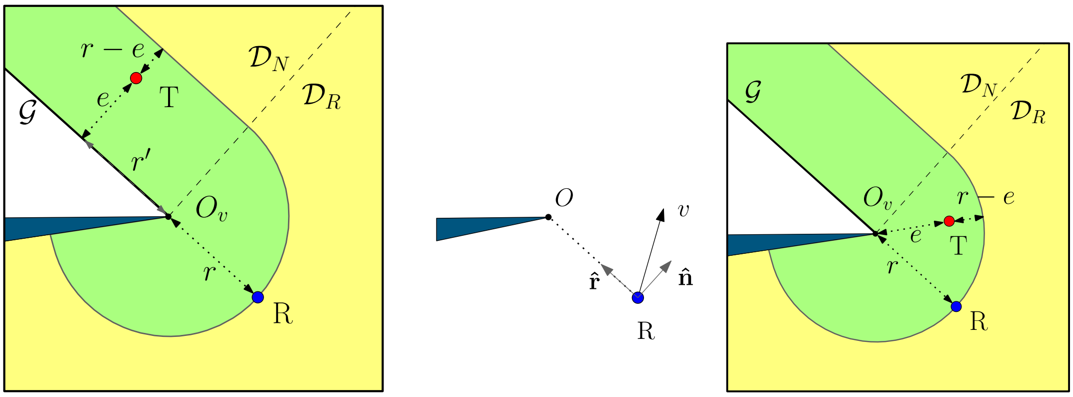
gaps when $t_{r.v} < t_{esc}$, otherwise t_{esc} is optimized. t_{esc} is also optimized for FoV or range gaps.

The concepts defined here, are independent of the dimensionality of the Euclidean space they occupy. They can be used not only in 2-D tracking as shown in this Chapter, but also in 3-D as will be shown in Chapter 5.

3.4 Computing risk analytically for 2-D

For the rest of the Chapter we will focus on 2-D tracking (Figure 3.6). A typical tracking scenario of following a person in an office environment is shown in Figure 3.6a. A geometrical abstraction is shown in Figure 3.6b. In the 2-D context the escape gaps are actually curvilinear-linear edges. Let us now derive the expressions for Equation 3.4 for a single escape edge. As mentioned earlier the target has to be prevented from escaping through occlusion, range and FoV escape edges. Let us first consider the more interesting type of gap edges, occlusion edges. Let $\mathcal{G}$ represent an occlusion edge.

3.4.1 Occlusion edges



(a) (b)

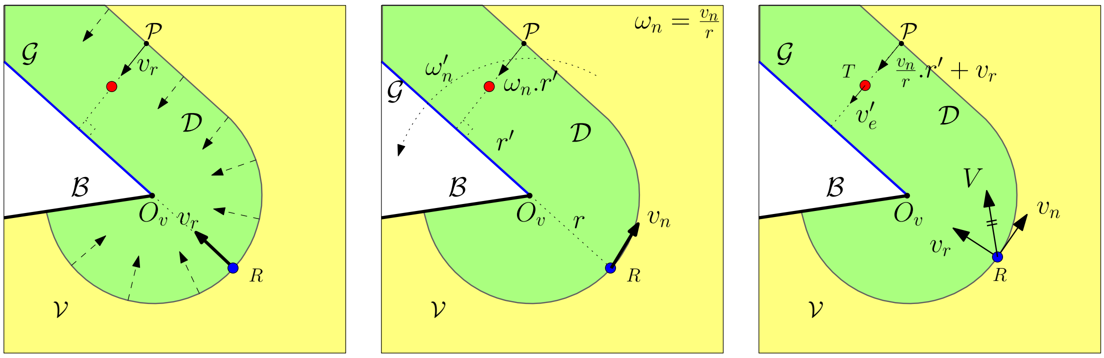
Figure 3.7. Calculating $t_{r.v}$ for occlusion edges. $\mathcal{D}_N$ and $\mathcal{D}_R$ are written outside $\mathcal{D}$ for clarity, although they represent components of $\mathcal{D}$.

To get an analytical formula for Equation 3.4 & Equation 3.5, let us define some parameters as shown in Figure 3.7. Let $\mathcal{O}_v$ denote the obstacle vertex abutting $\mathcal{G}$ (Figure 3.7). Based on the proximity to either the $\mathcal{G}$ or the $\mathcal{O}_v$, we can divide the region in $\mathcal{D}$ into $\mathcal{D}_N$ and $\mathcal{D}_R$ (Figure 3.7b).

$$\begin{aligned}\mathcal{D}_N &= \{\mathbf{q} \in \mathcal{D} | nearest(\mathbf{q}, \mathcal{G}) \in (\mathcal{G} - \mathcal{O}_v)\} \\ \mathcal{D}_R &= \{\mathbf{q} \in \mathcal{D} | nearest(\mathbf{q}, \mathcal{G}) = \mathcal{O}_v\}\end{aligned}$$

where $nearest(\mathbf{q}, \mathcal{G})$ computes the closest point on the $\mathcal{G}$ to $\mathbf{q}$.

CASE I: $\mathcal{D}_N$ Let e and r denote the shortest distance of the target and the tracker from $\mathcal{G}$ respectively. r' measures the distance from $\mathcal{O}_v$ to the foot of the perpendicular



(a) v_r (b) v_n (c) combined effect

Figure 3.8. The effect of tracker motion on $\mathcal{D}$.

dropped from the target to $\mathcal{G}$. Assuming the target is inside $\mathcal{D}$, the distance of the target from the outer boundary of $\mathcal{D}$ is

$$Dist(T, \mathcal{D}) = r - e$$

As mentioned earlier, $\mathcal{V}$ and hence $\mathcal{D}$ depends on the tracker's motion. We decompose the tracker's velocity ($\mathbf{v}$) into components parallel (v_r) and perpendicular (v_n) to $\mathcal{G}$. Let $\mathcal{P}$ be a point on $\partial\mathcal{D}$ that is closest to the target. $\mathcal{P}$ has the highest relative velocity of all points on $\partial\mathcal{D}$ towards the target. We shall choose $\mathcal{P}$ to compute our estimated vantage time.

As shown in Figure 3.8, v_r moves the tracker closer to $\mathcal{O}_v$ in Figure 3.8a. This shrinks $\mathcal{D}$ towards $\mathcal{G}$ at the rate of v_r providing an additional velocity of v_r towards the target. v_n causes $\mathcal{G}$ to rotate about $\mathcal{O}_v$ with the angular velocity $\omega = v_n/r$. Acting as a rigid body, $\mathcal{D}$ rotates with the same angular velocity. The angular velocity ω, gives a linear velocity $(v_n(r/r'))$ to $\mathcal{P}$ towards the target.

Denoting the velocity component of the target along the shortest path to $\mathcal{G}$, (v'_{e}), we get the relative velocity of $\mathcal{D}$ to the target as

$$V_{eff}(T,\mathcal{D}) \;=\; \frac{v_n}{r}r' + v_r - v'_{\mathrm{e}}$$

This gives the risk as

$$t_{r.v} = \varphi = \frac{Dist(T,\mathcal{D})}{V_{eff}(T,\mathcal{D})} = \frac{r-e}{\frac{v_n}{r}r' + v_r - v'_{\mathrm{e}}} \tag{3.6}$$

The choice of v_n and v_r should be restricted to the condition where the $\mathcal{D}$ "gains" on the target, $(v_n r' + v_r r - v'_{\mathrm{e}} r > 0)$. Moreover, as mentioned earlier, in order to be conservative in our approach, this optimization should be constrained by the condition, $t_{r.v} < t_{esc}$. We define t_{esc} as

$$t_{esc} = \frac{Dist(T,\mathcal{G})}{V_{eff}(T,\mathcal{G})} = \frac{e}{v'_{\mathrm{e}} - \frac{v_n}{r}r'}$$

The constraint then reduces to,

$$\begin{aligned} t_{esc} - t_{r.v} &> 0 \\ ev_r + v_n r' - rv'_{\mathrm{e}} &> 0 \end{aligned}$$

This makes the optimization as,

$$\begin{aligned} \mathbf{v}^{\star}(v_r, v_n) \;&=\; \arg\min_{(v_r,v_n)} \left(\frac{r-e}{\frac{v_n}{r}r' + v_r - v'_{\mathrm{e}}}\right) \\ \text{subject to} \quad & |\mathbf{v}^{\star}| < V \\ & ev_r + v_n r' - rv'_{\mathrm{e}} > 0 \\ & v_n r' + v_r r - v'_{\mathrm{e}} r > 0 \end{aligned} \tag{3.7}$$

Hence, for a single gap edge $\mathcal{G}$, target tracking reduces to minimizing the risk function φ. Note that from Equation 3.6, assuming all other parameters constant,

φ is a function of (v_n and v_r). Thus the action to minimize the local estimate of φ would be to move along the negative gradient of φ computed with respect to v_n and v_r:

$$-\nabla\varphi = \frac{\varphi}{v_{\text{eff}}}\left(\frac{r'}{r}\hat{\mathbf{n}} + \hat{\mathbf{r}}\right). \tag{3.8}$$

In Equation 3.8, $\hat{\mathbf{n}}$ and $\hat{\mathbf{r}}$ are unit vectors in the tangential and radial directions respectively, and $v_{eff} = v_r + v_n(r'/r) - v'_{\text{e}}$ is the effective velocity in the direction along the shortest path from the target to $\mathcal{G}$.

The tracker's action $\mathbf{v}$ with respect to $\mathcal{G}$ is simply $-\nabla\varphi$. Equation 3.8 shows that the direction of v is $(1/\sqrt{r^2 + r'^2})(r'\hat{\mathbf{n}} + r\hat{\mathbf{r}})$. It depends only on r and r', which intuitively measures the tracker's and the target's abilities to swing the visibility line $\mathcal{G}$ against each other. When r is smaller than r', swinging is effective. Thus, the tangential component gets higher weight. When r is larger than r', the opposite holds. The magnitude of $\mathbf{v}$ acts as a weight when there are multiple gap edges. It depends on all three quantities, r, r', and e. In particular, when e is small with respect to a gap edge $\mathcal{G}$, φ becomes large. Thus, $-\nabla\varphi$ becomes large according to Equation 3.8, and the corresponding action gets higher weight.

CASE II: $\mathcal{D}_R$ This corresponds to region $\mathcal{D}_R$ marked in Figure 3.7b, and the closest point in $\mathcal{G}$ to the target is the occlusion vertex $\mathcal{O}_v$. The e and r is now defined with respect to the circular section of $\mathcal{D}$, and v'_{e} is along the radial direction to $\mathcal{O}_v$. The tangential component v_n has no effect on this part of $\mathcal{D}$. By definition Equation 3.4,

$$\begin{aligned}\varphi &= \frac{Dist(T,\mathcal{D})}{V_{eff}(T,\mathcal{D})} \\ &= \frac{r-e}{v_r - v'_{\text{e}}}\end{aligned} \tag{3.9}$$

Again we should restrict v_r by $v_r - v'_{\mathrm{e}} > 0$. This makes the optimization as,

$$\begin{aligned} \mathbf{v}^{\star}(v_r, v_n) \quad &= \quad \arg \min_{(v_r, v_n)} \left(\frac{r-e}{v_r - v'_{\mathrm{e}}} \right) \\ \text{subject} \quad \text{to} \quad & \quad |v_i^{\star}| < V \\ & \quad e v_r - r v'_{\mathrm{e}} > 0 \\ & \quad v_r > v'_{\mathrm{e}} \end{aligned} \tag{3.10}$$

The corresponding gradient of φ is then

$$-\nabla\varphi = \frac{\varphi}{v_{\mathrm{eff}}}\hat{\mathbf{r}}, \tag{3.11}$$

where v_{eff} is still the effective velocity in the direction along the shortest path from the target to $\mathcal{G}$, but this time, it is directed towards $\mathcal{O}_v$ and is equal to $v_r - v'_{\mathrm{e}}$.

An interesting observation is that Equation 3.11, can be obtained from Equation 3.8 by placing $r' = 0$ *i.e.* when physically the moment arm of the target is zero. Clearly in such a situation swinging does not help and so we get an action directed in v_r direction. Together, Equation 3.8 and Equation 3.11 reveal that the tracker's action is continuous over the entire domain, if the target's action is continuous too. This gives an important advantage in practice for smooth control of the tracker.

Guarrantee of tracking

Let us now show mathematically that the target tracking is guaranteed for a single escape edge once the target is outside $\mathcal{D}$ for that $\mathcal{G}$. Although we derive the proof for a 2-D scenario, similar arguments could be proposed in the 3-D as well.

In Figure 3.9, the dashed lines denote the boundary of $\mathcal{D}$, $(\partial\mathcal{D})$. Since $\mathcal{D}$ completely envelops the gap edge by definition, the only way for the target to escape is to enter $\mathcal{D}$. We can prove the target is kept in view if we show that at each step,

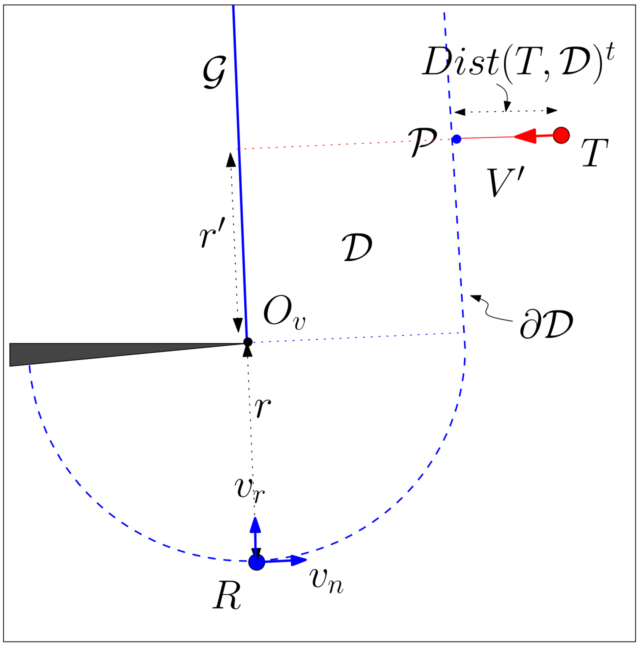


Figure 3.9. Deriving tracking guarrantee for a single occlusion edge.

the distance between $\partial\mathcal{D}$ and T is non-decreasing for each time step. We show below that even though the maximal velocities of the tracker and the target are the same, the tracker can "gain" on the target.

Theorem 1 *For each time step Δt, for $V' \leq V$, $Dist(T,\mathcal{D})^{t+\Delta t} \geq Dist(T,\mathcal{D})^t$, where $Dist(T,\mathcal{D})^t$ denotes the shortest distance between $T$and $\mathcal{D}$ at time t.*

Proof: Let us assume that the target moves towards $\partial\mathcal{D}$ at its maximum speed (V') as shown in the Figure 3.9. The case of $V' < V$ is trivial. We focus here on the case where $V' = V$ and $V = \sqrt{{v_n}^2 + {v_r}^2}$. Let the velocity of $\mathcal{P}$ be denoted by $v_{\partial\mathcal{D}}$.

$$\begin{aligned} v_{\partial\mathcal{D}} &\geq V' \\ v_r + v_n\frac{r'}{r} &\geq V' \end{aligned}$$

$$\begin{aligned} {v_r}^2 + {v_n}^2\frac{r'^2}{r^2} + 2v_r v_n\frac{r'}{r} &\geq {v_r}^2 + {v_n}^2 \qquad \text{taking sq} \\ v_n\frac{r'^2 - r^2}{r^2} + 2v_r\frac{r'}{r} &\geq 0 \end{aligned} \tag{3.12}$$

If $r' > r$, Equation 3.12 holds as the LHS is a positive number. For $r' < r$, if the tracker chooses its velocities as,

$$\begin{aligned} v_r &= V\frac{r}{\sqrt{r^2 + r'^2}} \\ v_n &= V\frac{r'}{\sqrt{r^2 + r'^2}} \\ \frac{v_r}{v_n} = \frac{r}{r'} \end{aligned} \tag{3.13}$$

Combining Equation 3.12 with Equation 3.13 gives,

$$\begin{aligned} 2\frac{v_r}{v_n}r'r + r'^2 - r^2 &\geq 0 \\ r^2 + r'^2 &\geq 0 \end{aligned} \tag{3.14}$$

in which case Equation 3.12 holds again.

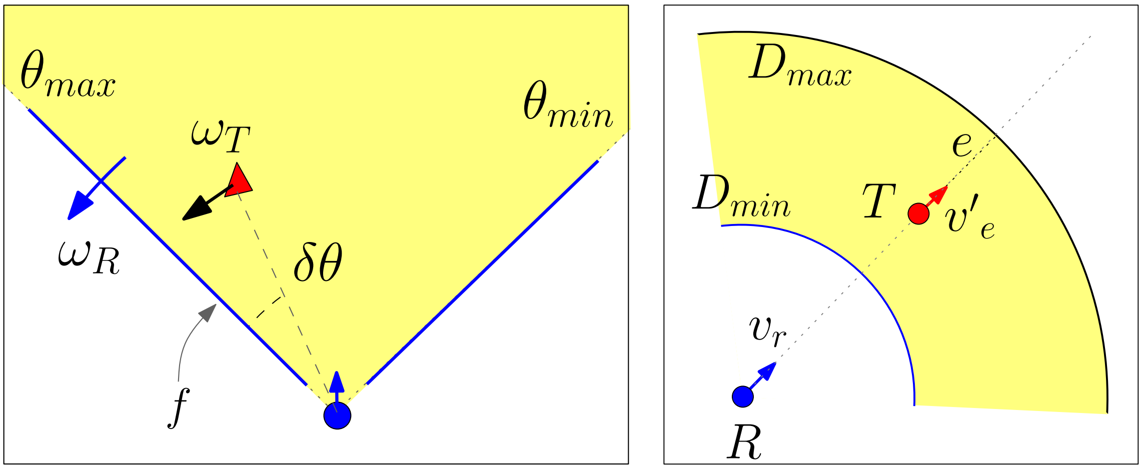
Hence there is at least one choice of v_r, v_n for the tracker to move $\partial\mathcal{D}$ faster than $\mathbf{v}'$. This means that at each time step there is at least one action from the tracker, s.t.

$$\begin{aligned} (v_{\partial\mathcal{D}} - V')\Delta t &\geq 0 \\ v_{\partial\mathcal{D}}\Delta t - V'\Delta t &\geq 0 \\ Dist(T, \mathcal{D})^{t+\Delta t} - Dist(T, \mathcal{D})^{t} &\geq 0 \end{aligned} \tag{3.15}$$

This proves that the target will not be able to reach $\partial\mathcal{D}$ once it is outside $\mathcal{D}$ as the distance between them can be consistently prevented from decreasing.

3.4.2 Visibility limitations

Apart from environmental occlusions, limits of the visibility sensors also pose a risk from which the target can escape. In general there are two kinds of sensor limitations, *i.e.* limits on the maximum and minimum sensing range (*range edges*) and limits on the field of view *fov edges*. These edges are special because they cannot be eliminated by the tracker itself. The tracker needs external sensors or additional trackers to be able to eliminate them. In the absence of the ability to eliminate these edges the best strategy is to delay the target's escape through these edges, *i.e.* by maximizing t_{esc}.



(a) FoV Limits (b) Range Limits

Figure 3.10. Handling visibility sensor limitations.

Field of View Edges Most of the visibility sensors like lasers or camera have a limited field of view (FoV). This imposes an additional constraint on the tracking strategy to keep the target within the boundaries of FoV, preferably around its center.

The FoV can be modeled as an annular sector with the tracker at the center, the radii given by the visibility spanning from θ_{min} to θ_{max} (Figure 3.10a).

For guarding against FoV constraints, we use the same approach of manipulating the *FoV edges* (f in Figure 3.10a) away from the target. FoV can be manipulated either by rotating the visibility sensor or moving the tracker base away from the target. Although a combined motion can be modeled, for simplicity we just use the rotation primitive. This is more natural in case the visibility sensor has an additional degree of freedom over the tracker, *e.g.* a pan mechanism, the angular velocity of the panning (ω_R) is then the action of the tracker. On the other hand, if the sensor is attached rigidly to the tracker base, the turning of the tracker itself acts to rotate the FoV. In that case we treat ω_R as the rotation of the tracker.

As we deal with the angular motion, it is natural to derive t_{esc} using these rotational parameters. Based on the target's motion, we can approximate its angular velocity, ω_T towards FoV. This gives $V_{eff}(T, f) = \omega_T - \omega_R$, and $Dist(T, f) = \delta\theta$. Treating t_{esc} as the risk,

$$\varphi = \frac{\delta\theta}{\omega_{eff}} \tag{3.16}$$

where $\omega_{eff} = \omega_T - \omega_R$. The tracker's angular velocity ω_R, is chosen to maximize this t_{esc} by choosing action along $\nabla\varphi$,

$$\nabla\varphi = \frac{\varphi}{w_{eff}}\nabla\omega_R \tag{3.17}$$

Range Edges As with FoV edges, the tracker cannot eliminate range edges as they are induced by its own sensor limitations. In Figure 3.10b, D_{max} and D_{min} show the visibility range limits.

In a similar approach as above, for *range edges* we use t_{esc} as the risk. In this case e is the distance towards the nearest point in the range edges. From Figure 3.10*b* we can calculate the φ as,

$$\varphi = \frac{e}{v_r - v'_{\mathrm{e}}} \tag{3.18}$$

where v'_{e} is the velocity component of the target towards the escape edge D_{max}, and v_r is the tracker's velocity in pushing the range edge away from the target. The tracker chooses actions based on the local gradient,

$$\nabla\varphi = \frac{e}{{v_{eff}}^2}\nabla v_r \tag{3.19}$$

where $v_{eff} = v'_{\mathrm{e}} - v_r$ The analysis for D_{min} is identical in which case, the tracker would actually back away from the target to guard the D_{min}.

An interesting thing to note is that the behavior generated by the range edges alone, makes the tracker move towards the target. This is exactly the visual servo behavior. This shows that visual servo is a special case of vantage tracking when there are no occlusions.

3.4.3 Qualitative performance analysis

We now show the performance of the algorithm with respect to existing approaches and analyse the results qualitatively. We first compare the risk based approach to a simple implementation of visual servo *Visual Servo tracker* where the tracker tries to minimize its distance to the target. Subsequently, we compare our vantage tracker to another risk based approach *SDE tracker* [3], which maximizes the shortest distance of escape from the tracker's visibility.

Vantage Tracker vs Visual Servo Tracker

To illustrate the fundamental difference in the tracking approach from a simple visual servo based tracker, we run both the algorithms in a small room which has one exit in the northern wall, Figure 3.11. For the vantage tracker, we only compute the risk for occlusion edges and not of FoV in this example to show the effect of occlusions clearly. In the first and second column, the red robot is the target while the blue robot is the tracker. The trail of the tracker shows the tracking behavior. The third column shows the tracker's view of the tracking scene. In this local visibility, the blue lines show occlusion edges while the green segment shows the motion decision taken. We look at the actions taken by both the algorithms once when the exit is closed (Figure 3.11(*a*-*c*)) and when the exit is opened subsequently (Figure 3.11(*d*-*f*)). In all cases the red robot (target) and the blue robot (tracker) start at the same points.

Closed exit In the first case where the exit is closed, Figure 3.11(*a*-*c*), the whole room is visible to the tracker. The standard visual servo action is to move towards the target as shown in Figure 3.11(*a*). However, in the objective of keeping the target in view, no action is required as the target cannot escape the tracker's view anywhere inside the room. As there is no occlusion edge, there is no risk of losing the target and hence there is no action generated for the vantage tracker. In such a scenario, the vantage tracker comes up with the smarter alternative of staying put Figure 3.11(*b*,*c*).

Open exit Since the visual servo does not take the environment into account while planning, its action does not change when the exit is opened (Figure 3.11(*d*)). On the other hand, the vantage tracker sees two new occlusion edges. This generates a risk

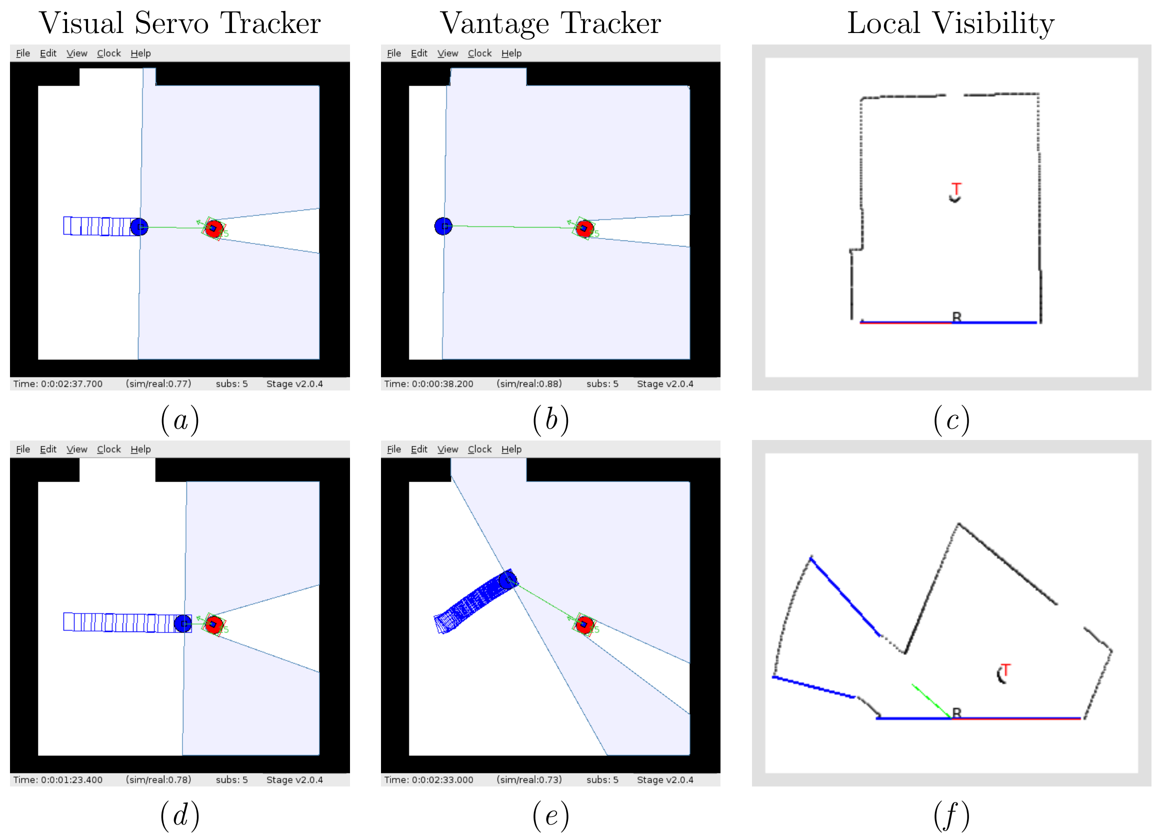

Visual Servo Tracker
Vantage Tracker
Local Visibility
File Edit View Clock Help
Time: 0:0:02:37.700 (sim/real:0.77) subs: 5 Stage v2.0.4
Time: 0:0:00:38.200 (sim/real:0.88) subs: 5 Stage v2.0.4
T
R
(a)
(b)
(c)
Time: 0:0:01:23.400 (sim/real:0.78) subs: 5 Stage v2.0.4
Time: 0:0:02:33.000 (sim/real:0.73) subs: 5 Stage v2.0.4
(d)
(e)
(f)

Figure 3.11. Comparing the difference in nature of visual servo based tracker to relative vantage tracking.

of losing the target through these edges. The vantage tracker then moves towards the source of the edges, the exit, to prevent the target's escape (Figure 3.11(*e*,*f*)). This action prevents the foreseeable escape of the target through the exit in the future.

Vantage Tracker vs SDE Tracker

The benefits of the new risk function (Vantage Tracker) are best illustrated in comparison with a related risk function introduced in earlier work [3] (SDE Tracker). For occlusion edges, the SDE tracker risk function is a monotonic function of the ratio r/e and completely ignores r'. To simplify the presentation, we assume that the tracker and the target have the same velocity bounds in all the following examples.

We implement the algorithm for a single occlusion edge (in C++ using the geometric library LEDA), to qualitatively compare its performance with previous shortest distance to escape (SDE) based methods. We implement the SDE tracker in [3] to compare our vantage tracker with. In these experimental screen-shots, the shaded region indicates $\mathcal{V}$. A small blue circle marks the tracker position. A filled black triangle marks the target position. The associated arrows indicate the tracker's and target's velocity directions. Again the target and the tracker start from the same positions for both algorithms.

Effect of relative position SDE trackers only take e into account for deciding tracking decisions. Moving the target parallel to the edge does not affect its decision (Figure 3.12(a & c)). The closer the target is to the occlusion vertex, the effectiveness of the tracker's swinging action (v_n) decreases proportionately. SDE trackers do not consider this fact and loses the target eventually. On the other hand, vantage tracker adapts to the changing position of the target and gives more weight to v_r when swinging becomes ineffective (Figure 3.12(d)).

Balancing current vs future risk SDE based risk formulation tries to maximize the shortest distance, which in effect decreases the current risk. This preoccupation with current risk limits its actions towards improving the future risk. Now consider scenario (Figure 3.13). The target is very close to the gap edge, and thus e is small. As a result, the SDE based risk function generates a motion more towards v_n to swing the gap edge away from the target. However, the situation is in fact not that critical.

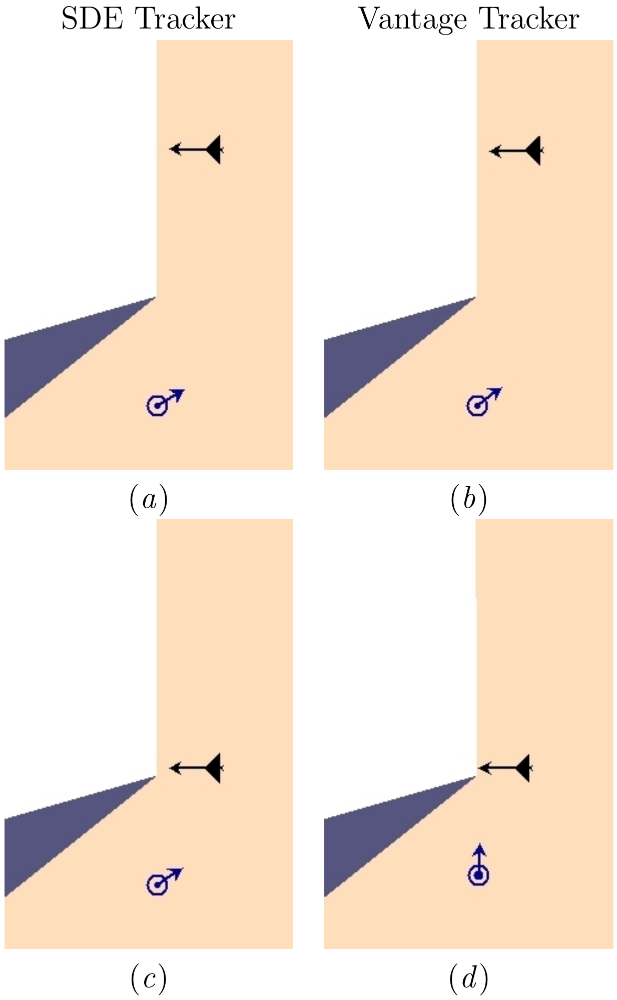


Figure 3.12. Comparing the SDE tracker vs Vantage tracker in response to change in relative position of the target.

The target is still a small distance away from the gap edge, leaving some time for maneuvering. More importantly, the tracker is slightly closer to the occlusion point than the target. A small swing is sufficient to keep the target visible. Too much swing in the tangential direction reduces the motion in the radial direction and increases the future escape risk. The preoccupation with the current risk eventually causes the SDE tracker to lose the target. The vantage based risk formulation handles this

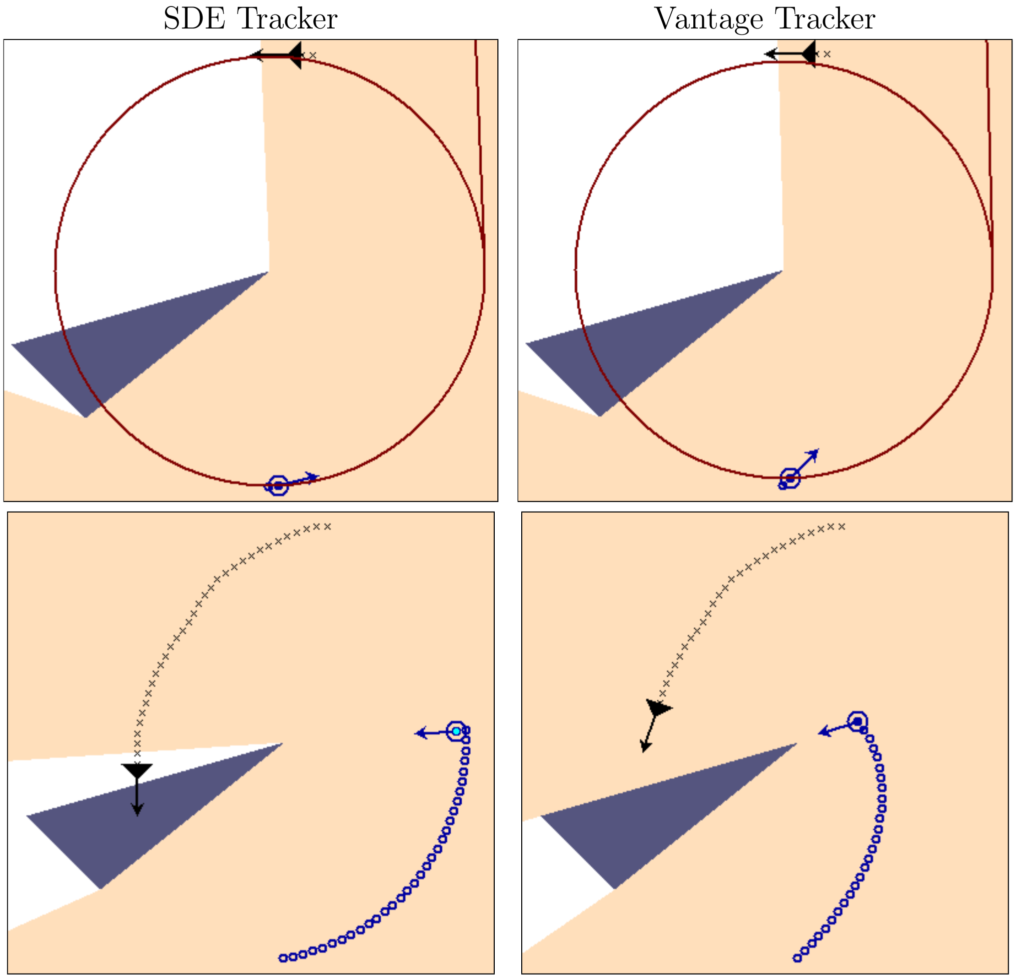
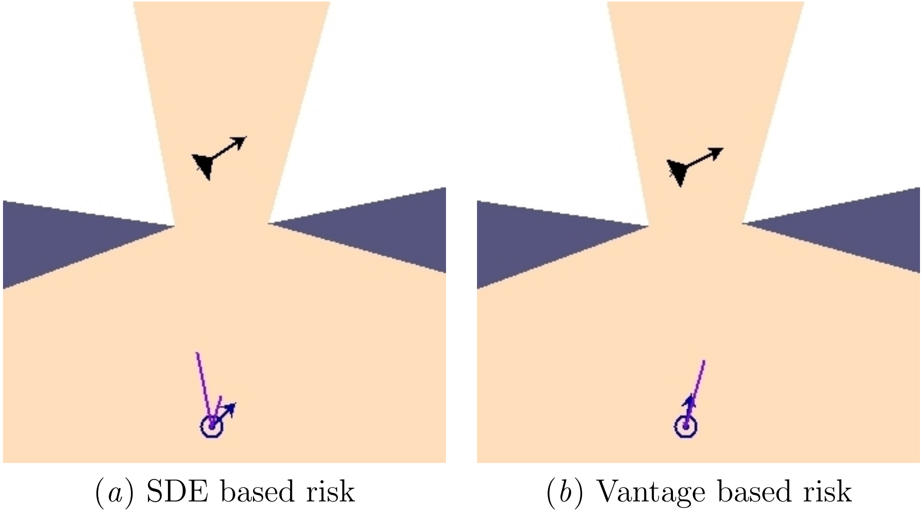


Figure 3.13. A scenario in which too much swinging increases future risk.

situation much better. It always keeps the target visible while trying to position itself in a better location, until it "eliminates" the gap edge in the end (Figure 3.13). This example shows that the proposed relative vantage based risk formulation can balance the current risk vs future risk more effectively.

Effect of target heading The vantage risk takes into account the target's velocity. In Figure 3.14(a & b), we compare the risk assignment based on SDE to that on



(a) SDE based risk

(b) Vantage based risk

Figure 3.14. The effect of using the target's velocity information on the risk and tracker motion decision. The purple segments are proportional to the amount of risk perceived by the tracker and is pointed towards its corresponding occlusion edge.

vantage. We see that a SDE based tracker assigns higher risk to the closer edge (here the left occlusion edge) when clearly the target heading suggests that the right edge has to be guarded against. Taking into account this heading information, the vantage tracker is able to assign a higher risk to the right edge.

3.5 Handling Multiple Edges

The target can escape $\mathcal{V}$ though any of the escape edges (occlusion, fov or range) and hence the total risk (Φ) of losing sight of the target is a combination of the risks from individual escape gaps. Let φ_i represent risk due to a i^{th} gap $\mathcal{G}_i$. Theoretically, φ_i is not independent of each other. However, to simplify the formulation, we assume independence of φ_i. We show from our experimental results that such a simplification does not break down the algorithm. We approximate Φ by the expected risk of all the gaps.

$$
\begin{aligned}
\Phi &= E[\varphi_i] \\
&\approx \sum_i p_i \varphi_i
\end{aligned}
$$

where p_i is the probability of the target's escape through the i^{th} escape edge, $\mathcal{G}_i$.

Let $\mathbf{v}^\star$ be the optimal action that minimizes the total risk Φ,

$$
\begin{aligned}
\mathbf{v}^\star &= \arg\min_v \Phi \\
&\approx \sum_i p_i (\arg\min_v \varphi_i) \\
&= \sum_i p_i v_i^\star \qquad (3.20)
\end{aligned}
$$

where $v_i^\star$ are the optimal action taken for each of the individual escape edges.

3.5.1 Prediction

A visibility set may contain many gap edges. Based on the target's motion patterns, we can identify the important ones and improve tracking performance. Let the *heading probability* p_g be the probability of the target headed to a gap edge $\mathcal{G}$. To estimate heading probabilities, we need the current target velocity $\mathbf{v}'$. At any time, we maintain an estimate of $\mathbf{v}'$ by storing a short history of the target trajectory and extrapolating. Many other methods for velocity estimation are possible. For simplicity, the uncertainty in estimating the direction θ of $\mathbf{v}'$ is assumed to follow a Gaussian distribution $f(\theta)$. The variance of the Gaussian indicates our confidence in estimating the target behavior. Other distributions, even non-parametric ones, can be used instead of the Gaussian, depending on the method of velocity estimation.

Our method for computing heading probabilities is general and works with any distribution. We derive the probability p_g for an occlusion edge $\mathcal{G}$. The approach is identical for computing p_g for FoV and range edges.

It is natural to assume that the target will exit a gap $\mathcal{G}$, if it is headed to $\mathcal{G}$. In other words, suppose that λ_θ is the ray originating from the current target position and having direction θ. The target will exit $\mathcal{G}$ if λ_θ intersects $\mathcal{G}$. Thus, p_g can be estimated from the target's estimated velocity distribution and the angle subtended by $\mathcal{G}$:

$$p_g = \int_{\Theta_\mathcal{G}} f(\theta)\, d\theta,$$

where θ lies in the angular range $\Theta_\mathcal{G}$ if and only if λ_θ intersects $\mathcal{G}$ (Figure 3.15). This seems reasonable, unless we consider a gap edge subtending zero angle, *e.g.*, the one marked as $\mathcal{G}_0$ in Figure 3.15a. It is a distinct possibility that the target may exit $\mathcal{G}_0$. We must relax our initial assumption and incorporate this situation. To do this, we expand every gap edge by a pre-defined distance δ and call the resulting region the *gap zone*:

$$G(\mathcal{G}) = \{\mathbf{q} \in \mathcal{V} \mid Dist(\mathbf{q}, \mathcal{G}) \leq \delta\},$$

where $Dist(\mathbf{q}, \mathcal{G})$ denotes the shortest distance from $\mathbf{q}$ to $\mathcal{G}$. Now the heading probability of $\mathcal{G}$ depends on the angle subtended by its gap zone instead of gap edge. In general, adjacent gap zones may overlap, and the probability in overlapping region must be split evenly among all gap zones involved. Taking all these into account, we have the following formula for computing the heading probability of $\mathcal{G}$:

$$p_g = \int_{\Theta_G} f(\theta)/h(\theta)\, d\theta, \tag{3.21}$$

where Θ_G is the angular range subtended by the gap zone G of $\mathcal{G}$ and $h(\theta)$ is the number of gap zones that λ_θ intersects. Note that $h(\theta) \geq 1$.

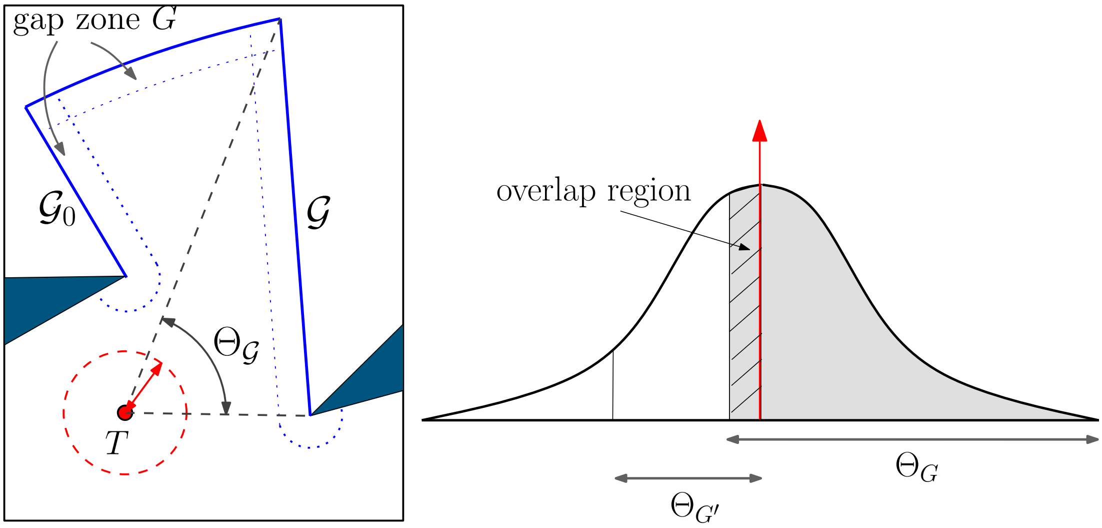


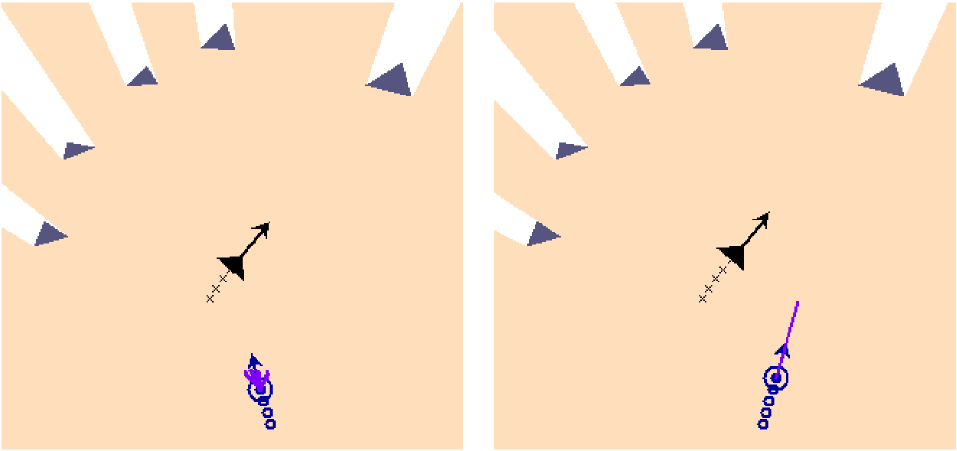
Figure 3.15. Estimating heading probabilities.

The threshold δ for determining the gap zone basically says that the target may exit $\mathcal{G}$ whenever it comes within a distance δ of $\mathcal{G}$. It can be chosen according to our understanding of target behaviors. In our experiments, we chose δ to be the distance that the target can reach with maximum velocity in one time step. This is an aggressive choice, indicating high confidence in the target motion model.

Effect of prediction

The tracker's prediction of the target's motion can help it focus its attention to more critical escape gaps. We show the improvement in the tracking performance due to target motion prediction.

Too much clutter can confuse the tracker without proper prediction Good velocity prediction helps the tracker to focus on the important gap edges and ignore clutter (extraneous edges) and improve tracking performance. Consider the example



(a) Distance based prediction (b) Velocity based prediction

Figure 3.16. The prediction based on velocity information helps in focusing on more important escape edges.

in Figure 3.16. It compares our new tracking strategy with the one in [3], which does not use velocity prediction. Each image in Figure 3.16 shows several small line segments rooted at the current tracker position. Each segment corresponds to the heading probability of a gap edge. The length of the segment is proportional to the heading probability, and its orientation points to the gap edge associated with the heading probability. For the SDE tracker, all the gap edges are weighted with probabilities proportional to the SDE of the tracker to that edge. For the vantage tracker, the gap edge which the target is headed has a distinctively large heading probability, indicated by a long segment.

Wrong prediction can be overcome by fast update rate When the target makes abrupt turns, the linear velocity prediction is usually inaccurate. Our tracking strategy may cause the tracker to make the wrong move. Consider the example in Figure 3.17. The target makes several abrupt turns. However, the velocity prediction

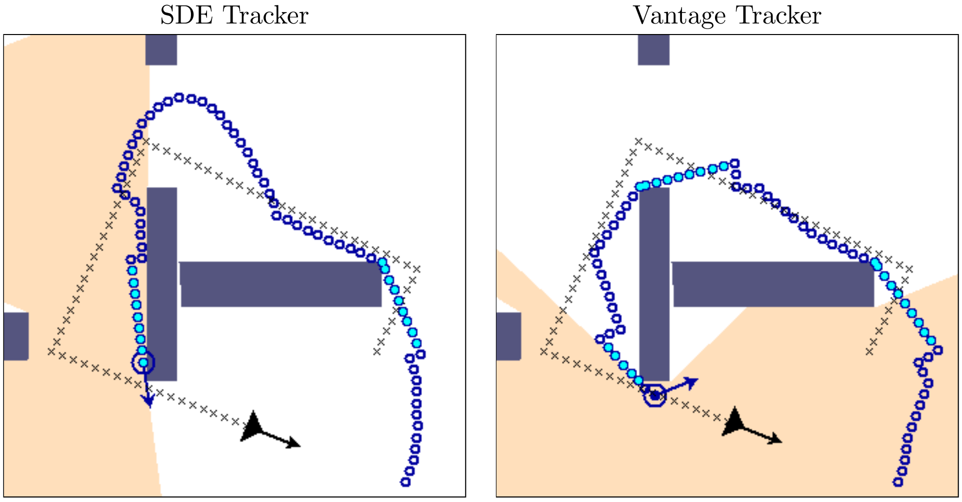


Figure 3.17. An example in which the target makes abrupt turns.

is reasonable for most of the time. Despite the wrong moves, our tracking strategy follows the target to the end and performs better than the SDE tracker, which loses the target midway.

3.6 Adding Constraints

The choice of $\mathbf{v}^\star$, as in Equation 3.20, has to satisfy many constraints $(\mathcal{C}_i)$: physical constraints like *reachability, motion dynamics* or planning constraints like *obstacle avoidance, stealth* etc. The constraints are projected into the position space. We define a *feasible region* as, $\mathcal{L}$ (x), that satisfies all the constraints $\mathcal{C}_i$ (Figure 3.18),

$$\mathcal{L}(x) = \mathcal{R} - \bigcup_i \mathcal{C}_i,$$

The local greedy optimization then becomes choosing an action $(\mathbf{v}^\star)$, that minimizes Φ while keeping the tracker in $\mathcal{L}$ for the time step Δt,

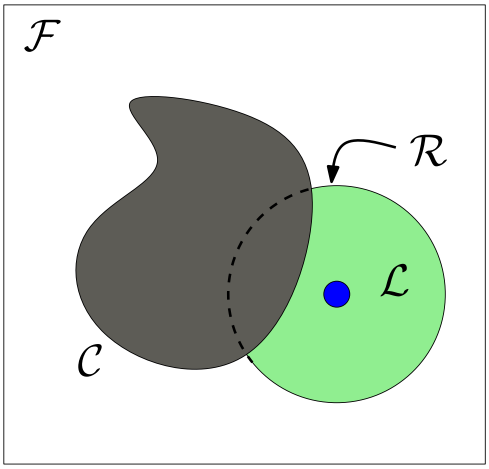


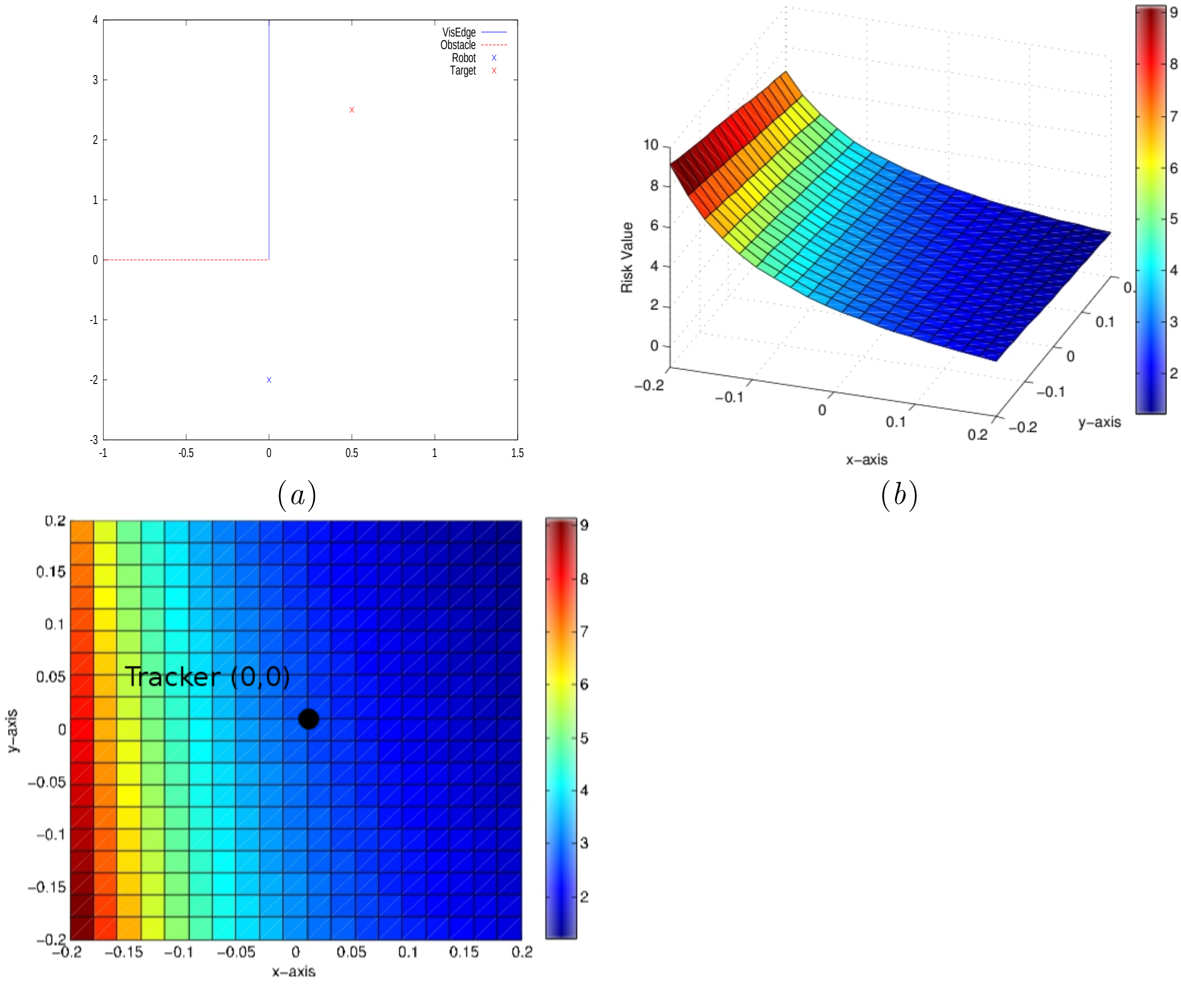
Figure 3.18. Feasible region, $\mathcal{L}$

$$\mathbf{v}^{\star} = \arg\min_{\mathbf{v}} \Phi \qquad s.t. \quad \mathbf{v}^{\star}\Delta t \in \mathcal{L} \tag{3.22}$$

3.6.1 Locally optimal constrained action

We plot the risk function for the proximity of the tracker position in Figure 3.19. Without loss of generality the origin is chosen at $\mathcal{O}_v$, while y-axis coincides with the occlusion edge. The positions of the tracker and the target are shown in Figure 3.19(a). The tracking scenario is akin to the target being in $\mathcal{D}_N$ in Figure 3.7. For an action computed from Equation 3.6, risk plot is generated in the tracker's neighborhood as shown in Figure 3.19b. We see that the risk function is smooth, continuous, and monotonic in the neighborhood of the tracker position $\mathbf{x}$. Consider the linear approximation of φ at $\mathbf{x}$:

$$\varphi(\mathbf{x} + \Delta x) \approx \varphi(\mathbf{x}) + \nabla\varphi(\mathbf{x}) \cdot \Delta x. \tag{3.23}$$



(*a*) (*b*)

(*c*)

Figure 3.19. MATLAB risk plot. The negative risk gradient is towards the top-right corner

Minimizing function Equation 3.23 is equivalent to

$$\min_{\Delta x} \nabla\varphi(\mathbf{x}) \cdot \Delta x \qquad \text{subject to } \Delta x \in \mathcal{L}$$

The feasible region can be approximated by a list of convex polygons, then the problem reduces to linear programming [91]. The minimum solution $\Delta x = \mathbf{v}^{\star}\Delta t$ must lie at the vertex of one of the convex polygons. By projecting all the vertices in the feasible region along $\mathbf{v}^\star$, we can find the vertex with minimum risk. The optimal heading for the tracker is taken to be along this vertex.

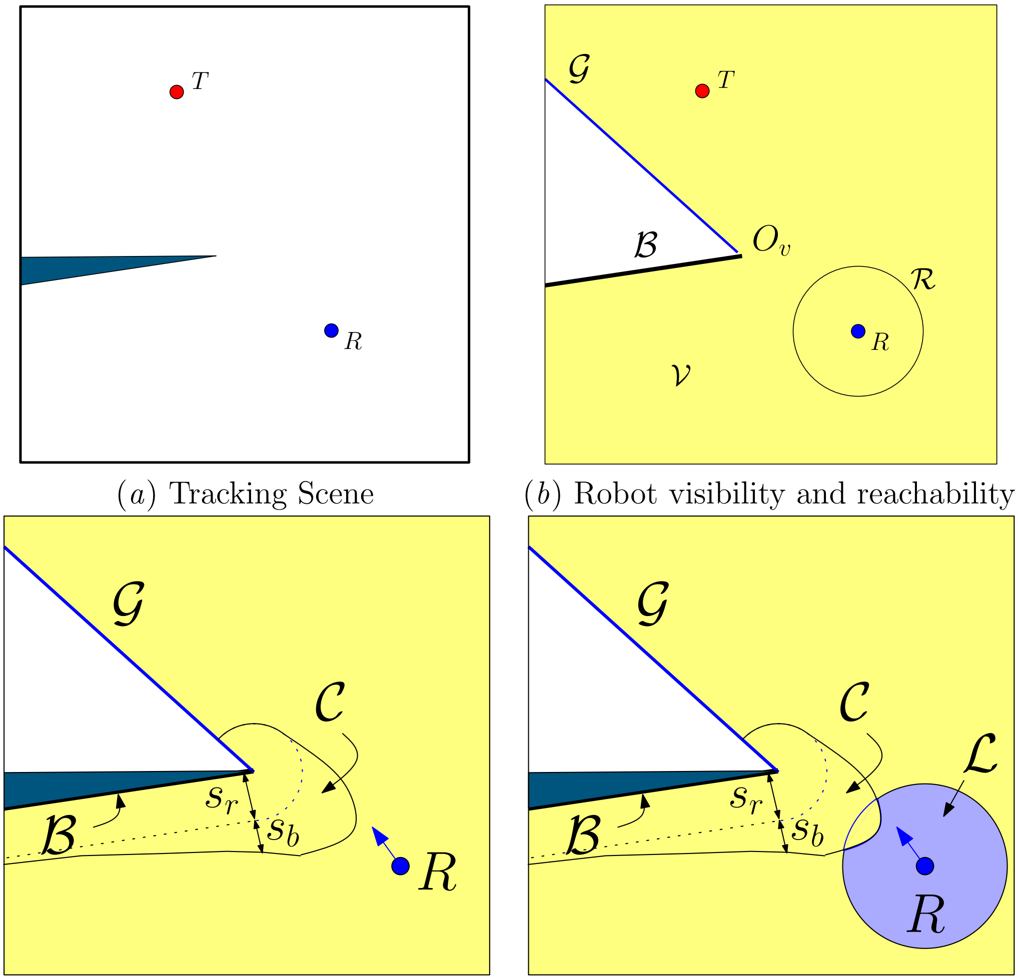
As an example, we develop the constraint for obstacle avoidance below. Similar approach is used for mission constraints like localization, safe navigation or stealth, all of which restrict $\mathcal{R}$. In the Chapter 4 we develop a mission constraint *stealth* in detail, that requires the tracker to track the target while preventing its discovery by the target.

3.6.2 Obstacle avoidance

We show the case of obstacle avoidance as an example of constrained optimization as described above. We approximate the tracker's size by the radius (s_r) of its bounding circle. Also, depending on the tracker's current velocity, there is a finite braking distance s_b, $s_b = \int_{\Delta t} a(x)dt$ based on the max deceleration of the tracker. Combined s_r and s_b denote a region around the obstacle ($\mathcal{C}$) that must be avoided for safe navigation.

$$\mathcal{C}(x) = \{q \in \mathcal{V} : d(q, \mathcal{B}) \leq (s_r + s_b)\} \tag{3.24}$$

The choice of motion then is restricted to the feasible region $\mathcal{L} = \mathcal{R} - \mathcal{C}$ as shown in (Figure 3.20d). The problem of obstacle avoidance becomes a problem in real world. Fortunately, the Equation 3.24 can be represented quite easily as a polyline by choosing a small set of samples from the range based sensor. This makes the computation of the obstacles quite fast.



(a) Tracking Scene

(b) Robot visibility and reachability

(c) Obstacle dilation $(s_r + s_b)$

(d) Feasibility region $\mathcal{L}$

Figure 3.20. Obstacle Avoidance

3.6.3 Local target recovery

Our algorithm tries to keep the target in view for as long as possible. But many times it is impossible to guarantee this. In case the target steps out of view, the tracker can try to retrieve it. This brings the problem into the domain of target searching in an unknown environment. In our case, we try a simple two step strategy of recovering the target, Figure 3.21. Let the possible position of escape be P and r'

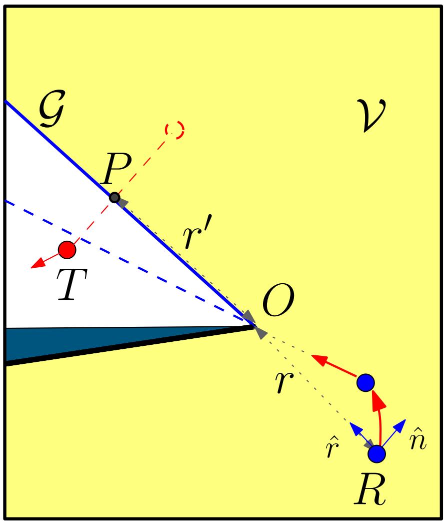


Figure 3.21. Local target recovery strategy

be the distance OP. If the target escapes with a high r' that means that it is easier to recover by swinging, On the other hand when r' is small swinging does not help and the tracker should move more towards the occlusion vertex, O of the escaped edge. Based on this observation we propose a simple recovery velocity as

$$\mathbf{v} = \frac{r\hat{\mathbf{r}} + r'\hat{\mathbf{n}}}{\sqrt{r^2 + r'^2}} V$$

We execute this strategy for a fixed number of steps. In case the tracker fails to recover the target, the tracker moves to eliminate the gap, $\mathcal{G}_i$.

However, there might be cases where after reaching the last known target position, the tracker either does not find any target. At that moment we stop the algorithm. In such cases choosing the next most probable target's exit could help. Here the problem switches from target tracking into target searching for which any general search strategies can be applied [33, 96].

Table 3.1. Performance comparison of the SDE and the vantage tracking strategies.

Env.	Total No.	SDE Tracker	
	Target Steps	No. Steps Visible (%)	No. Times Lost (Steps Lost)
Maze	82	35 (43%)	2 (11,12)
City Blocks	156	78 (49%)	6 (14, 15, 16, 15, 8, 10)
		Vantage Tracker	
	Target Steps	No. Steps Visible (%)	No. Times Lost (Steps Lost)
Maze	82	74 (90%)	1 (8)
City Blocks	156	131 (84%)	2 (13, 12)

3.7 Experimental Results

In the following, we show quantitative results in simulation to show the effectiveness of the vantage tracker. We show that vantage tracker consistently out-performs the previous approaches both for a simplistic sensor and motion model as well as when sensing and mobility limits are introduced.

3.7.1 Tracking in Polygonal Environments

In this set of experiments we compare the performance of the risk formulation without visibility or mobility constraints, *i.e.*, the tracker has omni directional vision and holonomic mobility. As a point tracker in a polygonal world, we do not have to address the obstacle avoidance problem either. We compare the Vantage tracker to the SDE tracker as before. To have a fair comparison, we provided the SDE tracker the same emergency actions that the vantage tracker uses, though they are not in the original work.

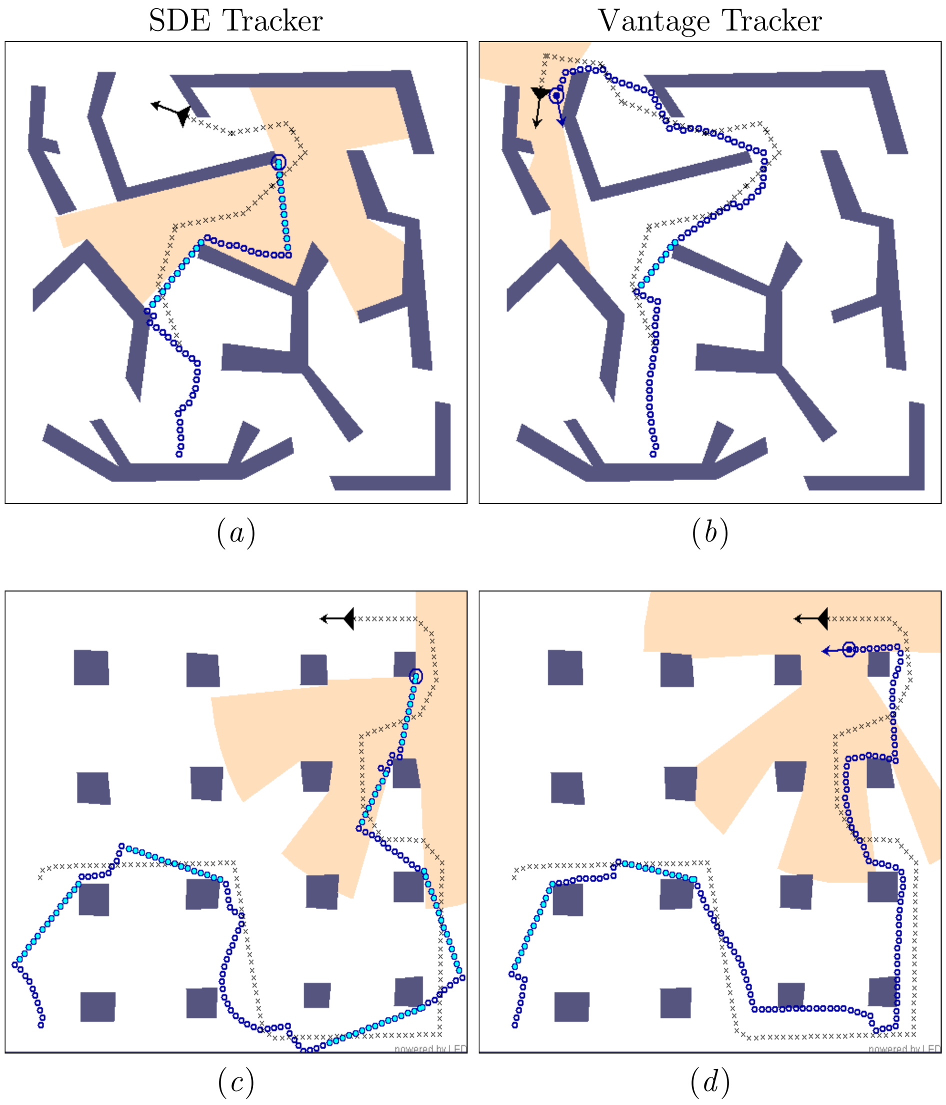


Figure 3.22. Two environments with complex geometry. (a,b) show the tracking path for the *Maze* environment while (c,d) are results for the *City Blocks* experiment. Black crosses depict the target's path, while the blue void circles show the tracker's trajectory. The portions of the tracker's trajectory where the target is lost is marked by filled cyan circles.

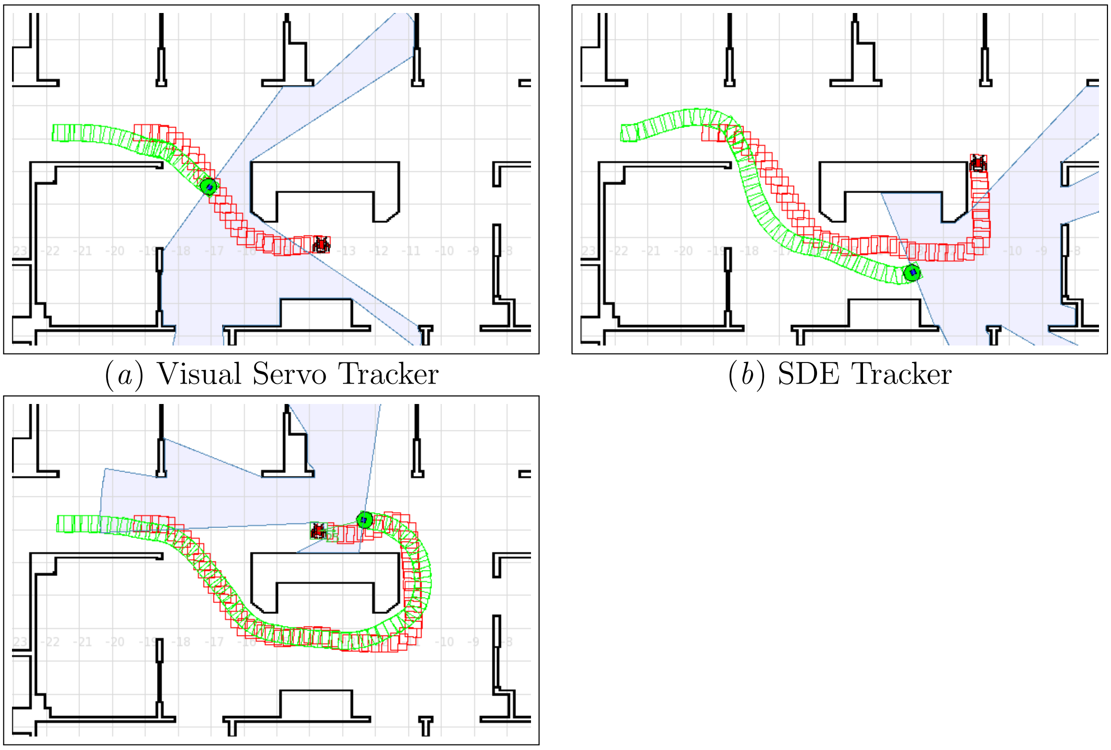
a) Maze. (Figure 3.22(*a,b*)) This environment brings together various geometric features, such as long corridors, open spaces, and sharp turns. The target takes a long and winding path. Even with emergency actions, the SDE tracker loses the target midway. The vantage tracker follows the target to the end. It loses the target once, but recovers it quickly through emergency actions.

b) City blocks. (Figure 3.22(*c,d*)) This example mimics city blocks in an urban environment. The SDE tracker has lots of difficulty in this environment. It loses the target many times for extended periods (see Table 3.1) and fails to follow the target to the end. The vantage tracker has much improved performance.

Detailed performance statistics on these two environments are shown in Table 3.1. Column 2 of the table lists the length of the target trajectory in time steps. For the SDE tracker, column 3 lists the number of steps that the tracker has the target visible as well as the number as a percentage of the total number of target steps. Column 4 lists the number of times that the target is lost *and* recovered with emergency actions, as well as the durations for which the target is lost. The comparison in these two environments shows that the vantage tracker (i) less likely loses the target, (ii) has the target visible for much longer total duration, and (iii) always follows the target to the end. All these indicate better performance.

3.7.2 Tracking in Realistic Office Environments

Next we implement the algorithm in a more realistic simulation in Player/Stage [97]. As we model a realistic model of a differential drive Pioneer tracker and a SICK-lms200 laser sensor with FoV (-90:90)deg and a fixed maximum range of 8m. The scene is an indoor office environment. The algorithm now has to consider the sensing constraints and mobility constraints.


(*a*) Visual Servo Tracker

(*b*) SDE Tracker

(*c*) Vantage Tracker

Figure 3.23. The green tracker is trying to follow the red target. The trails show their actual path. The light blue shaded region denotes the tracker's visibility. Target is lost in (*a*) and (*b*), whereas in (*c*) the target is still in tracker's view.

In (Figure 3.23 & Figure 3.24), we compare more comprehensively the performance of the different algorithms: the visual servo, SDE tracker and our vantage tracker. The trackers start from the same location and tries to track a target executing a fixed path. The target's path is unknown to the tracker. In fig.3.24(*a*), we compare the performance of the trackers using the metric of SDE. The plots stop as soon as the target is lost by the tracker. We see that visual servo loses the target first around step

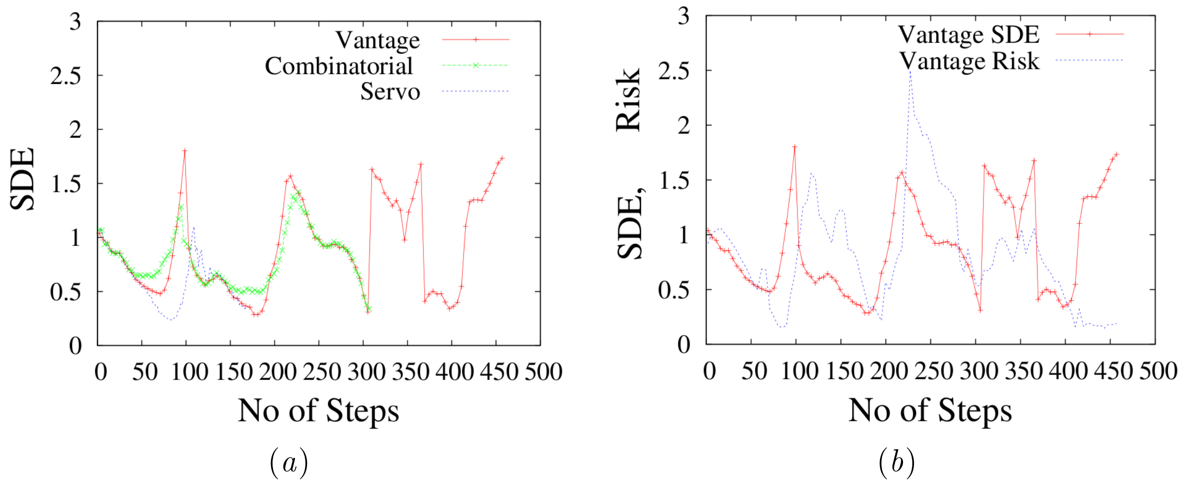
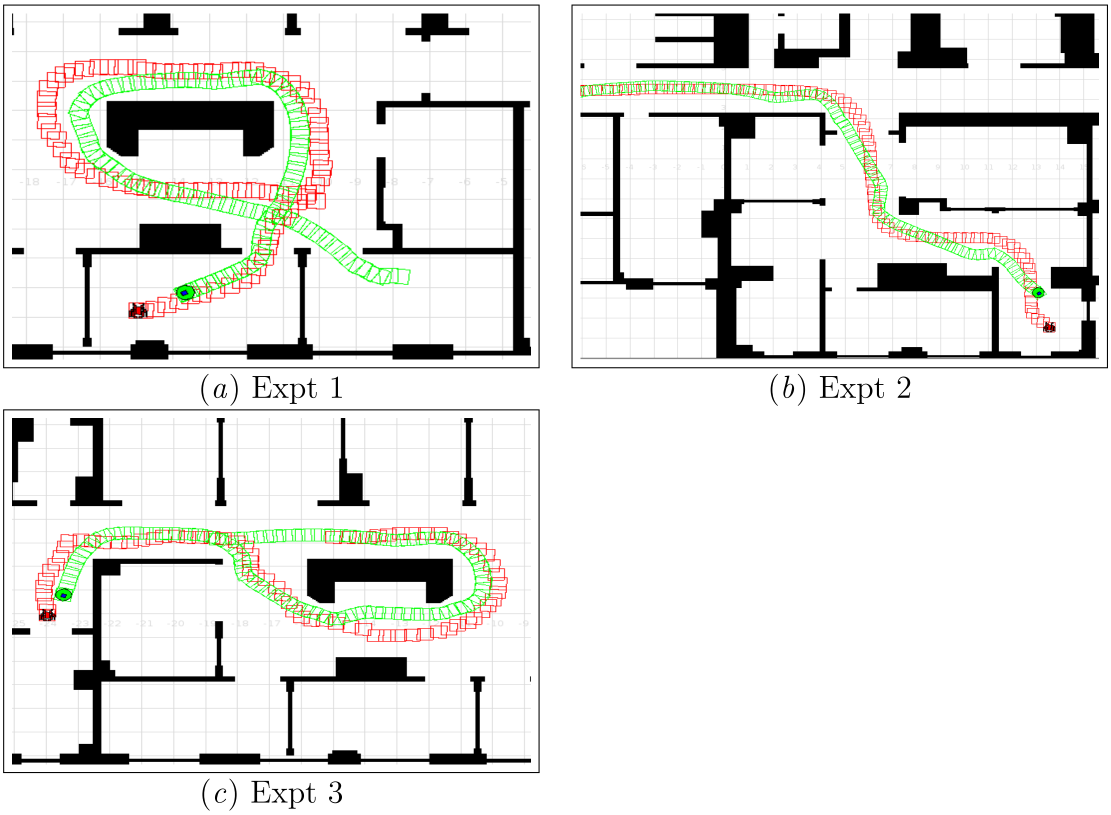


Figure 3.24. (*a*) Plotting SDE for various algorithms, (*b*) Plotting SDE and the risk value for the vantage tracker.

#160, (3.23*a*,3.24*a*) and then the SDE tracker around #300 (3.23*b*,3.24*a*) where as the vantage tracker manages to continue till the end (3.23*c*,3.24*a*). The visual servo tracker ignores the environment due to which it loses out early. The SDE tracker performs better, but it focuses more on the short term goals of immediate target loss. This is seen in fig.3.23*b* as a highly curved path due to the swinging actions. The vantage tracker balances the short term and long term goals better. Around step #300 when the SDE tracker loses out, the vantage tracker is much closer to the occlusion edge than the SDE tracker at that time step and is better positioned to handle the occlusion edge. Fig.3.24*b* plots the vantage risk value over the same path and this plot shows why the vantage tracker performs better. The risk plot shows peaks in the risk value when the SDE starts to fall (as the target comes closer to the occlusion edge). Anticipating this risk early allows the vantage tracker to improve its future position and keep the target safely in view.



(*a*) Expt 1

(*b*) Expt 2

(*c*) Expt 3

Figure 3.25. The simulation experiment paths taken by the target (red) and the path of the vantage tracker (green).

Quantitative comparison We run the algorithms for three other target paths, shown in Figure 3.25, and compare them for the percentage of the number of steps in which target was visible. For longer runs, local target recovery modes were activated. The visual servo tracker tried to regain the target by moving directly towards the last position seen, while the SDE and the vantage tracker both followed the local recovery algorithm mentioned earlier in this paper.

The results are shown in Table 3.2. Column 2 of the table lists the length of the target trajectory in time steps. For the each strategy, the first column lists the number of steps that the tracker has the target visible as well as the number as a

Table 3.2. Performance comparison of visual servo, SDE and vantage trackers.

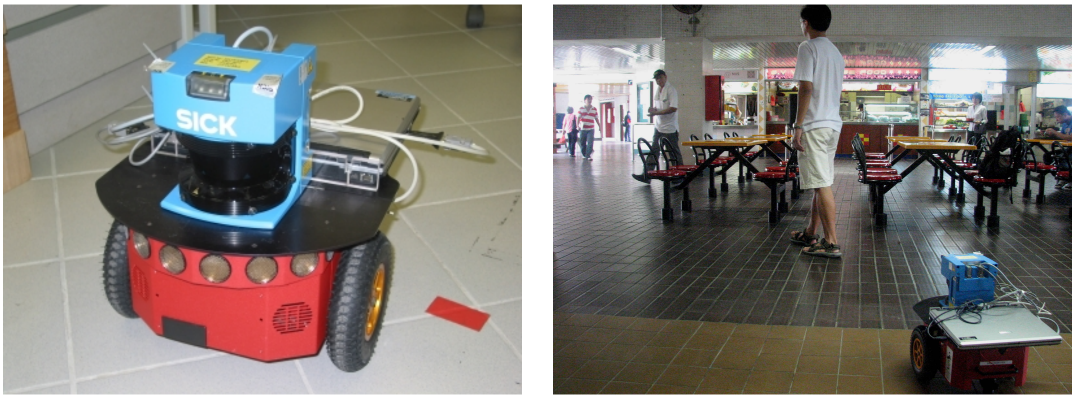
		Visual Servo		SDE		Vantage	
Expt. No.	Target Steps	Visible Steps (%)	No. Times Lost	Visible Steps (%)	No. Times Lost	Visible Steps (%)	No. Times Lost
1.	1200	268 (22%)	2, lost	230 (19%)	2, lost	1015 (85%)	4
2.	1700	467 (39%)	6, lost	295 (17%)	1, lost	705 (42%)	5
3.	1200	309 (26%)	4, lost	237 (20%)	1, lost	399 (33%)	5

percentage of the total number of target steps. The next column lists the number of times that the target is lost *and* recovered with emergency actions. In case the target was not recovered even after executing emergency actions, it is marked as *lost.* The comparison in these two environments shows that even in the presence of constraints, the vantage tracking performs better.

3.8 Hardware Implementation

Building a successful tracking system requires implementation of both the *Target detection* and *Target following* modules. While the previous sections deal with simplistic models of sensing and motion, these hardly hold true in the real world. The problem of reliable target identification is bypassed as the simulator can be queried for the exact position of the target. In reality however, uncertain and noisy sensor data makes target identification and localization unreliable. The tracking algorithm has to handle scenarios of false detection and no detection.

For the synthetic environment the tracking algorithm does not have to deal with the physical characteristics of the tracking robot. The robot was treated as a point object and it could move arbitrarily close to obstacles. In reality the robot has a



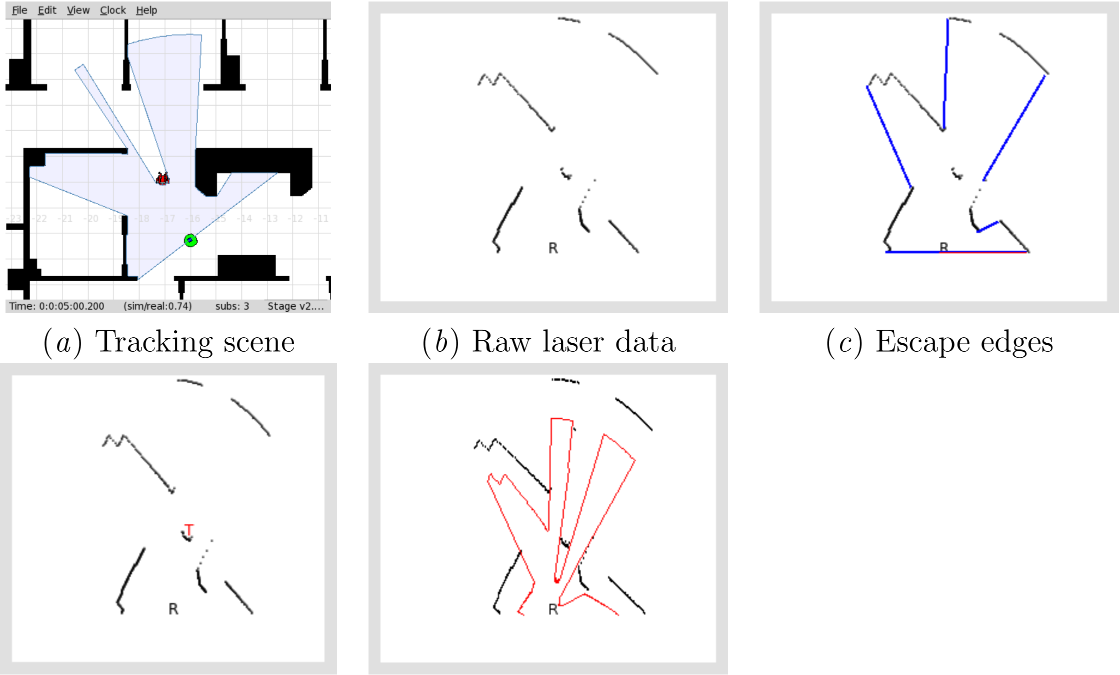
(*a*) Tracking tracker (*b*) Canteen environment

Figure 3.26. Pioneer 3D-X with an on board SICK lms-200 as a tracker. It was deployed in the school cafeteria to test the effectiveness in a cluttered and dynamic environment.

finite size and may have kinematic and dynamic constraints that need to be considered. Moreover, the robot controls are no longer accurate and there is uncertainty in robot's motion. Safe robot navigation is necessary not only for the sake of the robot but also to prevent damaging the environment. In fact safety becomes a critical issue when introducing the robot in human environments. All these issues provide serious challenges in implementing a robust tracking robot. Below we present the implementation approach for our vantage tracker.

The tracking algorithm is implemented on a Pioneer P3-DX differential drive tracker. A SICK-lms200 is mounted on the tracker. The laser returns 361 readings on a field of view of 180deg at the resolution of 0.5deg. The maximum range of the tracker is 8m. The control algorithm runs on a Pentium M Processor @1.5GHz laptop running Player server v-2.0.5 [97] on Linux. The algorithm runs at 10Hz. The target is a person walking around the lab corridors in the presence of other people.

For illustration purposes, a snapshot of the algorithm running in the stage simulator is shown in Figure 3.27a. The green circle is the tracker and the red object is the target. Figure 3.27(b-e), are shown in the local frame of the tracker with the origin at R.


File Edit View Clock Help
Time: 0:0:05:00.200 (sim/real:0.74) subs: 3 Stage v2....
R
R
T
R
R

(a) Tracking scene (b) Raw laser data (c) Escape edges

(d) Target detection (e) Obstacle dilation

Figure 3.27. Snapshots of the implementation at various stages.

Visibility polygon The output of the sensor is a radial scan of range values (Figure 3.27b). The data points with max-range readings form the range edges. We detect the continuity of the remaining data point to its neighbors by thresholding the range value change in adjacent data points. These changes represent the occlusion edges and their location can be extracted from the corresponding data points. Two additional edges are added at the orientation of the min/max angular limits to form the FoV limits. This is shown in Figure 3.27*c*.

Target identification The problem of robust target detection and identification from noisy data is an important problem that has been addressed in significant details by the computer vision and data processing research community. Since this thesis focuses on the motion planning aspect of the tracking problem, the target detection module has been possibly unfairly trivialized. Very simple and ad-hoc mechanisms have been incorporated during the implementation that provides reasonable target detection. However, our tracking framework is general enough to incorporate advance detection techniques that handle uncertainty models of the sensor, probabilistic frameworks for detection and maintaining target hypothesis seamlessly into the implementation. In fact such approaches are essential for the deployment of such tracking systems in real applications for realistic time durations. The following discussion summarizes the efforts to build a simple target detection mechanism robust enough to validate the motion strategies for our tracking framework.

Data points into clusters (Figure 3.27*b*). These clusters represent physical objects in the sensing range, *e.g.* walls, furniture, other people, *etc.*. In fact, one of the clusters is the target. We start with a known target, and focus on the subsequent target matching. In our experience, we find that a simple nearest neighbor match gives reliable target identification for reasonable target behaviors, even in a crowded environment. We perform the identification as follows.

The clusters are filtered based on average human leg size with some tolerance to give a set of potential target clusters. These potential clusters can be the chair or table legs, pipes or other human legs. Given the target's maximum speed, we can estimate a bound on the target's future position in time Δt, $V' \ \Delta t$. Within this bound, we choose the cluster closest to the target's previous position. False negatives are handled by increasing this bound by a fixed number of steps before declaring the target lost. In fact, this even helps in cases of momentary occlusion when someone walks between the target and the tracker.

Clearly, this method will fail if $V' \ \Delta t$ is too large. However, the practical success of this simple technique can be found in : (a) small Δt as the algorithm runs at a high frequency of about 10Hz, (b) target speed being slow enough, average human walking speed 1m/s (giving the tolerance level of about 10cm) and (c) low false negative rate for cluster detection. The cluster based target detection is more reliable than shape based feature detection (*e.g.* arc based leg detection) since leg features may partially occlude, eclipse or fuse with each other in the process of walking.

We found that maintaining a list of the non-target clusters in addition to the target using a simple nearest neighbor match, decreases the rate of false positives. Of course, algorithms like EKF, MHT can make the target detection much more robust.

Obstacle avoidance The clusters computed earlier create motion obstructions in addition to the visibility occlusions. We utilize an implementation of applying Equation 3.24 to the data points directly. This saves us considerable computation in extracting the actual shape of the cluster to compute $\mathcal{C}$. To achieve this we perform a radial transformation in to move each data point towards the tracker by $(s_r \ + s_b)$. s_b is estimated based on the relative velocity and the maximum relative acceleration towards the data point. The result is shown in Figure 3.27*e*.

Robot Motion control The optimal velocity $\mathbf{v}^\star$, generated from the algorithm does not take into account the non-holonomicity of the tracker base. We apply a simple low level control on the tracker velocity that tries to achieve $\mathbf{v}^\star$ (similar to [3]). From the structure of the risk function we see that Φ is locally smooth. Due to this, $\mathbf{v}^\star$ also changes slowly and the controller is stable.

Uncertainty in sensing and execution As the algorithm uses only local geometric information available to the tracker's visual sensors, it does not require a global map and thus bypasses the difficulty of localization with respect to a global map. Noisy sensor data and the uncertainty in tracker's control can affect the performance of the target's relative localization and especially the target's velocity estimate. In the hardware experiments we found that reliability of the target's velocity was quite poor. Still the tracker was able to successfully track the target by focusing more on the worst case scenarios. Moreover, uncertainty in sensing and motion control does not accumulate, because the tracker's action is computed using sensor data acquired in the current step only. This improves the reliability of tracking.

3.8.1 Experimental Results

We implement the system as shown in Figure 3.26. In the following we show snapshots of the video from the experiments. All the videos and additional results are uploaded at *http://guppy.mpe.nus.edu.sg/ mpeangh/tirtha/PhD/thesis.html.* Individual videos can be accessed from *http://guppy.mpe.nus.edu.sg/ mpeangh/tirtha/PhD/[Video-id].mov.* Alternate encodings in mpeg is also available as *http://guppy.mpe...../[Video-id.avi].*

Dynamic Environment Expt In Figure 3.29, a box is pushed between the target and the tracker to occlude the target. Since, the vantage tracker actively tries to avoid future possible occlusions, it is able to adapt to the changing environment (Figure 3.29*b-1*). A point to note is that the vantage tracker does not model the motion of the environment but just re-plans its motion at a high frequency. This makes the performance of the tracker independent of the dynamic nature of the environment. Later the box stops and the target starts to move (Figure 3.29c) and the tracker is able to successfully follow the target till the end (Figure 3.29d).

Guarding FoV Fig.3.30, shows the tracker guarding the target against its FoV by turning on the spot.

Cluttered Environment In fig.3.31, the tracker follows the target through the department lobby. The chairs in the lobby generate a large number of escape edges that the tracker has to handle. Note that the tracker tends to generate motion decisions biased towards left 3.31(c) as there are larger number of escape edges there. Due to the online nature and fast computation of the risk the tracker is able to successfully follow the target around.

Canteen crowd (Figure 3.32)

The canteen environment is inherently complex with the table and chair legs having a similar signature as the target person's legs. Moreover in lunch hour the crowds appear and dissolve changing the environment significantly and reducing the visibility and mobility. In the presence of humans the tracker must also plan to avoid moving human obstacles.

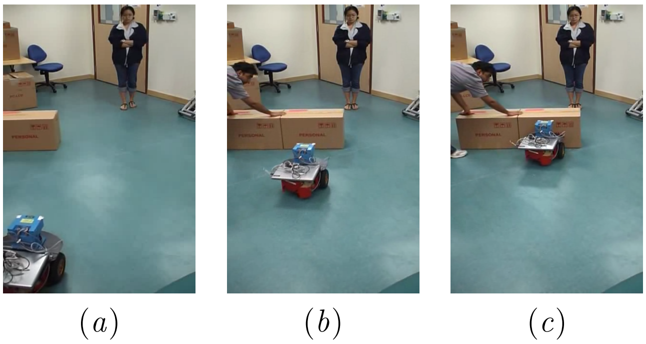


Figure 3.28. Visual Servo : Since the tracker does not take into account the environment information, it moves straight ahead towards the target (b) and loses the target to the occluding box (c). (Video-id: VisualServo-MovingBox)

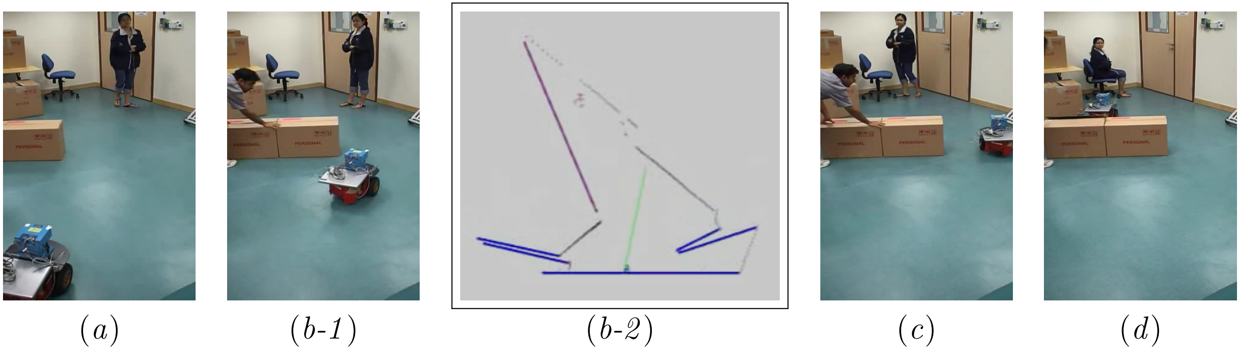
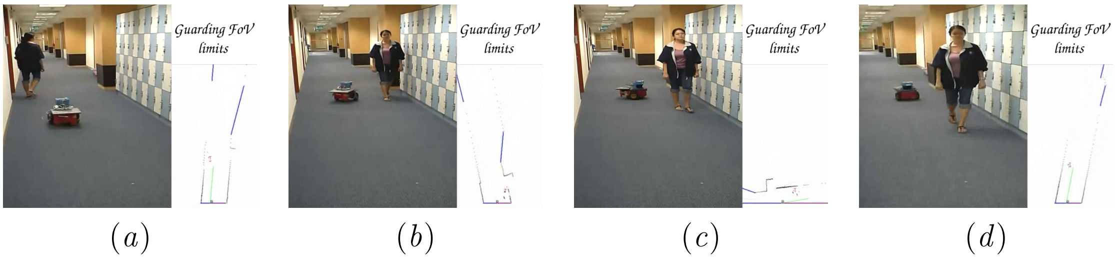


Figure 3.29. Vantage tracker : (*b-2*) shows the tracker's local perception of the environment. The target is marked by T, the blue lines are the occlusion edges, red line is the most critical occlusion and the green segment starting from R denotes the tracker's motion decision. The tracker sees the target too close to the occlusion and swings out. (Video-id: Vantage-MovingBox)

Temporary occlusion (Figure 3.33)

A challenging aspect of following the target in a crowd is when someone walks in between the tracker and the target. In this video snapshots the tracker is following the target in green t-shirt as seen in Figure 3.33(*a*) when it faces a temporary occlusion by a lady (in purple) walking across as seen in Figure 3.33(*b*-*d*). Since the tracker



(a) (b) (c) (d)

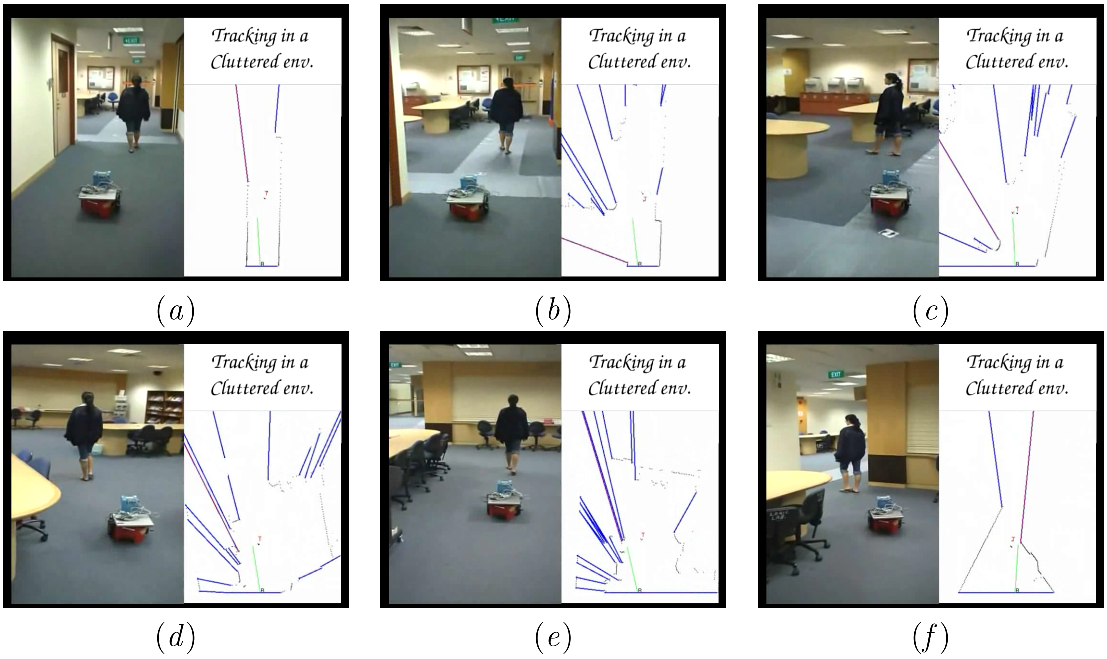
Figure 3.30. When the target doubles back, the tracker has to guard against its fov limits and makes a turn as well. (Video-id: Vantage-FoV)

has range information about the target and the algorithm is run at a high frequency, it can recover from these temporary occlusions in such a dynamic environment.

3.9 Summary

In this chapter, the concept of relative vantage is developed that captures the risk of losing the target from sight and an tracking algorithm is proposed that optimizes it in the local context. Incorporation of the relative positioning of the tracker and the target in the tracker's local visibility improves the tracking performance as shown in controlled experiments in simulation w.r.t previous trackers like SDE and visual servo. In fact, we can show that in the absence of obstacles or visual occlusions, the proposed approach boils down to the visual servo tracker.

The proposed tracking approach makes it easy to accommodate multiple hardware constraints like FoV, and mission constraints like obstacle avoidance and stealth. This eases the implementation on a real hardware, details of which are discussed. Such a tracking robot is shown to work well in a real crowded environment such as a school cafeteria. The tracker exhibits robust behavior for temporary occlusions.


Tracking in a Cluttered env.
Tracking in a Cluttered env.
Tracking in a Cluttered env.
Tracking in a Cluttered env.
Tracking in a Cluttered env.
Tracking in a Cluttered env.

(a) (b) (c)

(d) (e) (f)

Figure 3.31. The tracker tracks the target in cluttered environment. Due to the clutter of chairs the tracker has to guard against a lot of potential gap edges.(Video-id: Vantage-cluttered)

In the Chapter 4, we develop in detail a mission constraint of stealth, and show how such a constraint is incorporated into the tracking framework.

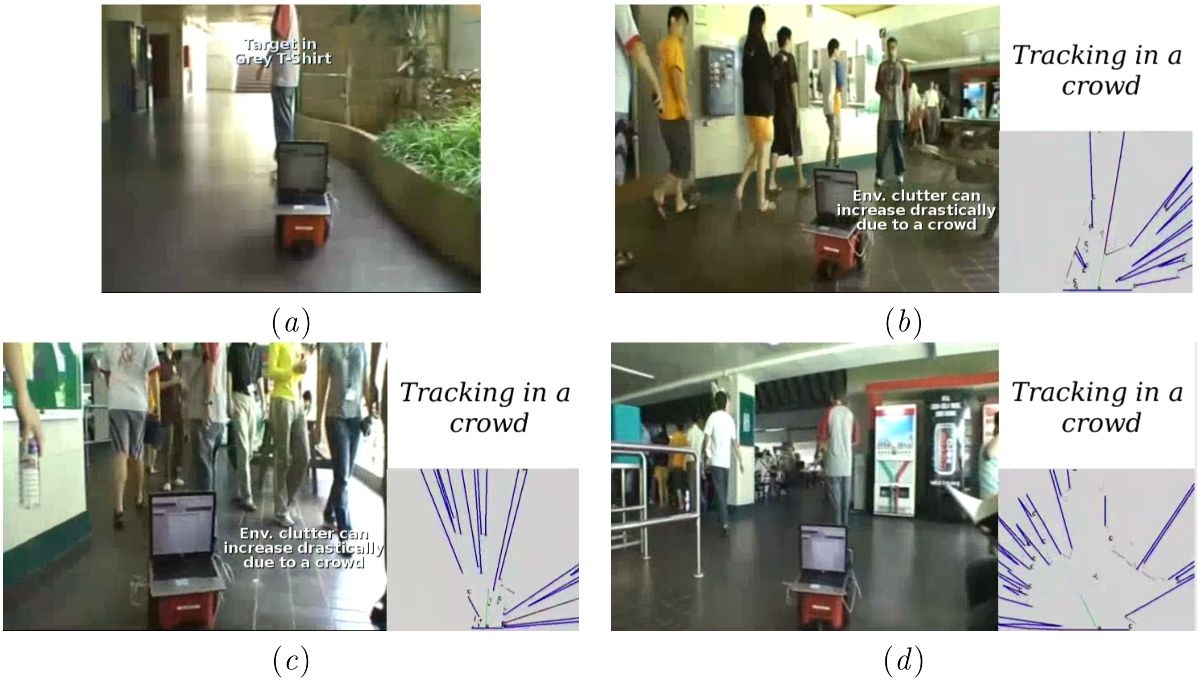


Figure 3.32. The tracker tracks the target in crowded canteen environment. (Video-id: Vantage-crowd)

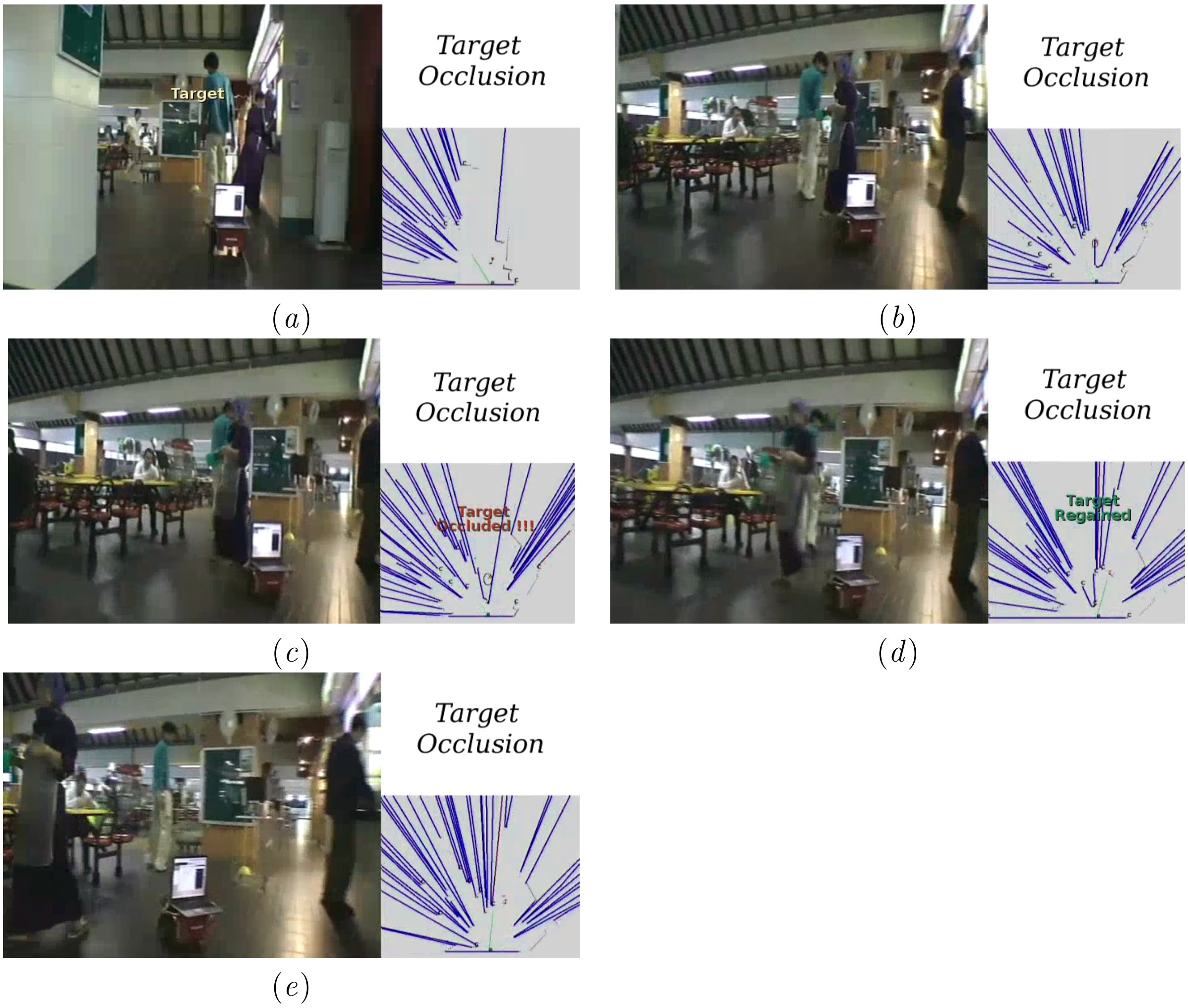


Figure 3.33. Handling temporary occlusions in a dynamic environment. (Video-id: Vantage-ladyOcclude)

CHAPTER 4

2-D STEALTH TRACKER

In this Chapter, we develop motion strategies for an autonomous mobile robot to track a moving target among obstacles and, at same time, remain hidden from the target. This problem is relevant in many applications. For example, in military surveillance, a robot tracker follows a target to acquire information and may endanger itself if exposed. Other examples include graphic animation and monitoring animals in the wild. In all these applications, it is important that a robot tracker not only follows the target, but also avoids detection by the target.

More specifically, both the tracker and the target are equipped with visual sensors. The tracker has two objectives: *tracking*—keeping the target inside the field of view—and *stealth*—staying outside the target's field of view. We call such a tracker a *stealth tracker*. The obstacles in the environment provide motion and visibility constraints as in the normal tracking problem. As described in Chapter 3, we follow the tracking framework that integrates the physical and operational requirements into the tracking problem. In this Chapter, we develop the requirement of maintaining stealth into a planning constraint in visibility based tracking.

Stealth in Tracking Ideally the tracker always moves to keep the target in the middle of its visibility region so that the target cannot escape easily. At the same time, the tracker is required to stay outside the target's visibility region to remain hidden. If the target has a smaller sensing range than the tracker, the tracker can always stay just out of the target's range, which is equivalent of tracking with a minimum distance constraint. We consider the more interesting problem where both the target and the tracker have the same range. This forces the tracker to exploit the local environment and execute intelligent motion to hide within the target's visibility. However, if the target has the same sensing capability as the tracker, *i.e.*, the line of sight visibility relationship is symmetric: if the tracker sees the target, the target sees the tracker. This apparent dilemma can be resolved due to asymmetry in sensing. Although the tracker and the target have the same visual sensors, the visibility relationship may not be exactly symmetric. Since detection requires processing of the visibility data, unless the target is initially aware of the tracker, it may not try to detect the tracker even if the tracker is in fact visible. In reality, detection algorithms do not perform very well near the boundaries of the sensor's limits due to sensor noise. Moreover, the tracker might be partially occluded due to the obstacles in the environment making such a detection difficult. So we assume that the tracker can operate safely near the boundary of the target's visibility region and quickly run outside if it detects risk of exposure. This provides the tracker the "slack" needed to achieve its dual objectives of tracking and stealth.

The stealth tracking problem introduced here is related to the more common tracking problem without the stealth requirement. However, the stealth requirement makes the problem more difficult. We have already mentioned the conflict between the dual objectives of tracking and stealth. In addition, good tracking strategies may not work for stealth tracking at all. For example, in a star-shaped environment, without the stealth requirement, the tracker can simply stay in the middle of the environment, but this does not work for stealth tracking. Fortunately, as we will see, a suitable formulation of the problem translates the stealth requirement as a constraint on the motion of the tracker. The resulting algorithm is almost as efficient as that of tracking without the stealth requirement.

Stealth tracking with mobile trackers have relatively little research results. A few notable works are mentioned in the following. In [73], the tracker displays stealth behavior during navigation. The observer and the goal positions are given and the tracker tries to find a path that minimizes exposure to the observer. Such a tracker has been developed into a real world stealth tracking tracker. Covert tracking has been explored for known and unknown environments in [74, 75]. A visibility model has been developed based on the distance transform from the known observer locations and the tracker tries to follow a path that minimizes the exposure measure for the whole path. In [98], the motion of a tracker is planned in spatio temporal domain for known terrain using models of the observer's motion. For known target and known environment, a detection map is generated and an interception route to the target is planned with minimum exposure.

All the above work focuses on finding a path to a goal position while avoiding the visibility of the observer. As maintaining the view of the observer is not essential, the line of sight visibility detection is rigidly followed *i.e.*, the tracker and target detect each other immediately when they have a line of sight. In our case we try to keep the target in view all the time, for which we have to soften the line of sight stealth requirement. We exploit the limitations of the target's visibility due to occlusion and try to keep close to the target's occlusion edges.

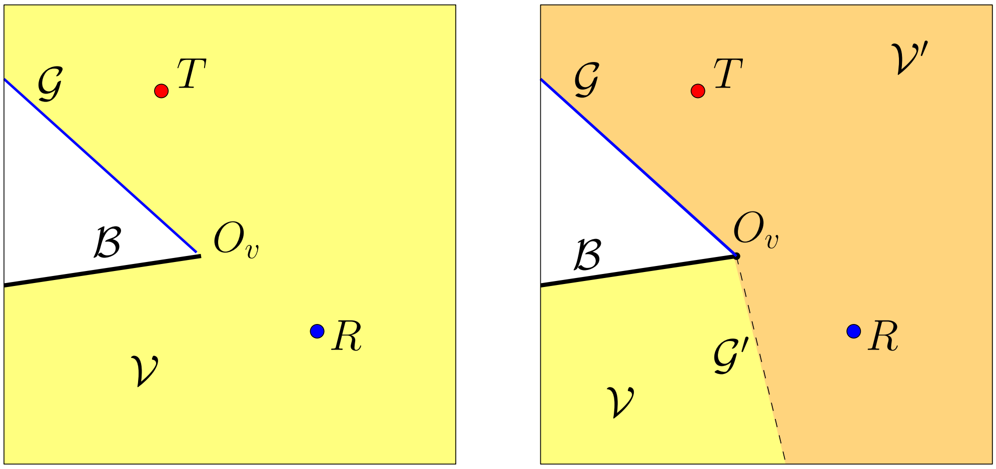
4.1 Problem Formulation

We develop the stealth constraint for tracking and analyze the behavior of the resultant stealth tracker in different environments. As before, the environment and the target's motion are unknown to the tracker. For simplicity of formulation and analysis, we shall restrict ourselves to a pure geometric framework. In general, extending the tracker to real world will follow the steps similar to those discussed in the Chapter 3.

As in the previous formulation, the tracker,R, and the target, T, are point objects in a planar environment, $\mathcal{W}$. The environment, $\mathcal{W}$, tracker visibility, $\mathcal{V}$ and mobility, $\mathbf{v}$, and the target's velocity $\mathbf{v}'$, are modeled as in Chapter 3. For simplicity, we assume omni-directional visibility and mobility for both the target and the tracker. Extensions could be made as in the case of standard tracking without stealth without loss of generality. The tracker visibility can be computed either by a rotational plane sweep algorithm [91] which has a complexity of $O(nlogn)$, n being the number of segments or by thresholding the range data from a sensor, complexity $O(n)$, n being the number of data points, as the range data are sorted during sensing. We assume similar line of sight visibility model for both the target and the tracker.

4.1.1 Target visibility

In an unknown environment, the complete target's visibility might not be possible to compute based on the target's position. The tracker can only compute a subset of the target's actual visibility that is contained inside the tracker's own visibility $\mathcal{V}$. Let $\mathcal{V}'$ denote this portion. Simply stated, $\mathcal{V}' \subset \mathcal{V}$, (Figure 4.1). Stealth is defined against the target's visibility, $\mathcal{V}'$:



(a) Tracker's visibility $\mathcal{V}$ (b) Target's visibility $\mathcal{V}'$

Figure 4.1. Target's visibility, $\mathcal{V}'$, shown in the darker shade, is computed inside $\mathcal{V}$. $\mathcal{G}'$ is generated along $\mathcal{V}'$ boundaries.

$$\mathcal{V}' = \{\mathbf{q} : \mathbf{q} \in \mathcal{V} \mid \overline{T\mathbf{q}} \subset \mathcal{V} \text{ and } dist(T, \mathbf{q}) \leq D_{max}\}$$

where D_{max} is the max range of the target's visibility.

4.1.2 Stealth constraint

As can be seen, $\mathcal{V}'$ is bounded by the tracker's visibility edges as well as occlusion edges inside the tracker's visibility due to the obstacles in the environment. Such an occlusion edge is called *stealth edge* and is shown in Figure 4.1(b) by $\mathcal{G}'$. Stealth edge denotes the boundary that demarcates the tracker-target line of sight and hence separates regions from which the tracker can hide from the target as compared to regions from where the tracker can track the target. The tracker has two contradictory objectives, tracking and stealth. For perfectly symmetrical visibility models between the target and the tracker, absolute stealth tracking cannot be achieved. However,

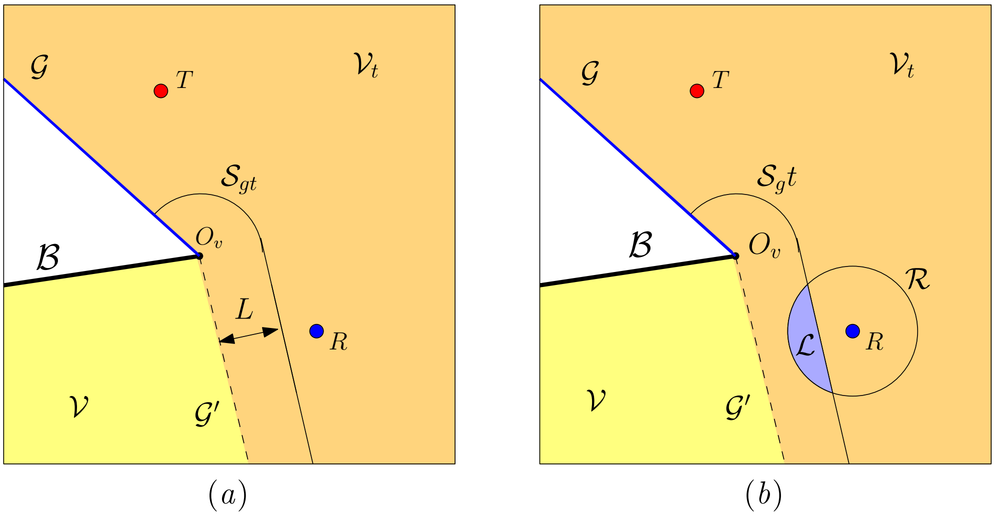


Figure 4.2. Stealth tracking formulation. (a) The stealth region is maintained at a distance L from $\mathcal{G}'$ (b) The feasible region $\mathcal{L}$ is the intersection of $\mathcal{R}$ and $\mathcal{S}$

in reality the visibility relationship is not exactly symmetric. The target may not be initially aware of the tracker and may not detect the tracker even if the tracker is inside the visibility region, $\mathcal{V}'$. Moreover, sensing and detection is not accurate at the sensor edges.

Lookout region In order to stay out of the target's attention, the tracker must stay near the stealth edges. We define a *lookout region* close to the stealth edge from where the tracker will be able to escape the $\mathcal{V}'$ within a determined time span, Δt (Figure 4.5). Since there is an upper bound on the tracker's maximum velocity, this translates into a distance threshold (L) that must be maintained by the tracker to prevent detection. $L = V\Delta t$. Based on L, the *lookout region* of a stealth edge $\mathcal{G}'$ is

$$\mathcal{S}_g = \{\mathbf{q} \in \mathcal{V}' \ \mid dist(\mathbf{q}, \mathcal{G}') \leq L\}$$

The total lookout region $\mathcal{S}$ is then the union of $\mathcal{S}_g$ over all stealth edges ($\mathcal{G}'$) of $\mathcal{V}'$. The stealth constraint then limits the tracking motion to within $\mathcal{S}$.

Problem Statement

The problem of stealth tracking can then be stated as: *For a current target position,* $\mathbf{x}'$ (t)*, and tracker position* $\mathbf{x}$ (t)*, choose a locally optimal tracking action,* $\boldsymbol{v}$ *(t), such that :* $\mathbf{x}'(t+1) \in \mathcal{V}(\mathbf{x}(t+1))$ *and* $\mathbf{x}(t+1) \in \mathcal{S}(t+1)$*.*

4.2 Stealth Tracking Algorithm

4.2.1 Overview

The main essence of stealth tracking is to choose motion strategies to improve the tracker's view while the tracker remains hidden. We apply the tracking framework developed in Chapter 3 for stealth tracking in unknown and dynamic environments. The stealth requirement is developed as a planning constraint and integrated into the local greedy tracker as described before. The optimal direction to move is given by minimizing a risk function that tries to maximize the time for which the target will be kept in view.

$$\mathbf{v}^{\star} = \arg\min_{\mathbf{V}} [\Phi] \qquad s.t. \quad \mathbf{v}^{\star}\Delta t \in \mathcal{L} \tag{4.1}$$

As before the velocity is constrained by the feasible regions. The feasible region consists of the reachable region, $\mathcal{R}$ limited by the lookout regions $\mathcal{S}$. If there are additional constraints, *e.g.* obstacle avoidance $\mathcal{C}_j$, $\mathcal{L}$ is computed as :

$$\mathcal{L} = \mathcal{R} \quad \bigcap \quad \mathcal{S} \quad - \quad \bigcup_j \mathcal{C}_j \tag{4.2}$$

In the following we propose algorithms to compute the target's visibility and the lookout regions. We also show that overall such a stealth tracking algorithm has the complexity of $O(n)$.

4.2.2 Computing the target's visibility

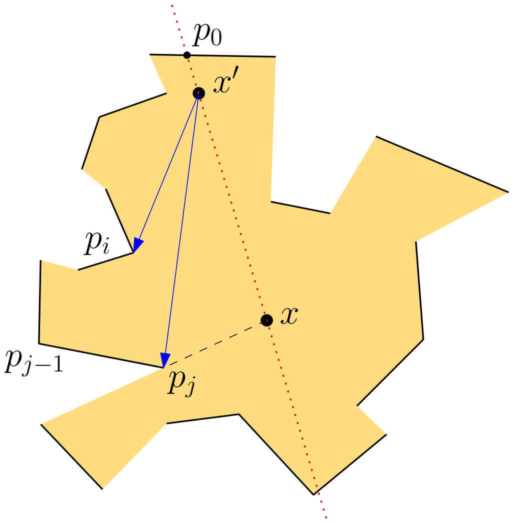


Figure 4.3. Computing the target's visibility region from the tracker's local visibility. P is the portion of the visibility polygon to the left of the red dotted line and P' is the right portion.

The target's visibility $\mathcal{V}'$, is computed within the tracker's visibility $\mathcal{V}$. Since $\mathcal{V}$ is a simple polygon, $\mathcal{V}'$ can be computed in optimal $O(n)$ time [99, 100, 101]. However, a simpler algorithm is possible here, because $\mathcal{V}$ is star-shaped.

Our basic idea is to walk along the boundary of $\mathcal{V}$ and compute the vertices of $\mathcal{V}'$ incrementally. To initialize, we use the line that goes through $\mathbf{x}$ and $\mathbf{x}'$ to divide $\mathcal{V}$ into two halves, P and P' (see Figure 4.3). We now describe the algorithm for P, the left half.

Let p_0 be the point where the ray $\overrightarrow{\mathbf{x}\mathbf{x}'}$ intersects the boundary of $\mathcal{V}$. We number the vertices $p_0, p_1, p_2, \dots$ of P in counter-clockwise order, starting from p_0. Every vertex of P is visible to R, because P is a subset of the visibility region of R. The first vertex of the target's visibility region $\mathcal{V}'$ is p_0, which is visible to T because $\mathbf{x}'$ lies on the line segment $\overline{\mathbf{x}p_0}$. Now we walk along the boundary of P and visit the vertices $p_1, p_2, \dots$ in this order. Let p_i be the latest vertex of P that is visible to T. For every new vertex p_j encountered, where $j > i$, if $\overrightarrow{\mathbf{x}'p_j}$ lies to the right of $\overrightarrow{\mathbf{x}'p_i}$, we simply move to p_{j+1}, because p_j must be blocked by edges adjacent to p_i. If $\overrightarrow{\mathbf{x}'p_j}$ lies to the left of $\overrightarrow{\mathbf{x}'p_i}$, we claim that p_j is visible to T. To see this, we have to show that no boundary edge of $\mathcal{V}$ intersects $\overline{\mathbf{x}'p_j}$. Suppose, for the purpose of contradiction, that some edges of P intersect $\overline{\mathbf{x}'p_j}$. Let e be the intersecting edge closest to Talong $\overline{\mathbf{x}'p_j}$. If e belongs to the polygonal chain between p_0 and p_i, the chain must cross $\overline{\mathbf{x}'p_i}$. This is impossible, because p_i is visible to T. If e belongs to the polygonal chain between p_i and p_j, then some vertex on the chain (excluding p_i and p_j) must be visible to T. This contradicts the fact that p_i is the latest vertex visible to T. If e belongs to the polygonal chain after p_j, the chain must cross $\overline{\mathbf{x}p_j}$. This is also impossible, because p_j is visible to R. Of course, none of the edges in P' can intersect $\overline{\mathbf{x}'p_i}$, otherwise, they would block the visibility line through $\mathbf{x}$ and $\mathbf{x}'$. Hence, p_j is visible to T. We compute the intersection of the ray $\overrightarrow{\mathbf{x}'p_i}$ and the edge $\overline{p_{j-1}p_j}$ and add both the intersection point and p_j to $\mathcal{V}'$ as new vertices. We then move to p_{j+1} and continue until all vertices are visited.

The right half of $\mathcal{V}$ can be processed similarly, except that now, we walk along the boundary in clockwise order. We then merge the two halves of $\mathcal{V}'$.

Our initialization step takes $O(n)$ time, where n is the number of vertices in $\mathcal{V}$. When walking along the boundary of $\mathcal{V}$, we encounter each vertex exactly once and process it in constant time. Hence the following theorem.

Theorem 2 *Let n be the number of vertices of the tracker's visibility region $\mathcal{V}$. Given a target position $\boldsymbol{x}'$, the visibility region of T within $\mathcal{V}$ can be computed in $O(n)$ time.*

4.2.3 Computing Feasible Region

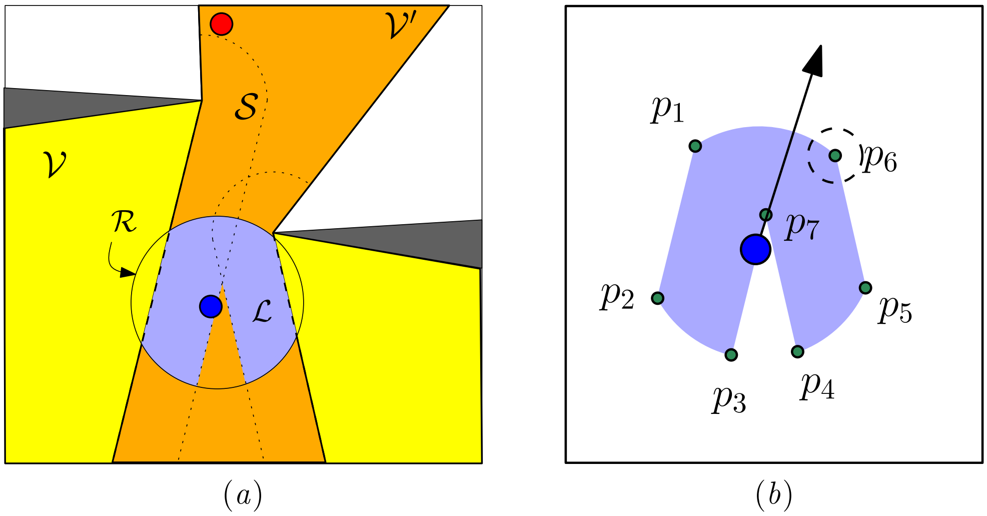


Figure 4.4. Feasibility region (a) $\mathcal{L}$ is computed based on all $\mathcal{G}'$ (b) Assuming polygonal approximation of $\mathcal{L}$, the minimum risk lies on one of the vertices.

In a successful stealth tracking strategy, the tracker's new position must satisfy the following constraints :

- $\mathbf{x} \in \mathcal{R}$. The tracker is limited by its reachability region.
- $\mathbf{x}' \in \mathcal{V}$. Since the environment is not known to the tracker, the tracker should maintain the target in view.

- $\mathbf{x} \in \mathcal{S}$. The tracker has to maintain stealth.

We combine these constraints into a feasible region $\mathcal{L}$ that the tracker can step into being the intersection of reachable regions $(\mathcal{R})$ and lookout regions $(\mathcal{S})$.

$$\mathcal{L} = \mathcal{R} \quad \bigcap \quad \mathcal{S}$$

In Figure 4.4, the intersection of $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{V}$, produces $\mathcal{L}$ for the next time step.

To obtain the feasible region, we first compute the intersection $I = \mathcal{V} \bigcap \mathcal{R}$. Let $p_i, i = 1, 2, \ldots, n$ be the vertices of $\mathcal{V}$ in counter-clockwise order. Since $\mathcal{V}$ is star-shaped, we represent it as a list of triangles $\mathbf{x}p_ip_{i+1}$ for $i = 1, 2, n - 1$. We then intersect each triangle with $\mathcal{R}$ and obtain a new convex shape in constant time. So I can be computed in $O(n)$ time.

The set I is basically the visibility region $\mathcal{V}$ clipped by the boundary circle of $\mathcal{R}$. Usually the time step Δt is small. Thus $\mathcal{R}$ is also small and contains only a small constant number of obstacle vertices and edges, if any. By merging consecutive convex shapes in I whenever possible, the resulting I has only constant size.

For each gap edge $\mathcal{G}'$ of $\mathcal{V}'$, there is a lookout region $\mathcal{S}_g$, which may be quite complex if many obstacles are close together. We compute a simpler set $\mathcal{S}'_g$, which consists of a possibly clipped rectangle adjacent to $\mathcal{G}'$ and two circular sectors that cap the rectangle (Figure 4.5). The width of the rectangle and the radius of the circular sectors are both L, which is the distance threshold for satisfying the stealth requirement. Compared with $\mathcal{S}_g$, the set $\mathcal{S}'_g$ ignores all the obstacles except the two at the end points of $\mathcal{G}'$. The set $\mathcal{S}'_g$ is sufficient for computing the feasible region, because $\mathcal{S}_g \cap I = \mathcal{S}'_g \cap I$. A point in the obstacle is certainly outside $\mathcal{V} \supset I$.

It is important to observe that $\mathcal{S}'_g$ can be represented as a union of at most three convex shapes, each of which has constant size. Assuming I has constant size, we can

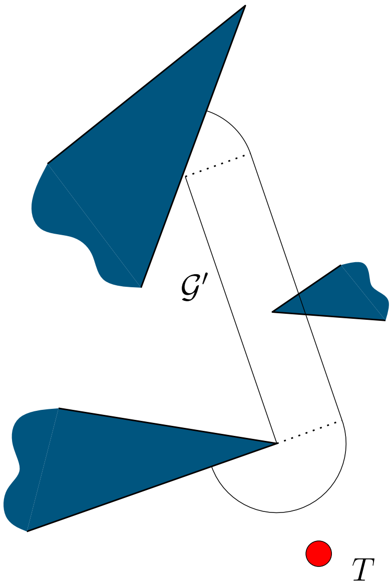


Figure 4.5. Comparing $\mathcal{S}_g$ and $\mathcal{S}'_g$. The obstacle in the middle right is ignored in $\mathcal{S}'_g$.

intersect every convex shape in I with $\mathcal{S}'_g$ in constant time. There are at most $O(n)$ gap edges, and thus we can compute the feasible region in $O(n)$ time. The feasible region is represented as a list of convex shapes approximated by convex polygons, all having constant sizes. Again, if we assume that $\mathcal{R}$ is small enough, then $\mathcal{R}$ intersects only a constant number of lookout regions, and the feasible region has a constant size. We summarize the result in the lemma below.

Lemma 1 *The feasible region for the tracker's position can be computed in $O(n)$ time, where n is the number of vertices in the tracker's visibility region $\mathcal{V}$.*

4.2.4 Constrained Risk

The risk function has the same goal as before: to move the tracker to have better view of the target and keep the target away from the escape gaps. The tracker now optimizes the risk inside $\mathcal{L}$. As done in Chapter Section 3.6, we linearize the risk around the tracker position $\mathbf{x}$.

As the feasible region consists of a list of convex polygons, the problem reduces to linear programming [91]. The minimum solution $\Delta\mathbf{x} = \mathbf{v}^{\star}\Delta t$ must lie at the vertex of one of the convex polygons. By projecting all the vertices in the feasible region along $\mathbf{v}^{\star}$, we can find the minimum in $O(c)$ time, where c is the number of vertices describing the feasible region. We now set the tracker's new position $\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta\mathbf{x}$.

The computation of the total Φ and the optimal velocity $\mathbf{v}^{\star}$ depends on the number of $\mathcal{G}$s and hence can be computed in $O(n)$, from $\mathcal{V}$ of n sides. Combining the above result with Lemma 1 gives the following theorem.

Theorem 3 *Let n be the number of vertices in the tracker's visibility region $\mathcal{V}$. The best action for the tracker to minimize the risk function φ within the feasible region, $\mathcal{L}$, can be computed in $O(n)$ time at each time step.*

Without preprocessing, we cannot hope to achieve running time better than $O(n)$. However, as the tracker's visibility region changes at each time step, there is little opportunity for preprocessing.

4.3 Experiments

The experiments seek to demonstrate and discuss qualitative aspects of the stealth tracker. To test the effectiveness of our tracking algorithm, we implemented it in C++ using the LEDA library and ran it with different environments in simulation. Any kind of objective function can be used depending on the mission objective. The risk function used for the experiments is taken from [3], which maximizes the shortest distance to escape. In Figures 4.6–4.10, we show five representative experiments to illustrate the behavior of the algorithm. In these figures, dark blue regions indicate obstacles. Red crosses mark the target's trajectory. Blue boxes mark the tracker's trajectory. Two circles mark the target's and the tracker's current positions. The tracker's visibility region is marked with thick blue lines. The target's visibility region is shaded in light red. The lookout region is marked with thin black lines.

4.3.1 Stealth behavior: target turning a corner

This example shows the tracker's behavior when the target turns round a corner in a corridor. Initially the tracker stays near the lower right corner of the obstacle inside a lookout region to maintain stealth (Figure 4.6*a*). When the target makes the turn, a new lookout region develops, and the line of sight pivots about the upper right corner of the obstacle. The tracker follows the gradient of the risk function to take advantage of this. It swings out to improve its visibility (Figure 4.6*b*). Finally the tracker goes towards the upper right corner and stays there to maintain stealth (Figure 4.6*c*).

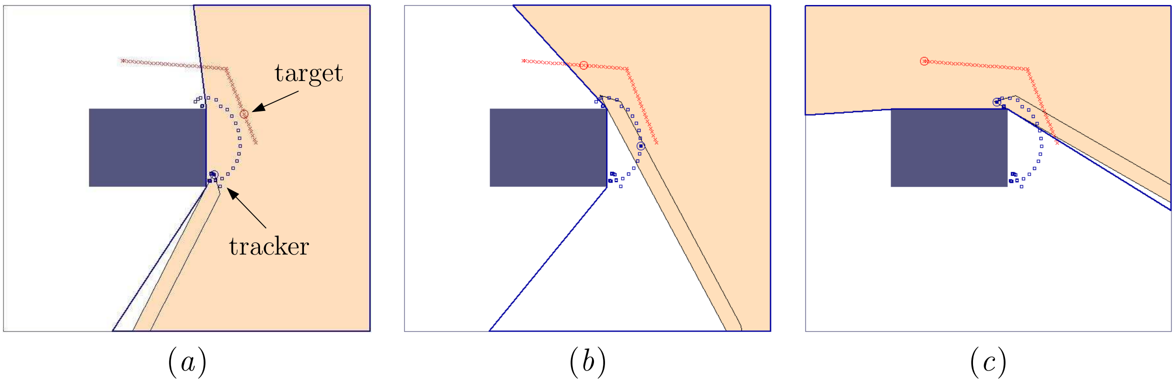


Figure 4.6. The target turns around a corner.

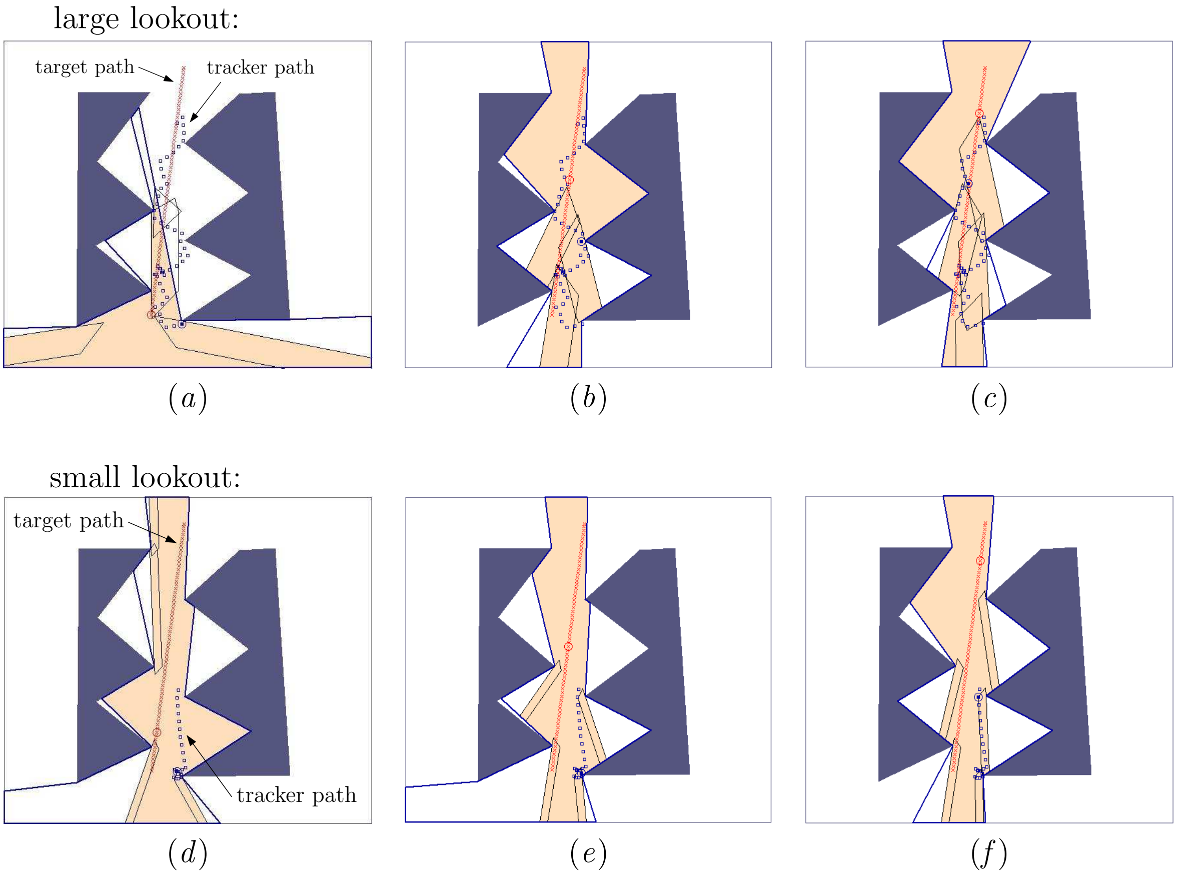


Figure 4.7. The tracker's behavior changes due to different sizes of lookout regions.

4.3.2 Effect of lookout region

The tracker's behavior and success may change drastically, depending on how much risk of exposure that it is willing to take. Recall that the distance threshold L controls the size of lookout regions and reflects the tracker's estimate or willingness to take risk of exposure. If L is large, the tracker has more room to maneuver, usually resulting in more successful tracking.

Consider the environment shown in Figure 4.7. The target moves roughly along a straight path through the zigzag pathway. When L is large, lookout regions from two sides of the pathway often merge and the tracker can jump from one side to the other, using obstacles on both sides for cover (Figures 4.7*a*–*c*). It thus follows the target very closely.

For the same environment and target motion, the tracker's performance worsens when L is small. It is unable to move from one side to the other, because the lookout regions are small and do not merge. It is stuck in one lookout region and has to wait for another lookout region to come close as a result of target motion (Figures 4.7*d*–*e*). However, by then, it falls behind the target by a large distance (Figure 4.7*f*). Although the tracker does not lose the target here, the target can escape easily by making a turn at the end of its straight-line motion. No simple emergency action can recover the target, because the distance between the target and the tracker is too large.

4.3.3 Stealth behavior in cluttered environment: forest

Imagine a target going straight along a road passing through a dense forest. If the tracker follows behind the target on the road, it risks exposure. The figures show the path that the tracker chooses when faced with such a situation. In general, the tracker stays on the side of the road near the obstacles to avoid the risk of exposure. It tries to trail the target as closely as possible, given the constraint that it must stay inside the lookout region of some gap edge (Figure 4.8*a*). When the lookout regions of two gap edges merge, the tracker immediately switches to the new lookout region to further reduce the risk function (Figure 4.8*b*).

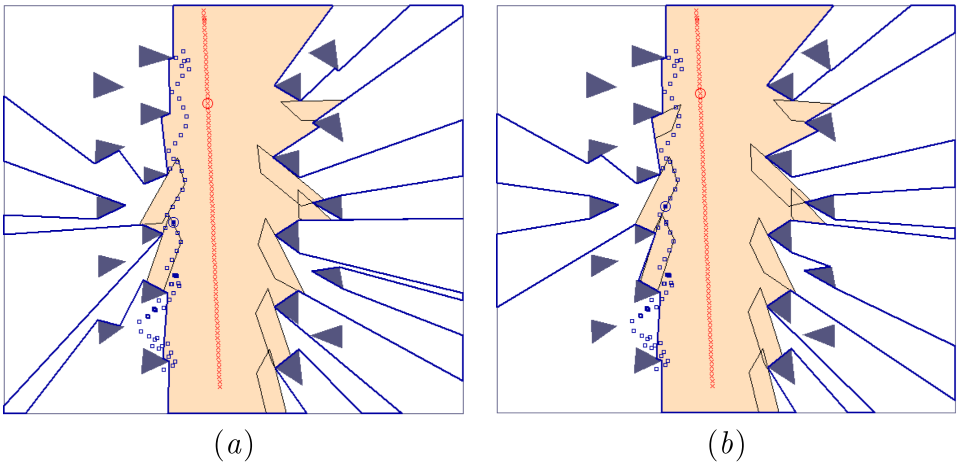
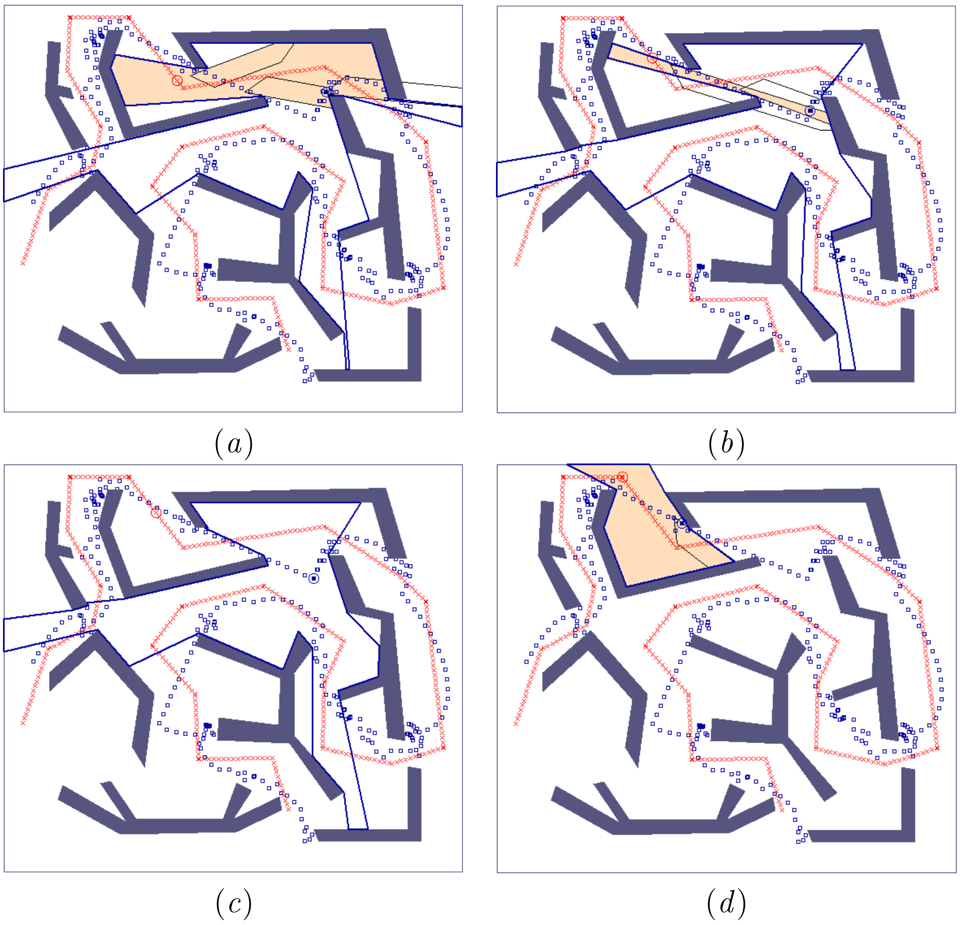


Figure 4.8. The tracker switches lookout regions in a forest like cluttered environment.

4.3.4 Stealth tracking in complex environments

Maze (Figure 4.9) The geometry of this environment is more complex than the others. The target also takes a long and winding path. The tracker is able to follow the target till the end, with the help of emergency actions. In Figure 4.9a–4.9b, the tracker moves almost entirely in the direction ∇e, because the target is very close to the gap edge and the current escape risk is high. At this moment, the tracker does not have the luxury to move along ∇r to reduce the future escape risk. Unfortunately the tracker still loses the target (Figure 4.9c). Taking the emergency action (see Section 3.6.3), the tracker runs to the vertex that abuts the gap edge from which that the target has escaped and regains the target (Figure 4.9d). The tracker loses the target eight times during tracking, but every time it regains the target after taking the emergency action.


(*a*)

(*b*)

(*c*)

(*d*)

Figure 4.9. Losing and regaining the target in the Maze environment.

City Blocks (Figure 4.10) This environment resembles urban areas with roads and housing blocks and gives another example with many obstacles. The target takes a long path, but the tracker successfully follows the target till the end, with a little help from emergency actions.

Table 4.1 shows the performance statistics of the tracker in the above environments. Column 2 lists the total time of target motion. Column 3 lists the length for which the target is visible to the tracker. Column 4 shows the number of times

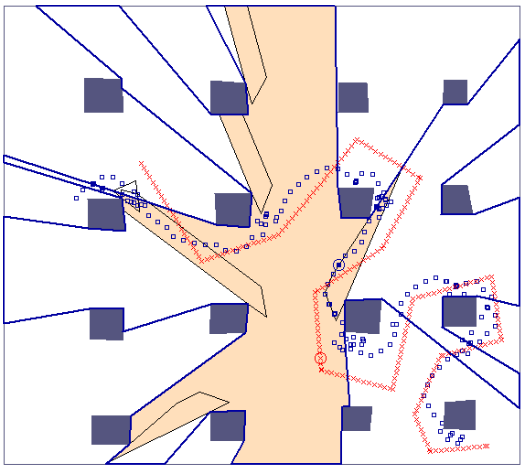


Figure 4.10. Tracking a target among many obstacles in an urban built up environment, with lanes and alleys.

the target is lost and regained. From the table, we see that the tracker loses the target in the two environments where there are many obstacles. In the City Block environment, the tracker loses the target two times, for a duration of one step each. Every time that the tracker loses the target, it recovers it in the next step by taking the emergency action. In the Maze environment, which is more complex, the tracker loses the target eight times. It always recovers the target, but it takes more time than that needed in City Block. When there are many obstacles causing occlusions, the tracker is more likely to lose the target. The emergency action, despite it simplicity, seems quite effective in such environments.

4.4 Discussion

We now discuss some limitations of our approach and possible solutions.

Table 4.1. Tracking performance.

Environment	Total No. Steps	No. Steps Visible	No. Times Lost & Regained
Corridor	102	102	0
Forest	214	214	0
Zigzag (large L)	130	130	0
Zigzag (small L)	130	130	0
Maze	730	663	8
City Blocks	468	466	2

"Discontinuity" in the lookout regions. Sometimes a lookout region may suddenly disappear. Consider the example shown in Figure 4.11. Initially the tracker is at $\mathbf{x}$, safely inside the lookout region of a gap edge $\mathcal{G}$. As it moves to the new position, $\mathcal{V}$ is reduced. Although the target has not moved at all, $\mathcal{G}$ now lies outside $\mathcal{V}$. The tracker can no longer ascertain the existence of $\mathcal{G}$ and the associated lookout region, and assumes that it is exposed. This happens, because the tracker uses only local information from the sensor and knows only the environment within $\mathcal{V}$. If the tracker had fused sensor data through history to produce a global map, it would know that it is inside the lookout region of $\mathcal{G}$. The use of only local information is clearly a limitation here.

This situation occurs only if the tracker is close to the obstacle vertex $\mathcal{O}_v$ that supports the line gap edge $\mathcal{G}$ (Figure 4.11). If the tracker crosses $\mathcal{O}_v$, the $\mathcal{G}$ disappears from $\mathcal{V}$. This situation can be addressed by truncating the lookout region $\mathcal{S}_g$ so that the tracker always remains closer to $\mathcal{O}_v$, than the interior of $\mathcal{G}$. This can be done by effectively removing the circular capped region around $\mathcal{O}_v$ of $\mathcal{S}_g$.

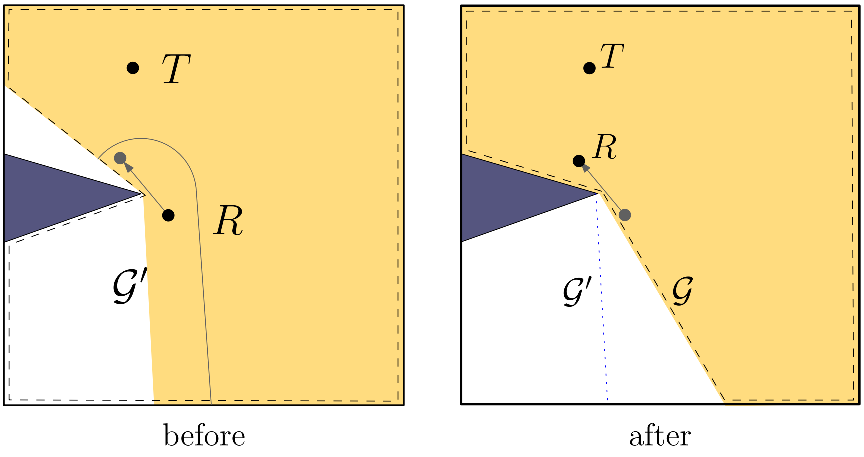


Figure 4.11. The target's visibility regions before and after the tracker's move. The dashed lines indicate $\mathcal{V}$. The shaded region indicates $\mathcal{V}'$.

4.5 Summary

We have introduced the stealth tracking problem, in which a tracker tries to track a moving target among obstacles and remain hidden from the target at the same time. As before, the tracking algorithm uses only local information from the tracker's visual sensors and assumes no prior knowledge of target motion or a global map of the environment. It defines a function that measures the target's escape risk and tries to minimize the risk function, subject to the stealth constraint, in order to achieve the dual objectives of tracking and stealth. The algorithm is efficient, taking O(n) time at each time step, where n is number of vertices of the tracker's visibility region. Experiments in simulation show that the algorithm performs well in difficult environments.

CHAPTER 5

MOTION STRATEGIES: 3-D

In this chapter we develop the vantage tracker for the 3-D environment. Moving from 2-D to 3-D space offers opportunities to improve tracking performance. The 3-D space is more flexible: the tracker gains one additional degree of freedom to maneuver, which potentially improves tracking performance. For example, a tracker helicopter follows and monitors a ground target. If the target turns around at the corner of a building, the helicopter may choose to fly over the building to keep the target visible, instead of following the target around the building. However, the same flexibility leads to several challenges. Just as the tracker, the target may also gain more room to maneuver and more easily escape from the tracker sensors' visibility region. In addition, the visibility relationships in 3-D are more complex than those in 2-D.

5.1 Problem formulation

The basic formulation of the problem remains the same as in Section 3.1. The tracking environment is now a 3-D Euclidean space cluttered with obstacles. Such environments may occur either indoors (e.g., tracking a human in a regular home or office environment) or outdoors (e.g., tracking a target in an urban environment).

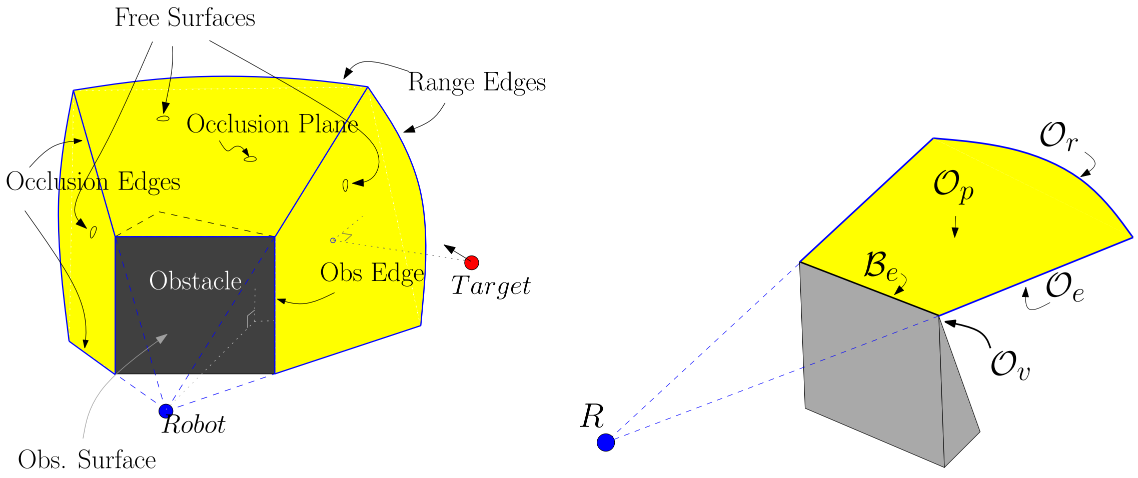
5.1.1 3-D Motion Model

The tracker and the target are modeled as free flying point objects with no motion constraints, except for bounds on their maximum speeds, V and V', respectively. As before, assume $V' \leq V$; otherwise, the problem becomes uninteresting if the target tries to escape by moving faster than the tracker. The tracker motion is modeled as a simple discrete-time transition equation. If at time t, the tracker at position $\mathbf{x}(t)$ moves with a velocity $\mathbf{v}(t)$, its position at time $t+1$ is given by

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{v}(t)\Delta t$$

with $|\mathbf{v}(t)| \leq V$ for all t. So the region $\mathcal{R}$, of the tracker in Δt is a sphere of radius $V\Delta t$ centered at $\mathbf{x}(t)$. The equations are the same as in 2-D except that the vectors $\mathbf{x}$ and $\mathbf{v}$ are now defined in 3-D euclidean space.

5.1.2 3-D Visibility Model



(a) 3-D occlusion surface

(b) Occlusion plane $(\mathcal{G})$ notations

Figure 5.1. 3-D Visibility model.

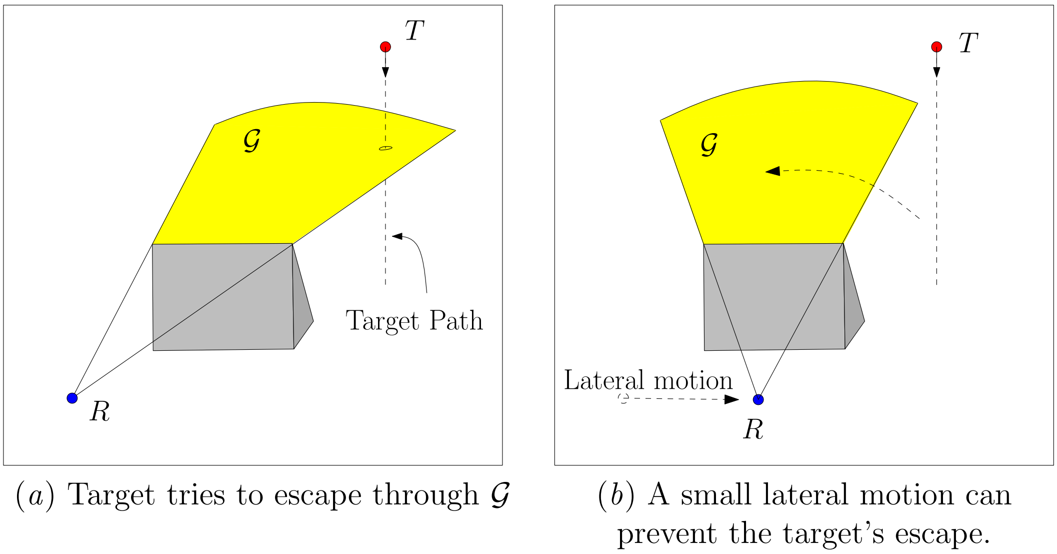
The assumptions on the environment and the target motion are the same: both are unknown to the tracker *a priori*, but we assume that the target is initially visible to the tracker. The tracker uses 3-D visual sensors, *e.g.*, cameras or laser range finders, for sensing the target and its surroundings. For simplicity, we assume omni directional sensing capabilities for the sensors, although field of view based models can be incorporated easily. Visibility is modeled here as simple line of sight sensing, bounded by a maximal range D_{max}. Thus, in an open space, the tracker's visibility region is a sphere of radius D_{max} with the center at $\mathbf{x}(t)$. Obstacles in the environment may obstruct visibility. Let $\mathcal{F}$ denote the space that is within the sphere and is free of obstacles. The set of all points $\mathbf{q}$ within $\mathcal{F}$ visible to the tracker defines the visibility region, $\mathcal{V}$, of the tracker at position $\mathbf{x}$:

$$\mathcal{V}(x) = \{\mathbf{q} \in \mathcal{F} \mid \overline{\mathbf{xq}} \subset \mathcal{F} \text{ and } dist(\mathbf{x}, \mathbf{q}) \leq D_{max}\}$$

where $\overline{\mathbf{xq}}$ denotes the line segment joining $\mathbf{q}$ and $\mathbf{x}$. Clearly, $\mathcal{V}$ is now a star-shaped generalized polyhedral. Note that although we use the same symbol $\mathcal{V}$ for the tracker's visibility as in the previous chapter 3, the visibility set is defined in 3-D.

The tracker plans its actions using local geometric information from its sensors. Based on a polyhedral approximation for the environment geometry, $\mathcal{V}$ takes the shape of a generalized polyhedron bounded by two types of surfaces (shown in Figure 5.1a): the surfaces that are the boundaries of the polyhedral obstacles, *obstacle surfaces*, and the surfaces that lie in $\mathcal{F}$, *free surfaces.* The free surfaces can be further divided into *occlusion surfaces* that are caused by the obstruction of visibility and *range surfaces* that are caused by the visibility limits D_{max}. The free surfaces pose the risk of losing the target.

In general, the occlusion surfaces can be made up of a number of *occlusion planes* concatenated to each other. Adjacent occlusion planes meet in the *occlusion edges*. Let us take the occlusion plane $\mathcal{G}$ as shown in Figure 5.1b. $\mathcal{G}$ consists of the interior of the occlusion plane ($\mathcal{O}_p$) bounded by a pair of occlusion edges ($\mathcal{O}_e$) at the lateral sides, obstacle edge ($\mathcal{B}_e$) in front and range edge ($\mathcal{O}_r$) at the rear. Let us term the vertex at which $\mathcal{O}_e$ and $\mathcal{B}_e$ meet as an *occlusion vertex* ($\mathcal{O}_v$).



(a) Target tries to escape through $\mathcal{G}$

(b) A small lateral motion can prevent the target's escape.

Figure 5.2. Increase in complexity of planning due to additional degree of freedom in 3-D tracking.

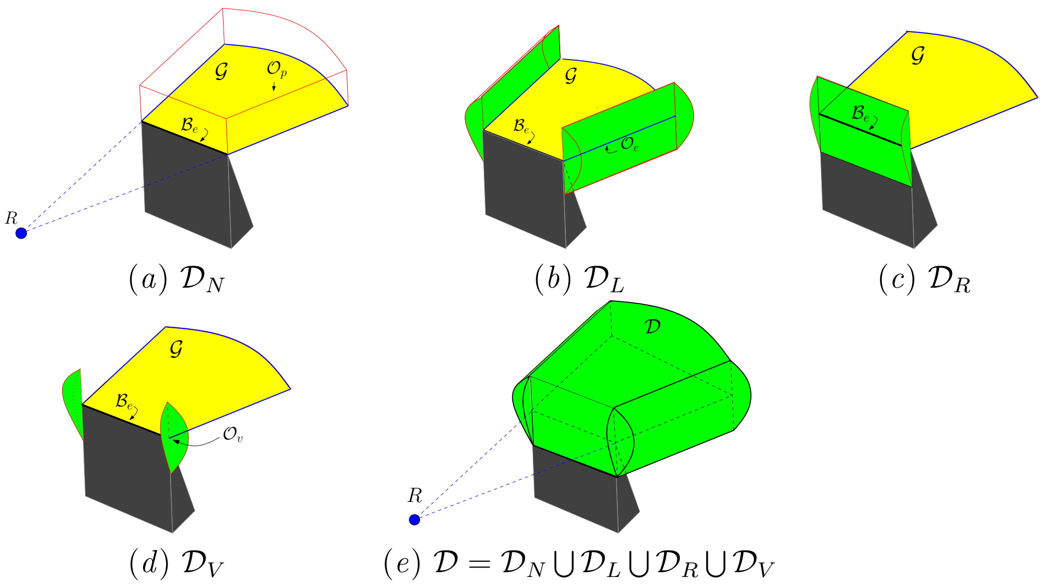
Complexity of 3rd dimension In the 3-D case, occlusion planes replace occlusion edges, and the tracker and the target gain one additional degree of freedom to maneuver. This introduces additional complexity in the tracking problem. To illustrate this, let us take a simple scenario as shown in Figure 5.2a. The gray object occludes the trackers view to above the occlusion plane. The target moves in a path that intersects this plane and escapes the tracker's visibility. To prevent this, the tracker has to manipulate the occlusion plane away from the target. The usual steps as seen from the 2-D formulation would be to swing the plane away from the target and move towards the obstacle in view of eliminating the occlusion plane. As the visibility plane is finite, the tracker also has the option of shifting the plane, by a lateral motion, such that the target's projected path does not intersect the occlusion plane in 3-D as shown in Figure 5.2*b*. Such a lateral motion is not automatically obvious by a simple extension of the 2-D approach. The effect of any other occlusion plane can be eliminated by making the occluding obstacle in Figure 5.2 arbitrarily short in the vertical direction.

In spite of the additional dimension, the intuition on balancing the short-term and long-term goals, as illustrated in the 2-D example, remains basically the same. Below we propose a carefully constructed risk function to capture this intuition for each free surface. The total risk is a sum of these risks, weighted by the probabilities of the target's escaping through the corresponding surfaces. The tracker's action is then a local greedy step to minimize the total risk.

5.2 Relative Vantage in 3-D

The concept of relative vantage and danger zone, $\mathcal{D}$ that partitions $\mathcal{V}$ into dangerous and safe zones remain the same as in Section 3.1. We construct a danger zone ($\mathcal{D}$), such that for any position of the target within that region, the tracker cannot eliminate $\mathcal{G}$ (by moving towards $\mathcal{B}_e$) before the target reaches $\mathcal{G}$, given the target's and tracker's current velocities. We have the same definition as in the 2-D case except that now the vectors are in 3-D and $\mathcal{G}$ represents a finite plane.

$$\mathcal{D} = \{\mathbf{q} : \mathbf{q} \in \mathcal{V} \bigwedge time(\mathbf{q}, \mathcal{G}) < time(\mathbf{x}, \mathcal{G})\} \tag{5.1}$$



(a) $\mathcal{D}_N$ (b) $\mathcal{D}_L$ (c) $\mathcal{D}_R$

(d) $\mathcal{D}_V$ (e) $\mathcal{D} = \mathcal{D}_N \bigcup \mathcal{D}_L \bigcup \mathcal{D}_R \bigcup \mathcal{D}_V$

Figure 5.3. Vantage Zone $\mathcal{D}$ for a single occlusion plane

As shown in Figure 5.1b, the occlusion plane ($\mathcal{G}$) can be seen as a union of $\mathcal{O}_p$, $\mathcal{O}_e$, $\mathcal{B}_e$, $\mathcal{O}_r$ and $\mathcal{O}_v$. $\mathcal{O}_r$ exists due to the sensor range limitation and we shall address it later in Section 5.3.2. Based on the proximity to these geometric features, $\mathcal{D}$ can be partitioned into four regions, shown in Figure 5.3.

- $\mathcal{D}_N$: The region in $\mathcal{D}$ that is closest to the interior of $\mathcal{G}$, i.e. $\mathcal{O}_p$.

  $$\mathcal{D}_N = \{\mathbf{q} \in \mathcal{D} | nearest(\mathbf{q}, \mathcal{G}) \in \mathcal{O}_p\}$$

  This region is depicted in Figure 5.3a.

- $\mathcal{D}_L$: The region in $\mathcal{D}$ nearest to the occlusion edge $\mathcal{O}_e$.

  $$\mathcal{D}_L = \{\mathbf{q} \in \mathcal{D} | nearest(\mathbf{q}, \mathcal{G}) \in \mathcal{O}_e\}$$

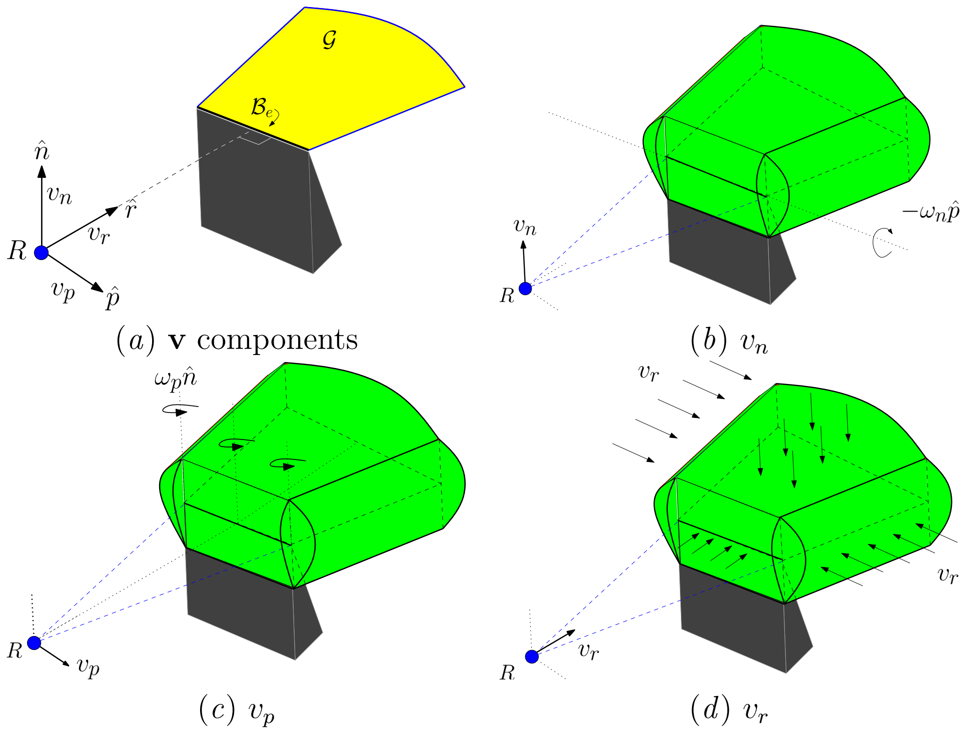
$\mathcal{D}_L$ is depicted by semi-circular cylindrical pieces abutting the occlusion edges, Figure 5.3b.

- $\mathcal{D}_R$: The region in $\mathcal{D}$ closest to the obstacle edge $\mathcal{B}_e$, as shown in Figure 5.3c. $\mathcal{D}_R$ is a semi-circular cylindrical piece abutting $\mathcal{B}_e$.

$$\mathcal{D}_R = \{\mathbf{q} \in \mathcal{D} | nearest(\mathbf{q}, \mathcal{G}) \in \mathcal{B}_e\}$$

- $\mathcal{D}_V$: The regions in $\mathcal{D}$ nearest to the occlusion vertex, $\mathcal{O}_v$, as shown in Figure 5.3d. $\mathcal{D}_V$ is a pair of spherical sectors.

$$\mathcal{D}_V = \{\mathbf{q} \in \mathcal{D} | nearest(\mathbf{q}, \mathcal{G}) = \mathcal{O}_v\}$$



(a) $\mathbf{v}$ components

(b) v_n

(c) v_p

(d) v_r

Figure 5.4. The effect of tracker velocities v_n,v_p and v_r on $\mathcal{D}$

Effect of tracker velocity $\mathcal{G}$ depends on the tracker's position w.r.t. the obstacle, and so the tracker can manipulate $\mathcal{G}$ by its motion. Let us introduce a reference frame on the tracker $\hat{\mathbf{n}}$, $\hat{\mathbf{p}}$ and $\hat{\mathbf{r}}$, shown in Figure 5.4*a*, to simplify the analysis. The tracker motion normal to $\mathcal{G}$, along $\hat{\mathbf{n}}$, increases the plane's distance to the target by swinging about $\mathcal{B}_e$ (Figure 5.4*b*). Similarly, motion parallel to $\mathcal{B}_e$, along $\hat{\mathbf{p}}$, increases the distance of the $\mathcal{O}_e$ to the target by swinging $\mathcal{O}_e$ laterally about $\mathcal{O}_v$ (Figure 5.4*c*). Both these velocity components help in achieving the short term goal of preventing the target's immediate escape. On the other hand, motion towards $\mathcal{G}$, along $\hat{\mathbf{r}}$, improves the future tracking capability of the tracker. Moving closer to the obstacle allows $\mathcal{G}$ to be swung more effectively (similar to 2-D swinging).

Tracking approach In order to maintain a good relative vantage over the target, the tracker must manipulate $\mathcal{D}$ such that the target is pushed out of $\mathcal{D}$ if it is inside, else move $\mathcal{D}$ as far from the target as possible. A good estimate of the relative vantage, is the shortest amount of time, $t_{r.v}$, the target needs to reach the boundary of $\mathcal{D}$. $t_{r.v}$ (vantage time) is positive if the target is inside $\mathcal{D}$ and negative if outside. Let $\mathcal{P}$ be the point through which the target exits $\mathcal{D}$. $t_{r.v}$ can then be approximated by,

$$t_{r.v} = \frac{Dist(T, \mathcal{P})}{V_{eff}(T, \mathcal{P})} \tag{5.2}$$

where $Dist(T, \mathcal{P})$ and $V_{eff}(T, \mathcal{P})$, represent the relative distance and velocity of the target to $\mathcal{P}$.

Depending on the tracker's motion, $\mathcal{P}$ can lie either in $\mathcal{D}_N$, $\mathcal{D}_L$, $\mathcal{D}_R$ or $\mathcal{D}_V$. Optimizing $t_{r.v}$ over all the four regions involves finding an optimal velocity $\mathbf{v}^\star$ such that,

$$\mathbf{v}^\star = \arg\min_{\mathbf{v}}(t_{r.v}) \quad \forall \mathcal{P} \in \mathcal{D}_N, \mathcal{D}_L, \mathcal{D}_R, \mathcal{D}_V \tag{5.3}$$

To simplify computation, let us approximate $t_{r.v}$ by its lowest bound,

$$t_{r.v} = min(t_{\mathcal{DN}}, t_{\mathcal{DL}}, t_{\mathcal{DR}}, t_{\mathcal{DV}}) \tag{5.4}$$

Once $t_{r.v}$ is found, the gradient of $t_{r.v}$ computed at the current values gives the optimal direction to move. The optimization then reduces to finding $\mathbf{v}^\star$ that minimizes $t_{r.v}$, and moving one step along the direction:

$$\nabla t_{r.v} = \frac{\partial t_{r.v}}{\partial v_r}\hat{\mathbf{r}} + \frac{\partial t_{r.v}}{\partial v_n}\hat{\mathbf{n}} + \frac{\partial t_{r.v}}{\partial v_p}\hat{\mathbf{p}} \tag{5.5}$$

The minimization is done analytically or by a few steps of the Newton-Raphson routine.

5.3 Computing risk analytically

Let us now compute the analytical forms for $t_{r.v}$ Equation 5.2.

5.3.1 Occlusion Planes

Risk parameters As in 2-D, we define the *danger zone* that partitions $\mathcal{V}$ into regions with relative vantage. The risk of the target's escape through $\mathcal{G}$ depends on its shortest distance of escape (SDE) to the plane, as denoted by e in Figure 5.5a. The smaller the e is, the higher is the risk of the target's escaping. The risk also depends on how well the tracker can manipulate $\mathcal{G}$ away from the target. The effectiveness of manipulation depends on the relative positioning of the tracker and the target w.r.t. $\mathcal{G}$. Let r be the distance between the tracker and $\mathcal{B}_e$, and r' be the distance between the target and $\mathcal{B}_e$, projected into the occlusion plane, shown in Figure 5.5a.

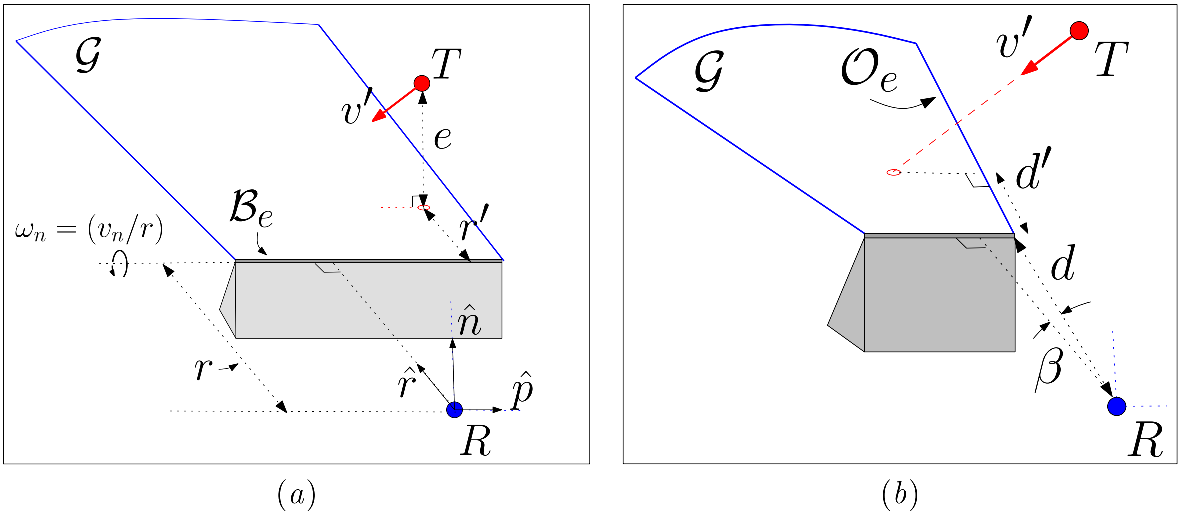


Figure 5.5. Parameters involved in the Risk Formulation

If $r' > r$, the tracker has an advantage in swinging $\mathcal{G}$ farther away from the target given the same velocity components of the tracker and target normal to $\mathcal{G}$. Similarly in Figure 5.5b, if d' is the distance of the target's projection on $\mathcal{O}_e$ from $\mathcal{O}_v$ and d the corresponding measure for the tracker, $d' > d$ gives the tracker an advantage in swinging $\mathcal{O}_e$ about $\mathcal{O}_v$.

Relative velocity As shown earlier, the tracker can manipulate $\mathcal{D}$ by its motion. v_r causes $\mathcal{D}$ to shrink towards $\mathcal{G}$ (Figure 5.4d), v_n causes $\mathcal{D}$ to swing about $\mathcal{B}_e$ (Figure 5.4b), in the normal direction to the $\mathcal{G}$ by an angular velocity ω_n and v_p shifts $\mathcal{D}$ along $\mathcal{B}_e$ by swinging $\mathcal{O}_e$ laterally by an angular velocity ω_p (Figure 5.6). From Figure 5.5(a & b) and Figure 5.6,

$$\begin{aligned}\omega_n &= -(v_n/r)\hat{\mathbf{p}} \\ \omega_p &= ((v_p \cos\beta - v_r \sin\beta)/d)\hat{\mathbf{n}}.\end{aligned}$$

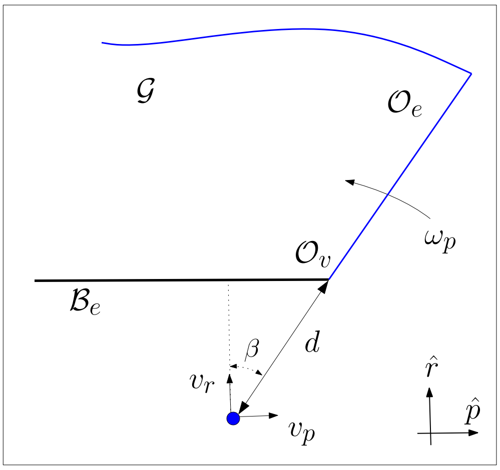


Figure 5.6. Computing ω_p

In order to find the analytical expressions for $t_{r.v}$ in the four regions, let us take $V = V'$. This does not take away the generality of the derivation as we can introduce a scale factor η such that, $V = \eta V'$. Let us define the distance of the tracker to $\mathcal{G}$, as D_{safe}: $D_{safe} = Dist(R, \mathcal{G})$. For the following, we take $\eta = 1$, which gives $D_{safe} = r$.

Computing $t_{\mathcal{DN}}$

Figure 5.7 shows a portion of $\mathcal{D}_N$. Let the target's velocity along the normal direction be v_n'. The relative velocity of the planar $\mathcal{D}_N$ surface w.r.t. the target is $v_{eff} = v_r + \omega_n r' - v_n'$ and effective $Dist(T, \mathcal{P}) = D_{safe} - e$, giving,

$$t_{\mathcal{DN}} = \frac{r - e}{v_r + \omega_n r' - v_n'}$$

Note that $t_{\mathcal{DN}}$ does not depend on v_p. This is to be expected since moving parallel to the $\mathcal{G}$ does not change the tracking parameters and hence the risk.

The optimal action to minimize $t_{\mathcal{DN}}$ is given as,

$$v^*_{\mathcal{DN}}(v_r, v_n, v_p) \quad = \quad \arg \min_{(v_r, v_n, v_p)} \left(\frac{r - e}{v_r + \omega_n r' - v_n'} \right)$$

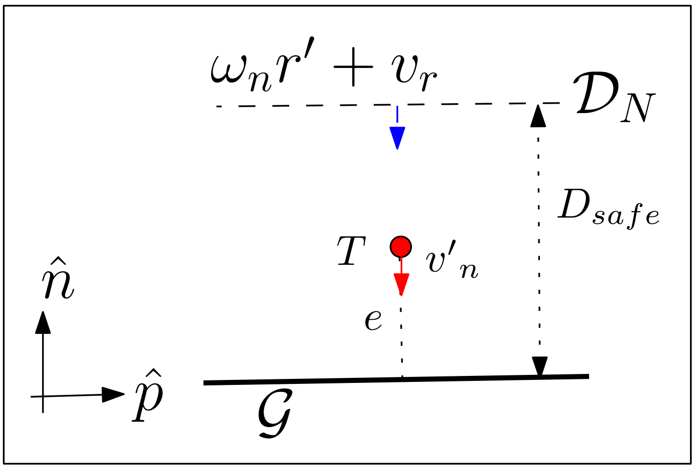


Figure 5.7. Computing $t_{\mathcal{DN}}$

$$= \frac{t_{\mathcal{DN}}}{v_{eff}} \left(\frac{r'}{r} \hat{\mathbf{n}} + \hat{\mathbf{r}} \right) \tag{5.6}$$

Interestingly, the result takes the same form as risk optimization in 2-D, Section 3.4.1. This shows that in a tracking scenario with infinite occlusion planes, (infinite polygons and infinite sensing ranges), the 3-D tracking can be treated as a 2-D tracking problem for individual occlusion planes.

Computing $t_{\mathcal{DL}}$

Let us ignore v_r for now. Figure 5.8 shows a portion of $\mathcal{D}_L$. The resultant motion of the target and $\mathcal{D}_L$ can be shown to lie along section AA'. If $\mathcal{P}$ lies on $\mathcal{D}_L$, projecting the target velocity and the distance to a section AA' that is perpendicular to $\mathcal{O}_e$, does not change the ratio of the distance to the velocity. We can then safely compute $t_{r.v}$ by just considering the motion in AA' plane. Let us introduce a coordinate system fixed at occlusion vertex $\mathcal{O}_v$ $\{\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}\}$ as shown in Figure 5.8. Let the target position

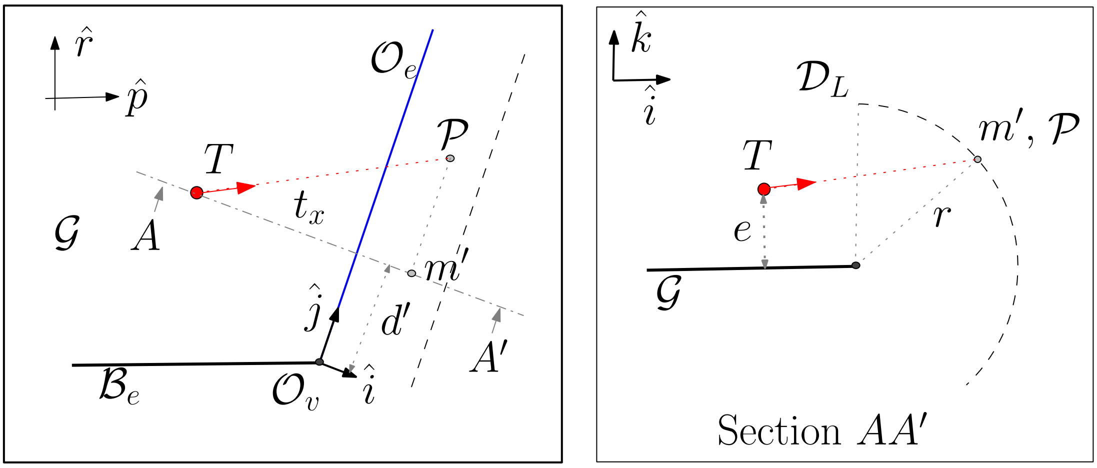


Figure 5.8. Computing $t_{\mathcal{DL}}$

w.r.t. the new coordinates be denoted by $\mathbf{x}'_{DL}$. Then,

$$\mathbf{x}'_{DL} \quad = \quad -t_x\hat{\mathbf{i}} + d'\hat{\mathbf{j}} + e\hat{\mathbf{k}}$$

The effective velocity, v_{eff}, is a resultant of three components: the normal and lateral swing, the shrinking of $\mathcal{D}$ and the velocity of the target $\mathbf{v}'$.

$$\begin{aligned} v_{eff} &= \omega_n\hat{\mathbf{p}} \times \mathbf{x}'_{DL} + \omega_p\hat{\mathbf{k}} \times \mathbf{x}'_{DL} - \mathbf{v}' \\ &= (-\omega_p d' + \omega_n e \sin\beta)\hat{\mathbf{i}} + (-\omega_p t_x - \omega_n e \cos\beta)\hat{\mathbf{j}} + \omega_n r'\hat{\mathbf{k}} - \mathbf{v}'_0 \end{aligned}$$

where β is the angle $\hat{\mathbf{j}}$ makes with $\hat{\mathbf{r}}$. The effect of v_r is to shrink $\mathcal{R}$ towards $\mathcal{O}_e$. This gives the condition,

$$\mid \mathbf{x}'_{DL} + v_{eff} \times t_{r.v} \mid_{ik} = D_{safe} - v_r t_{r.v}.$$

Solving for $t_{\mathcal{DL}}$ gives,

$$t_{\mathcal{DL}} \quad = \quad \frac{-(-t_x v'_x + e v'_z + r v_r) \pm \sqrt{(-t_x v'_x + e v'_z + r v_r)^2 - 4({v'_x}^2 + {v'_z}^2 - {v_r}^2)({t_x}^2 + e^2 - r^2)}}{2({v'_x}^2 + {v'_z}^2 - {v_r}^2)} \tag{5.7}$$

Where, $t_{\mathcal{DL}}$ is chosen to be the lower positive value. In the proposed approximation it is not necessary to solve Equation 5.7 exactly, and minimizing $t_{\mathcal{DL}}$ is done using a few iterations of any minimization routine. We use the Newton-Raphson minimization due to its simplicity.

$$v^*_{\mathcal{DL}}(v_r, v_n, v_p) \quad = \quad \arg \min_{(v_r, v_n, v_p)} (t_{\mathcal{DL}}) \tag{5.8}$$

As the actual optimization formulation is transient and changes at each time step, we just take a few steps in minimizing $t_{\mathcal{DL}}$. In the next time step the current values are chosen as initial parameters. As the objective function itself changes at a lower frequency than the computation cycle, the solution tends to converge towards the relevant local minima.

Computing $t_{\mathcal{DR}}$ and $t_{\mathcal{DV}}$

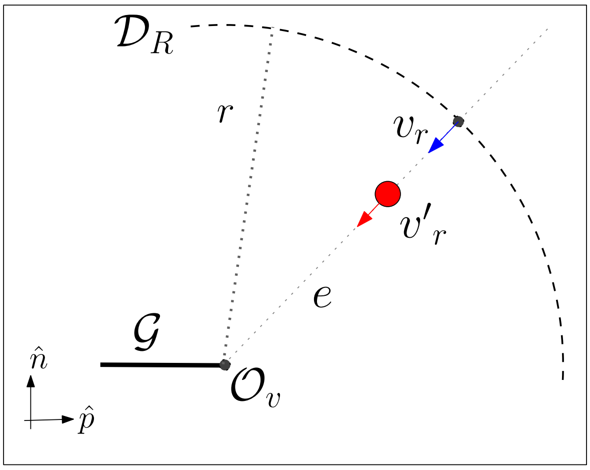


Figure 5.9. Computing $t_{\mathcal{DR}}$

$\mathcal{D}_R$ and $\mathcal{D}_V$ behave in a similar way that, neither ω_n nor ω_p cause any relative velocity of the surface towards the target in this case, as shown in Figure 5.9. Taking the radial component of velocity and distance, $v_{eff} = v_r - v'_r$ and $dist(target, \mathcal{P}) = D_{safe} - e$ giving,

$$t_{\mathcal{DR}} = t_{\mathcal{DV}} = \frac{r - e}{v_r - v'_r}$$

$t_{\mathcal{DV}}$ is given by the same expression for its corresponding case.

Minimizing $v^*_{\mathcal{DR}}$,

$$\begin{aligned} v^*_{\mathcal{DR}}(v_r, v_n, v_p) &= \arg\min \left(\frac{r - e}{v_r - v'_r} \right) \\ &= \frac{t_{\mathcal{DR}}}{v_{eff}} \hat{\mathbf{r}} \qquad (5.9) \end{aligned}$$

The action is to go straight towards $\mathcal{B}_e$. v_p and v_n do not contribute to minimizing $t_{\mathcal{DR}}$.

5.3.2 Formulation for Range Edges

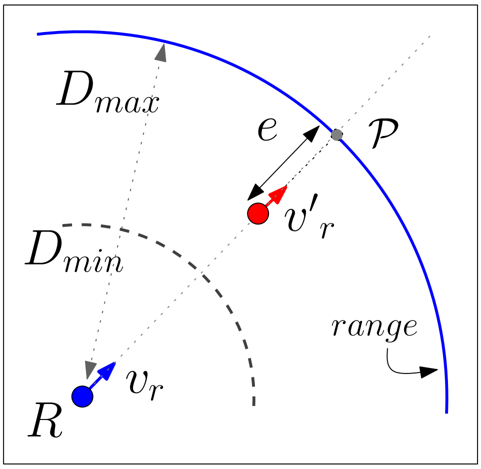


Figure 5.10. Computing $t_{r.v}$ for Range

The tracker cannot eliminate range surfaces by moving towards it. As before we use a measure of escape time, t_{esc}, to formulate the risk of the target's escape through $\mathcal{R}$,

$$t_{esc} = \frac{Dist(T, \mathcal{P})}{V_{eff}(T, \mathcal{P})}$$

In the definition of t_{esc}, we redefine $\mathcal{P}$ to be the nearest point along the radial direction to the range surface as shown in Figure 5.10. Then the effective velocity becomes $v_{eff} = v_r - v_r'$, and $dist(target, \mathcal{P}) = D_{max} - e$. e is then calculated from $\mathcal{P}$.

Minimizing t_{esc},

$$\begin{aligned} v_{range}(v_r, v_n, v_p) &= \arg\min\left(\frac{D_{max} - e}{v_r - v_r'}\right) \\ &= \frac{t_{esc}}{v_{eff}}\hat{\mathbf{r}} \end{aligned} \tag{5.10}$$

As mentioned earlier, the behavior generated by the range surfaces alone, makes the tracker move towards the target. This is the same as visual servo behavior. This again shows that visual servo is a special case of vantage tracking when there are no occlusions.

5.3.3 Handling Multiple Occlusions

Multiple occlusion planes are handled by weighing the individual actions by the probability of the target's escape through the particular occlusion plane,

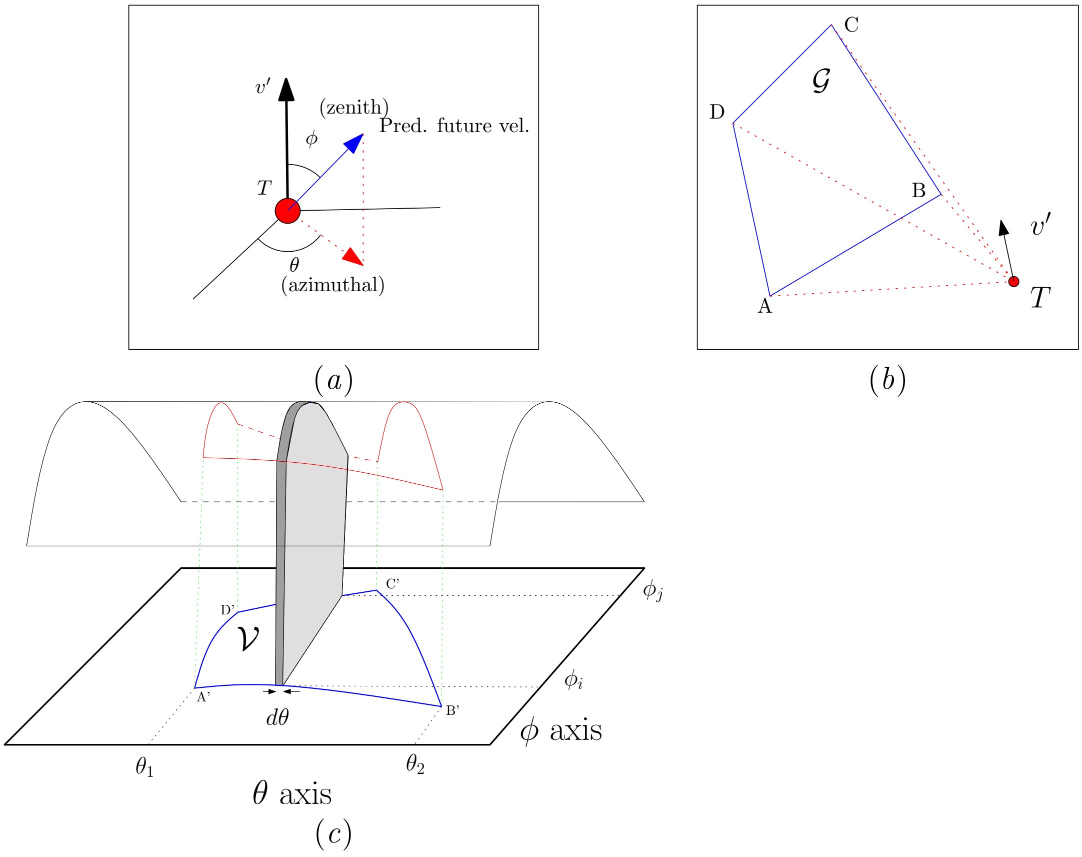
$$\nabla\Phi = \sum_i p_i\left(\frac{\partial t_{r.v}}{\partial v_r}\hat{\mathbf{r}} + \frac{\partial t_{r.v}}{\partial v_n}\hat{\mathbf{n}} + \frac{\partial t_{r.v}}{\partial v_p}\hat{\mathbf{p}}\right).$$

In the following section we show how the prediction of the target's escape is made for each escape plane.

5.4 Prediction

For finding the probability of the target's escape through any particular occlusion p_i, we predict the motion of the target by independent distributions $p(\theta)$ and $p(\phi)$ on its azimuth (θ) and zenith (ϕ) angle. We assume that the target speed remains constant, but in general, this can also be modeled by a distribution $p(s)$. The probability is measured in terms of the integral of the volume subtended by the plane,

$$p_i(\phi, \theta) = \int_{\theta_1}^{\theta_2} \int_{\phi_1}^{\phi_2} p(\phi) p(\theta) d\phi d\theta$$



(a)

(b)

(c)

Figure 5.11. (a) Spherical coordinates for the target velocity (b) Solid angle subtended by the occlusion plane $ABCD$ on the target (c) Escape Probability is the volume under the surface

For a locally predictable target, it is highly unlikely that the target would make abrupt changes in its velocity pattern in the immediate future. We model this by any distribution that places a higher likelihood of choosing a velocity closer to the current velocity. This gives us a probability distribution $p(\phi)$ on the zenith angle (ϕ) in the target spherical coordinates, shown in Figure 5.11*a*. Similarly the probability of the target's choice of any azimuthal angle (θ) is modeled as $p(\theta)$. We could also have a distribution over the choice of the speed of the target, but in the current analysis we assume that the target would not change its speed. The joint distribution for the predicted velocity direction is shown in figure.5.11*c*, where the $\mathcal{V}$ plane from Figure 5.11*b*) has been mapped to the $(\phi - \theta)$ plane. Linear extrapolation of the target's predicted motion in all possible directions gives us a measure of the target's probability of escape through all the occlusion surfaces.

The probability of the target's escape through any occlusion plane $\mathcal{G}$, is then the normalized volume under the probability distribution surface subtended by $\mathcal{G}$ at the target's current position.

$$p_i(\phi,\theta) = \int_{\theta_1}^{\theta_2}\int_{\phi_1}^{\phi_2} p(\phi)p(\theta)d\phi d\theta \tag{5.11}$$

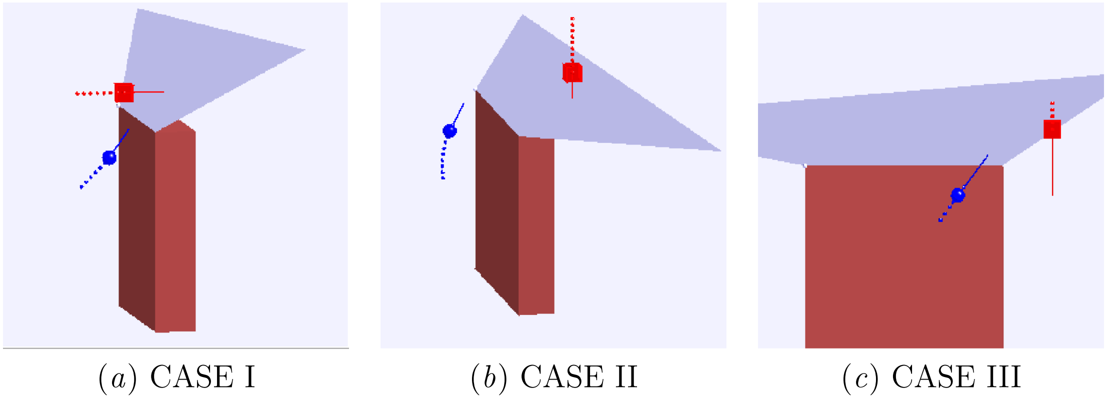
As an example, when $p(\phi)$ is a Gaussian distribution and $p(\theta)$ is uniform,the probability is calculated by the formula,

$$\begin{aligned} p_i &= \int_{\theta_1}^{\theta_2}(erf(\phi_j) - erf(\phi_i))d\theta \qquad (5.12)\\ \text{Where,}\quad erf(\phi) &= \frac{2}{\sqrt{\pi}}\int_0^{\phi} e^{-t^2}dt \end{aligned}$$

5.5 Experiments

We performed different tests on the algorithm by providing it with various environments and analyzing the results. In the first test, we check for specific behaviors of the algorithm with different configurations of the target in a simple environment, Figure 5.12. In the second test, we run the algorithm in a more complex urban environment amid sensor uncertainties, Figure 5.13, and analyzed its performance.

5.5.1 Qualitative Analysis : Single occlusion plane


(*a*) CASE I

(*b*) CASE II

(*c*) CASE III

Figure 5.12. Control Experiments to analyze the behavior of a single occlusion plane.

We create a simple scenario in Figure 5.12, where the target (red cube) tries to escape the visibility of the tracker (blue sphere), by moving behind the obstacle (maroon wall). To analyze the fundamental characteristics of the tracker motion in response to the target's motion against an occlusion plane, we turn off all the occlusion planes except the light blue plane at the top of the wall. The dotted lines depict the previous path executed by the target and the tracker, while the solid segments show their current heading. For all the cases, the tracker is placed in front of the wall. The target's path is unknown to the tracker. The tracker's velocity is generated at each step by Equation 5.3.

CASE I : When the target is placed in front of the wall, shown in Figure 5.12*a*, its shortest path of escape from the tracker's visibility passes through the top edge of the wall. This means that any amount of swinging of the occlusion plane by the tracker would be fruitless and the tracker should move towards this edge. This behavior is reproduced by the tracker, as v_r is the only component produced by Equation 5.3.

CASE II : Next, let the target be placed above the wall somewhere middle along its horizontal length, shown in Figure 5.12*b*. For such a position, the target's closest point of escape is its normal projection on the occlusion plane. In such a situation then, it makes sense to swing the occlusion plane away from the target. The algorithm manages to produce a combination of v_n and v_r to address the scenario. v_n helps in swinging the plane, and v_r helps in improving the vantage by moving closer to $\mathcal{B}_e$. This combination, that balances the long term and short term goals, generates a curved path as seen in the figure.

CASE III : If the target is placed not at the middle, but towards one end over the wall, a lateral swing can increase the shortest distance value in addition to the normal swinging motion. This is characteristic to 3-D environments where the tracker can prevent the target's escape by shifting the plane laterally away from under the target. In general, humans tend to show this kind of behavior in such a situation. This behavior is also shown by the algorithm as it generates a horizontal component v_p, in addition to v_n and v_p. This shows that the algorithm is able to exploit the additional dimension available to the tracker in 3-D. Moreover, this is achieved using only local information.

5.5.2 Realistic simulation

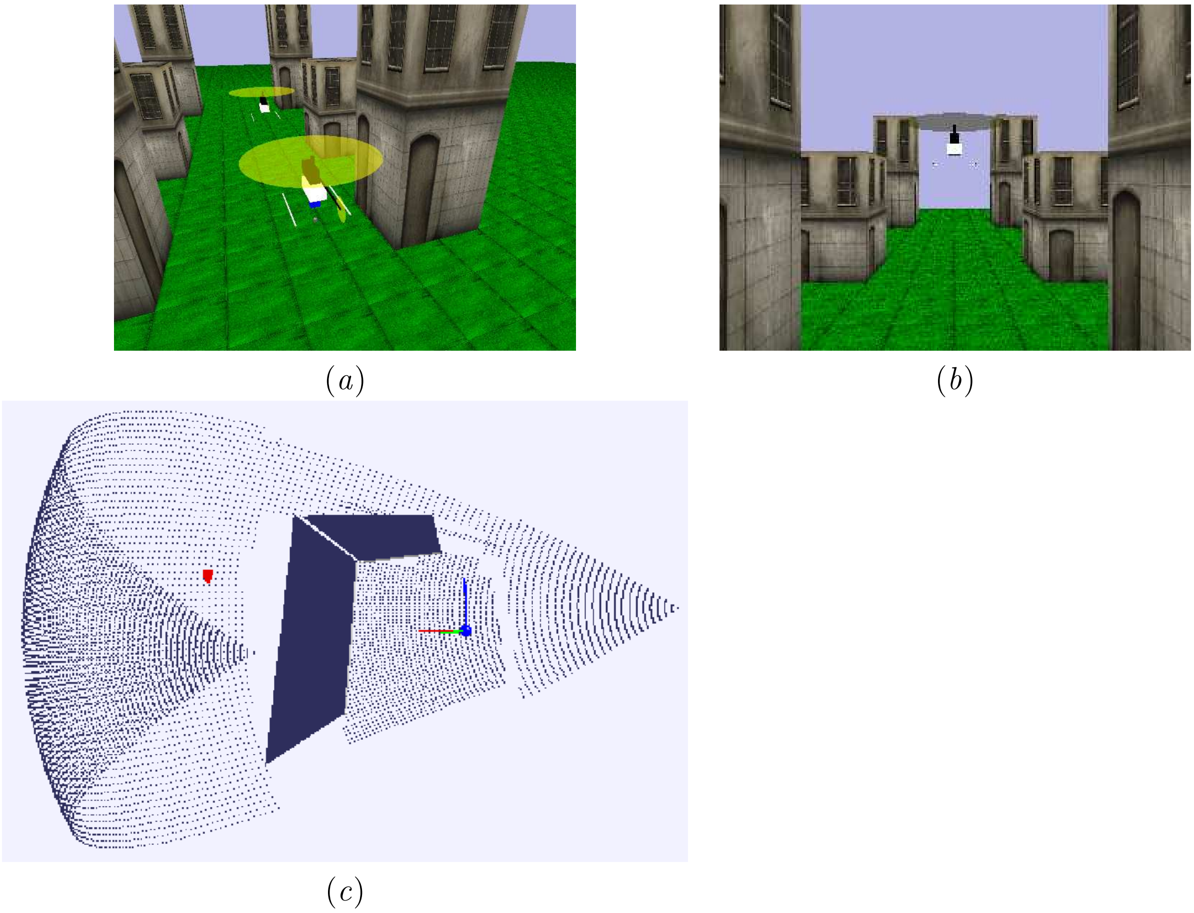


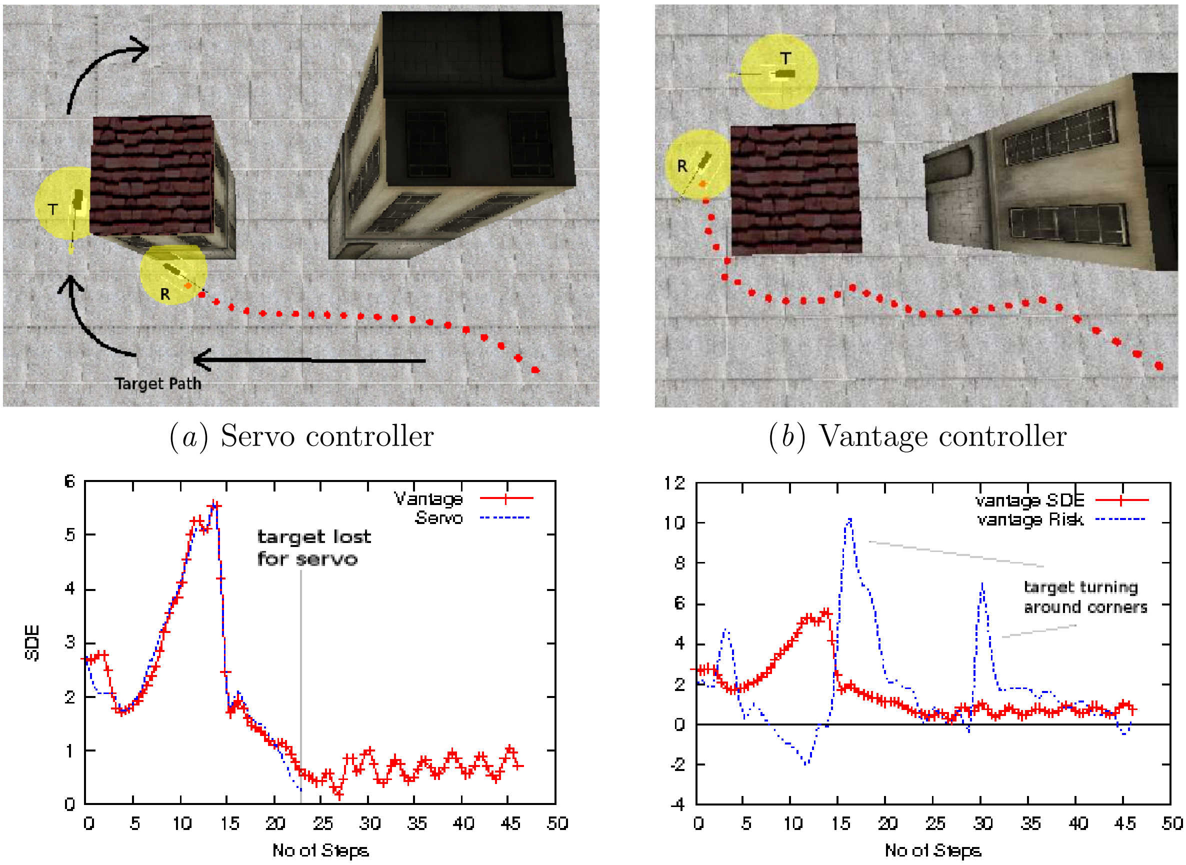
Figure 5.13. Realistic simulation setup using Gazebo. (a) Environment setup, (b) Robot viewpoint, (c) Extracting $\mathcal{G}$ from 3-D range scan

We next test the effectiveness of our algorithm in a realistic scenario by implementing it in the Gazebo tracker simulator [102]. Gazebo is a multi-tracker simulator for both indoor and outdoor environments in 3-D. It generates realistic sensor feedback, object collision, and dynamics. Using Gazebo, we build an urban environment in which a tracking robot helicopter tracks our target, another helicopter, shown in Figure 5.13. The environment has buildings of various sizes, separated by alleys and pathways. We mount a 3-D sweeping laser range finder on the tracker helicopter. In general, this can be replaced by any reliable vision system without much impact on our algorithm. The 3-D range data is processed to extract occlusion planes. Figure 5.13*c* shows an example. The set of points is the sensor data from the laser range finder. The dark blue planes are the occlusion planes obtained by the range discontinuity upon thresholding. The red dot indicates the target position.

Comparing with visual servo We compare our algorithm with the popular visual servo algorithm. To evaluate the performance, we compare the shortest distance to escape (SDE) from the target position to the nearest occlusion plane. Clearly, if an algorithm alway maintains superior SDE throughout, thereby keeping the target away from the possible escape regions, it can be considered to have a better tracking performance.

Figures 5.14*a* & 5.14*b*, show the tracking results for the servo algorithm and our vantage algorithm, respectively. The target executes an identical path, which is marked in Figure 5.14*a*. The tracker starts from the same position in the lower right part of the figures. The dotted paths show the tracker's paths under the control of the two algorithms.

The servo tracker loses the target at step 23, whereas the vantage tracker continues until we stop the simulation at step 46, at which time the target is still visible. The



(*a*) Servo controller

(*b*) Vantage controller

(*c*) SDE plots for Servo vs. Vantage

(*d*) Risk and SDE plot of Vantage

Figure 5.14. Experimental Results

SDE plots, in Figure 5.14*c*, show that the two trackers have comparable performance until around step 20. After that, the SDE for the servo tracker drops to 0, while the vantage tracker still maintains good SDE values and is able to continue tracking the target. The reason behind the success of the vantage tracker becomes clearer when we look at Figure 5.14*d*, which plots the risk values computed by the vantage tracker at each step. We see that the risk values peak for certain steps. Careful inspection reveals that such peaks occur whenever the target turns around a corner. For example, the target turns right sharply in steps 15–20, then again around step

30, and we have peaks in the risk plots accordingly. From the peaked risk values, the vantage tracker perceives the danger of losing the target in the near future and moves to reduce it, thereby successfully keeping the target visible. The servo tracker does not consider the effect of occlusion by obstacles and loses the target.

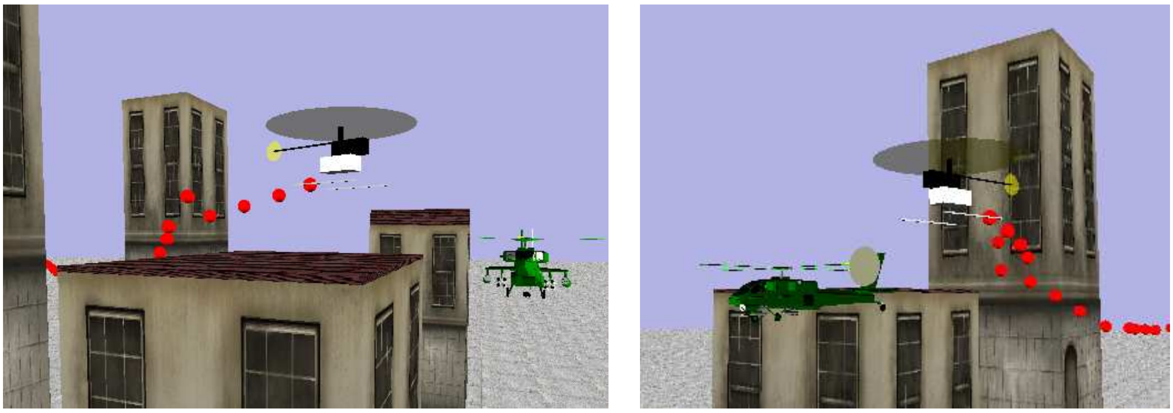


Figure 5.15. Characteristic motion of Vantage tracker in 3-D. Two views of the tracker's climbing action.

3-D climbing behavior In Figure 5.15, we show an example where the vantage tracker exploits the additional dimensionality in 3-D to its advantage. The target path is similar to that shown previous example, the only difference is now the height of the target path is similar to that of the shorter building (one on the left). The red balls show the path of the tracker. This triggers the lateral motion as in Figure 5.12*c*. As the target turns around the corner of the building, the tracker rises vertically and hovers over the top of the building. This gives it an advantage of being able to guard any side of the building against the target with equal ease. To the authors' best knowledge, the emergence of such a behavior achieved using only local information has not been shown before. Additional snapshots of the video are shown in Figure 5.16.

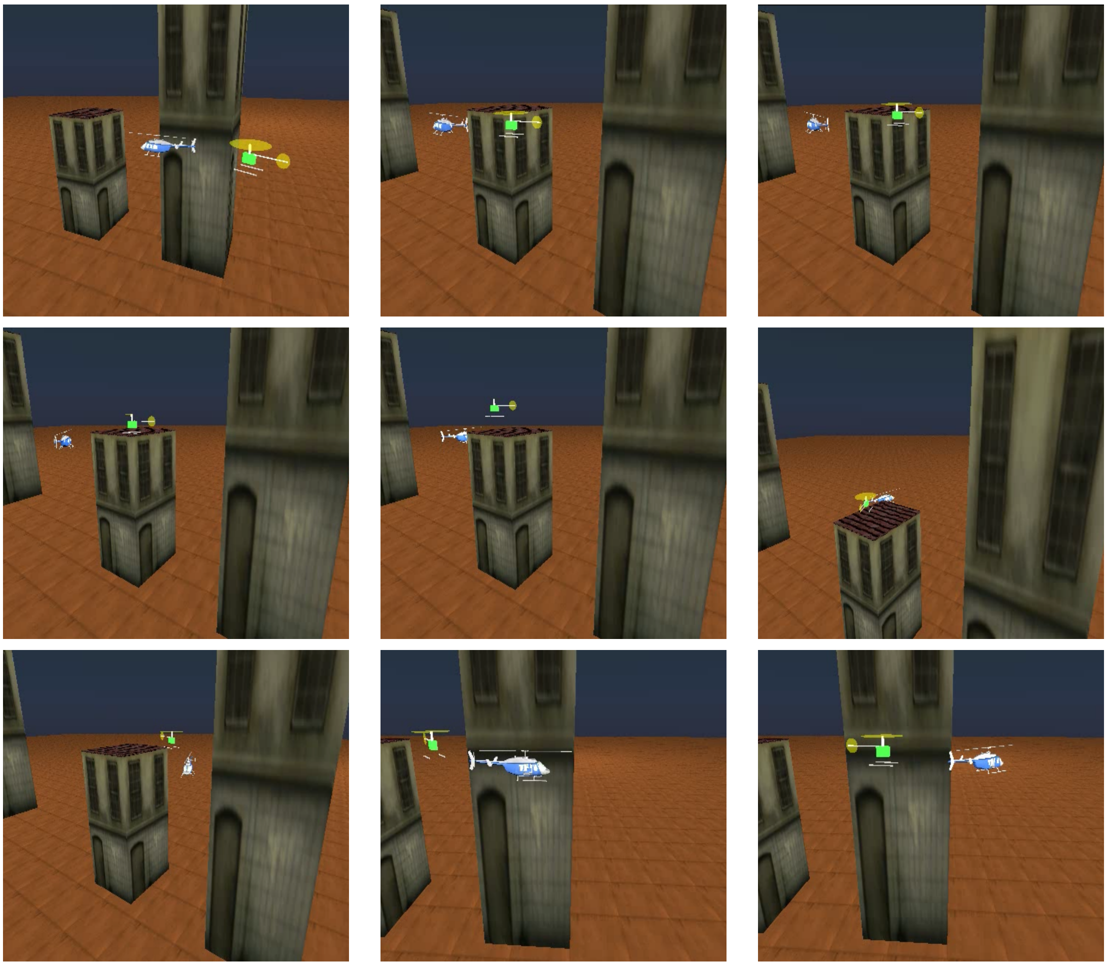


Figure 5.16. A set of snapshots showing the tracker's climbing behavior. The green tracker helicopter tracks the blue target helicopter using on board 3-D range sensor. We can see the tracker climb the shorter building and keep track of the target which is making a turn around the building. Again, when the target turns around the taller building, the tracker starts following it. (Video-id: Vantage3D-climb)

5.6 Summary

In this chapter, we propose a 3-D tracker that can keep in view an unknown target in an unknown environment. The risk function is based on the concept of relative vantage formulated in the earlier chapter 3. 3-D tracking proposed additional complexity due to an extra degree of motion available to both the target and the tracker. We formulate analytical forms for the risk function that can be minimized to compute the local optimal motion. Simulation results show that such a formulation is able to exploit the additional dimension to keep the target in view more effectively compared to visual servoing.

CHAPTER 6

CONCLUSION

This thesis focuses on the problem of generating motion strategies for tracking a moving target in an unknown and dynamic environment for both 2-D and 3-D. The tracker plans using only local information and has to take into account sensing, mobility and operational limitations. A general tracking framework is provided which integrates these limitations into the tracking problem as planning constraints for a local greedy online tracking algorithm that maximizes the relative vantage of the tracker with respect to the target in the locally sensed environment.

6.1 Contributions

In order to track in an unknown and dynamic environment an online local approach has been taken. On board sensors compute the local visibility and based on the escape gaps in the visibility a risk function is proposed that encodes the danger of losing the target through these escape gaps. Focusing only on local information instead of trying to build a global map keeps the tracking algorithm tractable in a complex or cluttered environment like crowded places. Moreover, such a tracker does not care about the boundedness of the environment or about loops in the environment.

In 3-D such an advantage is significant. As an example, a tool tip tracking using visual cameras during surgery would not require complete modeling of the pulsating organs which periodically might occlude the line of sight.

While the local planning approach evades the complexity of global planning, it performs better than the purely reactive approaches like visual servoing which does not incorporate the environment information. The intelligence in tracking comes from the risk function that includes the information about the environment and the relative position of the target and the tracker in it. By analyzing relative positioning of the target and the tracker in the environment, a relative vantage based risk formulation is proposed. Optimizing such a risk function allows the tracker to move itself towards a strategic location from where the escape gaps can be kept away from the target in an effective manner. Such a vantage tracker performs better that other risk based approaches that maximizes the shortest distance to escape for the target. The advantage of the vantage tracker which exploits the local information is that it is able to successfully balance the requirement of keeping the target in view for the immediate step while preventing the chances of visual occlusions in the future. In this way both short term and long term goals are achieved.

For both 2-D and 3-D we propose analytical formulations for risk function. This leads to a fast computation of the optimal motion to be taken and the algorithm can be run at a high frequency. The high frequency of sampling and planning makes the robot robust to dynamic obstructions and changes in the environment and allows rapid recovery from unexpected scenarios. As an example for the latter, the tracker can instantly modify its tracking behavior when a door is opened in a closed room and additional escape routes are exposed. Approaches that build a global map while tracking will have a lag before this information is propagated into the map and actions modified accordingly.

The thesis presents a general tracking framework where hardware and operational limitations can be incorporated into such a local planning approach. Such a framework makes implementing the tracker on a real robot possible. Limitations on sensing are incorporated into the visibility, while the reachable regions are limited by the mobility constraints. Additional mission requirements can be incorporated in a same way into the tracking problem. As an example, a stealth tracker is proposed. The stealth requirement is formulated into a stealth planning constraint by exploiting the target's estimated visibility in the environment. It is shown that the tracking behavior changes when this stealth constraint is added. An advantage of such an approach for an online local tracking approach is that such constraints can be added or removed at runtime. A higher AI loop or a human operator could add or remove operational requirements of stealth or human avoidance for different targets or environment.

For the 2-D formulation, this framework is utilized to build a tracking robot using only an on board laser sensor on a standard differential drive robot. The tracker was tested in crowded environments in the school cafeteria during lunch time. Crowds may occlude a significant portion of the environment and a robot that depends on the global information might have difficulty in localizing itself. Modeling the crowd behavior in a dynamic manner is extremely difficult using only the on board sensors of the tracker. The local information based tracking approach avoids this problem. The fast online re-planning helps the robotic tracker to recover from temporary occlusions. Moreover, the uncertainty in sensing and motion was bounded as the local information was extracted at each step and re-planning of the motion done making the tracking more robust to accumulation of errors. The tracking robot was able to successfully follow a person in the crowded cafeteria. Such a system can be easily upgraded into a prototype robotic personal porter for use in airports, railway stations or shopping malls.

In a 3-D environment, the visibility relationships are complex and the current tracking techniques are mostly based on visual servo approach. This thesis presents an intelligent vantage tracker which exploits the local information and computes a tracking motion in an online fashion. A relative vantage based risk generates intelligent tracking actions while keeping the computation load similar to that of visual servo. As an example, in simulation a robotic helicopter utilizes a vertical motion to avoid occlusion of the target due to the buildings in an urban scenario when advantageous. Such a behavior is generated based only on the locally sensed geometric parameters and no a-priori knowledge of the layout or the model of the obstacles in the environment is used. Another thing to note is generating such behaviors for environments with complex and cluttered generalized polygons still keeps the computation tractable.

6.2 Limitations

The target tracking approach proposed has limitations of a local approach. Since the planning is done in a local online fashion, the actions are not guaranteed to provide globally optimal motion paths. Without a global map, the tracker is not able to exploit environmental pathways that would ensure the maximizing of the total time for keeping the target in view. For example, the local optimization would not favor motion strategies for losing the target for a short duration eventhough it might improve the tracking significantly in the future. Effective motion algorithms to search and regain the target cannot be utilized due to the lack of a map. In addition due to the usage of limited history the tracker may get trapped unnecessarily into a series of oscillating maneuvers as discussed in Section 4.4.

There are situations, where continuous tracking is not desirable. E.g, in the monitoring of an elderly person, some privacy is necessary when the target goes to the washroom. The proposed approach to tracking cannot handle monitoring the target's location without keeping the target in sight. For such situations, the searching and tracking problem has to be combined by the target's location uncertainty which can then be tracked [56].

Since the algorithm does not keep a memory of the environment (does not build a map), it might generate transient occlusion gaps which physically lead to a dead end, e.g when the visibility rays are at grazing angle to an obstacle. This occlusion gap disappears when the incident angle decreases, and re-appears when angle increases again. Such spurious occlusion gaps can create wavy motion while tracking. This is a disadvantage of a limited temporal local information model.

6.3 Future Work

Incorporating Uncertainty Although the tracking framework proposed is general, the focus of this thesis has been on deterministic analysis of the actions from a given visibility polygon. The uncertainty in sensing and motion has not been incorporated explicitly into the formulation. A significant improvement of the tracking performance can be made by developing and incorporating probabilistic models for potential target features to identify clutter and filter them before generating the visibility polygon.

Multiple robots Multiple robot based vantage tracking is an interesting extension. A single tracker is bound to fail in certain cases. Additional robots can potentially increase the time the target is kept under surveillance. However, this increases the complexity of the problem as now the individual sensor information have to be fused in an intelligent manner to extract local geometrical feature. Also, the control of individual tracker quickly increase the dimensionality of the planning problem. This makes the problem challenging.

Computer vision based target disambiguation The major drawback of the tracking system is the assumption of reliable target detection, which is difficult in real environments. The tracking strategy assumes the target is visible and initialized in the beginning. It also does not focus too much on recovering the target, once it is lost for a long duration. Such a limitation can be addressed by having a robust target detection algorithm that can detect and disambiguate the target from the background. A vision based system can be integrated with the range data to make the target detection and recovery more robust. Combining the computer vision with laser recognition would lead to improved target identification and hence improved target tracking capabilities.

Stealth tracker in hardware Implementation of the stealth tracker presented on real hardware would be an interesting extension of this work. However, several significant issues must be investigated. For one, the identification of the target from partial occlusions as well as from an analysis of the *shadow regions* (regions not currently in view) would be an important component. Moreover, the physical structure of the tracking robot must also be incorporated into the planning aspect to determine the stealth regions it could physically be accommodated in.

3-D tracking in hardware The motion model for the 3-D tracker is a free flying holonomic model. In real applications, like gliders or helicopters, the kinematics and dynamics of the robot have to be taken into account. It would be interesting to integrate the non-holonomic motion models of such robots and see how the tracking performance is affected. The execution of such a strategy might reveal new 3-D maneuvers.

Using global information effectively Extending the concept of relative vantage beyond the local visibility, when the map of the environment is known is an interesting problem. If the criterion is to maximize the total time for which the target is kept in view, it may be in the tracker's interest to let the target move out of sight for a short duration while moving to a strategic location that significantly improves future tracking. This in conjunction with a multi-robot risk formulation can create a robust indoor surveillance system.

APPENDIX A

PUBLICATIONS

- **T. Bandyopadhyay**, N. Rong, M. Ang, D. Hsu, W. S. Lee. *Motion Planning for People Tracking in Uncertain and Dynamic Environments.* Workshop on People Detection and Tracking, ICRA-2009, ICRA-2010 (invited).

- **T. Bandyopadhyay**, D. Hsu. and Ang Jr. M.H. *Motion Strategies for People Tracking in Cluttered and Dynamic Environments.* Int. Symp. on Expt. Robotics, ISER-2008.

- **T. Bandyopadhyay**, Ang Jr., M.H., and D. Hsu. *Motion planning for 3-D target tracking among obstacles.* In Proc. Int. Symp. on Robotics Research, 2007.

- **T. Bandyopadhyay**, Y.P. Li, Ang Jr. M.H., and D. Hsu. *A greedy strategy for tracking a locally predictable target among obstacles.* In Proc. IEEE Int. Conf. on Robotics & Automation, pp. 2342-2347, 2006.

- **T. Bandyopadhyay**, Y.P. Li, Ang Jr. M.H., and D. Hsu. *Stealth tracking of an unpredictable target among obstacles.* In M. Erdmann and others, editors, Algorithmic Foundations of Robotics VI, pp. 43-58, Springer-Verlag, 2004.

BIBLIOGRAPHY

[1] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.

[2] S.A. Hutchinson, G.D. Hager, and P.I. Corke, "A tutorial on visual servo control," *IEEE Trans. Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996.

[3] Héctor H. González-Baños, Cheng-Yu Lee, and Jean-Claude Latombe, "Real-time combinatorial tracking of a target moving unpredictably among obstacles.," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002, May 11-15, 2002, Washington, DC, USA*, 2002, pp. 1683–1690.

[4] Samuel S. Blackman, *Multiple Target Tracking with Radar Applications*, Artech House, Norwood, MA, 1986.

[5] Yaakov Bar-Shalom, Ed., *Multitarget-Multisensor Tracking: Advanced Applications*, Artech House, 1990.

[6] Lawrence D. Stone, Carl A. Barlow, and Thomas L. Corwin, *Bayesian Multiple Target Tracking*, Artech House, 1999.

[7] A. Fod, A. Howard, and M.A.J. Mataric, "A laser-based people tracker," in *Proc. IEEE International Conference on Robotics and Automation ICRA '02*, 2002, vol. 3, pp. 3024–3029.

[8] Fedrik Gustafsson, Fedrik Gunnarsson, Kiclas Bergman, Urban Forssell, Jonas Jansson, Rickard Karlsson, and Per-Johan Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Trans on Signal Processing*, vol. 50(2), pp. 425–437, February 2002.

[9] D. Schulz, W. Burgard, D. Fox, and A. Cremers, "Tracking multiple moving targets with a mobile robot using particle filters and statistical data association," in *In Proc. International Conference on Robotics and Automation*, 2001.

[10] Michael Isard and Andrew Blake, “Condensation – conditional density propagation for visual tracking,” *Int. J. Computer Vision*, vol. 29-1, pp. 5–28, 1998.

[11] David Liu and Li Chen Fu, “Target tracking in an environment of nearly stationary and biased clutter,” in *In Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, Hawaii,, October 2001. 2001, pp. 1358 – 1363.

[12] Yaakov Bar-Shalom and Thomas E. Fortmann, *Tracking and Data Association.*, Academic Press, Inc., Orlando, Florida, 1988., 1988.

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

[14] Donald B. Reid., “An algorithm for tracking multiple targets.,” *IEEE Transactions on Automatic Control,*, vol. 24(6), pp. 843 – 854,, December 1979.

[15] Ingemar J. Cox and Sunita L. Hingorani., “An efficient implementation of reid’s multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18(2), pp. 138 – 150,, February 1996.

[16] R. Danchick and G. E. Newnam, “A fast method for finding the exact n-best hypotheses for multitarget tracking.,” *IEEE Transactions on Aerospace and Electronic Systems,*, vol. 29(2), pp. 555 – 560, April 1993.

[17] Kai O. Arras, Slawomir Grzonka, Matthias Luber, and Wolfram Burgard, “Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities,” in *In Proc. IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA,, May 19-23 2008.

[18] Alireza Behrad, Ali Shahrokni, and Seyed Ahmad Motamedi, “A robust vision-based moving target detection and tracking system.,” in *In the Proceeding of Image and Vision Computing Conference,*, University of Otago, Dunedin, New Zealand,, November 2001.

[19] J. Segen, “A camera-based system for tracking people in real time,” in *Proc. 13th International Conference on Pattern Recognition*, 1996, vol. 3, pp. 63–67 vol.3.

[20] Gian Luca Foresti and C. Micheloni., “A robust feature tracker for active surveillance of outdoor scenes.,” in *Electronic Letters on Computer Vision and Image Analysis,*, 2003, vol. 1, pp. 21 – 34,.

[21] Sohaib Khan and Mubarak Shah, "Tracking people in presence of occlusion," in *In Proc, Asian Conference on Computer Vision, Taipei, Taiwan,*, Jan 2000.

[22] Shigeyuki Sakane Ken Ito, "Robust view-based visual tracking with detection of occlusions," in *Proc. IEEE. Int. Conf. on Robotics & Automation*, 2001.

[23] R. Cucchiara, C. Grana, G. Tardini, and R. Vezzani, "Probabilistic people tracking for occlusion handling," in *Proc. 17th International Conference on Pattern Recognition ICPR 2004*, 2004, vol. 1, pp. 132–135 Vol.1.

[24] Jinshi Cui, Hongbin Zha, Huijing Zhao, and R. Shibasaki, "Robust tracking of multiple people in crowds using laser range scanners," in *Proc. 18th International Conference on Pattern Recognition ICPR 2006*, 2006, vol. 4, pp. 857–860.

[25] 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*, Orlando, FL, 15-19 May 2006 2006, pp. 557 – 562.

[26] M. Montemerlo, W. Whittaker, and S. Thrun, "Conditional particle filters for simultaneous mobile robot localization and people-tracking," in *IEEE International Conference on Robotics and Automation (ICRA)*, Washington, DC, 2002, ICRA.

[27] Jae Hoon Lee, T. Tsubouchi, K. Yamamoto, and S. Egawa, "People tracking using a robot in motion with laser range finder," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Beijing, 9-15 Oct. 2006 2006, pp. 2936–2942.

[28] S. LaValle, H. H. González-Baños, C. Becker, and J. Latombe, "Motion strategies for maintaining visibility of a moving target," in *In Proc. IEEE .Int'l Conf. on Robotics and Automation*, 1997, pp. 731–736.

[29] Alon Efrat, Hector H. Gonzalez-Banos, Stephen G. Kobourov, and Lingeshwaran Palaniappan, "Optimal strategies to track and capture a predictable target," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2003, pp. 3789–3796.

[30] T.D. Parsons, "Pursuit-evasion in a graph," *Theory and Applications of Graphs*, pp. 426–441, 1976.

[31] Masfumi Yamashita, Hideki Umemoto, Ichiro Suzuki, and Tsunehiko Kameda, "Searching for mobile intruders in a polygonal region by a group of mobile searchers," in *Symposium on Computational Geometry*, 1997, pp. 448–450.

[32] Leonidas J. Guibas, Jean-Claude Latombe, Steven M. LaValle, David Lin, and Rajeev Motwani, "A visibility-based pursuit-evasion problem," *Int. J. of Computational Geometry and Applications*, vol. 9, no. 4/5, pp. 471–, 1999.

[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 *In Proceedings IEEE International Conference on Robotics and Automation,*, 1997, pp. 737–742.

[34] J. Yu and S. M. LaValle., "Tracking hidden agents through shadow information spaces.," in *In Proceedings IEEE International Conference on Robotics and Automation,*, 2008.

[35] S. M. LaValle and J. Hinrichsen, "Visibility-based pursuit-evasion: The case of curved environments," *IEEE Transactions on Robotics and Automation,*, vol. 17(2), pp. 196–201,, April 2001.

[36] S. Lazebnik, "Visibility-based pursuit-evasion in three-dimensional environments," Tech. Rep., UIUC, 2001.

[37] Swastik Kopparty and Chinya V. Ravishankar, "A framework for pursuit evasion games in rn.," in *Inf. Process. Lett.*, 2005., vol. 96, pp. 114–122.

[38] S. Alexander, R. Bishop, and R. Ghrist., "Pursuit and evasion in non-convex domains of arbitrary dimensions.," in *In Proceedings of Robotics: Science and Systems,*, Philadelphia, USA,, August 2006.

[39] I. Suzuki and M. Yamashita, "Searching for a mobile intruder in a polygonal region," *SIAM Journal on Computing*, vol. 21, no. 5, pp. 863–888, October 1992.

[40] Brian P. Gerkey, Sebastian Thrun, and Geoff Gordon, "Visibility-based pursuit-evasion with limited field of view," *Intl. Journal of Robotics Research*, vol. 25, no. 4, pp. 299–316, Apr 2006.

[41] B. Tovar and S. M. LaValle., "Visibilty-based pursuit-evasion with bounded speed.," in *In Proceedings Workshop on Algorithmic Foundations of Robotics,*, 2006.

[42] T. Muppirala, S. Hutchinson, and R. Murrieta-Cid, "Optimal motion strategies based on critical events to maintain visibility of a moving target," in *Proceedings of the IEEE International Conference on Robotics and Automation, 2005. ICRA 2005.*, Barcelona, Spain, 18-22 April 2005 2005, pp. 3826 – 3831.

[43] Rafael Murrieta-Cid, A. Sarmiento, S. Bhattacharya, and S. Hutchinson., "Maintaining visibility of a moving target at a fixed distance: The case of

observer bounded speed," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation,New Orleans, USA.*, 2004, pp. 479–484.

[44] Rafael Murrieta-Cid, A. Sarmiento, and S. Hutchinson., "On the existence of a strategy to maintain a moving target within the sensing range of an observer reacting with delay," in *Proc IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2003,Las Vegas, USA.*, 2003, pp. 1184–1191.

[45] R. Murrieta, A. Sarmiento, and S.A. Hutchinson, "A motion planning strategy to maintain visibility of a moving target at a fixed distance in a polygon," in *IEEE Int. Conf. on Robotics & Automation*, 2003.

[46] Sourabh Bhattacharya, Salvatore Candido, and Seth Hutchinson, "Motion strategies for surveillance," in *Robotics : Science and Systems III, MIT press*, 2007, pp. 249–256.

[47] Rafael Murrieta-Cid, Héctor H. González-Baños, and Benjamín Tovar, "A reactive motion planner to maintain visibility of unpredictable targets.," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002, May 11-15, 2002, Washington, DC, USA*, 2002, pp. 4242–4248.

[48] R. Murrieta-Cid, B. Tovar, and S. Hutchinson, "A sampling based motion planning approach to maintain visibility of unpredictable moving targets," *Journal on Autonomous Robots*, vol. 19, no. 3, pp. 285–300, December 2005.

[49] R. Murrieta-Cid, L. Munoz-Gomez, M. Alencastre-Miranda, A. Sarmiento, S. Kloder, S. Hutchinson, F. Lamiraux, and J.P. Laumond, "Maintaining visibility of a moving holonomic target at a fixed distance with a non - holonomic robot," in *In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005. (IROS 2005).*, 2-6 Aug. 2005, pp. 2687 – 2693.

[50] J.-B. Hayet, C. Esteves, and R. Murrieta-Cid, "A motion planner for maintaining landmark visibility with a differential drive robot," in *Proc. WAFR 2008 to be published in Springer Tracts in Advanced Robotics*, 2009.

[51] S. Bhattacharya, R. Murrieta-Cid, and S. Hutchinson, "Optimal paths for landmark-based navigation by differential drive vehicles with field-of-view constraints," *IEEE Trans. on Robotics*, vol. 23, no. 1, pp. 47–59, Feb. 2007.

[52] B. Jung and G. S. Sukhatme, "Tracking targets using multiple robots: The effect of environment occlusion," *Autonomous Robots*, vol. 13, pp. 191–205, 2002.

[53] C. Becker, H.H. González-Baños, J.C. Latombe, and C. Tomasi, "An intelligent observer," in *Int. Symp. on Experimental Robotics*, July 1995, pp. 153–160.

[54] V. Isler, S. Kannan, and S. Khanna, “Randomized pursuit-evasion in a polygonal environment,” *IEEE Trans. on Robotics.*, vol. 21, no. 5, pp. 875–884, 2005.

[55] Volkan Isler, Sampath Kannan, and Sanjeev Khanna, “Randomized pursuit-evasion with local visibility,” *SIAM Journal on Discrete Mathematics*, vol. 1, pp. 26–41, 2006.

[56] D. Hsu, W.S. Lee, and N. Rong, “A point-based POMDP planner for target tracking,” in *Proc. IEEE Int. Conf. on Robotics & Automation*, 2008, pp. 2644–2650.

[57] S. Sachs, S. Rajko, and S. M. LaValle., “Visibility-based pursuit-evasion in an unknown planar environment.,” *International Journal of Robotics Research,*, vol. 23(1), pp. 3–26, January 2004.

[58] L. Guilamo, B. Tovar, and S. M. LaValle, “Pursuit-evasion in an unknown environment using gap navigation trees,” in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, 2004.

[59] B. Espiau, F. Chaumette, and P. Rives, “A new approach to visual servoing in robotics,” *Robotics and Automation, IEEE Transactions on*, vol. 8, no. 3, pp. 313 – 326, June 1992.

[60] N. Papanikolopoulos, Pradeep Khosla, and Takeo Kanade, “Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision,” *IEEE Trans. on Robotics and Automation*, vol. 9, no. 1, pp. 14–35, February 1993.

[61] F. Chaumette and S. Hutchinson, “Visual servo control, part i: Basic approaches,” *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, December 2006.

[62] F. Chaumette and S. Hutchinson, “Visual servo control, part ii: Advanced approaches,” *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 109–118, March 2007.

[63] C. Coue and P. Bessiere, “Chasing an elusive target with a mobile robot,” in *Proceedings. 2001 IEEE/RSJ International Conference on*, 2001, vol. 3, pp. 1370–1375.

[64] J.P. Hespanha, “Single-camera visual servoing,” in *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, 2000, vol. 3, pp. 2533 – 2538.

[65] Jason Rife and Stephen M. Rock, "Visual tracking of jellyfish in situ," in *Proc. of the 2001 International Conference on Image Processing*, Thessaloniki, Greece, October 2001, IEEE.

[66] C. Schlegel, J. Illmann, H. Jaberg, M. Schuster, and R. Worz, "Vision based person tracking with a mobile robot," in *In Proc. British Machine Vision Conference,*, 1998.

[67] Peter Nordlund and Tomas Uhlin., "Closing the loop: Detection and pursuit of a moving object by a moving observer.," *Image and Vision Computing,*, vol. 14, pp. 265 – 275, May 1996.

[68] Frank Hoeller, Dirk Schulz, Mark Moors, , and Frank E. Schneider, "Accompanying persons with a mobile robot using motion prediction and probabilistic roadmaps," in *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, Oct 29 - Nov 2 2007.

[69] Teck Chew Ng, J. Ibanez-Guzman, Jian Shen, Zhiming Gong, Han Wang, and Chen Cheng, "Vehicle following with obstacle avoidance capabilities in natural environments," in *In Proceedings IEEE International Conference Robotics and Automation,*, 26 April-1 May 2004 2004, vol. 5, pp. 4283– 4288.

[70] Hendrik Zender, Patric Jensfeltt, and Geert-Jan M. Kruijff, "Human- and situation-aware people following," in *In Proc. 16th IEEE International Conference on Robot & Human Interactive Communication*, Jeju, Korea, August 26 - 29 2007.

[71] J. Maver and R. Bajcsy, "Occlusions as a guide for planning the next view," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 5, pp. 417–433, 1993.

[72] Cheng.Yu. Lee, *Real-Time Target Tracking in an Indoor Envrionment*, Ph.D. thesis, Dept. of Aeronautics & Astronautics, Stanford University, Stanford, CA, USA, 2002.

[73] E. Birgersson, A. Howard, and G.S. Sukhatme, "Towards stealthy behaviors," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2003, pp. 1703–1708.

[74] M. Marzouqi and R. Jarvis, "Covert robotics: Covert path planning in unknown environments," in *Proc. Australian Conf. on Robotics & Automation*, 2003.

[75] M. Marzouqi and R.A. Jarvis, "Covert path planning in unknown environments with known or suspected sentry location," in *Intelligent Robots and Systems, International Conference on*, 2 - 6 Aug. 2005, pp. 1772 – 1778.

[76] Jung-Hee Park, Jeong-Sik Choi, Jimin Kim, and Beom-Hee Lee, “Roadmap-based stealth navigation for intercepting an invader,” in *Proc. IEEE International Conference on Robotics and Automation ICRA ’09*, 2009, pp. 442–447.

[77] P. Fabiani, H.H. González-Baños, J.C. Latombe, and D. Lin, “Tracking a partially predictable target with uncertainties and visibility constraints,” *J. Robotics & Autonomous Systems*, vol. 38, no. 1, pp. 31–48, 2002.

[78] A. Chella, H. Dindo, and I. Infantino, “A system for simultaneous people tracking and posture recognition in the context of human-computer interaction,” in *Proc. EUROCON 2005.The International Conference on Computer as a Tool*, 2005, vol. 2, pp. 991–994.

[79] H. Plantinga and C. Dyer, “Visibility, occlusion, and the aspect graph,” *Int. J. Computer Vision*, vol. 5, no. 2, pp. 137–160, 1990.

[80] Frédo Durand, George Drettakis, and Claude Puech, “The 3d visibility complex,” *ACM Trans. on Graphics*, vol. 21(2), pp. 176–206, April 2002.

[81] Luis O. Mejias, Srikanth Saripalli, Pascual Cervera, and Gaurav S. Sukhatme, “Visual servoing of an autonomous helicopter in urban areas using feature tracking,” *Journal of Field Robotics*, vol. 23, no. 3, pp. 185–199, 2006.

[82] R. Vidal, O. Shakernia, H.J. Kim, D.H. Shim, and S. Sastry, “Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation,” *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 662 – 669, Oct. 2002.

[83] Zhen Jia, A. Balasuriya, and S. Challa, “Sensor fusion based 3d target visual tracking for autonomous vehicles with imm,” in *Proc. IEEE International Conference on Robotics and Automation ICRA 2005*, 2005, pp. 1829–1834.

[84] Zlien Jia, A. Balasuriya, and S. Challa, “Visual 3d target tracking for autonomous vehicles,” in *Proc. IEEE Conference on Cybernetics and Intelligent Systems*, 2004, vol. 2, pp. 821–826.

[85] Khurram Shafiq Mubarak Shah. Fahd Rafi, Saad M. Khan, “Autonomous target following by unmanned aerial vehicles,” in *SPIE Defence and Security Symposium, Orlando FL.*, 2006.

[86] A. Ruangwiset, “Path generation for ground target tracking of airplane-typed uav,” in *Proc. IEEE International Conference on Robotics and Biomimetics ROBIO 2008*, 2009, pp. 1354–1358.

[87] H. Helble and S. Cameron, “3-d path planning and target trajectory prediction for the oxford aerial tracking system,” in *Proc. IEEE International Conference on Robotics and Automation*, 2007, pp. 1042–1048.

[88] Chengyu Cao and N. Hovakimyan, “Vision-based aerial tracking using intelligent excitation,” in *Proc. American Control Conference the 2005*, 2005, pp. 5091–5096 vol. 7.

[89] Gai Ming-jiu, Yi Xiao, He You, and Shi Bao, “An approach to tracking a 3d-target with 2d-radar,” in *Proc. IEEE International Radar Conference*, 2005, pp. 763–768.

[90] Srikanth Saripalli, James F. Montgomery, and Gaurav S. Sukhatme, “Visually-guided landing of an unmanned aerial vehicle,” *IEEE Transactions on Robotics and Automation*, vol. 19, no. 3, pp. 371–381, Jun 2003.

[91] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications.*, vol. 2nd Edition, Springer, Berlin, 2000.

[92] Q. Zhu, “Hidden markov model for dynamic obstacle avoidance of mobile robot navigation,” *IEEE Trans. on Robotics and Automation,*, vol. 7, no. 3, pp. 390–397, 1991.

[93] N.H.C. Yung and Cang Ye, “An intelligent mobile vehicle navigator based on fuzzy logic and reinforcement learning,” *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 29, no. 2, pp. 314–321, 1999.

[94] C.C. Chang and K.-T. Song, “Dynamic motion planning based on real-time obstacle prediction,” *Proceedings on IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2402 – 2407, 1996.

[95] M. Bennewitz, W. Burgard, and S. Thrun, “Using EM to learn motion behaviors of persons with mobile robots,” in *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, 2002.

[96] B. Gerkey, S. Thrun, and G. Gordon, “Visibility-based pursuit-evasion with limited field of view,” in *Proceedings of the AAAI National Conference on Artificial Intelligence*, San Jose, CA, 2004.

[97] T.H. Collett, B.A. MacDonald, and B.P Gerkey, “Player 2.0: Toward a practical robot programming framework.,” in *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005).*, 2005.

[98] Y. Ansel Teng, Daniel DeMenthon, and Larry S. Davis, "Stealth terrain navigation," *IEEE Trans on Systems, Man and Cybernetics*, vol. 23, no. 1, pp. 96–109, Jan/Feb 1993.

[99] H. ElGindy and D. Avis, "A linear algorithm for computing the visibility of polygon from a point," *J. Algorithms*, vol. 2, pp. 186–197, 1981.

[100] D.T. Lee, "Visibility of a simple polygon," *Computer Vision, Graphics, & Image Processing*, vol. 22, pp. 207–221, 1983.

[101] B. Joe and R.B. Simposon, "Corrections to Lee's visibility polygon algorithm," *BIT*, vol. 27, pp. 458–473, 1987.

[102] *http://playerstage.sourceforge.net*.