Visibility Maximization with Unmanned Aerial Vehicles in Complex Environments

by

Kenneth Lee

Bachelor of Applied Science (BASc), Mechanical Engineering
University of Waterloo (2008)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2010

Author .....
Department of Aeronautics and Astronautics
August 19, 2010

Certified by .....
Jonathan P. How
Richard C. Maclaurin Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Eytan H. Modiano
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Visibility Maximization with Unmanned Aerial Vehicles in Complex Environments

by

Kenneth Lee

Submitted to the Department of Aeronautics and Astronautics
on August 19, 2010, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Unmanned aerial vehicles are used extensively in persistent surveillance, search and track, border patrol, and environment monitoring applications. Each of these applications requires the obtainment of information using a dynamic observer equipped with a constrained sensor. Information can only be gained when visibility exists between the sensor and a number of targets in a cluttered environment. Maximizing visibility is therefore essential for acquiring as much information about targets as possible, to subsequently enable informed decision making. Proposed is an algorithm that can design a maximum visibility path given models of the vehicle, target, sensor, environment, and visibility. An approximate visibility, finite-horizon dynamic programming approach is used to find flyable, maximum visibility paths. This algorithm is compared against a state-of-the-art optimal control solver for validation. Complex scenarios involving multiple stationary or moving targets are considered, leading to loiter patterns or pursuit paths which negotiate planar, three-dimensional, or elevation environment models. Robustness to disturbances is addressed by treating targets as regions instead of points, to improve visibility performance in the presence of uncertainty. A testbed implementation validates the algorithm in a hardware setting with a quadrotor observer, multiple moving ground vehicle targets, and an urban-like setting providing occlusions to visibility.

Thesis Supervisor: Jonathan P. How
Title: Richard C. Maclaurin Professor of Aeronautics and Astronautics

Acknowledgments

I would like to thank my supervisor, Professor Jonathan How for the opportunity to work at the Aerospace Controls Laboratory. I appreciate his guidance and support throughout the two years of course work, projects, and this thesis, which have taught me an immense amount about controls, optimization, and aerospace systems hardware and software, about which I knew very little before starting at MIT. I feel much more attuned to the fields of aerospace and robotics because of this wonderful learning experience. I would also truly like to thank Dr. Luca Bertuccelli, who supervised me during my second year at MIT. We met when we began the visibility maximization project, and he guided me through the qualification exams, lab talks, projects, and the writing of this thesis. Amazingly he could be called up at any time to bring fresh insight into difficult problems, and point me in the right direction, for which I am really grateful. I want to thank Dr. Peter Cho at Lincoln Laboratory for providing funding support and entrusting me with this project which has become the subject of my thesis. I wish to thank everyone I've met in the Aerospace Controls Laboratory for all the help with homework, the lab talks, and the occasional fun events — it's been quite a memorable time that we've spent together. In addition, I want to thank Kathryn for helping make the lab's daily operations run really smoothly! Plus, my gratitudes towards friends, mentors, colleagues, and professors who have helped me get to this point and whom I've met during my stay, and to my family for their constant love and encouragement.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

1 Introduction **17**

1.1 Importance of Unmanned Aerial Vehicles 17

1.2 Autonomy in UAV Applications 18

1.3 Visibility Motion Planning Problem 19

1.4 Contributions of Thesis 20

2 Background **23**

2.1 Visibility Maximization Motion Planning 23

2.2 Optimal Control Formulation 25

2.3 Visibility Maximization Systems View 26

2.4 Modeling and Assumptions 27

2.4.1 Target Motion Models 28

2.4.2 Sensing Tasks and Sensor Models 30

2.4.3 Observer Models 32

2.4.4 Environment Models 33

2.4.5 Visibility Models 35

2.5 Literature Review of Visibility Maximization Motion Planning 39

2.6 Chapter Summary 40

3 Optimal Control for Visibility Maximization **41**

3.1 Block Diagram of Proposed Solution 41

3.2 Visibility Maximization Dynamic Programming Solver 42

3.2.1 Visibility Approximation Module 43

3.2.2 Path Planning Optimization Module 47

3.2.3 Summary of VMDP 51

3.3 VMDP Versus Optimal Control Solver 51

3.3.1 General Pseudospectral Optimization Software 51

3.3.2 Visibility Maximization in GPOPS 53

3.3.3 VMDP Versus GPOPS in Simple 2-D Environments 57
3.3.4 Performance and Computation Versus Resolution 60
3.4 Results for A Single Stationary Target 62
3.4.1 Complex Scenarios in 2-D Environments 62
3.4.2 Scenarios in 3-D and DEM Environments 65
3.5 Chapter Summary 66

4 Multiple Targets and Parametric Optimization 69
4.1 Multiple Target Formulation 69
4.1.1 Weights and Weighted Sum of Per-Target Visibilities 70
4.1.2 Maximizing the Minimum Per-Target Visibility 71
4.1.3 Diminishing Returns on Per-Target Visibility 72
4.1.4 Receding Horizon Approach for Complex Objectives 73
4.1.5 Multiple Target VMDP Numerical Results 74
4.2 Comparison Against Baseline Parametric Paths 79
4.2.1 Comparisons Between Parametric Optimizations 84
4.2.2 Comparisons Against Non-Parametric Optimization 85
4.2.3 Comparisons of Objective Functions 88
4.2.4 Effects of Visibility Approximation Error on Optimization . . 91
4.3 Chapter Summary 92

5 Moving Targets and Uncertainty in Motion Models 95
5.1 Extension of VMDP to Moving Targets 95
5.1.1 VMDP Revision for Moving Targets 96
5.1.2 Time-Dependent Visibility Table 96
5.1.3 Results for Known Target Trajectories 98
5.2 Robust Target Observation 103
5.2.1 Robust Formulations 104
5.2.2 Analysis and Numerical Results for Robust Visibility 106
5.3 Chapter Summary 107

6 Testbed Implementation 109
6.1 Testbed Modules 109
6.1.1 RAVEN Module 110
6.1.2 Test Environment 113
6.1.3 Visibility Planner Module and Real-Time Visibility Feedback . 116
6.2 Experimental Results 120

6.2.1 Ground Observer . . . 120
6.2.2 Aerial Observer . . . 125
6.3 Chapter Summary . . . 126

7 Conclusions and Future Work **129**
7.1 Conclusions . . . 129
7.2 Future Work . . . 132

A Visibility Formulation **135**
A.1 Definition of Visibility, and Necessary and Sufficient Conditions . . . 135
A.2 Visibility Models . . . 137
A.2.1 Planar Visibility Model with Point Target . . . 137
A.2.2 Planar Visibility Model with Target Region . . . 140
A.2.3 3-D Visibility Model . . . 142
A.2.4 Elevation Model in Visibility . . . 144
A.3 Target Region Intersection Calculation . . . 145
A.3.1 Translating Target Samples . . . 145
A.3.2 Uniformly Spaced Samples . . . 146

B Path Parameterization and Parametric Optimization **149**
B.1 Path Parameterization . . . 149
B.2 Parametric Optimization Methods . . . 151
B.2.1 Simulated Annealing . . . 151
B.2.2 Genetic Algorithm . . . 152
B.2.3 Cross Entropy . . . 152
B.2.4 Tabu Search . . . 152
B.2.5 Ant Colony Optimization . . . 152
B.3 Cross Entropy Implementation . . . 153

References **164**

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

1-1 Examples of UAVs used in intelligence, surveillance, and reconnaissance missions 18

2-1 Visibility maximization planning problem for an aerial observer monitoring a ground target in an urban setting 24
2-2 **VMP** system diagram 27
2-3 Input models for visibility calculation 28
2-4 Range-limited, field-of-view limited planar sensor 31
2-5 Dubins vehicle reachability set with constant speed and turn rate constraint 32
2-6 Obstacle representations in 2-D and 3-D 35
2-7 Visibility model with line-of-sight 37

3-1 Visibility maximization problem solver block diagram 42
3-2 Visibility table construction 45
3-3 Visibility table error versus resolution selection 47
3-4 Discrete graph versus trajectory-optimized paths 49
3-5 Pure pursuit controller geometry 51
3-6 Logistic sensor model 54
3-7 GPOPS solution for forward versus left facing sensor 55
3-8 GPOPS solutions for forward-facing sensor with occlusions 56
3-9 GPOPS solutions for forward-facing sensor with obstructions 58
3-10 GPOPS versus DP comparison without obstacles 59
3-11 GPOPS versus DP comparison with obstacles 60
3-12 Effect of DP resolution on performance and computation 63
3-13 Stationary target DP solutions in 2-D environments with left-facing sensor 64
3-14 Stationary target DP solutions in 2-D environments with forward-facing sensor 65

3-15 Stationary target DP solutions in 3-D environments 66
3-16 Stationary target DP solutions in DEM environments 67

4-1 DP solutions in 2-D environments, multiple stationary targets 76
4-2 DP solutions in 3-D environments, multiple stationary targets 77
4-3 DP solutions in DEM environments, multiple stationary targets . . . 78
4-4 DP solutions, comparing target weightings for 3-D environment . . . 80
4-5 DP solutions, comparing planning horizons for 2-D environment . . . 81
4-6 Parametric path shape and parameter examples 83
4-7 Comparison of parametric optimization routines for different curves . 86
4-8 Comparison of parametric optimization routines for different curves . 87
4-9 Examples of terminal penalties imposed on DP paths 87
4-10 Multiple stationary target closed contour visibility using DP versus CE 89
4-11 Examples of per-target visibility for different objective functions . . . 90
4-12 Receding horizon examples using max-min and diminishing returns metrics 91
4-13 Performance and computation versus resolution 93
4-14 Performance and computation versus resolution 93
4-15 Performance and computation versus resolution 94

5-1 Time-dependent visibility table 97
5-2 DP solutions in 2-D environments, moving target, left-facing sensor . 99
5-3 DP solutions in 3-D environments, moving target, left-facing sensor . 99
5-4 DP solutions for different target speeds 101
5-5 DP solutions in 2-D environments, multiple moving targets 102
5-6 DP solutions for targets moving together, away from each other . . . 103
5-7 Extremal versus non-extremal paths 104
5-8 Sampling methods for uncertain moving targets 105
5-9 Nominal versus robust visibility with target noise 107
5-10 Evolution of robust trajectory with increasing noise 108
5-11 Sensitivity to number of samples for robust target approximation . . 108

6-1 Testbed module diagram 110
6-2 State estimation using a motion capture system 111
6-3 Vehicle control computers 112
6-4 Color recognition by image processing software 113
6-5 Color thresholds for a blue object in one lighting setting 114

6-6 RAVEN XBee modem . . . 115
6-7 RAVEN indoor test environment . . . 116
6-8 Create ground robot . . . 117
6-9 RAVEN quadrotor . . . 118
6-10 Panasonic wireless network camera and onboard view . . . 119
6-11 Approximate rectilinear projection in camera image . . . 120
6-12 Pinhole camera projection . . . 121
6-13 Moving target with ground observer in RAVEN, 2 obstacles . . . 123
6-14 Color threshold detection, ground observer, 2 obstacles . . . 124
6-15 Measurement error sources . . . 125
6-16 Moving target with aerial observer in RAVEN, 2 obstacles . . . 127
6-17 Color threshold detection, aerial observer, 2 obstacles . . . 128

A-1 Line-of-sight visible region of the target . . . 136
A-2 Visibility shadow region . . . 139
A-3 Line-of-sight visible overlap region . . . 141
A-4 Sensor visibility overlap region . . . 141
A-5 Visibility and shadow regions in 3-D . . . 142
A-6 Light normal and shadow volume . . . 143
A-7 3-D sensor model with limited range and field of view . . . 144
A-8 Moving target region . . . 145
A-9 Sampling over the target region . . . 146
A-10 Visibility maximization with static targets and weight evolution . . . 147

B-1 Illustration of sampling parameter evolution in 1-D cross entropy optimization example . . . 153

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

3.1 Computation time and performance for DP versus GPOPS. 61
3.2 Performance and computation (GPOPS) over 200 trials 61
3.3 Performance and computation (DP) over 200 trials 61

4.1 Decision matrix for DP in max-min formulation 72
4.2 Parameters for parametric paths 82
4.3 Parameters for cross entropy visibility maximization 84
4.4 DP versus CE performance statistics over 400 trials 88

5.1 Percentage of time in loiter versus pursuit phases 100

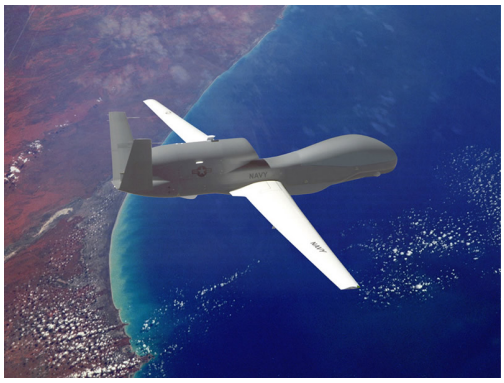
THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

1.1 Importance of Unmanned Aerial Vehicles

Unmanned aerial vehicles (UAVs) encompass a broad category of aircraft which do not have onboard flight crews, and can participate in a variety of missions requiring endurance and survivability such as persistent surveillance, precision strike, border patrol, search and rescue, and environment monitoring missions [1]. UAV aircraft include fixed-wing airplanes, helicopters, and balloons. One way to classify UAVs is according to operating domain [1]. Some operating domains include battle space awareness, force application, logistics, target/decoy, research, and civil and commercial UAVs. Battle space aware UAVs provide intelligence of the surrounding environment and troop movements; high-speed, high-altitude UAVs such as the RQ-4 Global Hawk [2] and RQ-7 Shadow [3] provide wide coverage, whereas troop-launched micro air vehicles (MAVs) provide local area surveillance [1]. Force application and combat UAVs, such as the MQ-1 Predator and MQ-9 Reaper, operate in high risk environments, serve also as surveillance platforms, and are equipped with weaponry to engage and eliminate hostile forces at safe range. Logistic UAVs are used for cargo delivery, including leaflets, fuel, and other supplies; their use is envisioned for the near future [1]. Target/decoy UAVs simulate enemy aircraft or missiles for human pilot combat training. Research UAVs act as testbeds for new aircraft designs and control/planning algorithms. Civil and commercial UAV applications, including bor-



(a) RQ-4 Global Hawk (*Source:* [7])

(b) RQ-7 Shadow (*Source:* [8])

Figure 1-1: Examples of UAVs used in intelligence, surveillance, and reconnaissance (ISR) missions.

der patrol, search and rescue, fire and environment monitoring (e.g. Aerosonde [4]), live aerial video feeds and photography (e.g. FULMAR UAV [5]), and even personal entertainment (e.g. AR.Drone [6]), are on the rise.

Alongside the growth of UAV applications, UAV operations continue a trend towards increased autonomy that will enable UAVs to fly with less supervision, less downtime, and more intelligent behavior [1]. Currently, UAVs are far from intelligent, but they do possess degrees of autonomy such as automatic piloting which enable UAVs to stay aloft with very little operator intervention [1, 9]. To further automate UAV systems, computers need to play an increasing role in higher-level decision making for UAVs, such as automatic UAV task assignment and flight route planning.

Current and future research in the worldwide UAV community is focused on bringing greater levels of autonomy into operational context, with many recent publications in this field [10–13]. The next section discusses autonomy in the context of UAVs, which will lead into the problem definition which is addressed in this thesis.

1.2 Autonomy in UAV Applications

Autonomous behaviors are typically decomposed into a hierarchy [14, 15]. Each layer in the hierarchy contains subsystems which enable a specific level of autonomy, often

labeled as "low-level control", "high-level control", or some intermediate level. The levels of autonomy refer to the separation between computation of commands and the actuation of the physical system.

"Low" levels deal directly with electro-mechanical actuation and sensing. Actuation influences the reference-tracking and stability properties of the physical system, while sensing provides feedback. Meanwhile, "high" levels deal with concepts less concerned with actuation and sensing, such as task selection, path planning, and human-machine interfaces, and are more mission-oriented in general. Intermediate control levels deal with converting high-level commands to low-level actuation, and relaying low-level sensing data to high-level planners.

Persistent Surveillance as a Motivating Example

An important application of autonomy is persistent surveillance using UAVs. Autonomy enables computers onboard surveillance UAVs to automatically calculate trajectories which provide sensor coverage of targets and account for given terrain and weather information, and to execute the flight maneuvers to follow the trajectory.

To enable autonomy in a persistent surveillance application, tools from control theory and artificial intelligence are explored. The next section describes the problem statement for persistent surveillance addressed in this thesis.

1.3 Visibility Motion Planning Problem

Maximizing visibility of targets by an aerial observer presently remains a pervasive problem. Whether in the pursuit of hostile agents, or ensuring that line of sight communication to a friendly agent is maintained, visibility problems have been studied extensively in the literature from geometry [16–18], control [19], estimation [20–23], and pursuit-evasion [24] perspectives. Visibility problems continue to attract attention due to the complexity of the optimization problem and the demand for faster, near real-time implementations.

This thesis addresses the *visibility maximization motion planning problem*. This

problem will be referred to as the Visibility Maximization Problem: *Given an aerial vehicle, maximize the fraction of time spent observing a number of stationary or moving targets, in an obstacle-rich environment and subject to constraints in sensing and line-of-sight visibility.*

1.4 Contributions of Thesis

Even with the large body of prior work mentioned earlier, there are many significant challenges still left to address. One limitation of the prior work is that only a subset of the full problem specification is investigated at a time. The combination of observer dynamics, multiple moving targets, sensor limits, 3-D and digital elevation model environments, a near-optimal path, and robustness is not considered. The work in this thesis proposes an approximate solution which incorporates the full set of models as well as empirically showing results that are intuitive, close to optimal for cases that can be readily verified, and robust to model errors.

This thesis provides a thorough introduction to the visibility maximization problem, a literature review of past work, details of modeling assumptions used to solve the problem, a new solver for the visibility maximization problem, comparisons with a state-of-the-art solver, and results of complex scenarios from simulation and a hardware in the loop testbed implementation.

Chapter 2 provides a background of the visibility maximization problem. It discusses models of the visibility system inputs, including targets, sensor, observer, environment, and visibility models that are used in the literature. It also provides a review of prior work related to the visibility maximization problem.

Chapter 3 details the new visibility maximization solver, including a formalization of the approximation scheme and the path planning optimization. The new solver is compared against an existing, state-of-the-art solver, using simple test cases as validation. More advanced cases, in particular static targets in complex environments, will show the merits of the proposed solver for addressing the visibility maximization problem.

Chapter 4 discusses multiple target visibility. It introduces different objective functions which lead to observation of multiple targets, including objectives that ensure sightings of difficult-to-see targets. It discusses two parametric analyses of multiple targets: the effect of changing weightings on targets for the weighted visibility objective function, and the effect of increasing the time horizon. It also describes a parametric optimization which can find simple flight contours such as circles, ellipses, and racetracks for maximizing visibility along those contours, which serve as a baseline to compare the performance of the new solver in complex environments. It presents performance results for different metaheuristic searches over the different contours, performance comparisons between the new solver and the parametric optimization, and shows performance-computation results as a function of the visibility approximation accuracy for the parametric solver.

Chapter 5 considers extensions of the new solver to moving targets and targets with uncertain motion. It considers the effect of the speed ratio between the target and observer. It also explores the effect of multiple moving targets, when their motions either diverge or move together. In addition it discusses a robust formulation for the visibility of targets with uncertain motion. It presents performance between the robust versus nominal formulations in the presence of target motion stochasticity.

Chapter 6 documents testbed implementation results in hardware. It describes the hardware and software architectures that enable a scale demonstration of the visibility maximization path planner. It shows that the algorithms developed in this thesis are valid. Actual camera measurements are taken and compared against the expected measurement from the algorithm for static and dynamic targets in complex 3-D environments.

Chapter 7 concludes the thesis, providing a summary of the results, and also outlines future work to advance the scope and performance of the new solver in the context of visibility maximization.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Background

This chapter presents the visibility maximization problem and provides a description of the methods and assumptions in the visibility maximization problem. Section 2.1 provides an overview of visibility maximization motion planning. Section 2.2 defines the visibility optimization problem in general mathematical form. Section 2.3 decomposes the problem into its constituent features. Section 2.4 presents the modeling assumptions and methods, including descriptions of the models for the target, observer, sensor, environment, and visibility. Section 2.5 reviews the relevant literature of the visibility maximization problem, including some of the assumptions, solution methods, and limitations of previous work.

2.1 Visibility Maximization Motion Planning

Visibility maximization motion planning is a problem where an observer's trajectory needs to be computed such that the trajectory maximizes visibility of a target. Figure 2-1 shows an example of a helicopter (an aerial observer) with a camera pursuing a car driving along city streets. The helicopter pilot is trying to maximize visibility by applying a sequence of control actions to maneuver the aircraft so that it follows a trajectory which maximizes the amount of time the car is kept in view.

This thesis adopts the following naming conventions relating to the visibility maximization motion planning problem. Other names and terminologies will be defined

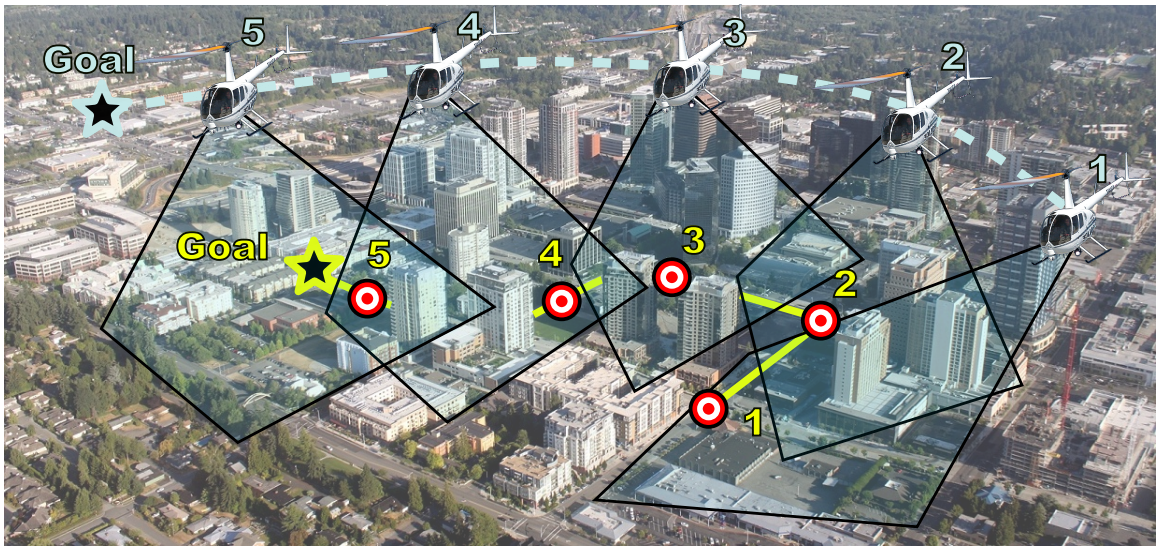


Figure 2-1: Visibility maximization planning problem for an aerial observer monitoring a ground target in an urban setting. *Photo source:* [25], [26]

in the rest of this chapter, or as they appear in the remaining chapters.

- *Actions* are the commands given to a vehicle. When actions are taken, a vehicle will change position, orientation, or some other state. A sequence of actions causes the vehicle to move along along a trajectory.

- *States* are variables which define the properties about a system, such as position and orientation.

- *Trajectories* or *paths* are a sequence of points which can be followed by the vehicle. When the trajectory or actions need to be found, motion planning is used to find a feasible path or list of actions. Motion planning can also be used to design paths or actions that minimize or maximize some objective, which might be the shortest time to a destination, or the greatest visibility of a target.

- *Visibility of a target* refers to the ability of a sensor to detect some property about the target. This property could be a picture or a video of the target acquired by a camera, or it could be a communications link, and others.

To find a maximum visibility path or actions, the problem is posed in an optimal

control framework. Optimal control returns the best sequence of control actions and a path for the vehicle to optimize some objective. The next section describes the optimal control problem for visibility in mathematical form.

2.2 Optimal Control Formulation

Optimal control is a general mathematical formulation for precisely stating an optimization problem with the goal of finding control actions or trajectories for a dynamic system. For the visibility maximization motion planning problem, the goal is to find control actions or a path of the observer that maximizes the accumulated visibility of targets.

Optimal control problems in general depend on time, t. Properties of the optimization, including the location of targets and the observer, are represented using *states* in vector form $\mathbf{x}(t)$ which define position and orientation for example and may also vary with time. The visibility maximization problem consists of targets with state vector $\mathbf{x}_T$, a sensor model $\mathbf{S}$, an observer with state $\mathbf{x}_A$, and an environment description $\mathcal{T}$, all of which may be time-varying. The *state space* contains all permissible values for the states. As an example, the observer's state in general 3-D coordinates is

$$\mathbf{x}_A(t) = [x_A(t), y_A(t), z_A(t), \phi_A(t), \theta_A(t), \psi_A(t)]^T \tag{2.1}$$

where (x, y, z) represents the position and (ϕ, θ, ψ) the orientation in Euclidean space which vary with time.

The goal of visibility maximization motion planning, also known as the *Visibility Maximization Problem* (**VMP**) problem, is defined as follows: find the *optimal control policy* $\mathbf{u}^*(t)$ for the observer which maximizes a *visibility reward* J_V subject to a set

of constraints

$$\textbf{VMP}: \begin{cases} \max\limits_{\mathbf{u}(t)} & J_V = \dfrac{1}{T}\displaystyle\int_0^T \mathcal{V}\left[\mathbf{x}_A(t), \mathbf{x}_T(t), \mathcal{T}(t), \mathcal{S}(\mathbf{x}_A(t), \mathbf{x}_T(t))\right] dt \\ \text{s.t.} & \dot{\mathbf{x}}_A(t) = \mathbf{f}\left(\mathbf{x}_A(t), \mathbf{u}(t)\right) \\ & \mathbf{C}(\mathbf{x}_A(t), \dot{\mathbf{x}}_A(t), \mathbf{x}_T(t), \dot{\mathbf{x}}_T(t), \mathbf{u}(t), \mathcal{T}(t)) \leq \mathbf{0} \\ & \mathbf{x}_A(0) = \mathcal{X}_{A,0}, \mathbf{x}_T(0) = \mathcal{X}_{T,0}, \dot{\mathbf{x}}_A(0) = \dot{\mathcal{X}}_{A,0}, \dot{\mathbf{x}}_T(0) = \dot{\mathcal{X}}_{T,0} \end{cases} \tag{2.2}$$

Here $\mathcal{V}(\cdot)$, which will be explained further in Section 2.4.5, is the instantaneous visibility as a function of the observer state $\mathbf{x}_A(t)$, target state $\mathbf{x}_T(t)$, the environment model $\mathcal{T}(t)$, and the sensor model $\mathcal{S}(\mathbf{x}_A(t), \mathbf{x}_T(t))$ which are functions of time t. The observer dynamics are in the form of a time derivative $\dot{\mathbf{x}}_A$. Initial conditions for the observer and target are specified by $\mathcal{X}_{A,0}$, $\mathcal{X}_{T,0}$, $\dot{\mathcal{X}}_{A,0}$, and $\dot{\mathcal{X}}_{T,0}$. The problem is finite horizon ($t_f = T$), which can represent the endurance limit of the aircraft for example. The visibility reward is normalized by $1/T$ and takes on values $J_V \in [0, 1]$, to ensure that different trajectories can be compared fairly between each other. A finite horizon must be considered otherwise the computation will not terminate.

An intuitive, systems-based model is presented in the next section. This systems-based model will prepare a discussion for the types of modeling that occur in the visibility maximization problem.

2.3 Visibility Maximization Systems View

Figure 2-2 shows a decomposed, input-output systems view of the visibility maximization motion planning problem. The figure places the inputs, the solver, and the outputs in order from left to right. This particular layout shows that five inputs need to be modeled before the problem solver can produce the desired outputs. The inputs are the target, sensor, observer, environment, and visibility models, and the outputs are the optimal control sequence and trajectory for the observer which maximizes visibility. The center block represents the *visibility maximization problem solver* or **VMP** solver. This module, much like a black box in model-based problem solving,

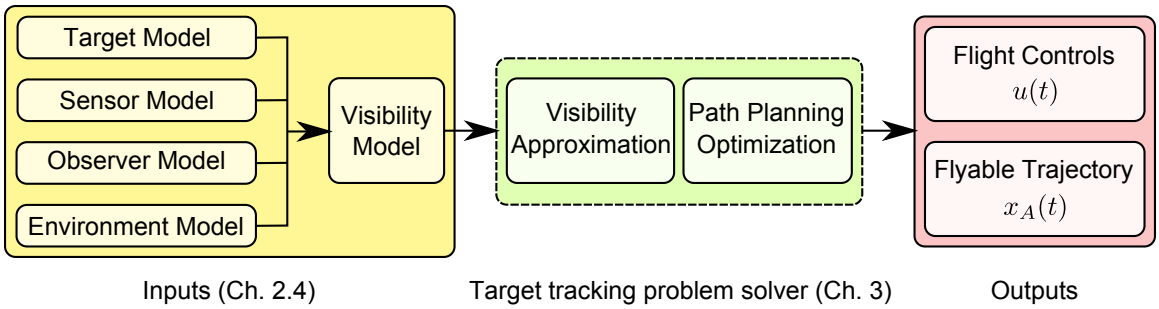


Figure 2-2: **VMP** system diagram, showing the flow of inputs to outputs through the problem solver.

encapsulates the complexities associated with the visibility maximization problem and can be reused for many variations on the inputs. This center block is concerned with two main questions: how to solve the problem, and how to solve it tractably.

The remainder of this chapter discusses the input models, some assumptions about each model, as well as the prior art which have addressed the visibility maximization motion planning problem. Chapter 3 describes the visibility maximization problem solver in detail.

2.4 Modeling and Assumptions

This section focuses on the five input models shown in Figure 2-3: target, sensor, vehicle, environment, and visibility. Each model captures assumptions that are made regarding each input, and need to be discussed individually. Once the models are known, they can be provided as input into the problem solver.

Figure 2-3 also shows the relations between the different inputs. The environment defines the possible locations for the targets and observer. The targets and observer determine if there is sensor visibility. All four determine if there is line-of-sight visibility between the targets and observer in the presence of occlusions and sensor limits.

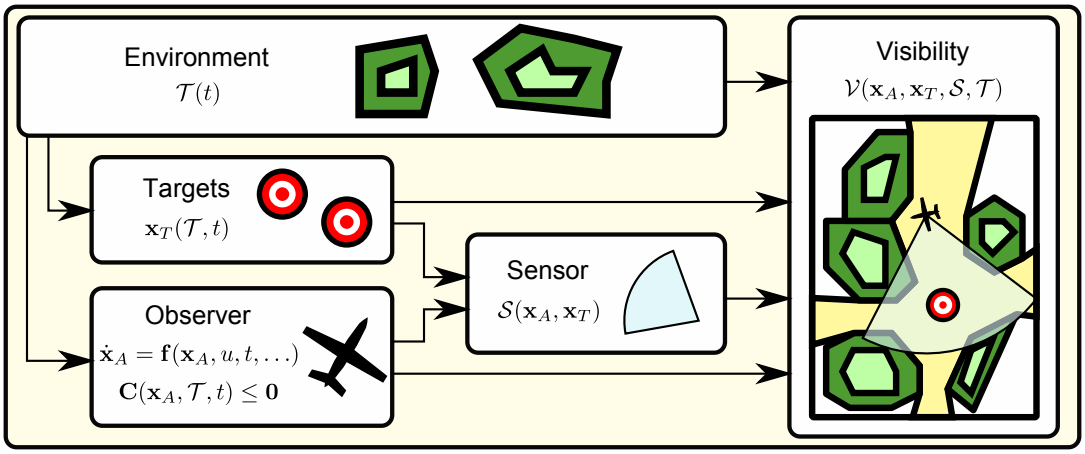


Figure 2-3: Input models for visibility calculation.

2.4.1 Target Motion Models

Targets include vehicles that are being pursued by a follower vehicle, buildings which need to be monitored from the air, landmarks that need to be captured in an aerial video, or wildlife whose movements need to be studied. Targets are characterized using the state space representation, $\mathbf{x}_T(t)$. The target state for an individual target in 3-D Euclidean space is

$$\mathbf{x}_T(t) = [x_T(t), y_T(t), z_T(t), \phi_T(t), \theta_T(t), \psi_T(t)]^T \tag{2.3}$$

Target motion models describe the movement of the target. Ref. [27] provides a discussion of deterministic target motion models which capture different assumptions on the target motion. Examples include constant velocity (linear function for position) described in continuous and discrete forms in Equation 2.4, constant acceleration (quadratic function for position), and constant change in acceleration (cubic polynomial for position) models.

$$\textbf{Continuous time form}: \begin{bmatrix} \dot{x}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \tag{2.4a}$$

$$\textbf{Discrete time form}: \begin{bmatrix} x_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ v_k \end{bmatrix} + \begin{bmatrix} \Delta T^2/2 \\ \Delta T \end{bmatrix} u_k \tag{2.4b}$$

Coordinated turns, with constant speed and constant angular rate Ω, can also be specified in continuous and discrete forms:

$$\textbf{Continuous time form}: \begin{bmatrix} \dot{x}(t) \\ \dot{v}_x(t) \\ \dot{y}(t) \\ \dot{v}_y(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\Omega \\ 0 & 0 & 0 & 1 \\ 0 & \Omega & 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ v_x(t) \\ y(t) \\ v_y(t) \end{bmatrix} \tag{2.5a}$$

$$\textbf{Discrete time form}: \begin{bmatrix} x_{k+1} \\ v_{x,k+1} \\ y_{k+1} \\ v_{y,k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & -\Omega\Delta t \\ 0 & 0 & 1 & \Delta t \\ 0 & \Omega\Delta t & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ v_{x,k} \\ y_k \\ v_{y,k} \end{bmatrix} \tag{2.5b}$$

These target motion models, in differential equation form, are required in the optimal control formulation in Eq. 2.2 to calculate visibility of a moving target with known motion.

The surveys [28] also describe common target models used to predict the trajectory and estimate the position of an aircraft. Aircraft flight paths including turns and wind effects are modeled. These trajectories are relevant for a ground observer monitoring an aircraft, such as a radar tracking application. They do not consider optimal observer trajectories for mobile observers. Ref. [16] considers two target motion models for a visibility maximization problem in the plane. The first model is a fully known, time-parameterized target trajectory. The second is an evader model, where the target is assumed to take actions which escape observation in the shortest amount of time. They do not consider uncertainty explicitly in the motion model.

Ref. [17] assumes worst-case target behavior in a UAV-based target tracking problem in an urban environment. This worst-case model is a planar diffusion model under a bounded speed assumption with known initial location. In their paper, the target's diffusion area under sensor coverage is to be maximized over the duration of the mission. This diffusion model may be conservative however, in that it could under-perform when the target trajectory is known. Ref. [24] uses a two-dimensional

occupancy grid map to represent the probability of a target being present at a particular state. The probabilities are updated with sensor measurements, and the map is used in a search and capture problem. Targets are treated as evaders whose predicted paths are assumed to be adversarial, meaning targets will minimize the time to escape the observer's sensor view. Their planning strategy is greedy with respect to maximizing the one-step probability of capturing the evader; their approach is suboptimal but they provide an upper bound for the expected time to capture.

Ref. [29] considers target motion with bounded but unknown speed and unknown direction of movement for a 3-D target tracking motion planning problem. Target actions are modeled as probabilities, but their true actions are adversarial and attempt to minimize time to escape observation. Ref. [19] consider target motion along a planar circular arc for the visibility maintenance problem, for a follower vehicle keeping sight of a leader vehicle. The follower treats the leader as an unknown but bounded disturbance for a reference-tracking controller. They provide guarantees for maintaining visibility for benign target motions such as a smooth arc. However, their approach may not be general enough if the target motion were more complex.

In this thesis, target motions are initially assumed to be fully deterministic and parameterized by time. This is a valid assumption for a wide range of applications, including vehicles moving along a road network at known speed, as well as stationary targets.

2.4.2 Sensing Tasks and Sensor Models

The second input model is the *sensor* $\mathcal{S}$. The sensor provides information about the target by taking measurements. Sensors include visible light cameras and radars. It is modeled using states $\mathbf{x}_{\mathcal{S}}$. Sensor models are concerned with constraints on the ability to take measurements, and are treated in a set-theoretic manner. The constraints are the set boundaries which define range and field-of-view limits, if and when such bounds are considered in the model.

The pinhole camera is a popular model for cameras [30, 31]. The pinhole camera model is simple: it provides an intuitive geometric description of a light ray's bearing

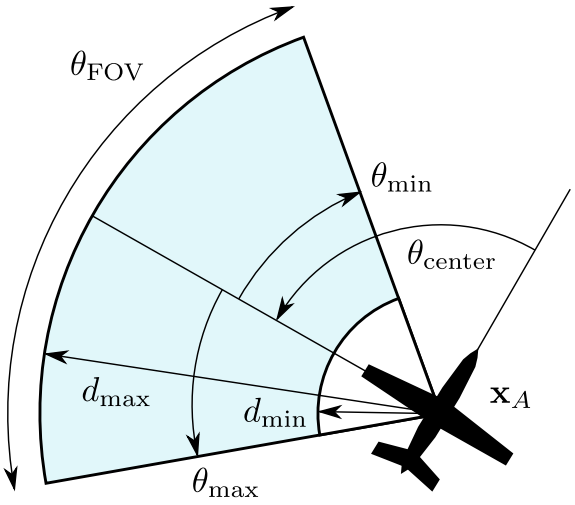


Figure 2-4: Range-limited, field-of-view limited planar sensor.

relative to the position and orientation of the sensor—the bearing measurements being determined from the set of pixels illuminated on the image sensor. The geometric model is a useful approximation that has been used successfully for applications such as vision-based navigation [32–34].

In this thesis, the sensor model is a variant of the pinhole camera, with fixed range and field of view limits. A 2-D projection is shown in Figure 2-4. Physically the sensor is not gimbaled, meaning it is fixed to the body of the observer: examples of such sensors include front-mounted cameras on a dashboard or a side-looking synthetic aperture radar on a surveillance UAV. The sensor constraints model optical sensors, which generally have directionality, field of view limits, resolution and range limits. For a planar sensor, the range limits are $[d_{\text{min}}, d_{\text{max}}]$ and the field of view limits are $[\theta_{\mathcal{S},\text{min}}, \theta_{\mathcal{S},\text{max}}]$. A 3-D sensor has both horizontal and vertical field of view limits.

The noisy measurement process is not considered in this thesis for two main reasons: first, there is a broad literature addressing the sensing problem in other domains (such as search) which leads to significantly different kinds of optimizations.[1] Secondly, the main goal of this work is to maintain the target in the sensor field of view, which requires only a geometric description of the sensor.

[1] Considerable attention has been paid to studying measurement processes in the context of target tracking. Some examples include [35–39].

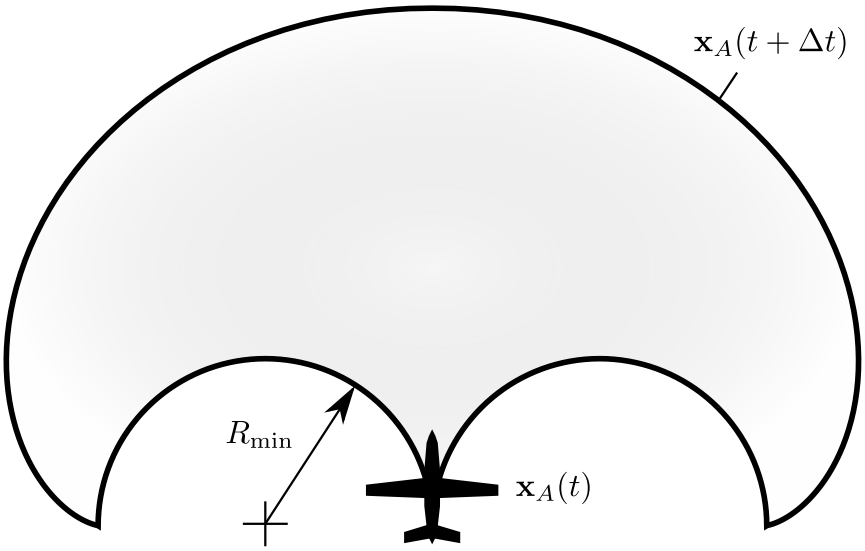


Figure 2-5: Dubins vehicle reachability set with constant speed and turn rate constraint.

2.4.3 Observer Models

The *observer* is a vehicle which carries the sensor. The vehicle can be a fixed-wing UAV, with constant speed and turn rate constraints, or it can be ground-based. Dynamics in general affect the transitions between the observer's states, $\mathbf{x}_A$. Many vehicles such as fixed-wing aircraft are dynamically constrained, so it is necessary to model the dynamics. In state space, dynamics are represented by a vector of ordinary differential equations.

Examples of ground vehicle models include skid-steer, unicycle, and bicycle models [40]. Examples of aircraft models include the Dubins model [41] and more complicated models [42]. Vehicles can be holonomic or non-holonomic [15]. Holonomic vehicles can be actuated directly over every degree of freedom. Non-holonomic vehicles are underactuated and do not have full control over its degrees of freedom. Limited steering automobiles and fixed-wing, fixed-engine aircraft are examples of non-holonomic vehicles.

For this thesis, the observer is assumed to be a fixed-wing UAV. A fixed-wing UAV can be modeled using a planar Dubins model [41]. This model is popular and widely used due to its simplicity for modeling fixed-wing aircraft.

A Dubins model is a vehicle with constant speed V_A, and a bounded control input u which changes the vehicle's heading angle, $u = \dot{\psi}$. The instantaneous heading ψ determines the change in position, $\dot{x}$ and $\dot{y}$. The dynamics are represented in vector form,

$$\dot{\mathbf{x}}_A(t) = \mathbf{f}\left(\mathbf{x}_A(t), \mathbf{u}(t), V_A\right) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \\ \dot{\phi}(t) \\ \dot{\theta}(t) \\ \dot{\psi}(t) \end{bmatrix} = \begin{bmatrix} V_A \cos\psi(t) \\ V_A \sin\psi(t) \\ 0 \\ 0 \\ 0 \\ u(t) \end{bmatrix} \tag{2.6}$$

The altitude z, roll (bank angle ϕ), and pitch (nose up, nose down angle θ) of the aircraft are assumed to be constant, so their rate of change is zero for all time.

The turn rate constraint $\omega_{\max}$ as a function of the minimum turning radius $R_{\min}$ and speed is given by

$$\omega_{\max} = \frac{V}{R_{\min}} \tag{2.7}$$

The turn rate constraint limits the control action. The turn rate constraint is the same in both turn directions.

$$\mathbf{C}\left(\mathbf{x}_A(t), \mathbf{u}(t)\right) = \begin{bmatrix} u(t) - \omega_{\max} \\ -u(t) - \omega_{\max} \end{bmatrix} \leq \mathbf{0}, \ \forall t \tag{2.8}$$

2.4.4 Environment Models

The fourth input model is an abstraction of the environment. The *environment* affects the targets, sensor, and observer each in at least one way, and so it must be modeled. The specific effects are as follows:

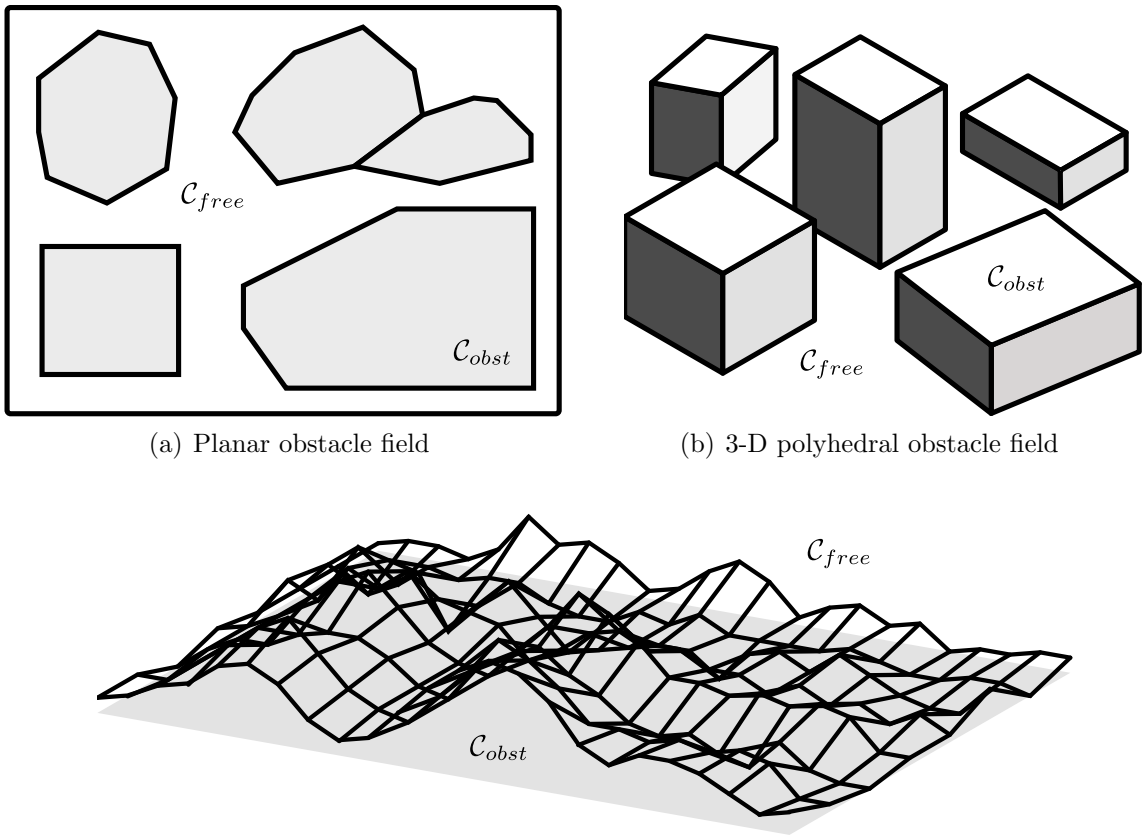
- For targets, the environment defines the targets' possible locations and movements.

- For the sensor, the environment determines if visibility exists between the targets and sensor.
- For the observer, the environment is a set of inadmissible states. Observer states are infeasible because they result in a collision, a violation of controlled space, a safety risk, and other factors.

Environments are modeled using geometric features. Other representations including 3-D scanning may be more accurate, but require more time to make visibility calculations. Therefore, environments are modeled with as few geometric features as possible to decrease computation time. In this thesis, the environment model is assumed to be known a priori. This is a realistic assumption for buildings, mountains, forests, and certain no-fly zones such as controlled airspaces over population centers or habitat protected areas; these are modeled and stored in Geographic Information Systems (GIS) databases. Several environment models have been considered in the literature. Ref. [16] uses a planar model of the environment to model obstacles and occlusions for planning visibility-rich paths. Obstacles are represented as polygons. The interior of these polygons deny passage, and the boundaries disrupt line of sight. Ref. [18] uses a 3-D model of an urban environment for visibility-based path planning for a UAV. 3-D polyhedrons, representing buildings in a city, obstruct passage of the UAV and occlude targets from sensors.

Figure 2-6 shows examples of 2-D, 3-D, and elevation models for environments. In this thesis, all three environment models are considered. In all three cases, the environments are modeled geometrically. *Computational geometry* provides mathematical definitions of boundaries of obstacles using vertices, edges, surfaces, and sets.

Unknown environments that are mapped as the observer is maneuvering through the environment are not considered in this thesis. The fact that the environment is unknown means that the observer must worry about exploration and mapping instead of solely maximizing visibility. Unknown environments can be mapped online using frameworks such as simultaneous localization and mapping (SLAM) [43]. Models are generated online from sensor data, such as LIDAR or optical cameras and pattern



(a) Planar obstacle field

(b) 3-D polyhedral obstacle field

(c) Digital Elevation Model (DEM)

Figure 2-6: Obstacle representations in 2-D and 3-D.

recognition techniques. For example, Ref. [44] uses 3-D scanning LIDAR to map the surfaces of roads, curbs, obstacles, and other vehicles from a perception-driven autonomous vehicle.

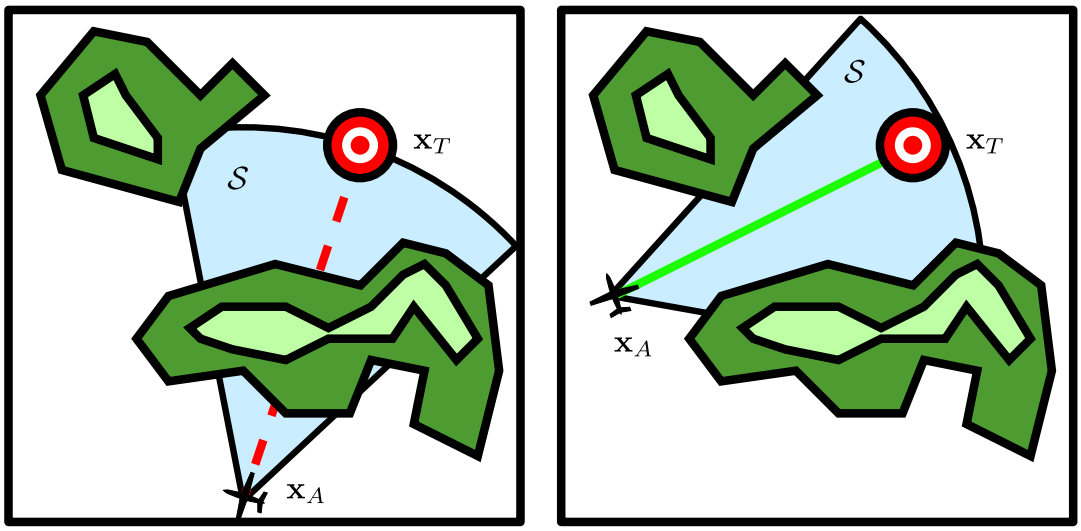
The next section describes the combination of the four models discussed so far, namely the target, sensor, observer, and environment, and how they relate together in the context of visibility.

2.4.5 Visibility Models

The fifth input model is visibility. *Visibility*, $\mathcal{V}$, is a binary variable which specifies whether the sensor can detect the target and is the objective to be maximized in the **VMP**. The visibility model considered in this thesis is complex, and a complete treatment is left in Appendix A. Described below are visibility models used in the

literature, as well as their applications. The model used in this thesis, its computational complexity, and the need for approximations to the visibility function are also discussed.

Ref. [45] considers visibility of a point-and-shoot sensor such as a laser range finder in cluttered environments. They model visibility as line of sight between the sensor and a polygonal target. They use visibility to define the set of sensor configurations which have a partial view of the target. They only consider placement and not dynamic movement of a sensor. Ref. [46] considers the *Art Gallery Problem* formulated by Klee, which asks the minimum number of omnidirectional, infinite range sensors and their positions required to cover a closed, planar environment with holes. They model two types of visibility: one is visibility of the workspace by each guard, and the other is visibility amongst the guards themselves. Visibility is treated as a constraint. There is no consideration of a dynamically-constrained observer. Ref. [47] considers the level sets of the volume of space visible to an observer, for determining the best placement of a sensor which covers the maximum volume. A gradient-based method is used, so their solution can be suboptimal if the level set function is non-convex. Vehicle dynamics, sensors, and targets are not considered, which are major limitations of their method, because the path planning problem is completely neglected. Ref. [48] investigates visibility maintenance and maximization from a UAV to track adversarial targets using iterative prediction. They assume a perfect information model, meaning they have full state knowledge of target and no-fly zone positions. They use a range-limited, field-of-view limited sensor to determine line-of-sight visibility to the target. Visibility is treated as a reward, but they also penalize large gaps in visibility. They use two methods, a greedy potential field and an A* informed search to find the best UAV path, where the best path considers multiple objectives including reducing loss of sight and maintaining an ideal separation. The primary limitation is the full state knowledge assumed about the target; no estimation is involved. Ref. [18] also treats visibility as a reward to be maximized along a trajectory flown by a UAV. They use a detailed line-of-sight visibility model accounting for atmospheric effects and probabilities of detection due to rasterization effects of the sensing array and



(a) Obstructed line-of-sight visibility (b) Unobstructed line-of-sight visibility

Figure 2-7: Visibility model with line-of-sight.

errors due to the image recognition process. The best UAV trajectory is determined using a probabilistic roadmap. However, probabilistic roadmap planners do not have any optimality guarantees.

Line-of-Sight Model

The visibility model chosen for this thesis is the *line-of-sight model*. Line-of-sight is an unbroken line segment joining the target and the sensor. Interruptions to line of sight are caused by obstructions which are treated as perfect absorbers of light waves. Maximizing visibility is the same as maximizing line-of-sight. Atmospheric effects and reflections are not considered. Figure 2-7 shows examples of obstructed and unobstructed line-of-sight visibility between a sensor and a target. The equation

$$\mathcal{V}_{\text{LOS}} = \left[\mathbf{x}_A \in \bar{\mathcal{Z}}(\mathbf{x}_T)\right] \tag{2.9}$$

$$\mathcal{V}_{\mathcal{S}} = \left[\mathbf{x}_T \in \mathcal{S}(\mathbf{x}_A)\right] \tag{2.10}$$

$$\mathcal{V}(\mathbf{x}_A, \mathbf{x}_T, \mathcal{T}, \mathcal{S}) = \mathcal{V}_{\text{LOS}} \bigcap \mathcal{V}_{\mathcal{S}} \tag{2.11}$$

states two conditions that need to be satisfied for visibility. The first is the inclusion of the observer's state $\mathbf{x}_A$ in the line-of-sight visibility region of the target, $\bar{\mathcal{Z}}$. The second is the inclusion of the target's state $\mathbf{x}_T$ in the sensor footprint, $\mathcal{S}(\mathbf{x}_A)$. The complexity in the equation lies in determining the boundaries of the line-of-sight visibility set.

Complexity of Visibility Calculations

Computational complexity is a measure of the tractability of an algorithm. A tractable algorithm is one that can be computed using a "small" number of operations. Fewer operations are desirable, because the number of operations is proportional to the computation time. Simple computations in each operation are also desirable.

As mentioned earlier, visibility is an involved calculation requiring a non-trivial amount of computation for even a single query. The computational complexity of one visibility calculation $\mathcal{V}$ is proportional to the number of geometric features in the environment. In planar environments, a single call evaluating visibility has an upper-bound or worst-case time complexity (denoted by big-O notation) in the number of occluding edges, $\mathcal{O}(n_{edge})$ [49]. In three-dimensional environments, the number of calculations is related to the number of planar surfaces, $\mathcal{O}(n_{surf})$. More efficient data structures (e.g. kD-trees) can be used to reduce complexity by lowering the number of surfaces checked in each visibility query [50, 51].

For digital elevation models sampled in a uniform grid, the linear-time computation for visibility proposed by Ref. [52] has time complexity which is a function of the number of grid points separating the two points of interest, $\mathcal{O}(m_x + m_y)$. However, for non-uniform altitude maps, such as longitude-latitude maps, triangle patches can be used to fill the surface, and time complexity becomes approximately $\mathcal{O}(m_x m_y)$ in a naïve implementation. The remainder of this chapter discusses the visibility maximization problem solver.

2.5 Literature Review of Visibility Maximization Motion Planning

The **VMP** continues to receive attention because it has many real-life applications, such as reconnaissance and patrol, and because of the difficulty of finding maximum visibility paths in the presence of sensor, observer, and environment constraints. Few attempts however have been made to solve the complete **VMP** with all modeling aspects fully considered. The literature shows that the problem of multiple moving targets, range-limited and field-of-view-limited sensors, dynamically-constrained observers, and 3-D and digital elevation model environments, has not been considered altogether at once, and their methods are not easily extended to capture this rich problem.

Approaches related to the **VMP** include visual servoing and visibility-based motion planning. These approaches were introduced in the previous sections ([16–19, 24, 29, 47]). The following presents more examples of existing solution methods that are relevant to the **VMP**. Ref. [53] treats visibility in the context of line-of-sight communications between a planetary rover and multiple airborne or orbiting observers, in a 3-D environment. They discuss efficient representations of the environment, using a non-uniform lattice to model mountainous terrain. The best trajectory is calculated using an A* graph search along adjacent nodes which are the centers of safe volumes. They do not consider sensor range or field of view limits, and they do not consider the dynamics of the observer, both of which increase the complexity of the problem significantly. Ref. [54] considers the polygon-visiting Dubins traveling salesman problem. Visibility is modeled as a constraint, where the observer must visit each target for at least an infinitesimal duration. Visibility is occluded by a 3-D environment representing buildings in an urban setting. The sets of states where the vehicle can see each target are represented as polygons. The shortest Dubins path which visits all polygons is desired. A genetic algorithm is used to determine the timing and sequence of alphabets consisting of left turns, right turns, and straight segments, resulting in the shortest path. The author does not consider field of view

limited sensors, which breaks the polygon assumption and changes the problem definition. Also, the traveling salesman formulation only ensures brief glances of targets if a solution is found, and does not maximize visibility of the targets.

Ref. [55] considers the optimal control formulation of the visibility maintenance problem for a static target and a field-of-view constrained sensor. Extremal paths are generated using Pontryagin's Maximum Principle. They do not consider path obstacles or occlusions, which would add a significant number of constraints to their problem. Also, they consider a static target only, which is a major simplifying assumption used in their method. Ref. [56] considers the visibility maximization for an eye-in-hand articulated manipulator with multiple degrees of freedom. They use a field-of-view constrained optical camera model for visual tracking of a stationary feature in the environment. A probabilistic roadmap approach is taken to find a feasible yet suboptimal trajectory for the manipulator. The major limitations are the lack of observer dynamics and a suboptimal path.

As seen above, many of these methods are incomplete in terms of the **VMP** for a UAV tracking targets in dense environments. Either the models are simplified, meaning the sensor model is not range-limited and field-of-view-limited, the environment is not modeled, or the solution method is not optimal. The remainder of the thesis addresses the proposed solution to the full **VMP** problem.

2.6 Chapter Summary

This chapter introduced the Visibility Maximization Problem (**VMP**). A systems-level decomposition of the visibility maximization was presented. The chapter also discussed models for the targets, sensor, observer, environment, and visibility, along with examples used in the literature. Approaches to the visibility maximization motion planning problem considered in the literature were also presented. The next chapter proposes a solution to the **VMP** including an evaluation against a state-of-the-art optimal control solver for stationary targets.

Chapter 3

Optimal Control for Visibility Maximization

This chapter presents the proposed solution method for the visibility maximization problem (**VMP**). Section 3.1 introduces the optimal control based visibility maximization dynamic programming solver, or **VMDP**, in the form of a block diagram. Section 3.2 describes in detail the methods used in the proposed solver. Section 3.3 presents comparison results between the new solver and a state-of-the-art optimal control solver, the General Pseudospectral Optimization Software (GPOPS) package, for simplified map geometries. Section 3.4 shows numerical results using the new solver for more complex scenarios that cannot be solved using GPOPS, including 3-D maps and digital elevation models, demonstrating its effectiveness in real-world problems.

3.1 Block Diagram of Proposed Solution

Figure 3-1 shows the block diagram of the Visibility Maximization Dynamic Programming solver, or **VMDP**, which is the proposed solution to the **VMP**. The solver addresses the limitations of previous work: it considers multiple targets, constrained sensors, dynamic observers, complex map geometries, optimality of the resulting flight path, and computational efficiency. Given the inputs described in Chap-

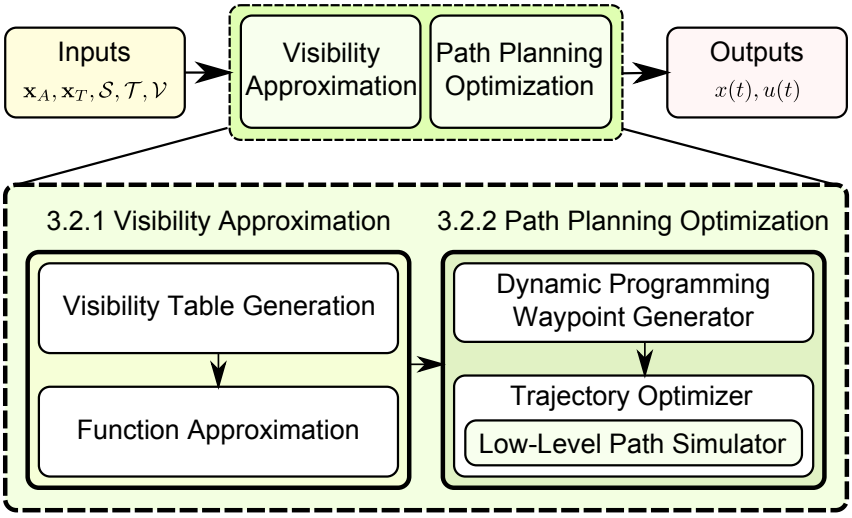


Figure 3-1: Visibility maximization problem solver block diagram.

ter 2, the **VMDP** generates a visibility table, which is used to compute a tractable approximation to the visibility function. Then, given the approximation, a dynamic programming path planner calculates optimal waypoints for the dynamic observer in the discrete graph. The waypoints are used for a waypoint-following control simulator which uses the same controller as the actual observer. The calculated waypoints are fed through a trajectory optimizer to find a feasible continuous path.

3.2 Visibility Maximization Dynamic Programming Solver

This section provides a complete description of the visibility maximization problem solver presented in Fig. 3-1, which is the main contribution of this thesis. The two sub-modules address the **VMP** question: how to find a path which maximizes visibility, and how to perform the computation tractably. Section 3.2.1 presents the Visibility Approximation Module which provides the path planner with a tractable visibility calculation. Section 3.2.2 presents the Path Planner, which uses the visibility approximation to determine optimal paths with respect to the approximation.

The main benefit of decomposition is to decouple the visibility objective from the

path planner, so that the individual problems can be solved more quickly. These benefits are explained in more detail in the following sections.

3.2.1 Visibility Approximation Module

To find the optimal trajectory in a multidimensional search space, the quantity in the cost function of Eq. 2.2, $\mathcal{V}(\mathbf{x}_A, \mathbf{x}_T, \mathcal{T}, \mathcal{S})$, must be called thousands if not millions of times. This computation is expensive, however an approximation can be used to reduce computation significantly. The Visibility Approximation Module calculates an approximation to the visibility function,

$$\mathcal{V}_{\text{approx}}(\mathbf{x}_A(t)) \approx \mathcal{V}(\mathbf{x}_A(t), \mathbf{x}_T(t), \mathcal{T}, \mathcal{S}) \tag{3.1}$$

in the **VMP**. It will be shown that the most complex inputs to the visibility calculation, including the sensor model $\mathcal{S}$ and environment model $\mathcal{T}$, no longer appear in the call to $\mathcal{V}_{\text{approx}}$, making the approximation considerably more tractable.

The Visibility Approximation Module provides the infrastructure to compute such an approximation $\mathcal{V}_{\text{approx}}$. It consists of two subcomponents. The first is a visibility table generated from exact visibility calculations. The second is a function approximation of visibility calculated using the visibility table.

The following sections detail the two subcomponents, as well providing an analysis of the computation speed and error between the visibility approximation and exact visibility to justify the use of the former.

Visibility Table

As mentioned in Section 2.4.5, the complexity of individual visibility calculations is dependent on the complexity of the map, such as the number of edges defining the environment. In addition to this complexity, the complexity of path planning over a forward search graph is exponential in the branching factor [15], making the combined complexity very high.

A *visibility table* $\mathcal{VT}$[1] is an array of uniformly spaced points spanning the continuous state space of the observer.

$$\begin{aligned}\mathcal{VT}(t) &= \left\{\mathcal{V}(x^{[i_x]}, y^{[i_y]}, \psi^{[i_\psi]}, \mathbf{x}_T(t), \mathcal{T}(t), \mathcal{S}(t))\right\}, \\ i_x &\in \left\{1, \ldots, N_{i_x}\right\}, i_y \in \left\{1, \ldots, N_{i_y}\right\}, i_\psi \in \left\{1, \ldots, N_{i_\psi}\right\}\end{aligned} \tag{3.2}$$

A $\mathcal{VT}$ provides a deterministic bound on the number of function calls to $\mathcal{V}$. It is also the resolution of the visibility table.

$$N_{\mathcal{VT}} = N_{i_x} N_{i_y} N_{i_\psi} \tag{3.3}$$

This bound on the number of calls is the primary advantage of using the visibility table. The bound can be selected based on the desired error, or the allowed computation time. The table is subsequently used for approximating visibility using inexpensive calculations. The visibility table in Eq. 3.2 is valid for a Dubins observer with three variable states[2] (x, y, ψ), where $(N_{i_x}, N_{i_y}, N_{i_\psi})$ are the number of discretizations in each dimension.

Figure 3-2 shows the construction of the visibility table over the state space of the observer. Visibility is computed exactly over a discrete set of observer positions and headings. The figure depicts visibility values with:

- the observer's positions $(x^{[i_x]}, y^{[i_y]})$ at the centers of each octagon,
- the sensor's orientations, as a function of the observer orientation $(\psi^{[i_\psi]})$, aimed in the same directions as each triangle, and
- the colors denoting the binary visibility value (light = visible, dark = not visible) given the observer position and sensor direction.

[1]The visibility table will sometimes be referred to as a *Skymap*. One can imagine points in the sky from which an aerial observer peers into the environment. Each of these points, along with the observer's orientation, is used to perform the geometric visibility calculation.

[2]Recall that although the state space of a vehicle in general 3-D coordinates contains $(x, y, z, \phi, \theta, \psi)$, three quantities (z, ϕ, θ) are constant over time (see Equation 2.6) and therefore do not need to be subsampled.

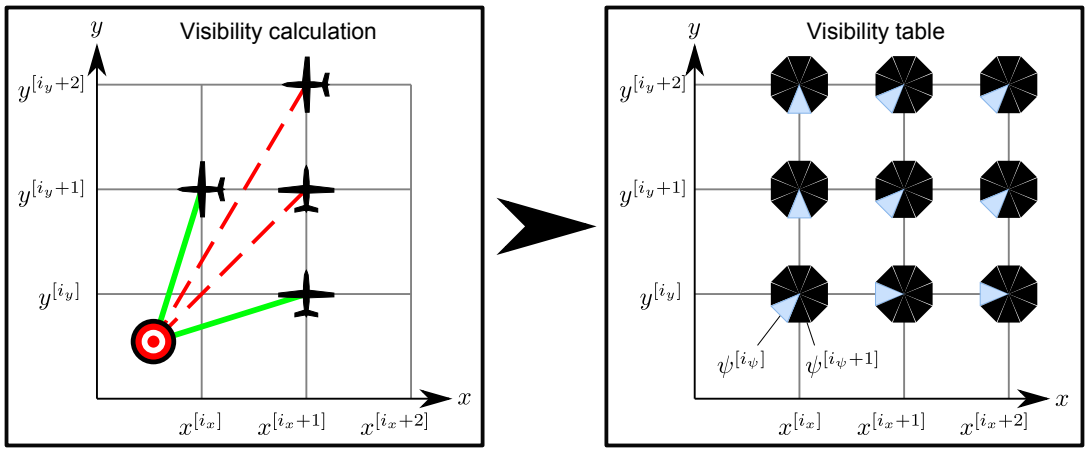


Figure 3-2: Visibility table construction, providing a look-up table for pre-computed visibility values between observer and target at a discrete set of observer positions and headings.

The visibility table has a drawback of discretization error. Any subsequent approximation utilizing the discrete table must interpolate values at the continuous states. In the **VMP**, the approximation affects the path planning stage, yielding solutions which are optimal to the approximation but not the exact visibility.

Visibility Function Approximation

In this section, interpolation over the visibility table is described. Interpolation provides an essential, time-saving feature for the **VMDP** solver. A function approximation $\mathcal{V}_{\text{approx}}$ can yield computational savings necessary for time-critical implementations.

Several interpolation schemes for $\mathcal{V}_{\text{approx}}$ are considered, such as nearest neighbor, polynomial, and spline interpolation [57]. Linear interpolation is used because it is simple, and it provides smooth transitions between visible and not visible points in space. The smoothness discourages states that are near visibility boundaries. Since the table of data is three-dimensional, trilinear interpolation is used. More complex interpolations are not necessary in this particular application, or are not suitable because the interpolation can exceed the $[0, 1]$ bounds on visibility.

The next section describes the computation time versus error tradeoff in using the

visibility approximation.

Analysis of Visibility Approximation

This section provides two evaluations of the visibility approximation. The first assessment is the complexity of calculating the visibility table. The table provides a computation benefit only when the number of calls to $\mathcal{V}_{\text{approx}}$ exceeds $N_{\mathcal{VT}}$. In other words, a table is not needed if more effort is required to build the table than to compute visibility directly. By evaluating the complexity of calculating visibility, it will be apparent that the table will be needed in almost every case.

The second consideration is the computation versus error analysis. To evaluate the error of the visibility approximation, the mean absolute error (MAE) is used:

$$e_{\mathcal{V},\text{MAE}} = \mathbf{E}|\mathcal{V}_{\text{approx}} - \mathcal{V}| \tag{3.4}$$

where the expectation is taken over a large number of Monte Carlo runs. The best discretization in x, y, and ψ are plotted[3] for each $N_{\mathcal{VT}}$. Figure 3-3 plots the histogram and mean of the MAE over 100 target initial locations, versus total resolution $N_{\mathcal{VT}}$. The figure also plots computation time for the corresponding $N_{\mathcal{VT}}$. The horizontal line is the average time using the exact calculation. This error analysis shows that reducing error requires increased computation, as expected. However, increasing computation leads to diminishing reductions in error since error decreases asymptotically. A good trade-off point can be selected which balances computation with error reduction. In this example 2×10^5 exact visibility calls appears to show good computation performance, and 4×10^5 shows the beginning of diminishing reduction in error. The resolution can be specified according to error tolerance.

[3]The best discretization means the best ratio of N_x, N_y, and N_ψ given a maximum number of calls $N_{\mathcal{VT}}$. The best ratios are determined in a separate analysis.

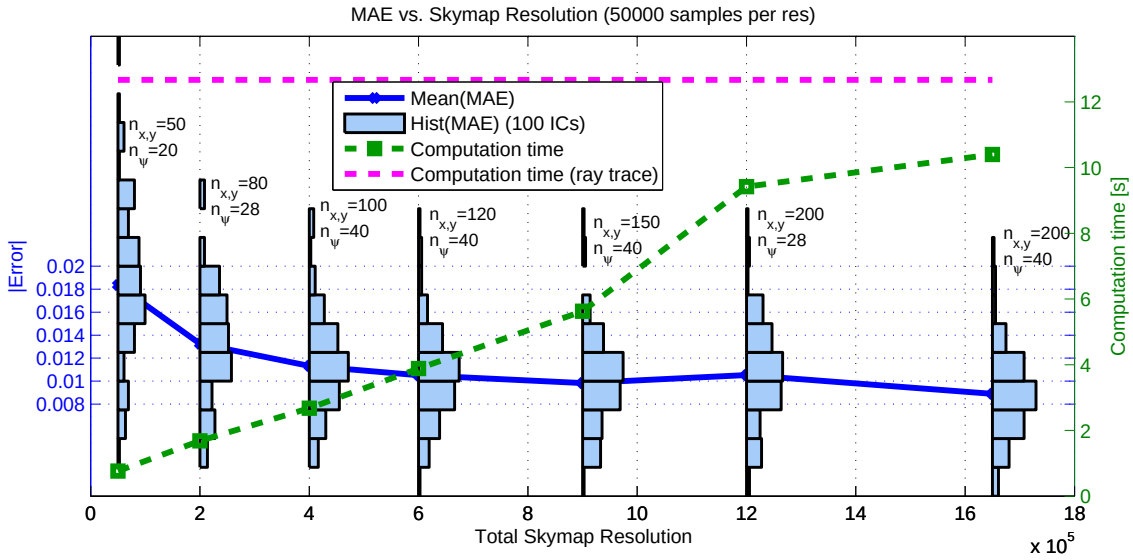


Figure 3-3: Visibility table mean absolute error and computation versus resolution selection. As resolution increases, the mean absolute error decreases asymptotically.

3.2.2 Path Planning Optimization Module

This section describes the path planner in the **VMDP**. Given that the nature of the **VMP** is an optimization problem, the solver incorporates an optimization module which is used to find the path which maximizes visibility. Two subcomponents make up the Path Planning Optimization Module. The first is a coarse search, using a dynamic programming waypoint generator. The second is a refined search using trajectory optimization. The coarse search is needed for tractable optimization. The refined search ensures feasibility. An extension considers parametric flight paths, which are simpler, more natural flight paths which can also be computed more quickly.

Dynamic Programming Waypoint Generator

The coarse search employs a method known as *dynamic programming (DP)* [58]. Dynamic programming has been used to solve discrete optimal control problems. Dynamic programming uses the Bellman principle of optimality to perform an exhaustive search over the state space, but is less expensive than a naïve uniform search. At each iteration, the best control actions are selected, by maximizing over the *cost to*

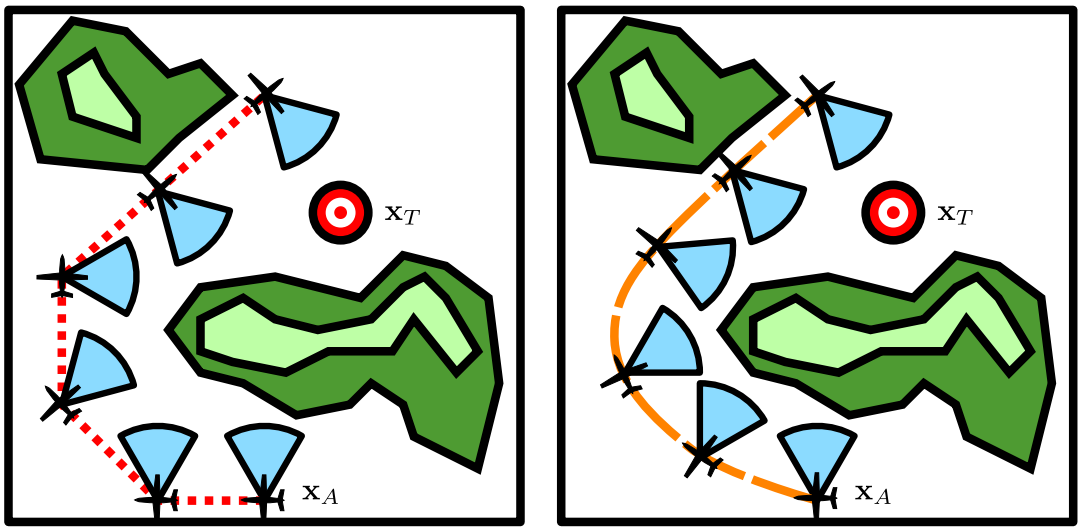
go values of all paths that reach a given state. The Bellman equation is stated below:

$$J^*(\mathbf{x}[k]) = \max_{\mathbf{x}[k],\mathbf{u}[k]} \left[J^*(\mathbf{x}[k+1]) + g(\mathbf{x}[k], \mathbf{u}[k], k)\right], \; k \in \{0, 1, \ldots, N\} \tag{3.5}$$

where at stage k, the optimal cost to go $J^*(\mathbf{x}[k])$ is the cost to go $J^*(\mathbf{x}[k+1])$ of the future subsequence, plus the *stage cost* $g(\mathbf{x}[k], \mathbf{u}[k])$ in performing the transition. In the **VMP**, the cost to go is the visibility accumulated between $\mathbf{x}[k+1]$ and $\mathbf{x}[N]$, the stage cost is the visibility accrued by executing an action $u[k]$, and the transitions are the observer's dynamics. The stages are taken from the final time $k = N$ backwards to the initial time $k = 0$, since the cost to go is known at the terminal time. This is possible if the goal state $\mathbf{x}[N]$ is known. Since the terminal state of a **VMP** is not known, it makes sense to start from the initial condition. A version of the principle of optimality holds for the forward search [59], known as the reversed principle of optimality. The resulting DP, known as a forward DP, replaces costs-to-go by *costs-to-arrive*, and future states by the state history.

The cost function must satisfy the monotonicity and contraction properties [60]. A function f is monotonic if $\forall x, \forall y$, s.t. $x \leq y$, then $f(x) \leq f(y)$. The visibility reward J_V is monotonic because it is an integral (or sum) of values of $\frac{\mathcal{V}dt}{T}$, where $\mathcal{V}$ lies between 0 and 1. A function f is a contraction if for some metric d and $\forall x, \forall y$ in the same metric space, then $\exists k$ s.t. $d \cdot (f(x), f(y)) \leq k \cdot d(x, y)$. The contraction property ensures the cost function is bounded. J_V is clearly bounded because it is normalized to values of $[0, 1]$ for all t. An additional memoryless assumption, in which past states and past actions cannot affect future rewards, must also hold [59]. The reward J_V is calculated only as a function of the current state and action, so this requirement also holds.

The most important feature of a DP path planner is its ability to find the optimal path in a discrete graph with dynamic constraints. Given a description of observer dynamics, constraints, and the visibility function, a DP solver finds the set of discrete controls which maximize the visibility objective. Thus, a discrete approximation to the **VMP** is solved using the dynamic programming path planner. Stochastic



(a) Coarse path generated by DP (b) Trajectory-optimized path

Figure 3-4: Discrete graph versus trajectory-optimized paths.

problems can also be handled by maximizing the expected instead of a deterministic reward. The output is a set of discrete control actions and states. The controls can be directly executed, or the state trajectory can be used as a list of reference waypoints. However, the control actions and states may be too coarse. The coarseness means it is possible to violate state constraints if the control actions are executed open-loop or if the state trajectory is tracked using a waypoint following controller. Hence, a trajectory optimization phase is proposed and is discussed next.

Trajectory Optimizer

The refined search reconsiders feasibility of the actual flight path. Figure 3-4 shows a DP-generated path and a trajectory optimizer path. There is a possibility of state constraint violations when the DP control or DP state trajectory is directly implemented. An additional optimization phase can be used for feasibility evaluation.

The trajectory optimizer poses the **VMP** directly. The **VMP** is a nonlinear optimization over the discrete control sequence $\{\mathbf{u}_1, \ldots, \mathbf{u}_N\}, \forall \mathbf{u}_k \in \mathbb{R}$ or the sequence of waypoints $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \forall \mathbf{x}_k \in \mathbb{R}^2$, at N time intervals. The optimization perturbs the control or waypoint sequence to find a feasible sequence that satisfies the

VMP constraints. The optimization can be solved using a nonlinear optimization solver.

Trajectory optimization can be slow because the number of decision variables N is large. Either the entire set of control actions or the entire waypoint list must be passed to the optimization. Each time a variable changes, the path needs to be simulated to ensure feasibility. The initial DP phase can accelerate the process by providing a good initial guess upon which to make improvements.

The actual flight trajectory is simulated using a low-level controller, given the control commands or reference waypoints.

Low-Level Control Simulator

The low-level control simulator returns a finer discretization of the action and state trajectories. This refinement is a much more accurate representation of the true flight path than the coarse path generated by the DP path planner.

Different controllers can be used to track a trajectory [61]. An open-loop controller will execute a smoothing of the set of controls provided by the DP. Alternatively, a waypoint-following controller will track the DP state trajectory by computing a new set of controls. In this thesis, the second type of controller is used.

One type of waypoint-following controller is the *pure pursuit controller*, with variants for aerial vehicles [62, 63] and ground vehicles [64]. It is an excellent controller for staying on paths. Figure 3-5 illustrates the pure pursuit controller geometry. The variant considered in this thesis uses a proportional steering controller on the heading error e_ψ. This heading error is measured with respect to a radial line segment $\overline{P_{\mathbf{x}_A} P_{L1}}$ with length L_1 which extends towards the reference line segment $\overline{P_A P_B}$. The reference line segment connects the previous and current waypoints, P_A and P_B, respectively. Once the observer is within L_1 of the current waypoint, the waypoint list is shifted so that the current waypoint becomes P_C and the previous waypoint is P_B.

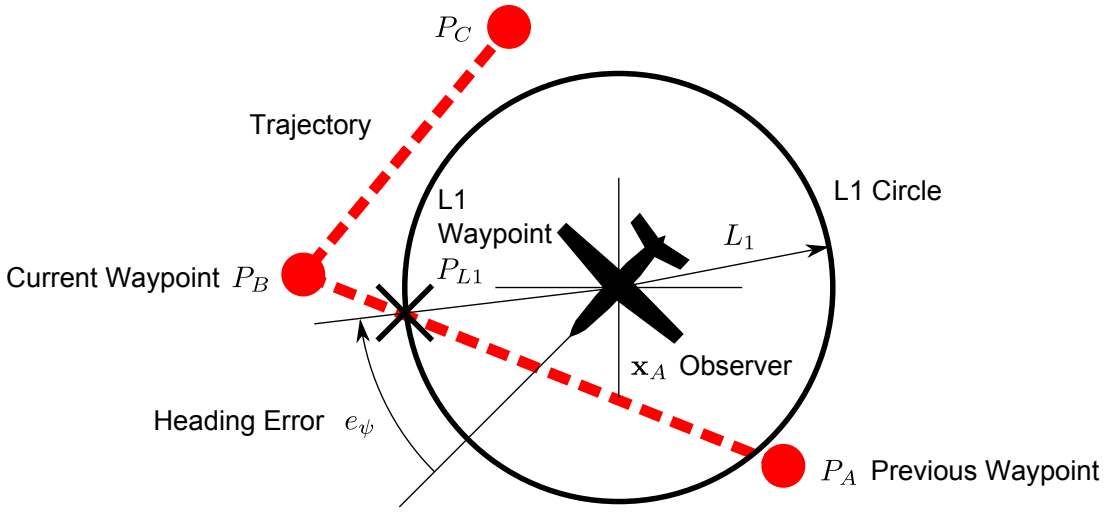


Figure 3-5: Pure pursuit controller geometry.

3.2.3 Summary of VMDP

This section described the proposed optimal control solver, the Visibility Maximization Dynamic Programming (**VMDP**) solver, for the visibility maximization problem. The next section discusses the performance of **VMDP** compared to a state-of-the-art optimal control solver.

3.3 VMDP Versus Optimal Control Solver

This section describes a state-of-the-art optimal control solver, the General Pseudospectral OPtimization Software (GPOPS), which can solve the **VMP**. This section presents GPOPS solutions to problems of increasing complexity. This section also compares the GPOPS solutions with the **VMDP** solutions.

3.3.1 General Pseudospectral Optimization Software

GPOPS, or *General Pseudospectral OPtimization Software*, is a MATLAB® package [65] for the *Gauss pseudospectral method* (GPM) [66] which can solve multi-phase optimal control problems with a general objective function and constraints, shown below. The software interfaces with a nonlinear optimization solver and solves for the

optimal control actions $\mathbf{u}$. State and control constraints are satisfied by the solution if one is found. For more information on GPOPS, see [65].

$$
\begin{aligned}
\min \;\; & J = \sum_{p=1}^{P} \left[\Phi^{(p)} \left(\mathbf{x}^{(p)}(t_0), t_0^{(p)}, \mathbf{x}^{(p)}(t_f), t_f^{(p)}; \mathbf{q}^{(p)} \right) + g^{(p)} \right] \\
\text{s.t.} \;\; & g^{(p)} = \int_{t_0^{(p)}}^{t_f^{(p)}} \mathcal{L}^{(p)} \left(\mathbf{x}^{(p)}(t), \mathbf{u}^{(p)}(t), t; \mathbf{q}^{(p)} \right) dt \\
& \dot{\mathbf{x}}^{(p)} = \mathbf{f}^{(p)} \left(\mathbf{x}^{(p)}, \mathbf{u}^{(p)}, t; \mathbf{q}^{(p)} \right) \\
& \mathbf{C}_{\min}^{(p)} \leq \mathbf{C}^{(p)} \left(\mathbf{x}^{(p)}(t), \mathbf{u}^{(p)}(t), t; \mathbf{q}^{(p)} \right) \leq \mathbf{C}_{\max}^{(p)} \\
& \phi_{\min}^{(p)} \leq \phi^{(p)} \left(\mathbf{x}^{(p)}(t_0), t_0^{(p)}, \mathbf{x}^{(p)}(t_f), t_f^{(p)}; \mathbf{q}^{(p)} \right) \leq \phi_{\max}^{(p)} \\
& \mathbf{L}_{\min}^{(s)} \leq \mathbf{L}^{(s)} \left(\mathbf{x}^{(p_l^s)}(t_f), t_f^{(p_l^s)}, \mathbf{x}^{(p_r^s)}(t_0), t_0^{(p_r^s)}; \mathbf{q}^{(p_r^s)} \right) \leq \mathbf{L}_{\max}^{(s)} \\
& (p = 1, \ldots, P), (p_l, p_r \in [1, \ldots, P]), (s = 1, \ldots, L)
\end{aligned} \tag{3.6}
$$

The GPOPS interface accepts handles to MATLAB functions: these functions describe the cost, cost gradients, state/control dynamics, constraints, phases, and events. It also accepts an initial guess. The inputs are mapped to a nonlinear programming (NLP) problem which can be solved using a NLP optimization package. GPOPS uses the SNOPT NLP solver [67] to calculate the optimal control trajectory. saturated systems: [65] discuss examples of bioreactor control and multiple-stage rocket maximum ascent. In these situations, the resulting control sequences are typically executed in open-loop fashion, which is reasonable if the dynamic models are accurate. However, for the **VMP**, GPOPS is not a real-time implementation due to its computational requirements. Each call to GPOPS requires many iterations with SNOPT. Also, GPOPS can terminate without finding a solution in more complex scenarios.

The next section describes the use of GPOPS to solve simplified versions of the target tracking problem in 2-D. Initially, no obstacles are considered. Then, the problem increases in complexity by considering occlusions (visibility-only constraints, not state constraints). Finally, the occlusions also behave as state constraints, obstructing the path of the observer.

3.3.2 Visibility Maximization in GPOPS

This section describes GPOPS solutions to the **VMP** in planar environments. The first problem is an obstacle-free environment with a constrained sensor. The second problem adds occlusions, but does not include obstructions.

Logistic Curve Representation of Sensor

The simplest **VMP** posed in GPOPS models only a dynamic observer and a constrained sensor. The target is stationary and there are no obstacles or occlusions. Maximizing visibility in this scenario corresponds to maximizing the time the range $d(t) = d(\mathbf{x}_A(t), \mathbf{x}_T(t))$ and bearing $\theta(t) = \theta(\mathbf{x}_A(t), \mathbf{x}_T(t))$ (both scalar functions of time) between observer and target lie within the sensor boundaries,

$$\max_{\mathbf{u}(t)} J_V = \frac{1}{T} \int_0^T \mathcal{V}(\mathbf{x}_A(t), \mathbf{x}_T(t), \mathcal{S})\ dt \tag{3.7}$$
$$\text{s.t. (constraints in Equation 2.2)}$$

where visibility is defined by the sensor boundaries:

$$\mathcal{V}(\mathbf{x}_A(t), \mathbf{x}_T(t), \mathcal{S}) = \mathcal{L}_{\mathcal{G}}\left(d(t), d_{\min}, d_{\max}\right) \mathcal{L}_{\mathcal{G}}\left(\theta(t), \theta_{\min}, \theta_{\max}\right) \tag{3.8}$$

Each $\mathcal{L}_{\mathcal{G}}$ is the product of two univariate logistic functions forming a continuous, smoothed mapping of $\mathbb{R}$ to $[0, 1]$, and was proposed for modeling binary constraints such as path planning obstacles in [68]:

$$\mathcal{L}_{\mathcal{G}}(x, x_{\min}, x_{\max}) = \left[1 + e^{-M(x - x_{\min})}\right]^{-1} \left[1 + e^{M(x - x_{\max})}\right]^{-1} \tag{3.9}$$

Figure 3-6 depicts the visibility $\mathcal{V}$ at different positions inside and outside the boundaries of the sensor. While the sensor visibility is intended to be binary, logistic or sigmoid functions remove discontinuities from the objective function, so that gradients exist everywhere in the state space except at the sensor origin. The logistic functions blur the sensor boundary, which can coarsely model spectral attenuation

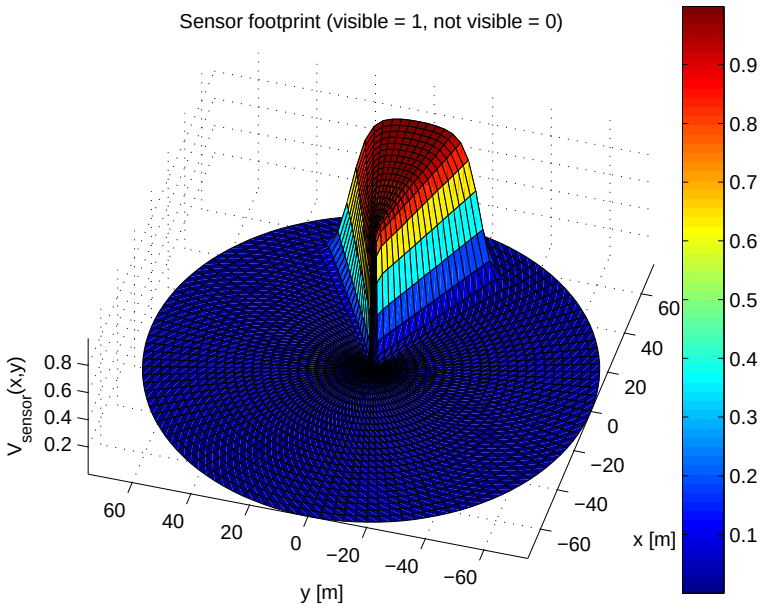
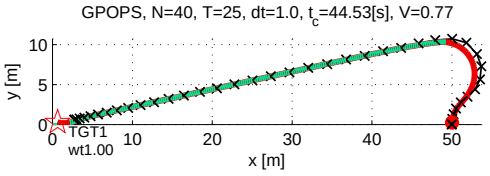


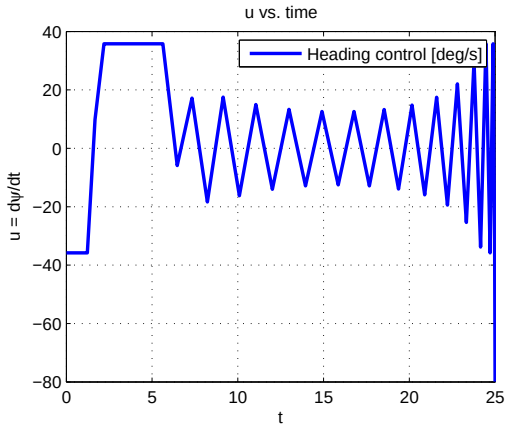
Figure 3-6: Logistic sensor model, range 60, FOV 36°

(for example, vignetting) or degradation (for example, chromatic aberration) at the edges of camera view [69], and model preferences to have targets centered in the sensor view.

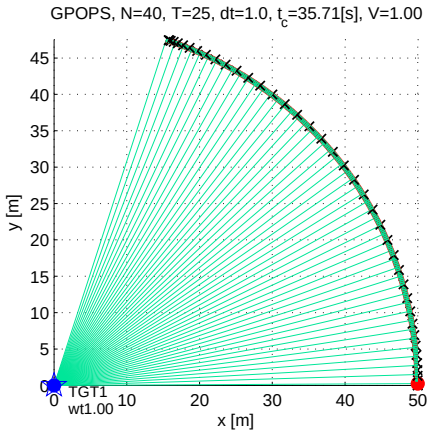
Figure 3-7 shows GPOPS results for two scenarios. The first is a forward-looking sensor, and the second is a left-looking sensor. In both cases, the observer tracks a stationary target at $(0, 0)$. The path of the forward-looking sensor in Figure 3-7(a) takes the observer directly towards the target, after making a right and left turn subject to the turn rate constraints. The path of the left-looking sensor in Figure 3-7(c) keeps the sensor pointed inwards while the observer orbits the target. The control actions $u(t) = \dot{\psi}(t)$ are also shown. One observed issue with GPOPS is twitching or chatter in the controls caused by numerical errors; from an execution standpoint these can be smoothed with a low-pass filter before being commanded to the controller.



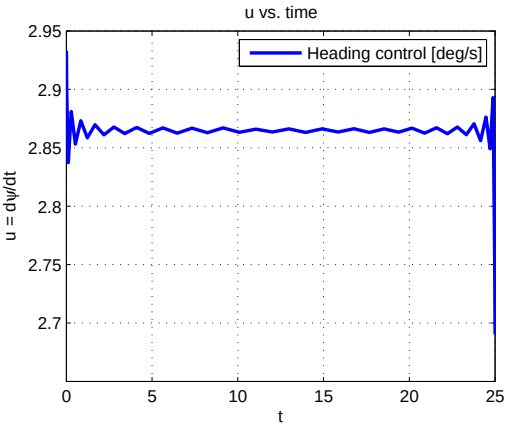
(a) x versus y, forward facing sensor



(b) u versus t, forward facing sensor



(c) x versus y, left facing sensor



(d) u versus t, left facing sensor

Figure 3-7: GPOPS solution for forward versus left facing sensor, 40 nodes, $\mathbf{x}_0 = [50, 0, \pi/2]^T$, $V = 2.5$ [m/s], $T = 25$ [s]. In (b) and (d), the twitching is a result of numerical integration issues.

Logistic Sensor in the Presence of Occlusions

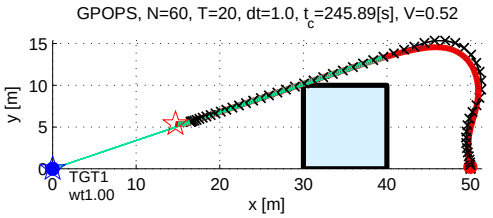
A more challenging visibility maximization scenario for GPOPS considers both the effect of sensor visibility and occlusions to the observer,

$$\max_{\mathbf{u}(t)} J_V = \frac{1}{T}\int_0^T \mathcal{V}(\mathbf{x}_A(t), \mathbf{x}_T(t), \mathcal{T}, \mathcal{S})\, dt \tag{3.10}$$

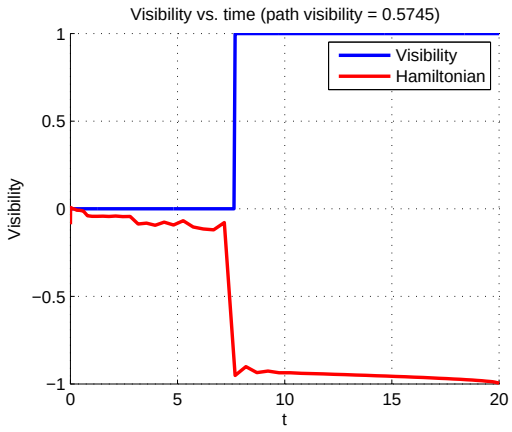
$$\text{s.t. (constraints in Equation 2.2)}$$

where $\mathcal{V}(\mathbf{x}_A(t), \mathbf{x}_T(t), \mathcal{T}, \mathcal{S})$ is defined in Appendix A.

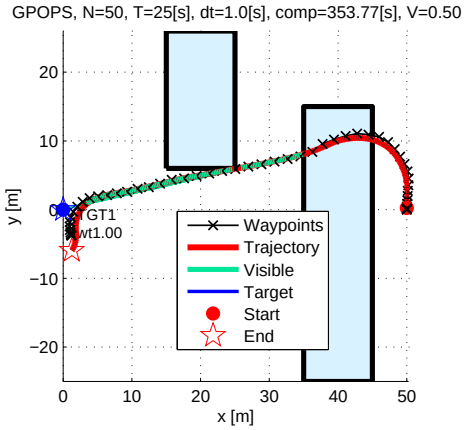
Figure 3-8 shows the paths for two scenarios: the first contains one occlusion



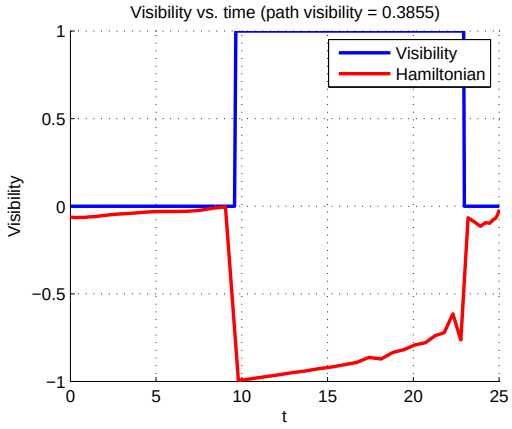
(a) x versus y, 1 occlusion



(b) Visibility versus t, 1 occlusion



(c) x versus y, 2 occlusions



(d) Visibility versus t, 2 occlusions

Figure 3-8: GPOPS solutions for forward-facing sensor with occlusions.

region, and the second contains two occlusion regions. In both cases, the sensor is forward-facing and the occlusions do not act as state constraints.[4] The best path is one which exits the occluded regions as quickly as possible, and maximizes the remaining time the target is visible.

[4]Examples of occlusions which do not act as obstructions include light cloud cover, fog, and low-lying buildings or geographic features.

Logistic Sensor, Occlusions, and Obstructions

The most complex 2-D scenario in GPOPS involves obstructions which do not allow the observer's path to cross inside them.

$$\max_{\mathbf{u}(t)} J_V = \frac{1}{T}\int_0^T \left[\mathcal{V}(\mathbf{x}_A(t), \mathbf{x}_T(t), \mathcal{T}, \mathcal{S}) - C(\mathbf{x}_A(t), \mathcal{T})\right]\, dt \tag{3.11}$$

s.t. (constraints in Equation 2.2)

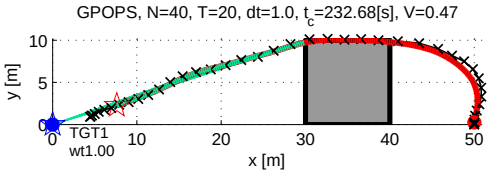
These obstructions are state constraints which can be modeled as hard or soft constraints. A hard constraint cannot be violated by a feasible path. A soft constraint $C(\mathbf{x}_A(t), \mathcal{T})$ places a very high or even infinite penalty in the cost function for any state violations.

Figure 3-9 shows examples of GPOPS with occlusions and obstructions. The observer has a forward-looking sensor. The paths suggest that the observer should reach the visible region as quickly as possible while avoiding the obstructions.

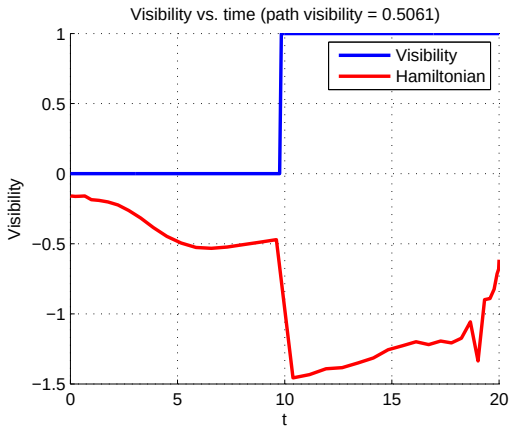
3.3.3 VMDP Versus GPOPS in Simple 2-D Environments

This section compares GPOPS and **VMDP** results in 2-D environments. The results validate the **VMDP** for complex 2-D and 3-D environments. They also show significant computational savings by using the **VMDP** solver compared to GPOPS, while incurring a small penalty in performance. Finally, they show that GPOPS produces suboptimal or infeasible plans in more advanced scenarios, making it inadequate for the **VMP**.

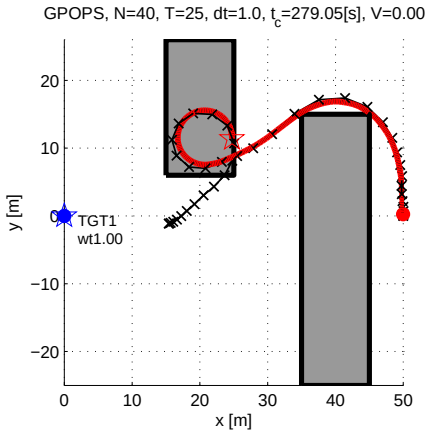
Figure 3-10 shows the obstacle-free scenario with forward and left-looking sensors. Figure 3-11 shows a single obstacle scenario with a forward-looking sensor. The GPOPS solutions for the obstacle-free, occlusion-present, and obstacle-present environments, where found, appear intuitively to maximize visibility by aiming the sensor directly at the target. The **VMDP** paths likewise maneuver the sensor to face the target, but appear to be sensitive in observer position.



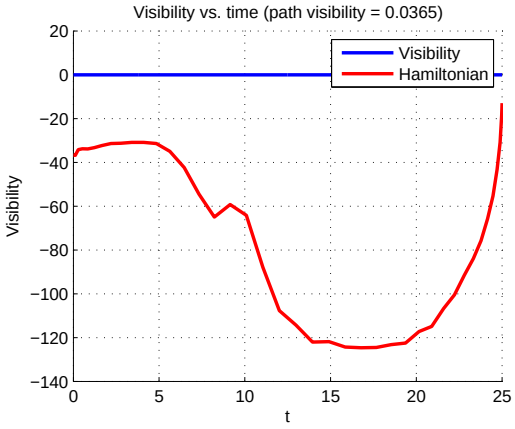
(a) x versus y, 1 obstacle



(b) Visibility versus t, 1 obstacle



(c) x versus y, 2 obstacles



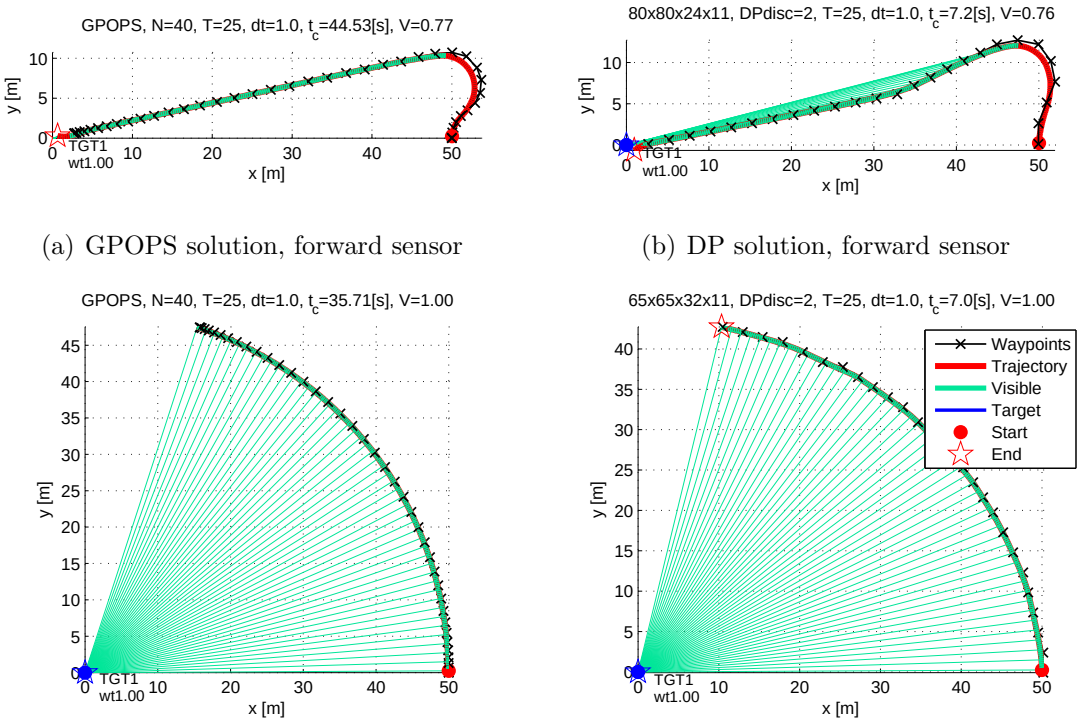
(d) Visibility versus t, 2 obstacles

Figure 3-9: GPOPS solutions for forward-facing sensor with obstructions; note GPOPS appears to be failing in the first **VMP** scenario, and has failed in the second **VMP** scenario.

Table 3.1[5] shows the **VMDP** solution is calculated faster, over an order or magnitude more quickly for more complex scenarios. Because of approximation error, there is an optimality gap present in the DP solutions. The optimality gap can be reduced by increasing the DP discretization resolution. The effects of DP discretization resolution are studied in Section 3.3.4.

Tables 3.2 and 3.3 show 200 total trials with random initial conditions, target locations, and obstacle fields in a 2-D environment. The initial condition is drawn in the upper right of the environment, the target in the lower left, and obstacles in

[5]In the table, DP resolution refers to $(n_{i_x}, n_{i_y}, n_{i_\psi}, n_{i_u}, N_{DP})$, and GPOPS resolution to (N_{PS} pseudospectral points)



(a) GPOPS solution, forward sensor

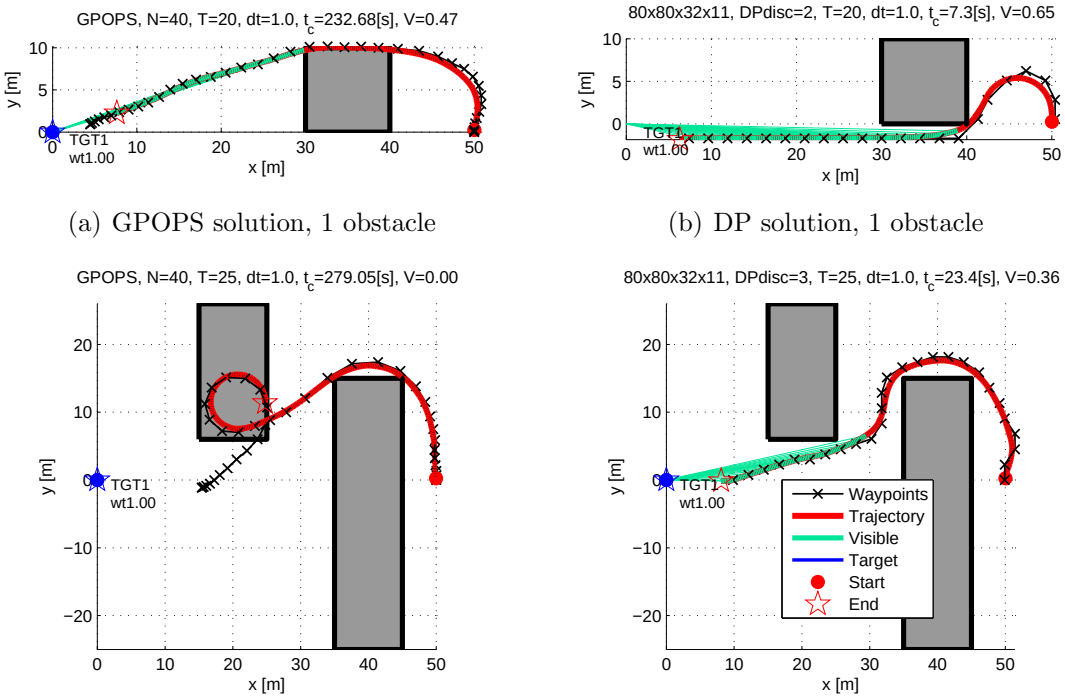
(b) DP solution, forward sensor

(c) GPOPS solution, left sensor

(d) DP solution, left sensor

Figure 3-10: GPOPS versus DP comparison without obstacles.

the center. The planning horizon is $T = 60$ [s]. GPOPS is given an initial guess of $[\mathbf{x}_A(0), \mathbf{x}_T(T)]$ each trial. These tables show that GPOPS performs poorly compared to DP in these constrained environments. GPOPS requires increasing amounts of computation as environments become more complex due to the number of constraints checked. DP, on the other hand, remains almost constant because this computation is performed once only over the entire discrete graph. Performance also decreases as the number of obstacles increases, as expected. While some GPOPS paths are feasible, these are mostly suboptimal because GPOPS terminates prematurely, often due to infeasibility. DP, on the other hand, finds solutions for almost every scenario. The few DP failures are caused by the initial condition being too close to an obstacle. In these cases it was impossible to avoid the obstacle due to the speed and turn rate constraints of the observer. The rightmost column in Table 3.3 also highlights the discrepancy between the discrete path returned during the planning phase, and the continuous approximation generated after running the dynamic simulation of the



(a) GPOPS solution, 1 obstacle

(b) DP solution, 1 obstacle

(c) Incomplete GPOPS solution, 2 obstacles

(d) DP solution, 2 obstacles

Figure 3-11: GPOPS versus DP comparison with obstacles.

observer. This discrepancy leads on average to 2 to 10 percent underperformance.

Most importantly at this point, GPOPS was unable to find adequate solutions, including many infeasible solutions, and therefore must be considered inadequate for the **VMP**. One cause is that GPOPS depends heavily on the initial guess. When the guess is not close enough to the solution, it will lead to suboptimal and infeasible solutions; and the solution obviously is not known in advance. The sensitivity increases as scenarios become increasingly complex, such as multiple obstacle environments. The initial guess sensitivity and other issues with the GPM were also observed in [70].

3.3.4 Performance and Computation Versus Resolution

This section discusses the effect of varying resolution on performance and computation time. Increasing resolution generally improves performance and increases computation time. The tricky aspect of discretization is that it may not necessarily be complete [15], meaning a feasible solution might not be found because discretization

Table 3.1: Computation time and performance for DP versus GPOPS.

Scenario	Solver	Resolution	Computation [s]	Visibility
No obstacles, forward	DP	(60,60,24,11,2)	4.3	0.72
	DP	(80,80,24,11,2)	7.2	0.76
	GPOPS	(40)	44.5	0.77
No obstacles, left	DP	(40,40,32,11,2)	3.2	1.00
	DP	(65,65,32,11,2)	7.0	1.00
	GPOPS	(40)	35.7	1.00
1 obstacle	DP	(80,80,32,11,2)	7.3	0.65
	GPOPS	(40)	232.7	0.47
2 obstacles	DP	(80,80,32,11,2)	23.4	0.36
	GPOPS	(40)	279.1	0.00

Table 3.2: Performance and computation (GPOPS) over 200 trials (50 trials per number of obstacles).

Scenario	Performance	Time [s]	% Feasible
1 obstacle	0.043 ± 0.027	47.8 ± 49.8	52
2 obstacles	0.029 ± 0.026	83.7 ± 82.7	30
3 obstacles	0.035 ± 0.025	97.8 ± 101.6	22
4 obstacles	0.033 ± 0.030	153.6 ± 158.1	12

Table 3.3: Performance and computation (DP) over 200 trials (50 trials per number of obstacles). Visibility table resolution is $n_x = 75, n_y = 75, n_\psi = 32$. DP resolution is 3 times finer in the x and y dimensions.

Scenario	Performance	Time	% Feasible	Error (actual vs. expected)
1 obstacle	0.49 ± 0.29	39.5	98	-0.009 ± 0.040
2 obstacles	0.42 ± 0.04	39.8	100	-0.023 ± 0.052
3 obstacles	0.42 ± 0.04	39.8	96	-0.013 ± 0.042
4 obstacles	0.34 ± 0.04	39.8	96	-0.022 ± 0.041

fails to capture any feasible transitions, or the transitions are suboptimal. Hence, resolution cannot be perfectly mapped to performance.

Figure 3-12 shows the effect of changing resolution on performance and computation time. Figure 3-12(a) shows a general performance-resolution correlation suggesting too low of a resolution impairs performance. Figure 3-12(b) verifies that increasing the resolution of one observer state affects computation linearly, and increasing it for two or three states has a multiplicative effect. A general rule-of-thumb is to choose a resolution that is sufficiently high and that does not significantly degrade computation. These results suggest that the problem can be solved multiple times using different resolutions and the best trajectory selected out of these trials.

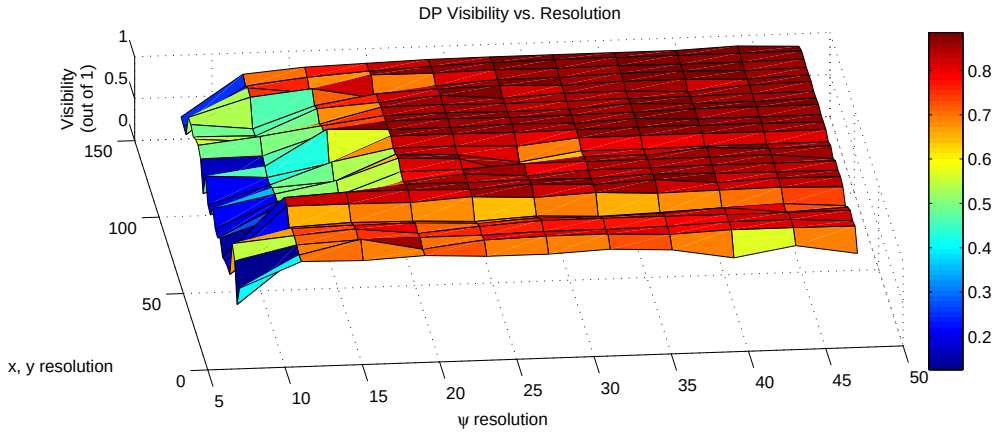
3.4 Results for A Single Stationary Target

This section presents results of paths generated by the **VMDP** for more complex environments, including cluttered 2-D environments, as well as 3-D and digital elevation models. These more complex scenarios represent real-world environments more accurately and assume a single stationary target in the environment.

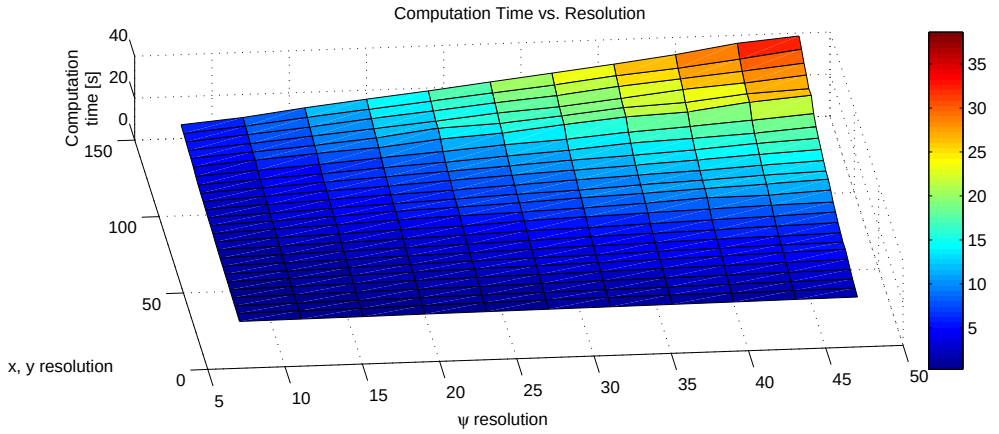
3.4.1 Complex Scenarios in 2-D Environments

This section presents numerical examples of single stationary target visibility maximization in complex 2-D environments. Figure 3-13 shows examples of complex 2-D environments. The resulting paths maneuver around obstacles and move into a loiter pattern around the target. The paths appear to maximize the time the sensor faces the target, in regions of the state space which are free of occlusions. Since the observer does not need to reach a goal state, the paths continue to loiter for as long as the planning horizon allows.

Mathematically, the loiter behavior appears to be an extremal behavior linked to the gradient of the visibility function. On short time scales, the observer moves in the direction which minimizes any decline in visibility: if the sensor model is smoothed using sigmoid functions, this direction vector will be tangent to the observer's present

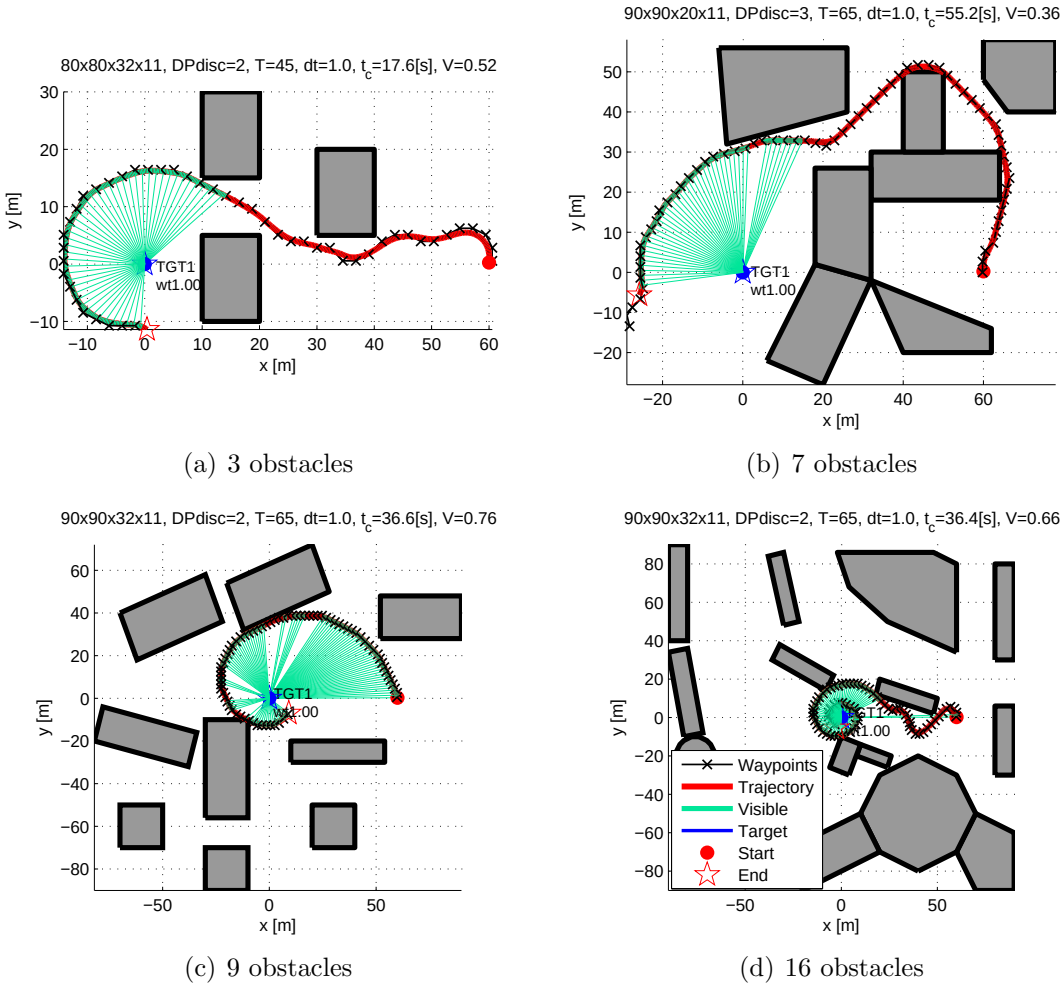


(a) Performance



(b) Computation

Figure 3-12: Effect of DP resolution on performance and computation. In (a), the DP appears to be sensitive to the heading discretization. Certain resolutions underperform due the resolution completeness sampling phenomenon. In (b), computation scales proportionally to the resolution of each state. Computation increases linearly with n_ψ and quadratically when n_x and n_y increase simultaneously.



(a) 3 obstacles

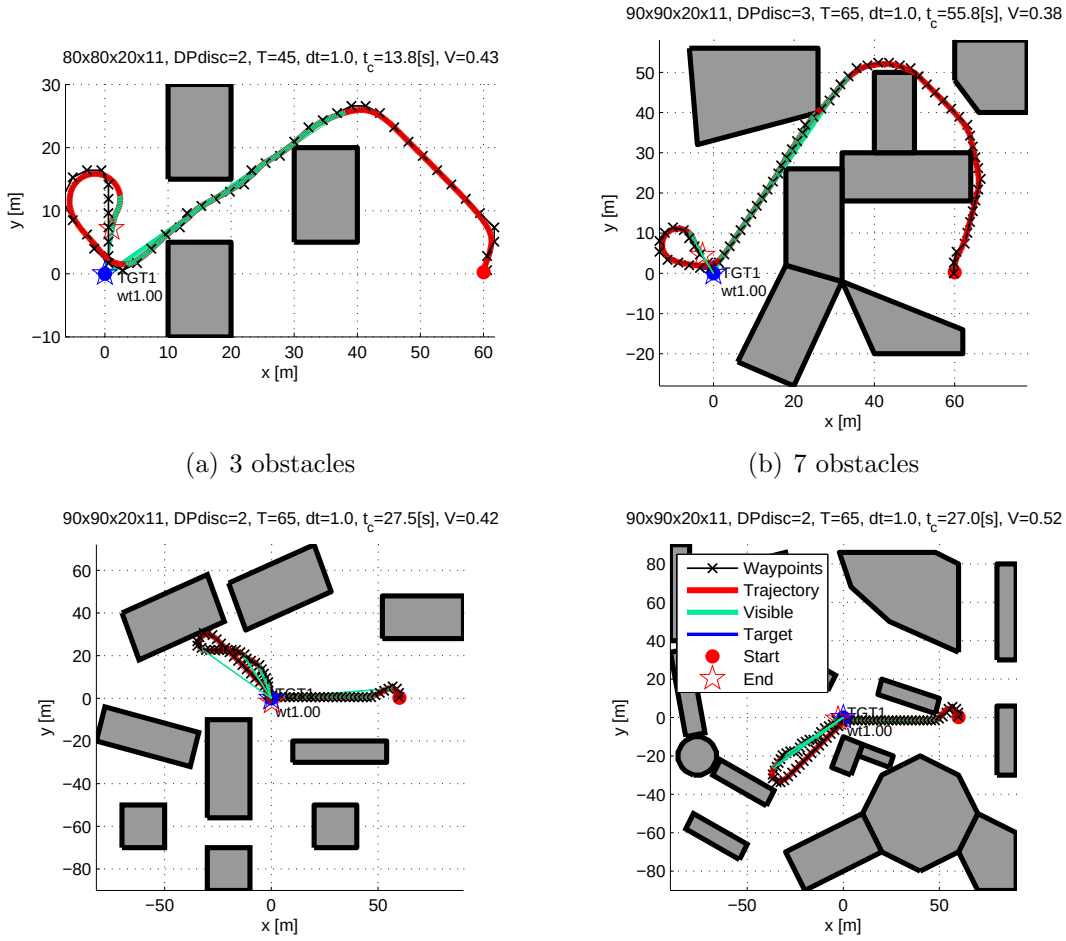
(b) 7 obstacles

(c) 9 obstacles

(d) 16 obstacles

Figure 3-13: Stationary target DP solutions in 2-D environments with left-facing sensor.

orbit around the target, or towards a different orbit depending on the reward scaling with distance to target. However, over longer time scales, because of occlusions and path constraints the solution can experience singular arcs. These arcs provide transitions from non-extremal states to extremal loiter sequences, or between two extremal loiter sequences. Figure 3-14 shows examples of forward-facing sensor paths. The resulting paths are multiple fly-bys, since the observer must maintain a constant speed. Each fly-by aims the vehicle (and the sensor) directly toward the target, then away from the target as quickly as possible to repeat another fly-by. The turns naturally satisfy turn rate constraints. For finite-horizon plans, the observer always ends very close to the target, since this approach maximizes the time the sensor sees



(a) 3 obstacles

(b) 7 obstacles

(c) 9 obstacles

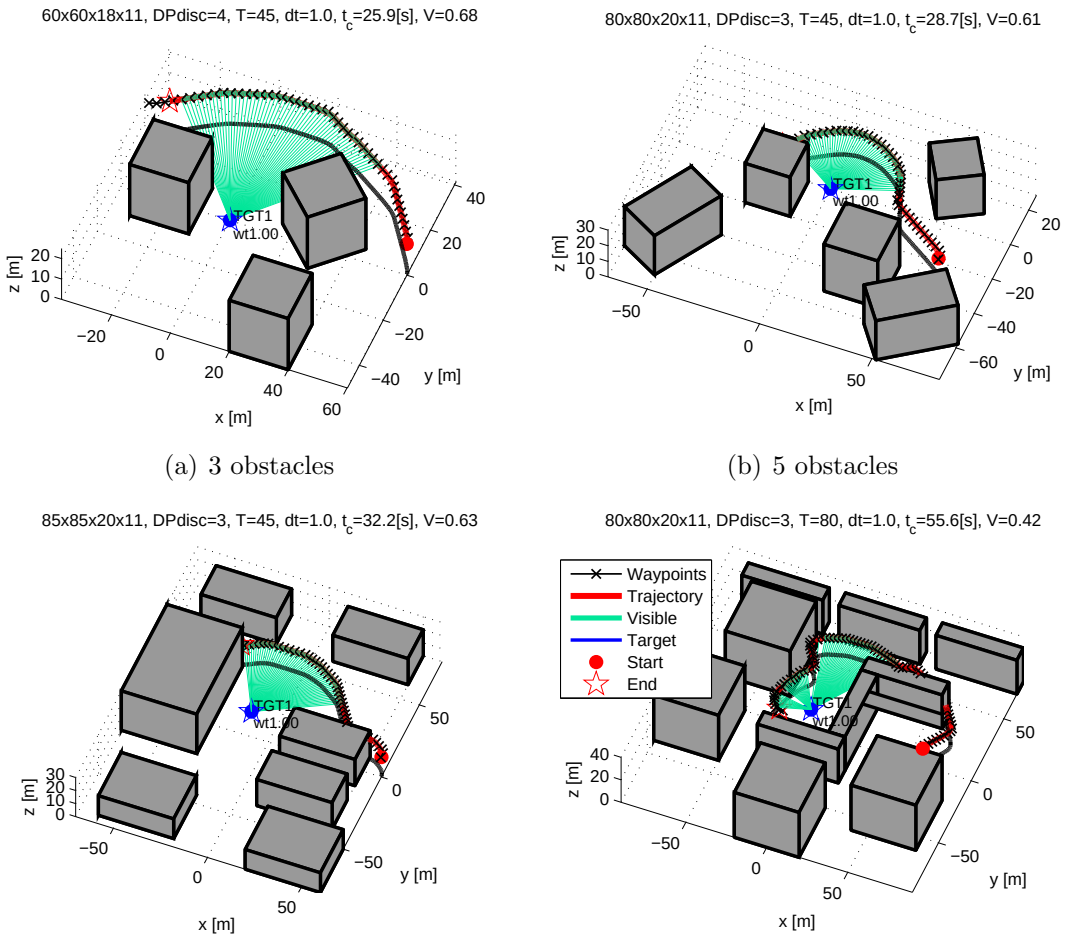
(d) 16 obstacles

Figure 3-14: Stationary target DP solutions in 2-D environments with forward-facing sensor.

the target.

3.4.2 Scenarios in 3-D and DEM Environments

This section shows numerical examples for complex 3-D environments. These cases represent an aerial observer equipped with a left-looking sensor with range and field of view constraints. The sensor is left-facing because this allows paths which orbit the target, leading to greater visibility fractions than a forward-facing sensor. The results are very similar to those on planar environments with the left-facing sensor. The observer enters into a loiter behavior about the stationary target while avoiding flight obstacles.



(a) 3 obstacles

(b) 5 obstacles

(c) 7 obstacles

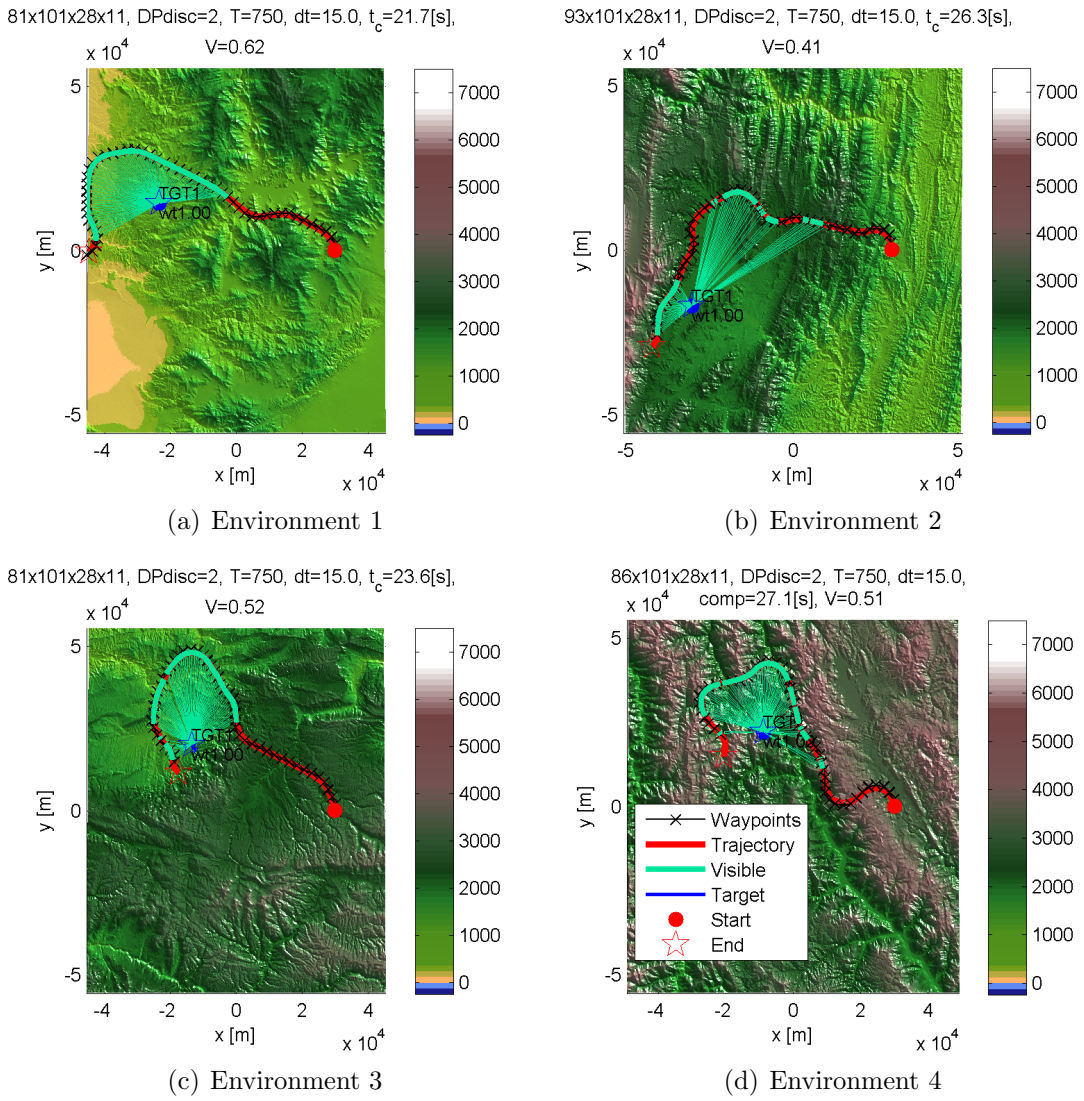
(d) 10 obstacles

Figure 3-15: Stationary target DP solutions in 3-D environments with left-facing sensor.

Figure 3-15 shows examples in 3-D urban environments. Each of these paths maximize visibility while performing obstacle avoidance. Figure 3-16 shows examples using digital elevation model environments. The observer navigates around flight obstacles and occlusions to orient the sensor towards the target. Again, the loiter behavior emerges.

3.5 Chapter Summary

This chapter presented the Visibility Maximization Dynamic Programming (**VMDP**) solver for the **VMP**. This chapter also presented a comparison with the General



(a) Environment 1

(b) Environment 2

(c) Environment 3

(d) Environment 4

Figure 3-16: Stationary target DP solutions in DEM environments with left-facing sensor

Pseudospectral Optimization Software (GPOPS), an optimal control solver that is able to solve elementary cases of the **VMP**. This chapter then provided examples of solutions generated by the **VMDP** for complex environments, including cluttered 2-D, 3-D, and digital elevation model scenarios. The next chapter discusses the multiple target formulation and the parametric **VMP** solver.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Multiple Targets and Parametric Optimization

This chapter discusses extensions of the **VMP** to multiple target visibility maximization. Section 4.1 presents formulations for target importance weighting and rare target visibility. It also discusses the effects of changing the trajectory design parameters such as the importance weights and the planning horizon on multiple target visibility. This chapter also introduces parametric optimization for finding closed contours. Section 4.2 describes how these contours are related to the steady-state trajectories from the non-parametric solution, and can be computed more quickly and are even desirable from a flight execution standpoint. This section also presents results comparing different parametric contours, parametric optimizers, and objective functions.

4.1 Multiple Target Formulation

While the previous chapter considered the single target visibility maximization problem, many real-life situations require that an observer maintain multiple targets in view. For example, a surveillance UAV may need to maximize visibility of two ships docked in separate regions of a harbor. This chapter considers the multiple target visibility maximization problem.

This section describes formulations for the multiple-target visibility maximization problem (MT-**VMP**). For multiple targets, where $N_T > 1$ denotes the number of targets, the *multi-target visibility metric* J_{VMT} can be posed several ways. These methods are presented below.

4.1.1 Weights and Weighted Sum of Per-Target Visibilities

The most basic multiple target formulation is to implement a *weighted (visibility) sum* $J_{VMT,WS}$, which simply weights the individual visibilities. Summing over the individual visibility metrics J_{V_m}, the weighted visibility metric takes the form

$$\begin{aligned} \max_{\mathbf{u}(t)} J_{VMT,WS} &= \sum_{m=1}^{N_T} (w_m J_{V_m}) \\ &= \sum_{m=1}^{N_T} \left(w_m \cdot \frac{1}{T} \int_0^T \mathcal{V}_m(t)\ dt \right) \end{aligned} \tag{4.1}$$

The weights w_m denote the relative importance of viewing target m. A higher weighting on a target places greater significance on the visibility reward from that particular target. The weightings can also be used to encourage harder-to-view targets to be visited by the observer, but there are no guarantees that every target will be visited. Visiting all targets cannot be guaranteed because it is not part of the constraints. On the other hand, if this constraint is specified, the problem might become infeasible.

The per-target weightings should satisfy

$$w_m \in [0,1] \ \text{ and } \ \sum_{m=1}^{N_T} w_m = 1,\ \forall m = 1, \ldots, N_T \tag{4.2}$$

to ensure the visibility metric J_{VMT} remains between $[0,1]$, to facilitate a fair comparison between trajectories.

4.1.2 Maximizing the Minimum Per-Target Visibility

While the weighted sum is a simple linear combination of per-target visibilities, it does not ensure that every target is visited. A nonlinear combination will not necessarily ensure every target is visited either. The choice of objective function needs to model the requirement of visiting every target at least once. To visit every target at least once, each target must be visible at some point given the constraints of the observer trajectory. If such a trajectory exists, one objective, the max-min formulation, assuming the optimization is able to find this trajectory, will automatically ensure that every target is visited. The set of visited targets will match the set of specified targets because the max-min formulation maximizes the minimum cumulative visibility of a target; if the minimum is greater than zero, this implies all targets will have a cumulative visibility greater than zero. However, a limitation exists in that if such a trajectory does not exist, or because the optimization fails to find it due to sampling incompleteness, then all feasible solutions become degenerate, having the same cost. When this happens, the target that cannot be seen should be removed from the optimization. Given this minor limitation, the *worst individual-target cumulative visibility* is formulated as:

$$J^*_{VMT,maxmin} = \max_{\mathbf{u}(t)} \left\{ \min_{\forall m=1,\ldots,N_T} \left(w_m \cdot \frac{1}{T} \int_0^T \mathcal{V}_m(t)\ dt \right) \right\} \tag{4.3}$$

This objective function considers the worst visibility, and not the visibility of the other targets. This function does not seek to maximize the remaining visibility as long as they exceed the worst visibility. In this sense, the objective function will under-perform if the desire is to simultaneously maximize visibility of all targets.

The DP framework, however, does not permit the max-min formulation. Consider the following example, without loss of generality: a graph has three nodes (start S, intermediate I, and goal G), and there are two transitions from start S to intermediate I with rewards (0.6,0.6) and (0.4,0.8), where $(J_1[k=1], J_2[k=1])$ are the rewards for viewing two targets. The one-step maximization would declare the transition (0.6,0.6) optimal, since it maximized the minimum of the per-target visibilities after

Table 4.1: Decision matrix for DP in max-min formulation. If one chooses (0.6,0.6) from S to I because it has a higher minimum after step 1, the result after step 2 will be worse than if one chose (0.4,0.8) instead. This example demonstrates the principle of optimality does not hold for the max-min formulation.

S to I	$\min[k=1]$	I to G	$J[k=2]$	$\min[k=2]$
$J_1[k=1](0.6, 0.6)$	0.6	$J_1[k=2](0.8, 0.4)$	(1.2,1.0)	1.0
$J_2[k=1](0.4, 0.8)$	0.4	$J_1[k=2](0.8, 0.4)$	(1.2,1.2)	1.2

the first step. However, say the final edge from I to goal G is $J_1[k = 2](0.8, 0.4)$. The result of taking the first choice is $(1.2, 1.0)$ while the second is $(1.2, 1.2)$. Clearly taking the first option reduces possible reward in the future. This example shows the memory effect which holds. If one were to apply the principle of optimality to the max-min objective, the result will be suboptimal. Despite this, the minimum will still be greater than zero, but it will not necessarily be the maximum achievable score. Table 4.1 summarizes the decision combinations and rewards of the above example.

4.1.3 Diminishing Returns on Per-Target Visibility

A different method which motivates paths that visit as many targets as possible is to use the notion of diminishing returns. Diminishing returns, in the context of economics, states that while additional effort can provide added reward, the marginal returns resulting from increasing effort becomes smaller and smaller, which suggests it makes more sense to allocate this effort elsewhere. In the context of multiple target visibility, the available effort (the observer's control actions) should be spent visiting all targets during a mission. And while this behavior is already seen in the max-min formulation, diminishing returns can be superior because it considers the marginal benefit of added visibility, whereas max-min is only concerned with maximizing the worst visibility. The *diminishing returns* metric $J_{VMT,dim}$ accomplishes the distribution of observer effort across multiple targets through the re-mapping of

the cumulative per-target visibilities:

$$\max_{\mathbf{u}(t)} J_{VMT,dim} = \sum_{m=1}^{N_T} 2w_m \left\{ \left[1 + \exp\left(-M \cdot \frac{1}{T} \int_0^T \mathcal{V}_m \, dt \right) \right]^{-1} - \frac{1}{2} \right\} \tag{4.4}$$

The sigmoid function $[1 + \exp(-MJ_{V_m})]^{-1}$ maps the cumulative visibility of a target, which corresponds to the amount of effort spent by the observer seeing the target, to a marginal or diminishing returns reward. The derivative of the sigmoid, representing the change in reward as a result of increased effort, is greatest when the cumulative visibility is zero. The tuning parameter $M > 0$ can be increased to raise the slope at zero cumulative visibility and emphasize visibility across many targets, or decreased to lower the slope to re-approximate the weighted sum formulation.

By an argument similar to max-min, the principle of optimality does not hold for the diminishing returns objective.

4.1.4 Receding Horizon Approach for Complex Objectives

To use the max-min and diminishing returns objectives, a receding horizon exhaustive search, known as visibility maximization with receding horizon (**VMRH**) is proposed. The benefits of a receding horizon approach is that the planning horizon can be reduced, enabling a brute-force exhaustive search to remain tractable, and the range of cost functions that can be used is expanded. Importantly, the **VMRH** can be used with objective functions that do not satisfy the principle of optimality. The receding horizon framework is stated in Algorithms 1 and 2. Algorithm 1 shows the receding horizon one-step (RHOS) planner. All plans $\mathcal{X}[k]$ for time step k, determined by executing $N_u[k]$ actions for each plan which then reach the set of states $\{X[k+1]\}$, are enumerated along with the costs $J_V[k+1]$ to form a finite reachability graph. The transitions $\{dX([k] \rightarrow [k+1])\}$ are constructed using Euler integration and is a function of the control action $u[k]$. The cost can encode a terminal state metric to alleviate the myopic nature of the receding horizon plan. While the optimal terminal penalty is the cost-to-go, this cannot be calculated since the terminal

Algorithm 1 Receding Horizon One-Step (RHOS): pseudocode for receding-horizon visibility maximization over a single time window.

INITIALIZE $k \leftarrow 1$, $X[k=1]$
while $k < N$ (receding horizon window) **do**
  EXPAND set of states $\{X[k+1]\} = \{X[k]\} + \{dX([k] \rightarrow [k+1])\}$
  EVALUATE $J_V[k+1]\left(\{X[k+1]\}\right)$
  $k \leftarrow k+1$
end while
return $\mathcal{X}^*[N] = \underset{\mathcal{X}[N]}{\arg\max}\{J_V[N]\left(\{X[N]\}\right)\}$

Algorithm 2 Receding Horizon Multiple Window (RHMW): pseudocode for receding-horizon visibility maximization over multiple windows.

INITIALIZE $X_{I,1}$
while $r < N_R$ (number of receding horizons) **do**
  $\mathcal{X}^*_{r+1} = \text{RHOS}(X_{I,r})$
  ASSEMBLE $\mathcal{X}_{1\rightarrow r+1}$
  SELECT $X_{I,r+1} = \mathcal{X}^*_{r+1}[K_{r+1}]$
  $r \leftarrow r+1$
end while
return $\mathcal{X}^*_{1\rightarrow N_R}$

state is unknown. Some possible heuristics include a repulsive field to guide paths away from obstacles, and attractive fields towards targets. The best plan $\mathcal{X}^*[N]$ is returned by the RHOS planner. Algorithm 2 shows the receding horizon multiple window (RHMW) planner. Each window r requires a call to the RHOS planner to generate one-step optimal plan $\mathcal{X}^*_r$. Window r uses an intermediate state $X_{I,r}$ at time K_r along the one-step optimal path as its initial condition, and each $\mathcal{X}^*_r$ is assembled to the full receding horizon plan $\mathcal{X}^*_{1\rightarrow r}$. This process repeats $N_R - 1$ times, building the multi-window plan $\mathcal{X}^*_{1\rightarrow N_R}$ until the entire plan reaches the desired time horizon.

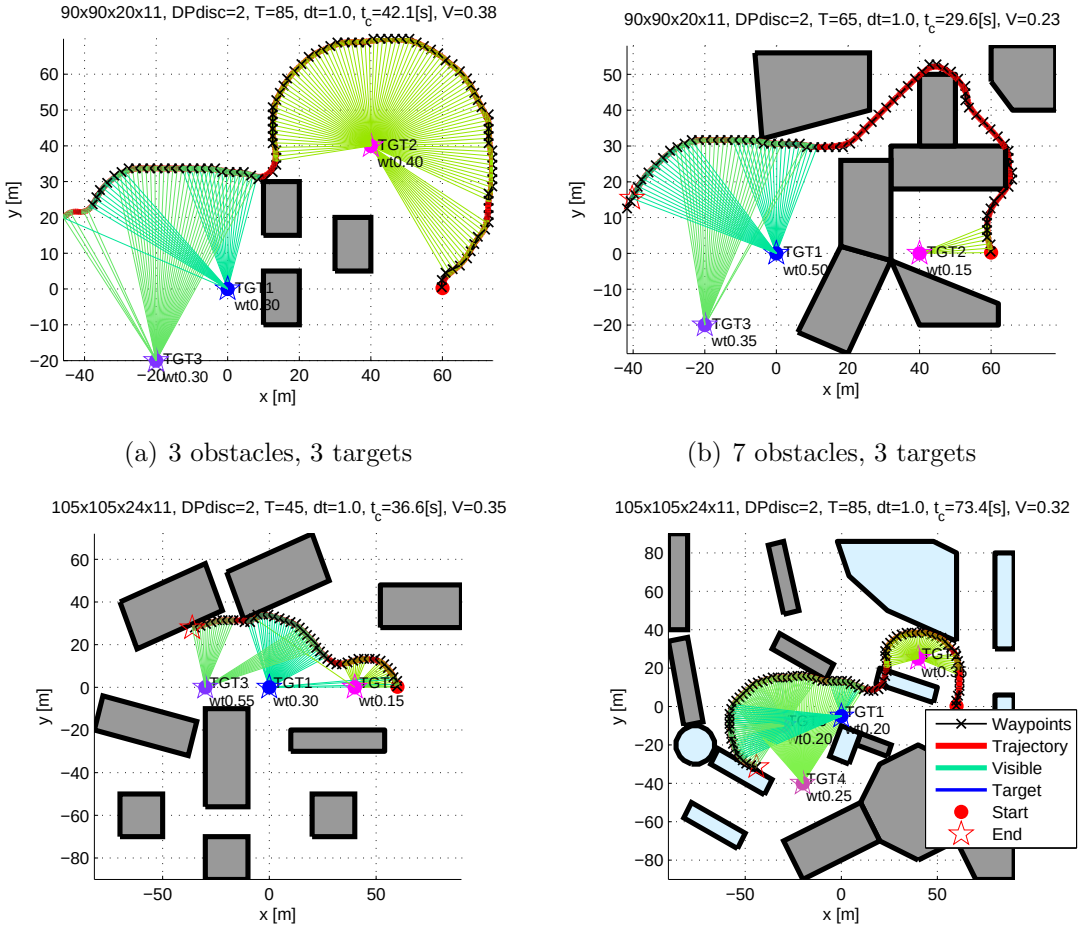
4.1.5 Multiple Target VMDP Numerical Results

This section presents results for multiple targets, in 2-D, 3-D, and DEM environments. Figure 4-1 shows results for multiple target scenarios in 2-D environments. Figure 4-2 shows results for multiple target scenarios in 3-D environments. Figure 4-3 shows results for multiple target scenarios using digital elevation model environments. The

observer behavior becomes complex in multiple stationary target scenarios. However, these trajectories can still be understood intuitively. All cases show the observer proceeding towards the most valuable target, or the most valuable cluster of targets. The observer will sometimes sacrifice high immediate visibility for visibility in the future horizon, when it is beneficial to do so. The importance weighting affects this behavior, so targets that are farther from the observer's initial condition but have sufficiently high weightings will still be visible along the observer's trajectory. Since the sensor is left-facing, the trajectories resemble loiter patterns similar to the patterns for individual target observation. The loiter patterns appear to conform around individual targets for parts of the trajectory to maximize the view of that particular target, but when targets are close enough together the loiter patterns will encompass the entire group of targets. Obstacles will occasionally affect the desirability of visiting particular targets. The desirability is reduced because occlusions remove portions of the observer's traversable space that provide line-of-sight visibility. Therefore, less time can be spent viewing these targets, making them undesirable to visit from a visibility maximization point of view. Visibility occlusions are less restricting: even though they obstruct visibility, the observer will still travel through no visibility occlusions to reach high visibility areas more quickly.

Computation time increases are the result of a number of factors, such as the number of obstacle edges and surfaces, the DP resolution, the planning horizon selection, and the number of targets in the environment. In general, 3-D environments with the same number of surfaces as 2-D edges will be more complex. The DEM models have the most edges and are therefore the most complex to solve. Another driving factor is the DP resolution, as shown earlier in Section 3.3.4.

When the time horizon and weights change, different behaviors emerge. These parameters can be chosen as needed. The next section describes the effects of changing the time horizon and weighting parameters.



(a) 3 obstacles, 3 targets

(b) 7 obstacles, 3 targets

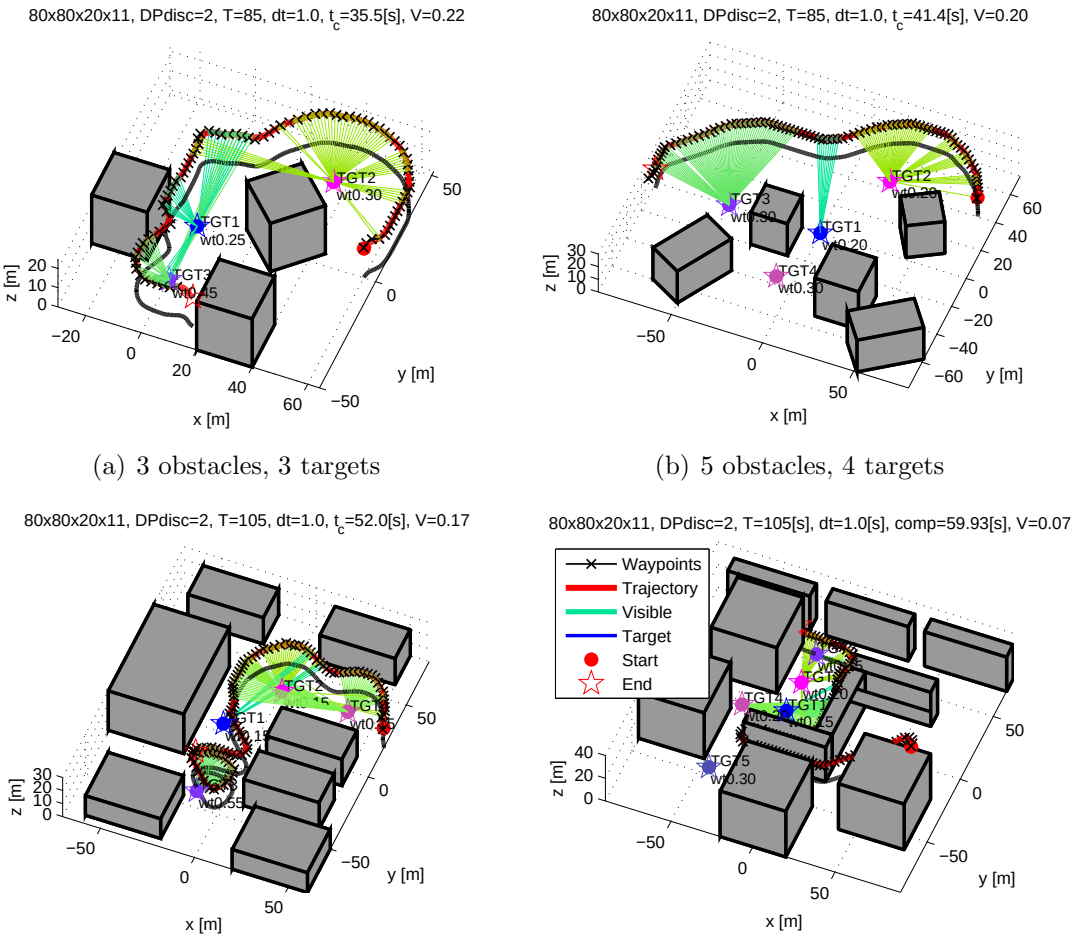
(c) 9 obstacles, 3 targets

(d) 16 obstacles (8 occlusions), 4 targets

Figure 4-1: DP solutions in 2-D environments, multiple stationary targets, left-facing sensor. In (d), the lightly-shaded obstacles are visibility occlusions only, and do not obstruct entry by the observer.

Varying Target Weights

Figure 4-4 shows the effect of varying the weights for a two target scenario. The weights represent the relative importance of viewing a target, and changing the weights can cause the observer behavior to switch from initially visiting only one target to visiting both targets. Also, the amount of time that a target is seen can be changed by increasing or decreasing the weights. Due to the number of variables of any particular problem, such as the initial conditions for the observer, the targets, the sensor, and the environment geometry, the effect of the weighting is not apparent. Also, while weights affect the resulting observer trajectory, the weights are not



(a) 3 obstacles, 3 targets

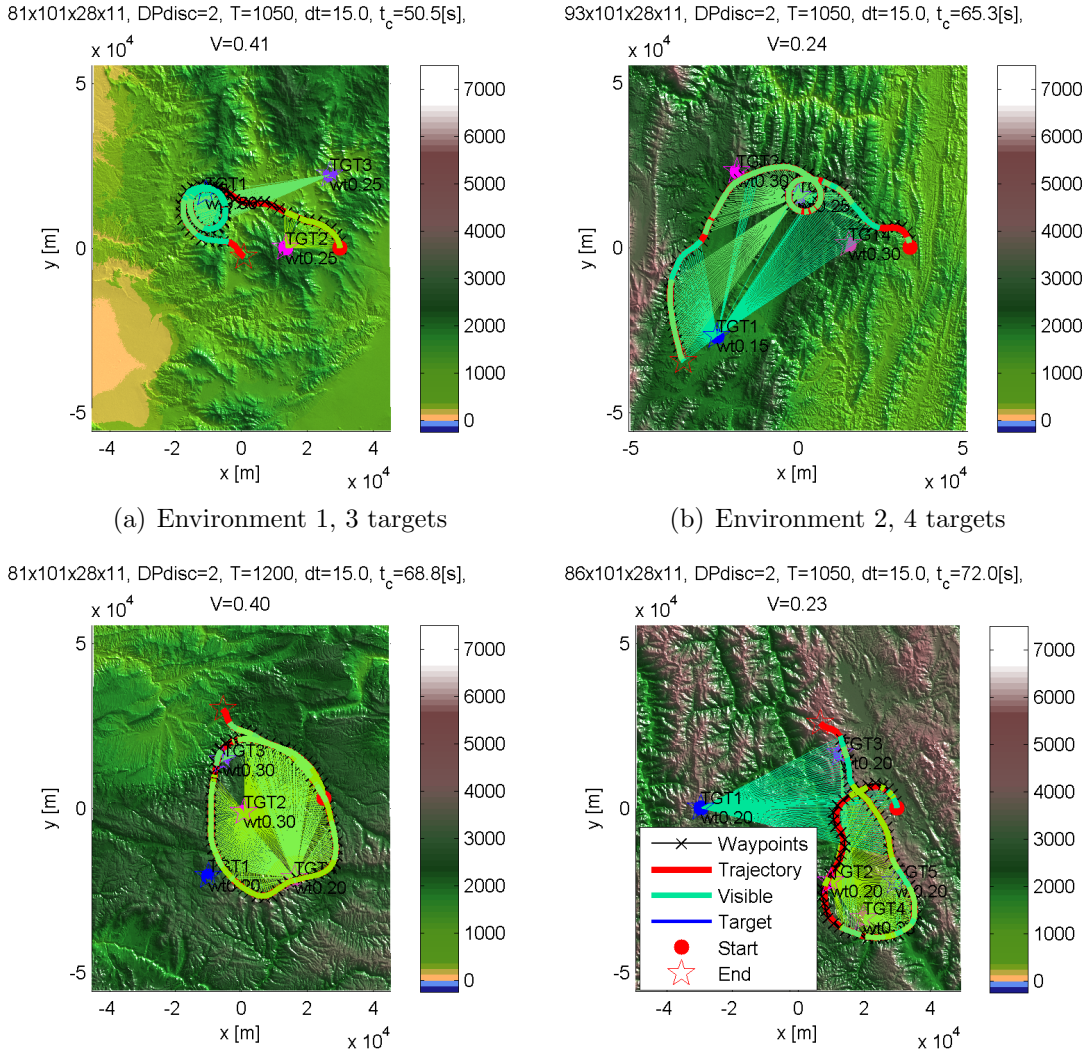
(b) 5 obstacles, 4 targets

(c) 7 obstacles, 4 targets

(d) 10 obstacles, 5 targets

Figure 4-2: DP solutions in 3-D environments, multiple stationary targets, left-facing sensor.

easily specified from the outset to achieve any particular behavior. From the example in Figure 4-4, several switching behaviors occur once particular weight ratios are reached. These switches can be identified in the figure when the visibilities of the targets jump discontinuously. The changes coincide with when the global optimum of the problem switches. The optimum switches because a previously suboptimal local optimum increases beyond the value of the prior optimum, once the weights have changed sufficiently. The presence of multiple switches reflects the potential existence of multiple local optimal in the observer action space. There also appear to be regions where the visibility of each target changes in a continuous fashion. These continuous changes correspond to the region around the global optimum growing, shrinking, or



(a) Environment 1, 3 targets

(b) Environment 2, 4 targets

(c) Environment 3, 4 targets

(d) Environment 4, 5 targets

Figure 4-3: DP solutions in DEM environments, multiple stationary targets, left-facing sensor.

moving at the same time, but remains more valuable than other local optima.

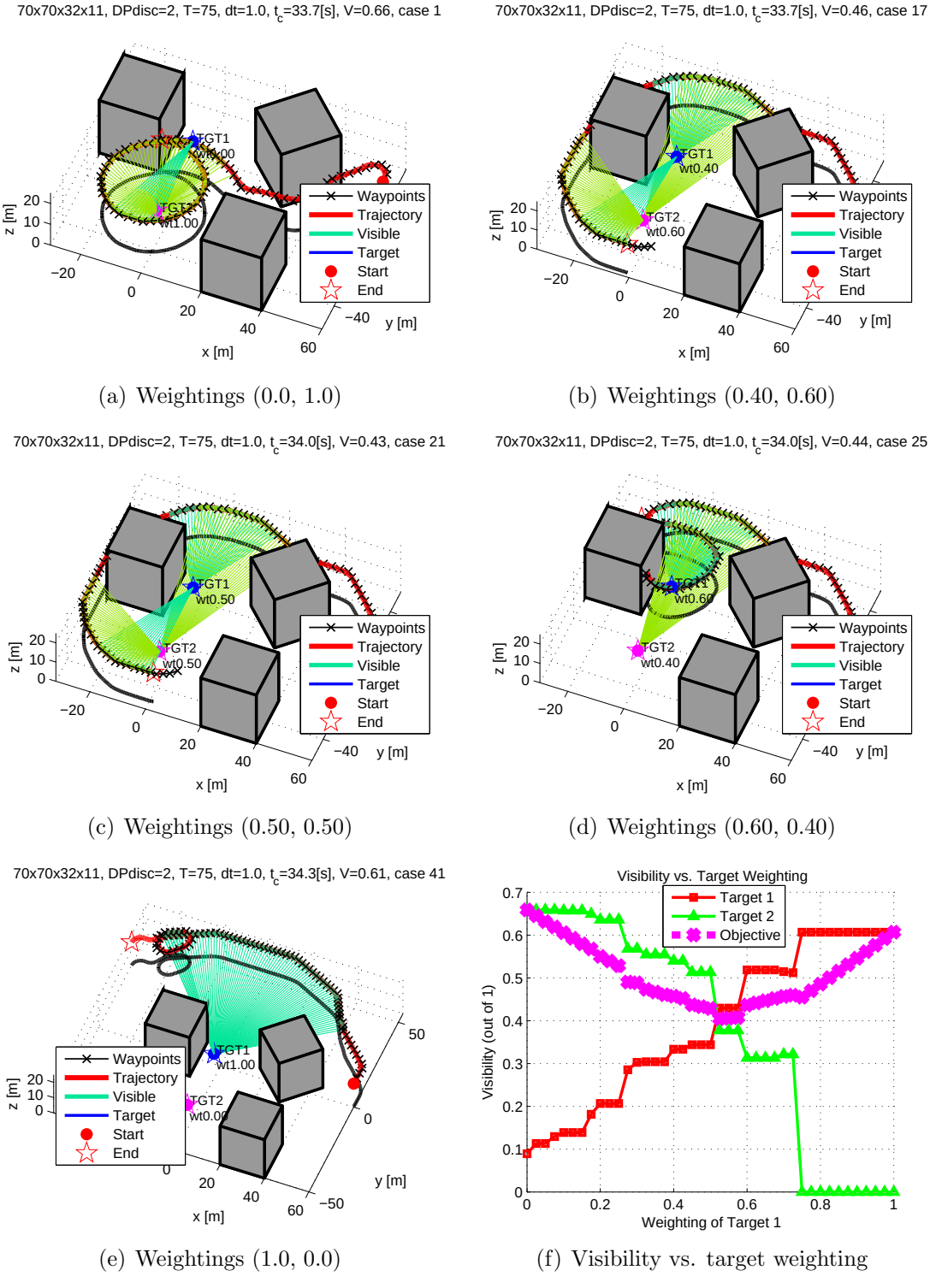
Varying Planning Horizon

Figure 4-5 shows the effect of varying the planning horizon. Time horizons can be specified if the exact execution time is known, or if computation time and memory need to be kept within limits. However, changing the planning horizon gives rise to different behaviors. The figure shows planning horizons of 40, 80, 120, 160, and 200 [s] in a 2-D environment with four targets. Short planning horizons yield paths which attempt to gain as much visibility in the limited near-term as possible, which is seen as a myopic and greedy behavior. Long planning horizons on the other hand exhibit periodicity or steady-state behavior: these plans repeat segments of the trajectory that are clearly favorable to the objective function, such as loitering around groups of targets with the highest visibility reward. Intermediate-length plans characterize the transition between myopic and steady state behavior. In this example, the transition phase emerges between time horizons of 80 to 180 [s]. During the transition, the trajectories may evolve to possess completely different time sequence behaviors. Also, in this example, at 200 [s] the path appears to repeat a large portion of its initial trajectory, signifying the onset of steady state paths.

The closed contours which have emerged as a result of increasing the planning horizon suggests a new type of optimization can be considered. The next section describes parametric optimization for visibility maximization, which models simple shapes with relatively few parameters that can be quickly optimized.

4.2 Comparison Against Baseline Parametric Paths

While GPOPS was unable to solve somewhat hard instances of the **VMP**, parametric optimization is capable of solving the **VMP**. Parametric optimization finds near-optimal solutions under restrictions on the types of paths that can be optimized. This class of paths includes parametric paths which have few parameters defining the path. Parametric paths can be closed contours which can be readily represented



(a) Weightings (0.0, 1.0)

(b) Weightings (0.40, 0.60)

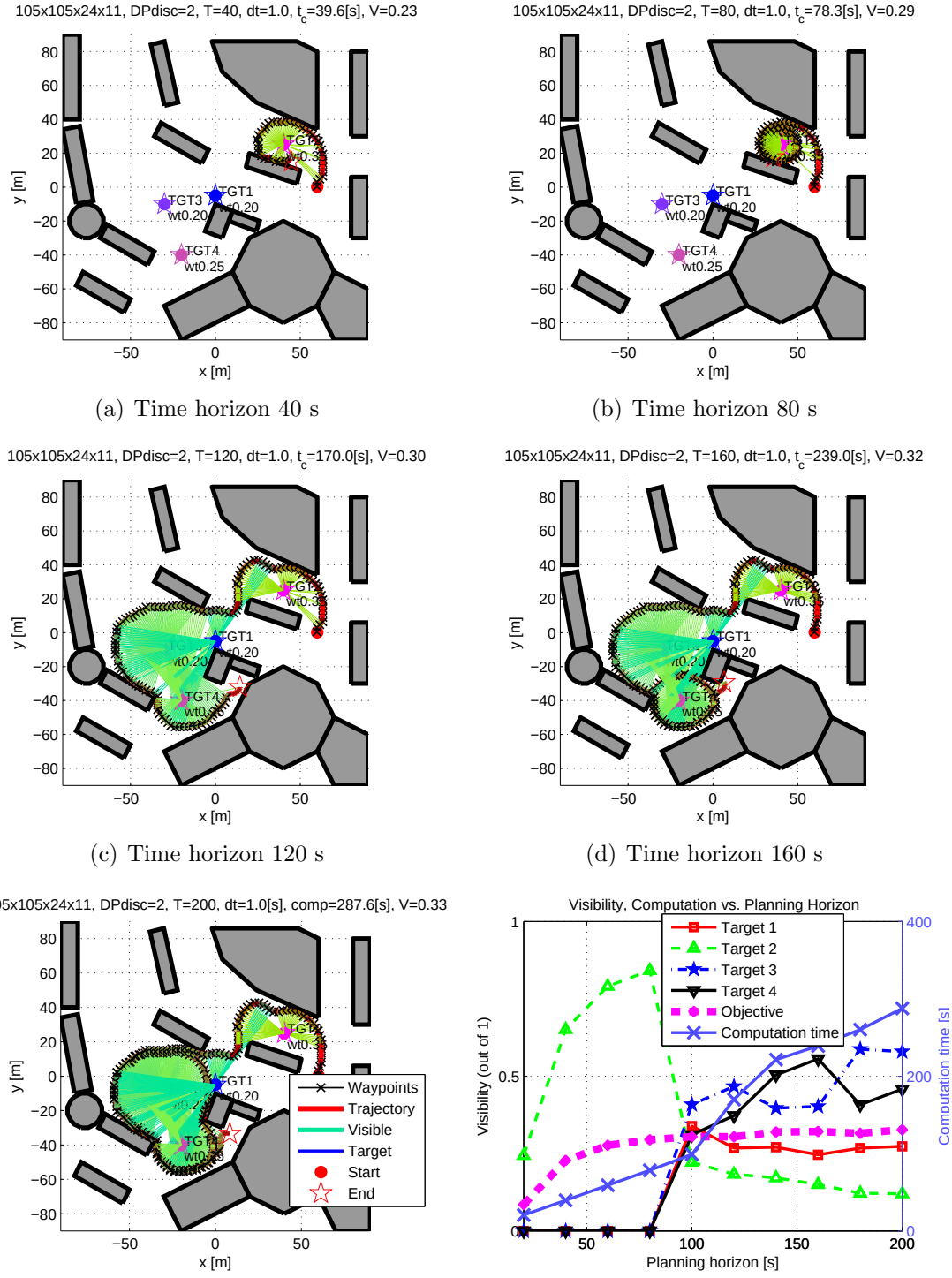
(c) Weightings (0.50, 0.50)

(d) Weightings (0.60, 0.40)

(e) Weightings (1.0, 0.0)

(f) Visibility vs. target weighting

Figure 4-4: DP solutions, comparing target weightings for 3-D environment.



(a) Time horizon 40 s

(b) Time horizon 80 s

(c) Time horizon 120 s

(d) Time horizon 160 s

(e) Time horizon 200 s

(f) Visibility, computation vs. planning horizon

Figure 4-5: DP solutions, comparing planning horizons for 2-D environment.

Table 4.2: Parameters for parametric paths. Note these parametric descriptions are not unique for these paths.

Path Type	Parameters	Description
Circle	$(x_c, y_c), r$	Center, radius
Ellipse	$(x_c, y_c), a, b, \phi$	Center, semi-major/minor axes, orientation
Line Segment	$(x_c, y_c), L, \phi$	Center, length, orientation
Racetrack	$(x_c, y_c), r, L, \phi$	Center, radius, length, orientation

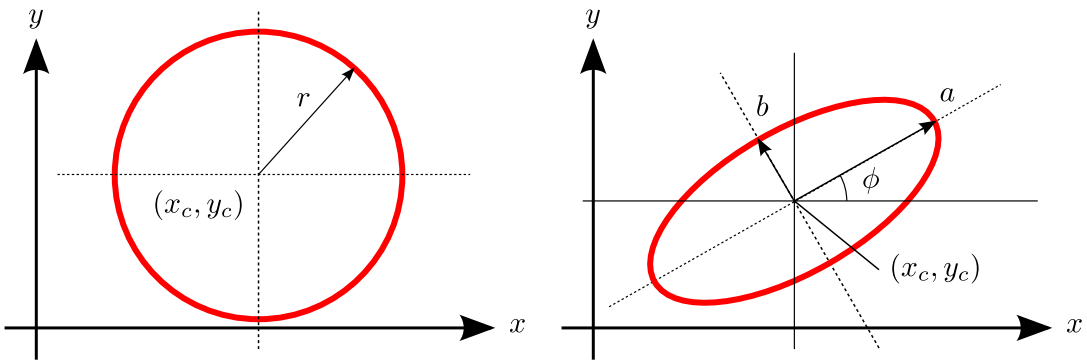
in closed form. Parametric paths typically have simple geometries that are suitable for flying, and are designed from the outset to be dynamically feasible. However, a disadvantage is that the resulting path generated by parametric optimization can only be optimal in the same family of contours.

This section describes parametric paths in more detail. Later, it provides comparison analyses amongst multiple solvers and the **VMDP** solver.

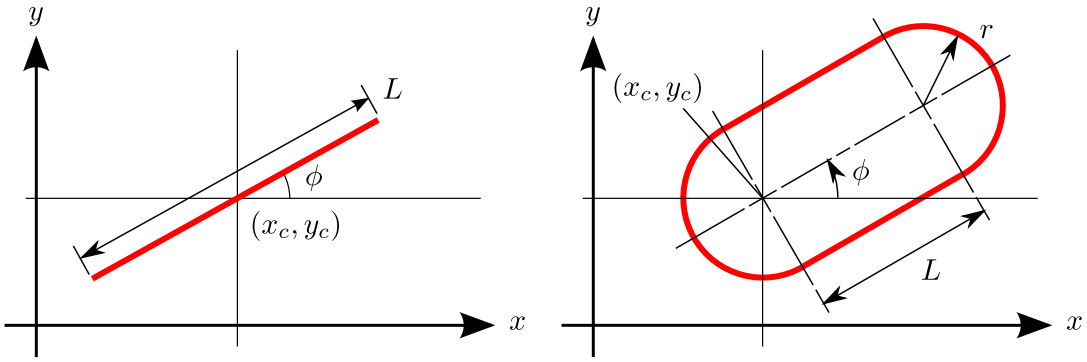
Parametric Paths

This section describes a new solution approach, and a baseline for validating the **VMDP** planner. Parametric visibility maximization or **VM-Par** finds parametric paths which maximize visibility. Figure 4-6 shows examples of parametric paths. Parametric paths include circles, ellipses, line segments, racetrack patterns, and arcs. These paths are fully described by a relatively small vector of parameters $\mathbf{x}$, such as the center and radius for a circle, or the start and end points for a line segment. Table 4.2 shows a list of parameters for the four parametric paths in Figure 4-6. In contrast, finding the optimal free-form path requires optimizing a parameter vector that is the size of the entire sequence of actions, which can have at least as many parameters as the number of time discretizations in a discrete approximation framework. Parametric optimization can be applied to the discrete action sequence using trajectory optimization as discussed in Section 3.2.2, but it is very difficult to obtain the optimal path in this manner without a good initial guess.

The parametric optimization formulation is posed as follows: the maximization of



(a) Circle (b) Ellipse



(c) Line segment (d) Racetrack

Figure 4-6: Parametric path shape and parameter examples.

visibility occurs with respect to $\mathbf{x}$ instead of $\mathbf{u}(t)$,

$$\max_{\mathbf{x}} J_V = \frac{1}{T}\int_0^T \mathcal{V}(\mathbf{x}_A(t), \mathbf{x}_T(t), \mathcal{T}, \mathcal{S})\ dt$$
$$\text{s.t. } \mathbf{x}_A(\theta) = A(\mathbf{x}, \theta), \quad \theta \in [0, 2\pi] \tag{4.5}$$
$$\text{(plus constraints in Equation 2.2)}$$

where θ denotes a *path parameter* (and not an angular state), and $A(\mathbf{x}, \theta)$ is a transformation matrix. It is possible to design classes of paths which subsume other classes, for example the *superellipse* [71] includes the circle, ellipse, two-way line segment, and racetrack. These "superclasses" typically have more parameters over which to optimize. If the initial condition is fixed, additional parameters may need to be specified. A full description of the path parameterization method is given in Appendix B.

Parameter-based optimization of a nonlinear, non-convex objective function such

Table 4.3: Parameters for cross entropy visibility maximization. Parameters correspond to ellipse optimization.

Variable	Symbol	Value
Sample min	$\mathbf{x}_{min}$	$[0.2W, 0.2H, 0.2\min(W,H), 0.0, 0.0]^T$
Sample max	$\mathbf{x}_{max}$	$[0.8W, 0.8H, \frac{\max(W,H)}{\sqrt{2}}, \frac{\max(W,H)}{\sqrt{2}}, \frac{\pi}{2}]^T$
Initial sample mean	$\mathbf{x}_\mu[k=1]$	$\frac{1}{2}(\mathbf{x}_{min} + \mathbf{x}_{max})$
Initial sample stdev.	$\mathbf{x}_\sigma[k=1]$	$\frac{1}{6}(\mathbf{x}_{max} - \mathbf{x}_{min})$
Number of samples	n	100
Update fraction	α	1.0
Best fraction	ρ	0.1
Converge tolerance	$\mathbf{x}_{\sigma,tol}$	$0.001\mathbf{x}_\sigma[k=1]$
Path discretization	N_θ	40
Execution time	$t_{\text{execution}}$	$0.5, 1.0, \cdots, 3$ [s]

as that of the **VMP** can be solved using the general class of algorithms known as metaheuristic searches. Metaheuristic search methods include simulated annealing (SA) [72], genetic algorithms (GA) [73], cross entropy (CE) [74], ant colony optimization (ACO) [75], and tabu search [76], each utilizing a unique set of heuristics to guide the improvement of initial guesses towards local or global optima. Appendix B.2 provides a more detailed description of these metaheuristic search methods.

4.2.1 Comparisons Between Parametric Optimizations

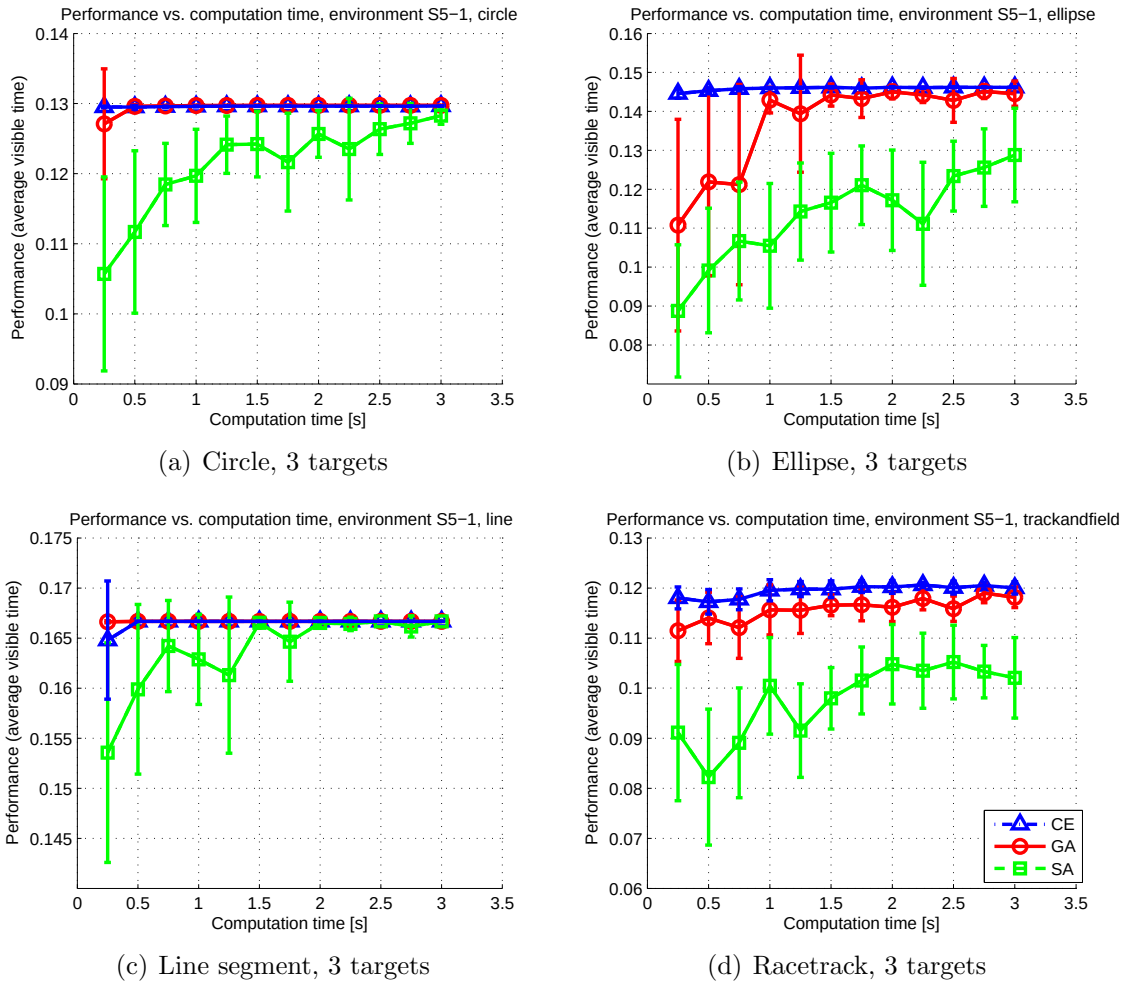
Figures 4-7 and 4-8 show comparison results between genetic algorithm, simulated annealing, and cross entropy parametric optimizers, using different parametric paths including circles, ellipses, one-way line segments, and racetracks. The MATLAB Global Optimization Toolbox Genetic Algorithm and Simulated Annealing implementations [77] were used, while a custom cross entropy implementation was developed, see Appendix B.3 for a thorough discussion on the method and implementation. The default values for the two MATLAB toolbox implementations were chosen. The computation time for every optimization trial is capped at increasing upper limits to ensure as fair a comparison as possible. 10 trials are run for each optimization, each time cap, and each parametric path type. Table 4.3 lists the parameters in Appendix B.3 selected for the cross entropy optimization for ellipses.

The results show that that cross entropy and genetic algorithms perform similarly, with the greatest average performance. Increasing computation time does not improve the results considerably, suggesting that both algorithms can consistently find a local optimum. Also, the performance scatter is small for both cross entropy and the genetic algorithm. The scatter is higher when computation time is low, as expected because the algorithm terminates before adequate improvement on the initial guesses have been made. On the other hand, simulated annealing performs poorly given the optimization time limit. Among the different path types, line segments and ellipses appear to perform the best. Note that the line segment is evaluated in one direction only, giving it higher scores. Ellipses provide additional degrees of freedom over the circle and racetrack, thus resulting in better performance. The next section shows the effect of relaxing the shape constraint to allow even more degrees of freedom to the path.

4.2.2 Comparisons Against Non-Parametric Optimization

This section describes comparisons between the non-parametric **VMDP** optimization with the parametric optimization. For this comparison, cross entropy is used to compute the performance baseline, using three different closed contours. Since *closed* means the start and end states are the same, to ensure a fair comparison between the parametric and DP paths, a terminal penalty on the DP solution is imposed so that the DP paths will also be closed. Also, similar to the parametric path design, the initial condition is no longer pre-specified. Figure 4-9 shows examples of closed DP contours under a terminal state penalty.

Figure 4-10 shows examples of maximum visibility paths in DEM environments for multiple stationary targets with equal weights, generated by the DP and CE optimization routines using the weighted sum metric. Table 4.4 shows this optimization performed over 400 sets of randomized target locations and 3 environments. As expected, the highest degree of freedom DP trajectory provides the highest visibility. In the first environment, the free DP on average performs 10% better than the circle, 6% over the ellipse, and 17% over the racetrack. The closed DP performs 2% better



(a) Circle, 3 targets

(b) Ellipse, 3 targets

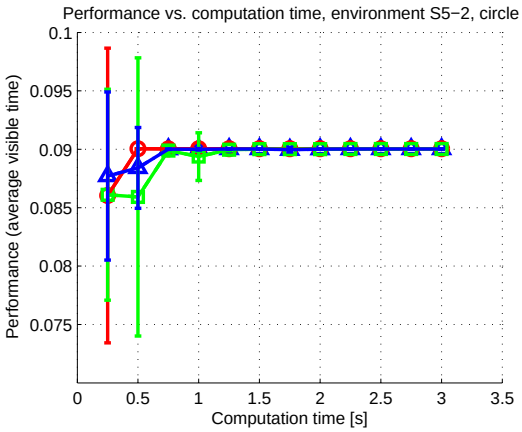
(c) Line segment, 3 targets

(d) Racetrack, 3 targets

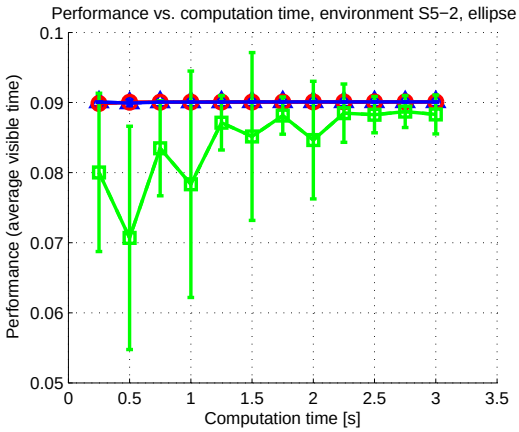
Figure 4-7: Comparison of parametric optimization routines for different curves (3 targets in DEM).

than the circle, 2% worse than the ellipse, and 8% over the racetrack. The results are similar in environments 2 and 3, except the closed DP path outperforms the ellipse on average. The ellipse shows similarities to the closed DP contour, and so it has similar performance. The circle and racetrack perform slightly worse than the other patterns because they are more constrained paths that cannot take advantage of the specific environment features.

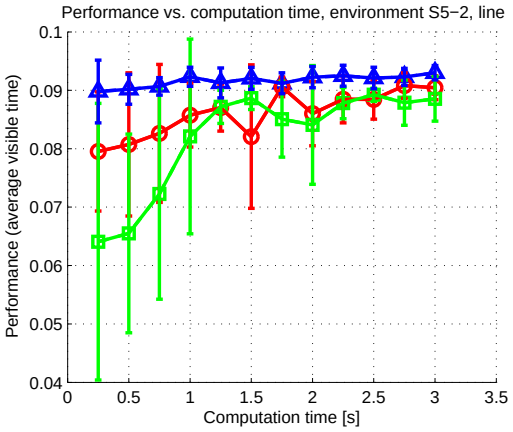
While the parametric paths underperform versus DP paths, one benefit is that the computation is faster. It is faster because there are fewer parameters to optimize. Another reason to consider parametric paths is the repetitive structure of long DP plans: as the planning horizon increases, these paths resemble closed contours. The



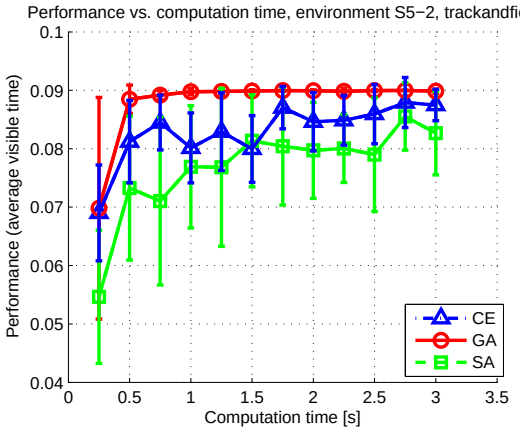
(a) Circle, 10 targets



(b) Ellipse, 10 targets

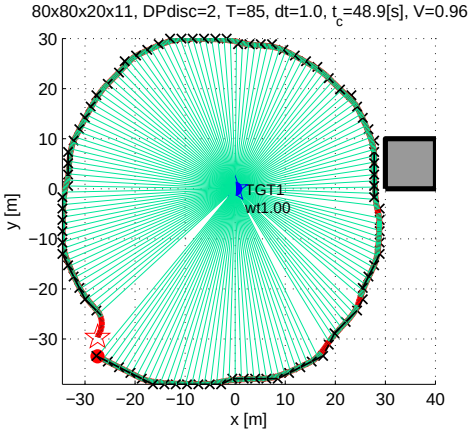


(c) Line segment, 10 targets

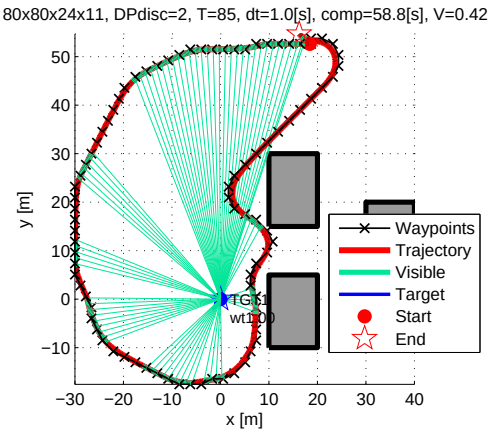


(d) Racetrack, 10 targets

Figure 4-8: Comparison of parametric optimization routines for different curves (10 targets in DEM).



(a) 1 obstacle, 1 target



(b) 3 obstacles, 1 target

Figure 4-9: Examples of terminal penalties imposed on DP paths for path closure with free initial condition.

Table 4.4: DP versus CE performance statistics over 400 trials, using the weighted sum metric.

	Performance		
Optimization	(Env't 1) (200 runs)	(Env't 2) (100 runs)	(Env't 3) (100 runs)
DP, open	0.512 ± 0.235	0.470 ± 0.141	0.485 ± 0.139
DP, closed	0.474 ± 0.238	0.453 ± 0.135	0.465 ± 0.135
CE, circle	0.464 ± 0.223	0.422 ± 0.161	0.443 ± 0.160
CE, ellipse	0.485 ± 0.232	0.443 ± 0.161	0.464 ± 0.160
CE, racetrack	0.438 ± 0.215	0.391 ± 0.157	0.422 ± 0.170

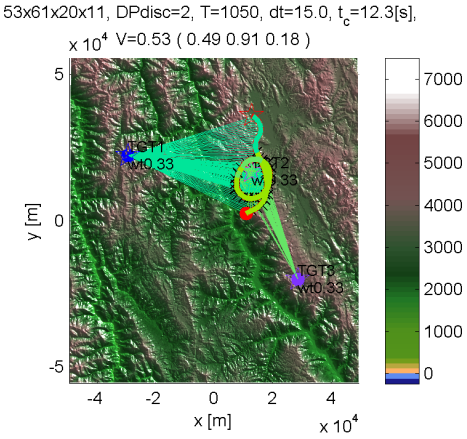
parametric paths provide a heuristic for designing these closed paths, and are useful if computation is limited. Also, note that the choice of terminal penalty should be such that it is sufficiently high so the penalty acts as a constraint.

4.2.3 Comparisons of Objective Functions

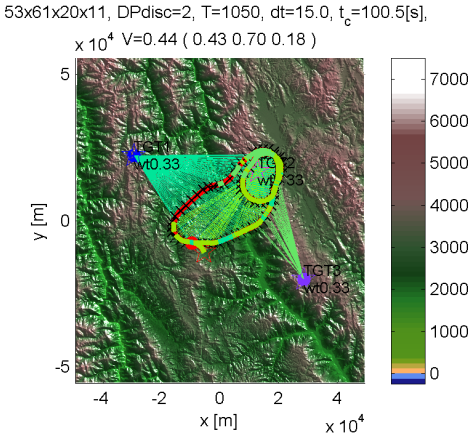
This section compares the weighted sum, max-min, and diminishing returns objectives in Eqs. 4.1, 4.3, and 4.4 for visibility maximization using the cross entropy solver. The purpose of the max-min and diminishing returns objectives is to elevate the importance of hard-to-see targets, so that every target can be seen in one pass of the observer's trajectory. One of the advantages of the parametric optimization is the ability to formulate objective functions which have the memory property, such as the diminishing returns objective.

Figure 4-11 shows the differences between the objective functions in DEM environments. Target 1 lies at the side of a mountain ridge and is therefore very difficult for the observer to see; target 2 is at the top of a mountain and is highly observable; and target 3 is moderately difficult to observe. Whereas the weighted sum metric does not make any attempt to ensure all targets are visible, both the max-min and diminishing returns metrics trade off substantial visibility of the easily-observed targets for marginal visibility of the most concealed target, by taking more sweeping and encompassing paths.

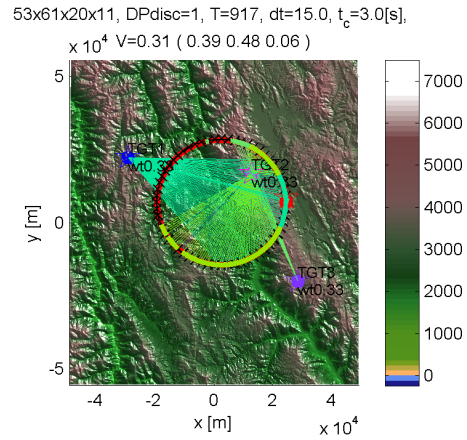
Figure 4-12 shows examples of the max-min and diminishing returns objectives using the receding horizon planner for a fixed initial condition. In the receding hori-



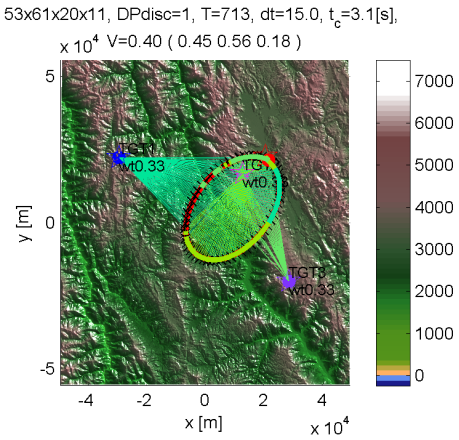
(a) DP, open



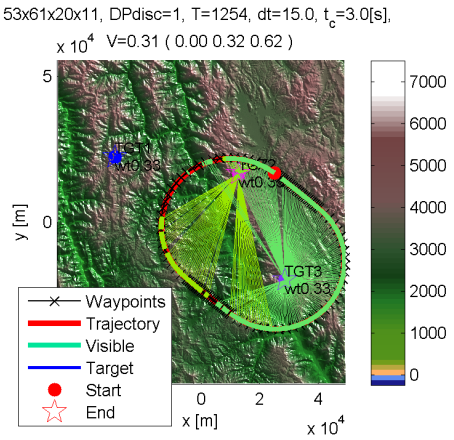
(b) DP, closed



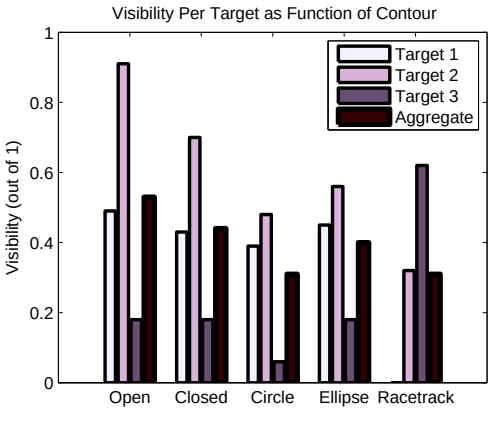
(c) CE, circle



(d) CE, ellipse

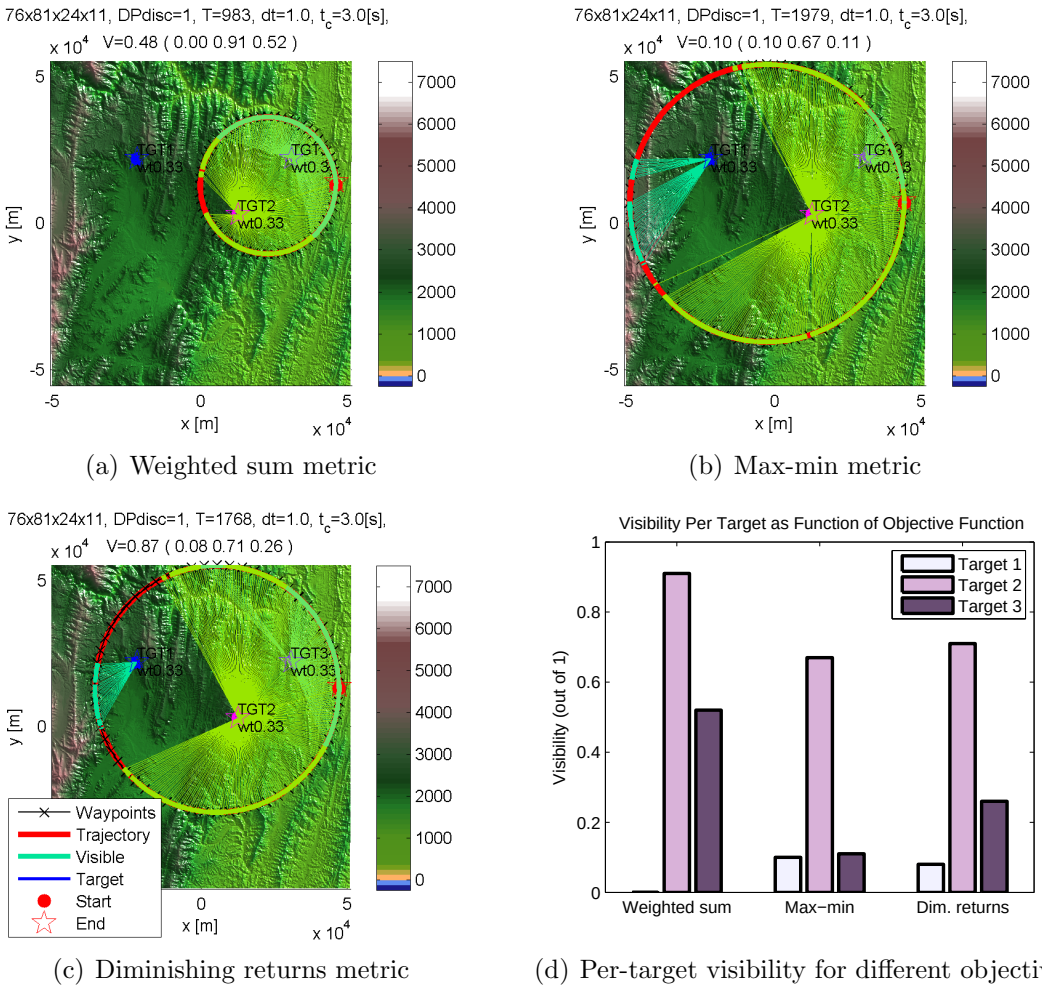


(e) CE, racetrack



(f) Per-target visibility for different curves

Figure 4-10: Multiple stationary target closed contour visibility using DP versus CE.



(a) Weighted sum metric

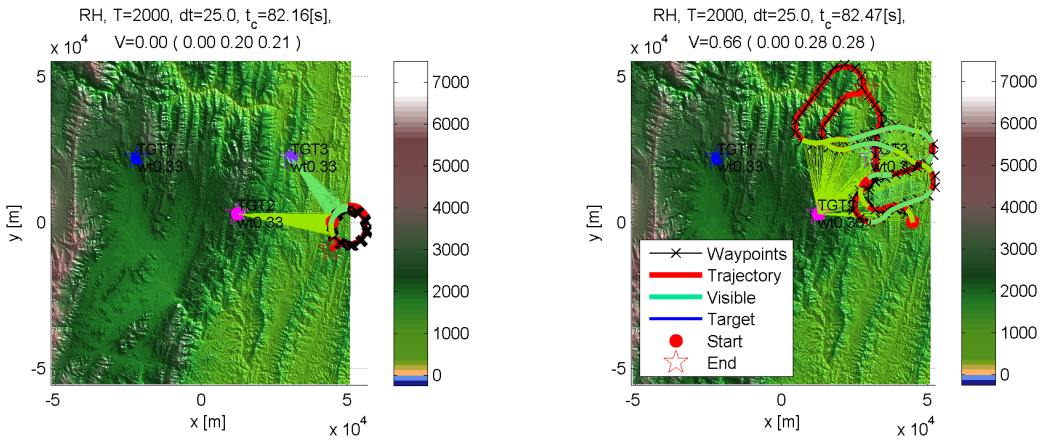
(b) Max-min metric

(c) Diminishing returns metric

(d) Per-target visibility for different objectives

Figure 4-11: Examples of per-target visibility for different objective functions; note that target 1 is very difficult to see.

zon framework, the terminal state cost encapsulates the anticipated cost left over the remaining time of the full plan. In the forward search visibility problem, the search space has dimensions in the number of time step discretizations and it is not possible to compute the cost exactly, and an approximation is hard to compute. One approximation is to project each branch forward maintaining the last heading and evaluating the visibility along this projected path, using the max-min or diminishing returns objective function. The results show that degeneracy in the max-min objective over a single time window (target 1 is not visible amongst any of the initial branches) results in indecision in the plan, because every path has a cost of zero even after applying the terminal cost. On the other hand, the diminishing returns objective still makes



(a) Max-min metric

(b) Diminishing returns metric

Figure 4-12: Receding horizon examples using max-min and diminishing returns metrics.

the best of the remainder of targets because it is not degenerate unless all targets are not visible during a receding horizon window, in which case the planner requires added guidance from the terminal cost to break the tie.

4.2.4 Effects of Visibility Approximation Error on Optimization

Figures 4-13–4-15 show three digital elevation model environments and Skymap / cross entropy optimization to find the best parametric path. For each environment, (a) shows an example path and (b) the difference in visibility between the best path computed using Skymaps and the online visibility computation. Figure 4-13(c) shows an example performance-computation plot, which is similar across environments. Four parametric flight paths (circles, ellipses, line segments, and arcs) are assessed. 100 random target locations are evaluated for each resolution and flight path combination.

Figure 4-13(c) shows that increasing the Skymap resolution reduces the optimality gap on average, and that online visibility calculation performs best on average as expected. The environment has little effect on the optimality gap. Consistent with Section 3.2.1, it takes more computation to reduce the error and optimality gap. Be-

cause the optimality gap shrinks asymptotically, a much lower resolution can provide the desired error tolerance.

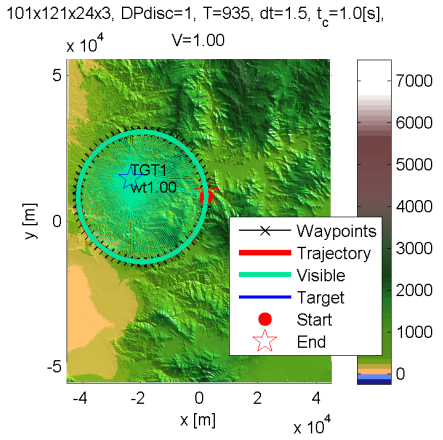
Figures 4-13(b), 4-14(b), and 4-15(b) show the performance difference with respect to the solution using online visibility computations. The performance difference $\Delta\mathcal{V}$ for two trials with the same initial conditions, is defined as $\Delta\mathcal{V} = \mathcal{V}_{\mathcal{VT}} - \mathcal{V}$ and it shows the optimality gap between the approximation and the online visibility computation. A red or negative delta shows the trials that perform worse, and a blue or positive delta represents an improvement. Each row of histograms in these figures corresponds to one $\mathcal{VT}$ resolution. The columns show different parametric paths. Descending rows show that as resolution increases, the performance difference improves on average. There is little relation between performance difference and the type of parametric path. These plots show the performance difference average moves towards zero as resolution increases, suggesting that at sufficiently high resolutions, there is little distinguishable performance from online computation. The scatter in performance can be attributed to the sampling nature of the optimization.

4.3 Chapter Summary

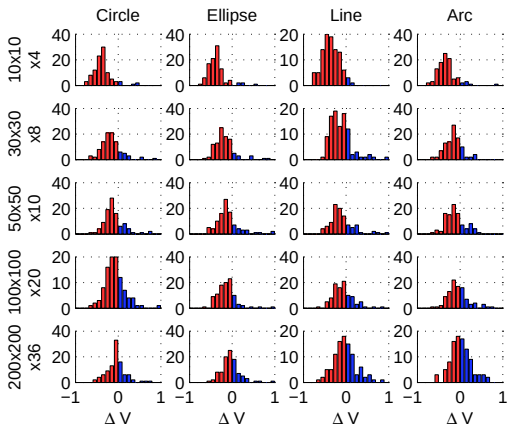
This chapter formulated the multiple target visibility maximization problem (MT-**VMP**), including the weighted sum, max-min, and diminishing returns metric, and results for multiple targets.

A variant of the **VMDP** was presented. This extension handled parametric paths which are more easily-described contours such as circles and ellipses. Faster computation and objective functions which were dependent on the state history could be accommodated. A comparison between **VMDP** and **VM-Par** showed the constrained paths produced less visibility than paths with more degrees of freedom. DP paths provided 10% more visibility than circles and 17% over racetracks in one environment over 200 random trials. An assessment of the different objectives showed that difficult-to-observe targets can be forced to be viewed without tuning weights.

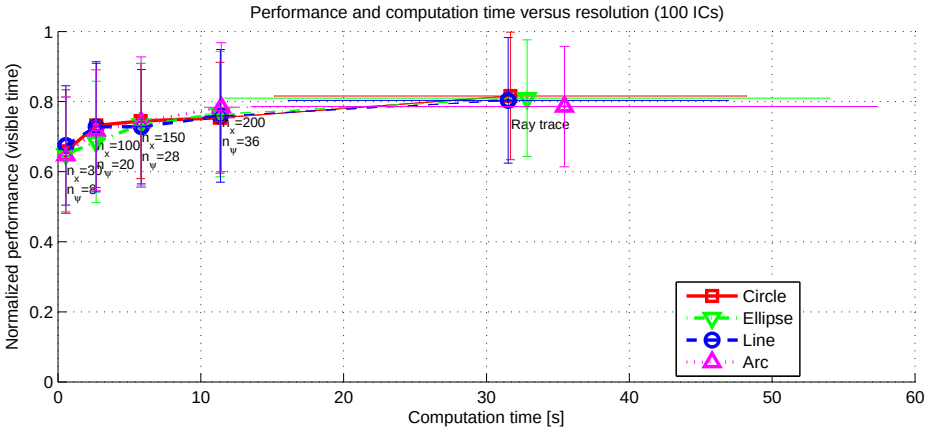
The next chapter discusses an extension of the **VMDP** to moving targets. It



(a) Environment and path example

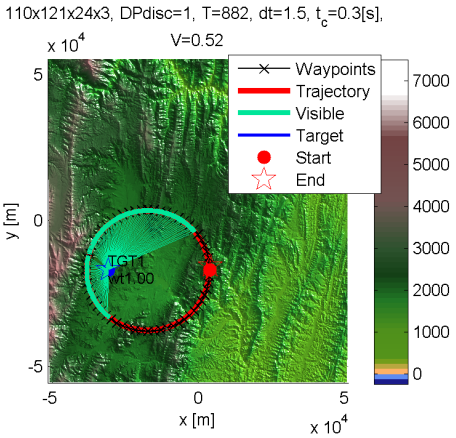


(b) Differences in visibility (red < 0, blue > 0)

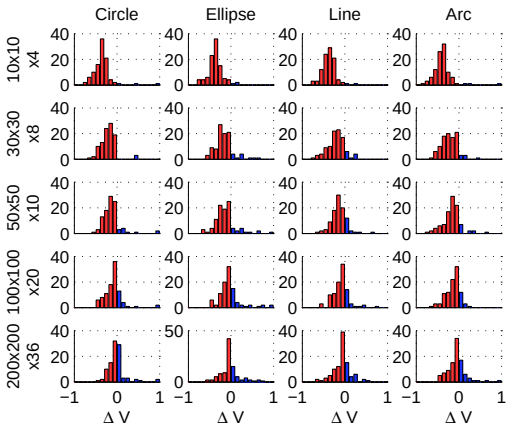


(c) Performance (normalized to online computation result)

Figure 4-13: Performance and computation versus resolution.

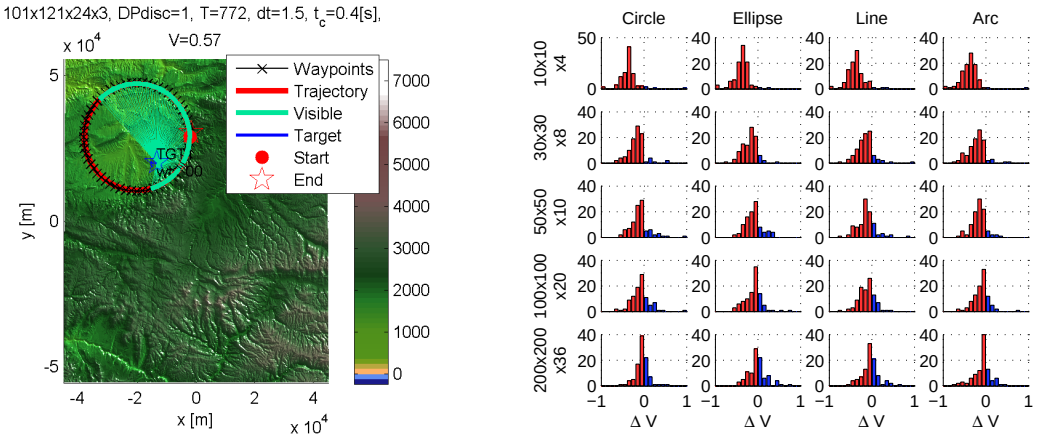


(a) Environment and path example



(b) Differences in visibility (red < 0, blue > 0)

Figure 4-14: Performance and computation versus resolution.



(a) Environment and path example

(b) Differences in visibility (red < 0, blue > 0)

Figure 4-15: Performance and computation versus resolution.

provides methods and results for multiple moving target visibility maximization, for both deterministic and uncertain target motions.

Chapter 5

Moving Targets and Uncertainty in Motion Models

This chapter considers visibility maximization of moving targets and uncertainty in target motion models. Section 5.1 discusses the extension of the stationary target **VMDP** to moving targets. Section 5.2 considers robustness and uncertain target motion using target uncertainty regions.

5.1 Extension of VMDP to Moving Targets

This section describes the extension of the **VMDP** to moving target trajectories. *Moving targets* represent a rich class of objects in the **VMP** that need to be kept visible, such as a car chased down a highway, ships monitored inside a bay, or wildlife pursued over hills and plains.

From the perspective of solving the **VMP**, moving targets add an additional dimension to the visibility model. Instead of a single time-invariant position for each target, the targets now execute a continuous trajectory where the position $\mathbf{x}_T$ varies with time, and so now visibility is a function of target positions which evolve with time.

5.1.1 VMDP Revision for Moving Targets

Recall the **VMDP** consists of a visibility approximation, followed by a path optimization phase. The extension of the **VMDP** to moving targets requires revisiting the visibility approximation, which had previously assumed that the target was stationary.

The most straightforward approach is to parameterize the approximation with time. This time parameterization means a new approximation is generated for each time step. If the approximations are created at discrete time intervals, the complexity of the moving target **VMP** becomes a constant multiple of a stationary target **VMP**.

The first step of the visibility approximation is to generate a visibility table. This table must now be constructed for each discrete time step.

5.1.2 Time-Dependent Visibility Table

Figure 5-1 shows the concept of the time-dependent visibility table. Recall that before, a visibility table was constructed for a stationary target. Now, one table is constructed for each position at each time step for the target.

The time-dependent visibility table $\mathcal{VT}[k]$ is a set of values of $\mathcal{V}$ evaluated by discretizing the target trajectory over finite intervals in time. The target trajectory is discretized over time, steps $k = 0, \ldots, K$.

$$\begin{aligned}
&\mathcal{VT}[k] = \left\{\mathcal{V}(x^{[i_x]}, y^{[i_y]}, \psi^{[i_\psi]}, t[k], \mathbf{x}_T[k], \mathcal{S}[k], \mathcal{T}[k])\right\}, \\
&i_x \in \{1, \ldots, N_{i_x}\}, i_y \in \left\{1, \ldots, N_{i_y}\right\}, i_\psi \in \left\{1, \ldots, N_{i_\psi}\right\} \\
&k \in \{0, \ldots, K\}
\end{aligned} \tag{5.1}$$

The number of function calls to $\mathcal{V}$ is

$$N_{\mathcal{VT},K} = N_{i_x} N_{i_y} N_{i_\psi} (K+1) \tag{5.2}$$

These tables can now be indexed by time k in addition to the discrete observer state indices. Also, these tables can now be used to construct the visibility function

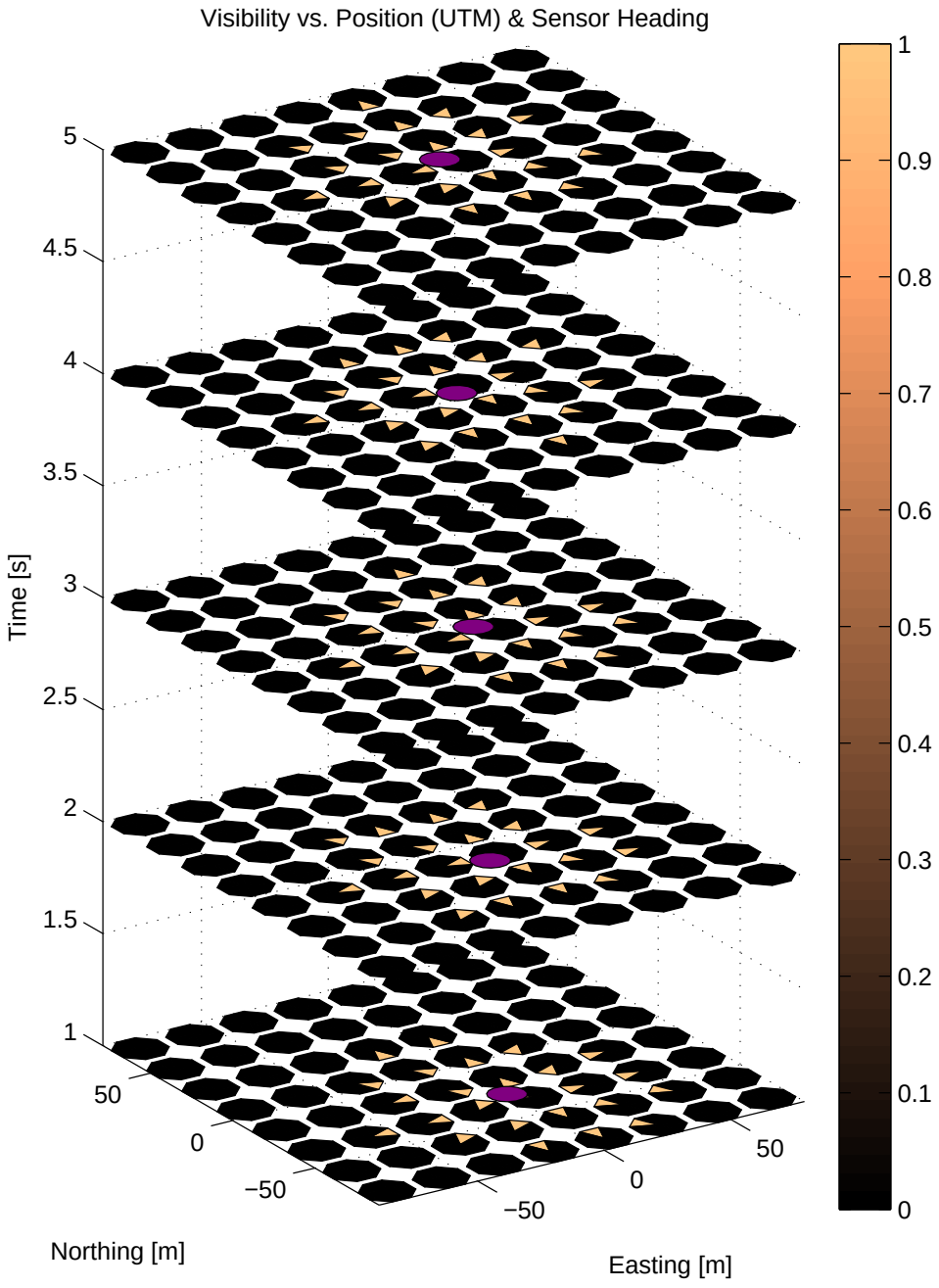


Figure 5-1: Time-dependent visibility table.

approximation $\mathcal{V}_{\text{approx}}(t)$.

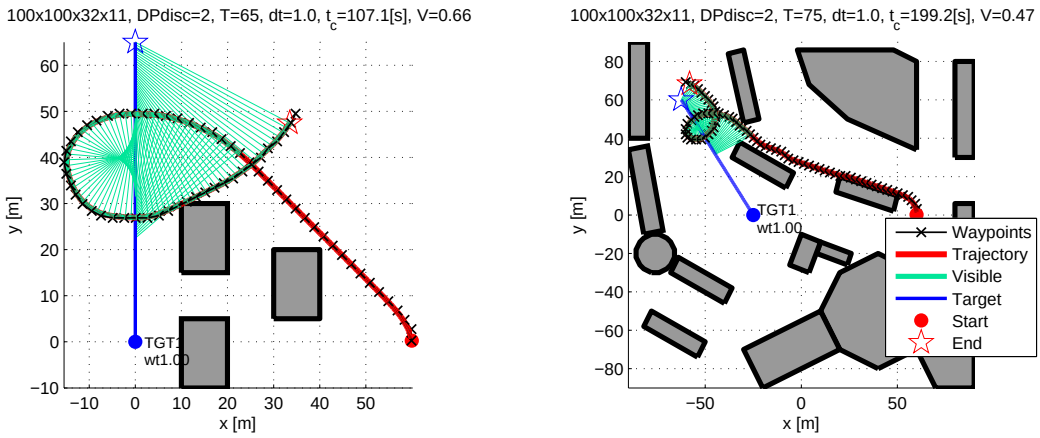
The next section discusses results of implementing the time-varying visibility approximation for maximizing visibility of targets along known trajectories.

5.1.3 Results for Known Target Trajectories

This section shows results of the moving target **VMDP** where the target motion is known a priori. Section 2.4.1 described target motion models. Results for single and multiple moving targets are presented below.

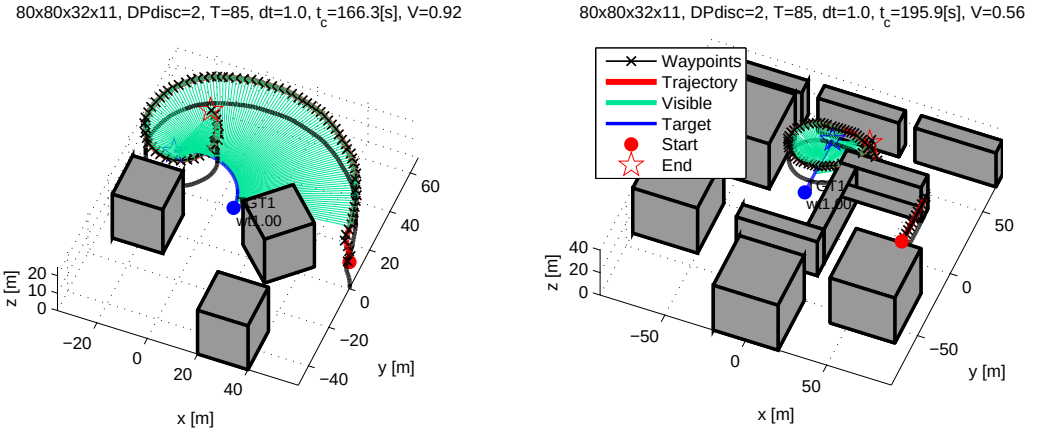
Single Moving Target

This section shows an observer tracking a single moving target. Figure 5-2 shows examples of a moving target observed using a left-facing sensor. Figure 5-3 shows moving target examples in 3-D scenarios. Whereas in the stationary target case, the behavior was typically a loiter pattern, the behavior for maximizing the visibility of a moving target is a mixture of loitering and pursuit phases. A loiter phase will be defined as the part of the trajectory where the target is visible and the observer is making a substantial turn. A pursuit phase is defined as the part of the trajectory where the target is still visible but the observer is not making a substantial turn, say $\dot{\psi} < 0.2u_{\max}$. Parts of the trajectory which do not provide visibility are course-correcting phases, which might occur as a result of initial conditions or the presence of occlusions and obstacles. The loiter phase is present when the target is slower than the observer. The loiter patterns are much closer to the target than the stationary loiter patterns because a part of the observer trajectory now has to be used to follow the target. The shape of the loiter pattern and pursuit pattern change when the speed ratio and the shape of the target's trajectory changes. Slower targets lead to larger loiter patterns and shorter pursuit phases, and faster targets cause the opposite to occur. When the target motion is curved, the loiter and pursuit portions of the observer trajectory also move in that direction. Figure 5-4 shows example trajectories where the speed ratios of the target to observer are varied. The target



(a) 3 obstacles (b) 16 obstacles

Figure 5-2: DP solutions in 2-D environments, moving target, left-facing sensor.



(a) 3 obstacles (b) 10 obstacles

Figure 5-3: DP solutions in 3-D environments, moving target, left-facing sensor.

moves in a straight line in a 2-D environment. When the target-to-observer speed ratio is small, as in Figures 5-4(a) and 5-4(b), the observer path approximately loiters around the target, as if the target were stationary. Over a long time horizon there will be a translation in the center of the loiter pattern, reflecting the motion of the target. As the target speed increases, the observer path must unwind more and more along the direction of the target's motion. The loiter circles begin to compress in size, and start to resemble tear drop shapes. Figure 5-4(d) shows that at a speed ratio of 80% the loiter phases have shrunk and given way to increasingly longer pursuit phases. Figure 5-4(f) shows that when the speed ratio is 160%, the observer no longer loiters

Table 5.1: Percentage of time in loiter versus pursuit phases (approximate). A switch between mostly loiter to mostly pursuit occurs somewhere between a speed ratio of 0.5 and 0.8.

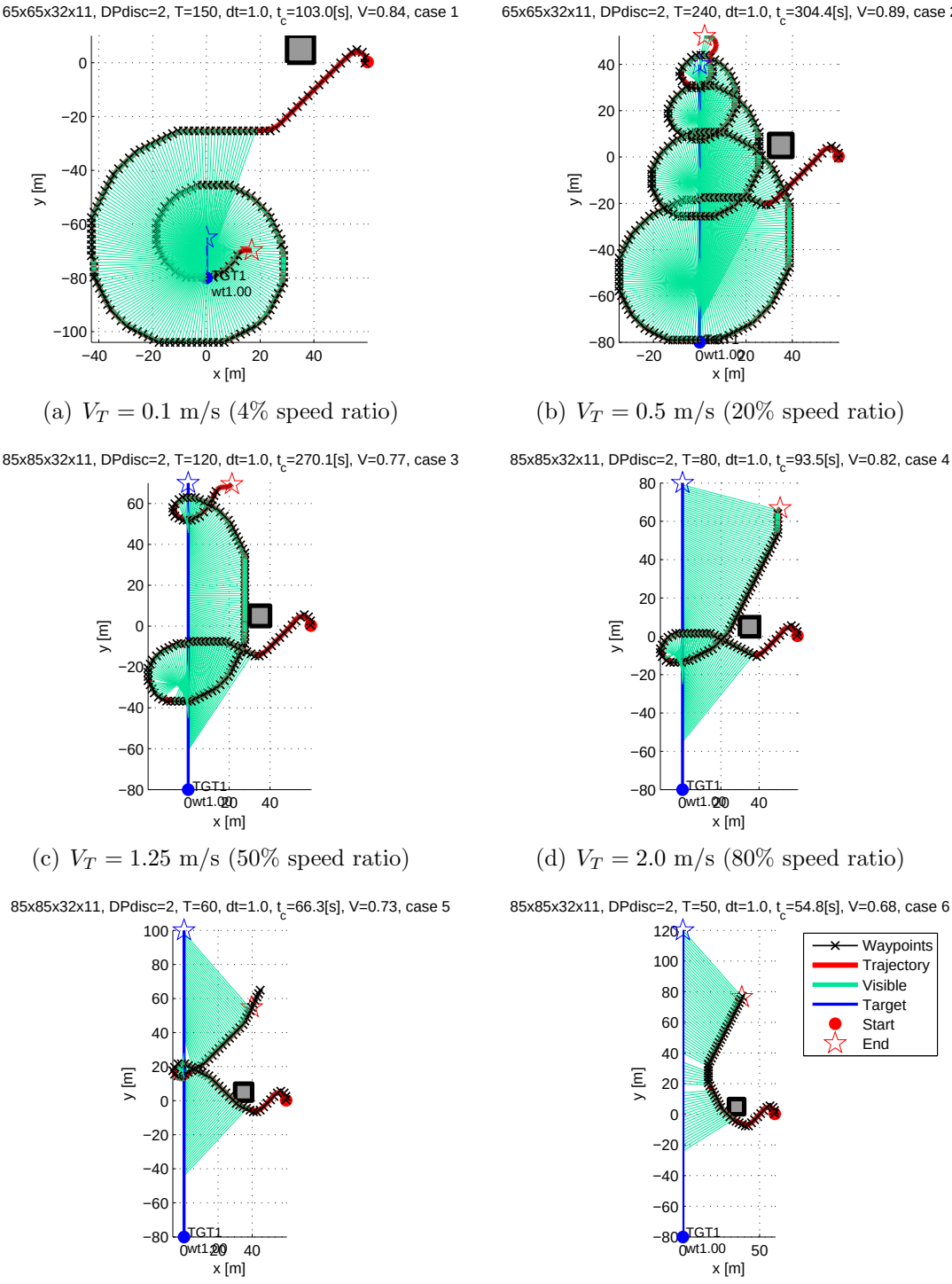
Speed ratio V_T/V_A	Loiter (%)	Pursuit (%)	Course Correction (%)
0.04	**78**	6	16
0.20	**77**	12	11
0.50	**61**	16	23
0.80	32	**50**	18
1.20	23	**50**	27
1.60	0	**68**	32

and instead executes a rendezvous or interception-like behavior. On the approach, the observer takes a line which gradually converges towards the interception point, which occurs around $(0, 40)$. Upon interception, the observer slingshots about 135 degrees to its right and prepares to watch the target as it moves away. Since the observer is slower than the target, the observer trajectory diverges in a manner such that the sensor line of sight can track the target for as long as possible, until the target disappears beyond the sensor's view. Table 5.1 shows an approximate percentage of time in the loiter, pursuit, and course correction phases.

Multiple Moving Targets

This section shows results for multiple moving targets. This is the most challenging case because if the targets move away from each other, the observer can only keep sight of one or the other. Multiple moving target plans are influenced by the reward of seeing, or opportunity cost of not seeing a particular target.

Figure 5-5 shows examples of multiple moving targets in 2-D. In these examples, the targets depart away from one another, making observation very difficult. Since the targets diverge, the observer can only see one of the targets for any amount of time. Thus, the observer executes two patterns. First, when the targets are still close, the observer performs a loiter-pursuit combination with the target cluster. Once the targets move sufficiently far apart, the observer switches to its second behavior which is the same behavior seen for the single moving target visibility maximization. The



(a) $V_T = 0.1$ m/s (4% speed ratio)

(b) $V_T = 0.5$ m/s (20% speed ratio)

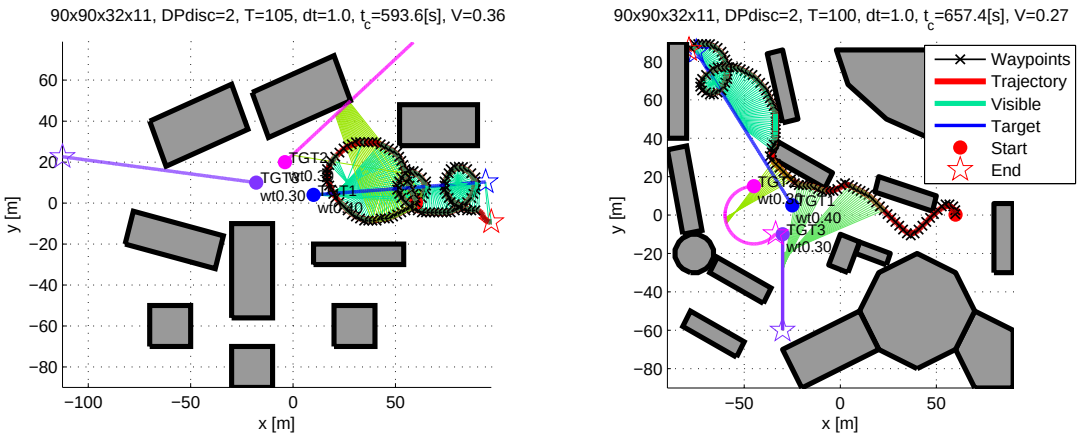
(c) $V_T = 1.25$ m/s (50% speed ratio)

(d) $V_T = 2.0$ m/s (80% speed ratio)

(e) $V_T = 3.0$ m/s (120% speed ratio)

(f) $V_T = 4.0$ m/s (160% speed ratio)

Figure 5-4: DP solutions for different target speeds; $V_A = 2.5$ [m/s].

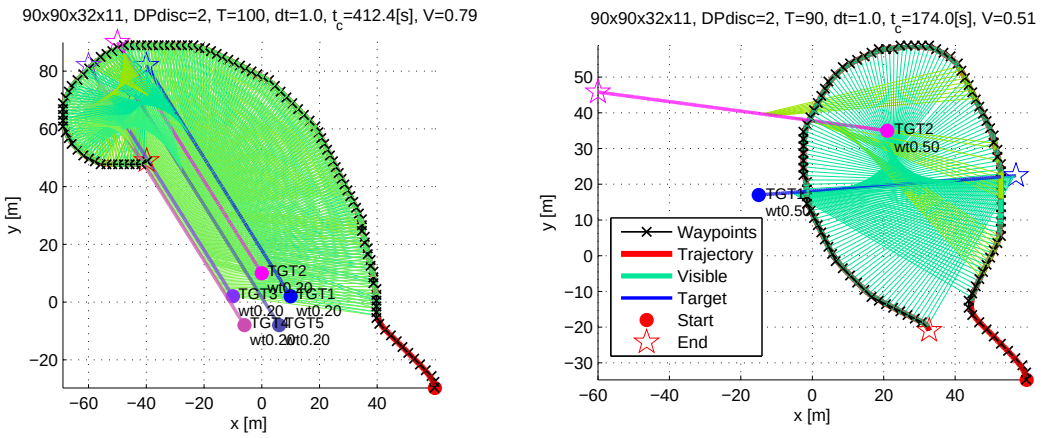


(a) 9 obstacles

(b) 16 obstacles

Figure 5-5: DP solutions in 2-D environments, multiple moving targets.

second behavior is again a combination of loiter and pursuit phases. The observer chooses only one target in both examples; this target in general will likely has the greatest importance or is the least occluded among all the targets. Figure 5-6 shows examples of targets moving together in the same direction and moving away from each other. The first case, where the targets move in unison, results in a tendency of the observer to keep the as many of the targets in view, with the area between the two targets constantly in view. This behavior is important for the robustness formulation considered later. The second case again shows the two-part behavior seen earlier for diverging targets. The first portion resembles the cluster observation pattern, while the second phase, when the observer sees the second target for the last time, is again a display of single moving target loiter-pursuit. It also appears that the first target can be seen relatively well throughout the first phase before reaching the second phase. As well, the initial conditions play a role in breaking the equal weight tie and the observer decides it is best to track the first target, which more favorably moves initially towards the observer. When more than one moving target is present, the observer must allocate its time between monitoring each target, depending on its relative importance in the objective function. When the targets are located close together and move in the same direction, the observer can monitor the targets simultaneously. However, when the targets move away from each other, the



(a) 5 targets moving together

(b) 2 targets moving away

Figure 5-6: DP solutions for targets moving together, away from each other.

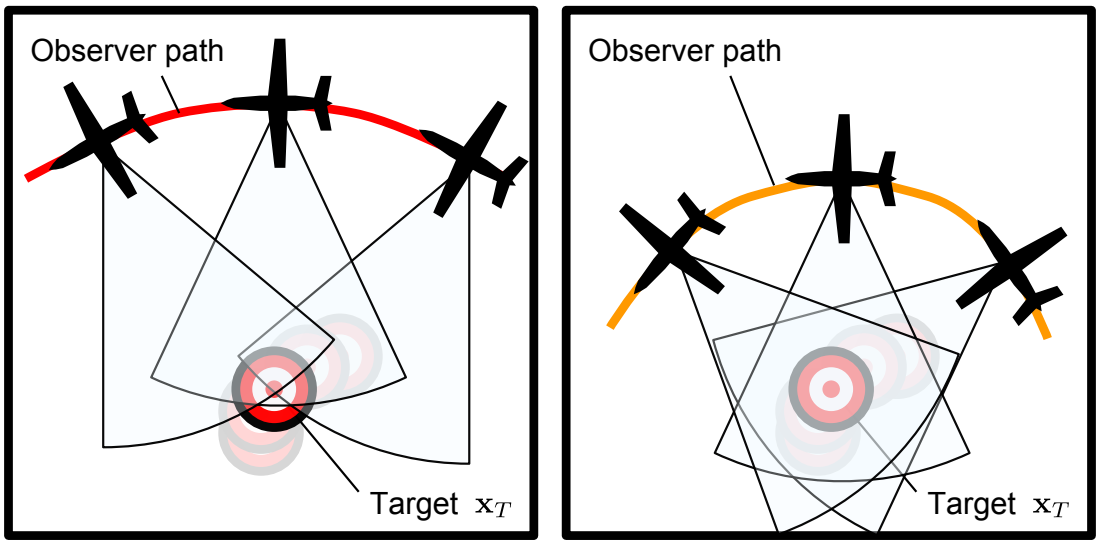
observer will have to spend time pursuing the targets individually. Sometimes it is too difficult or less rewarding to observe particular targets, in which case the observer will maneuver the sensor to the more easily viewed targets.

5.2 Robust Target Observation

This section investigates robust visibility trajectories. When a target's trajectory is not known exactly, or there can be disturbances in the observer's motion, a robust approach attempts to quantify these uncertainties. Doing so should make the observation sequence less susceptible to both forms of noise.

Figure 5-7 shows an example of an extremal and non-extremal path. *Less extremal* observer paths means the target will be kept closer to the center of the sensor's range and field of view, instead of the edges of view. A less extremal path is more robust to perturbations and to inaccuracies in modeling. In other words, disturbances to either the observer or the target trajectory do not affect the visibility along the path as severely as when a target is assumed to be just a single point.

In a sense, a target region can also capture the *uncertainty* about the target's position. The size of the uncertainty can be predetermined at the start of the plan, or updated using estimation and prediction techniques. Since the goal of this thesis



(a) Extremal (b) Non-extremal

Figure 5-7: Extremal versus non-extremal paths.

is not to perform estimation, uncertainty will be predetermined and allowed to grow according to the process noise of the target's motion, but cannot be reduced by measurement.

Target regions are also applicable to static targets, to generate less extremal paths, which tend to be sensitive to perturbations as will be seen in the following sections.

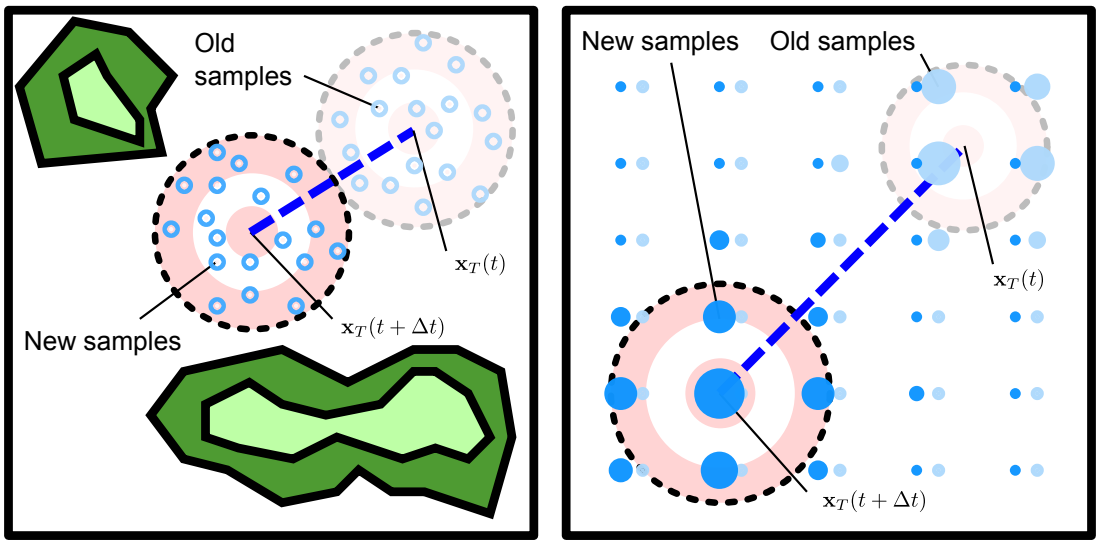
5.2.1 Robust Formulations

This section considers robust formulations to moving target trajectories whose trajectories are known, or unknown, possibly provided by a prediction mechanism.

Several robust formulations are possible, from the perspective of the target or the observer uncertainty. These include target regions and observer regions. The robustness factor is considered because optimal control tends to generate extremal paths which are sensitive to model uncertainty.

The robust formulation maximizes the integral of expected visibility. Expected visibility can be modeled using probabilistic distributions for the inputs.

$$\max_{\mathbf{u}(t)} J_{V,\text{robust}} = \int_0^T \mathbf{E}\left[\mathcal{V}(\mathbf{x}_A, \mathbf{x}_T, \mathcal{S}, \mathcal{T})\right] dt \tag{5.3}$$



(a) Translating samples

(b) Uniform samples with weight updates (larger circles have larger weights)

Figure 5-8: Sampling methods for uncertain moving targets.

The target region approach directly quantifies the uncertainty in target position as a function of time. The uncertainty can evolve with time, by assuming a particular target motion model (such as interacting multiple models, adversarial motions, and pop-up targets). The expected visibility is a function of an uncertain target position.

The multiple-target formulation, the MT-**VMDP**, is used to model target regions. There are two approaches: one represents the region using fixed weight samples whose spatial density represents the likelihood, or variable-weight samples; the other is to represent the likelihood at fixed locations in the environment.

The first method requires drawing new samples at each time step, in a manner similar to particle filtering [78]. Few samples are required to represent the probable area of detection. The complexity is $N_{\text{samples}} \times N_{\text{timesteps}}$. Figure 5-8(a) shows the translating samples. Further details are provided in Appendix A.3.

The second method requires updating the weights of the fixed target positions. Once this grid is specified, weights of each point on the grid can be assigned using a probabilistic model of the target being in the grid point. The complexity is $N_{\text{grid},x} \times N_{\text{grid},y}$. Figure 5-8(b) shows uniformly-spaced samples whose weights are updated with time.

5.2.2 Analysis and Numerical Results for Robust Visibility

This section shows results for robust visibility maximization. This section compares visibility along paths designed to track point versus region target trajectories, as well as point and region observer trajectories. The robust approach demonstrates its usefulness when observer or target trajectories are off-nominal. The sensor now maximizes its ability to detect an area rather than a point.

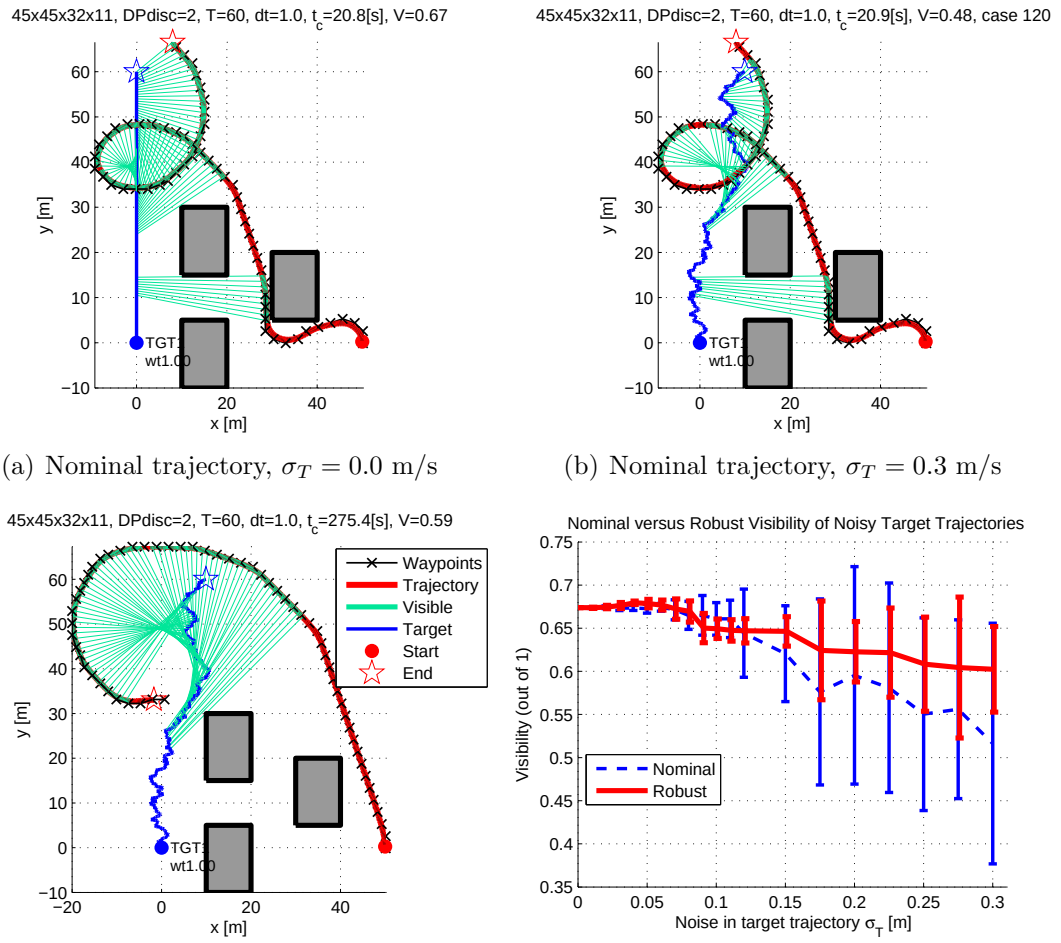
Figure 5-9 shows performance of the nominal and robust paths versus target uncertainty. The target uncertainty $\mathbf{w}_T(t)$ is modeled as a Wiener process [79] in the x position with standard deviation σ_T, and a constant velocity in y of $V_T = 1.0$ [m/s].

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 0 \\ V_T \end{bmatrix} + \mathbf{w}_T(t) \tag{5.4}$$

$$\mathbf{w}_T(t) \sim \begin{bmatrix} \mathcal{N}(0, \sigma_T^2) \\ 0 \end{bmatrix} \tag{5.5}$$

In the robust trajectory, the uncertainty region is approximated using 20 samples which are re-sampled every time step. Fewer samples decreases computation time, but also decreases accuracy, especially as the noise increases. Figure 5-10 shows that as target uncertainty σ_T increases, the observer takes broader paths to keep sight of the growing uncertainty region. The result is improved average performance and a reduction in performance variability.

Figure 5-11 shows the sensitivity of the robust plan to the number of samples used to approximate the target's uncertain motion. Thirty trials are run for each number of samples at three target uncertainty sizes, $\sigma_T = 0.25$ to 0.35 [m/s], over a time horizon of $T = 65$ [s]. From the figure, there appears to be little discernable improvement with increasing the number of samples from 4 to 32. The scatter is similar to the robust performance seen in Figure 5-9(d).



(a) Nominal trajectory, $\sigma_T = 0.0$ m/s

(b) Nominal trajectory, $\sigma_T = 0.3$ m/s

(c) Robust trajectory, $\sigma_T = 0.3$ m/s

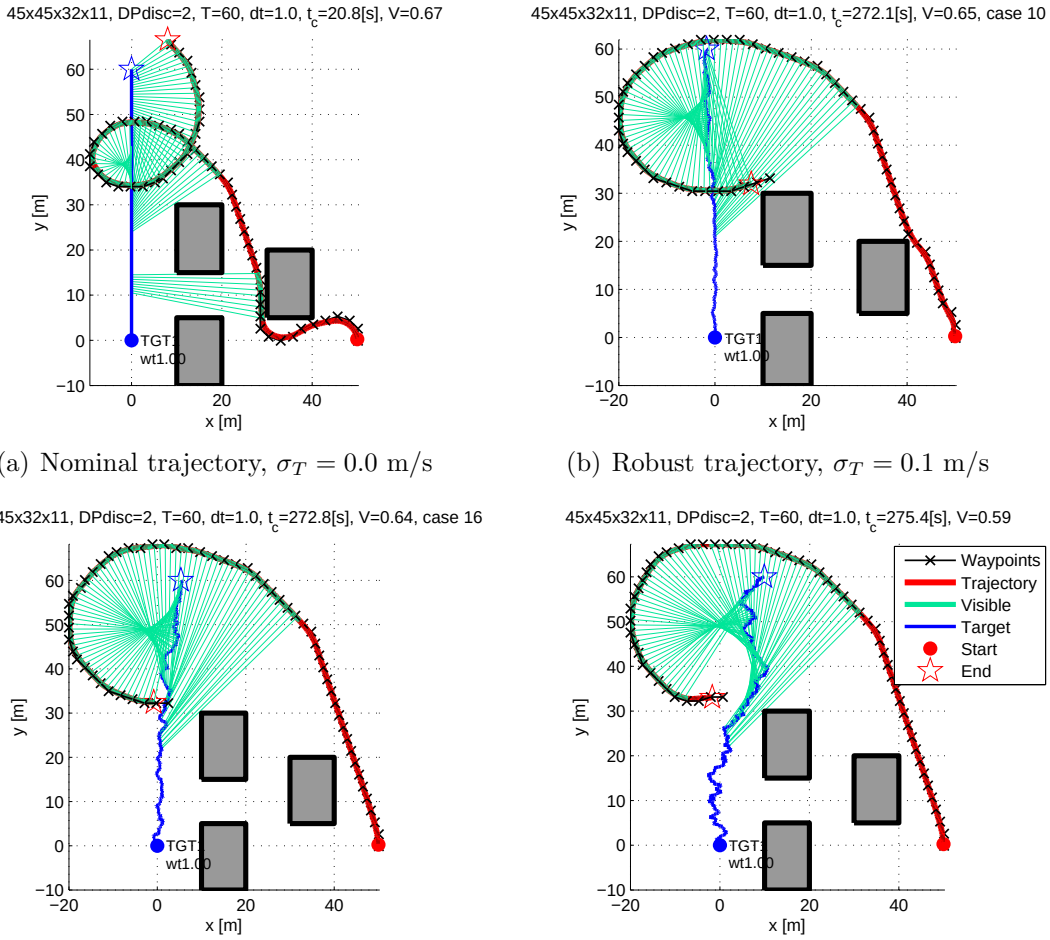
(d) Performance versus noise (20 trials per σ_T) (robust error bars offset slightly right for clarity)

Figure 5-9: Nominal versus robust visibility with target noise.

5.3 Chapter Summary

This chapter introduced the moving target formulation using time-varying visibility tables. It also presented numerical results for single and multiple moving target trajectories. It discussed an extension for robust target tracking using target region and observer neighborhood modeling, and empirically showed performance benefits in the presence of uncertainty.

The next chapter describes hardware implementations of the stationary and moving target visibility maximization algorithm.



(a) Nominal trajectory, $\sigma_T = 0.0$ m/s

(b) Robust trajectory, $\sigma_T = 0.1$ m/s

(c) Robust trajectory, $\sigma_T = 0.2$ m/s

(d) Robust trajectory, $\sigma_T = 0.3$ m/s

Figure 5-10: Evolution of robust trajectory with increasing noise.

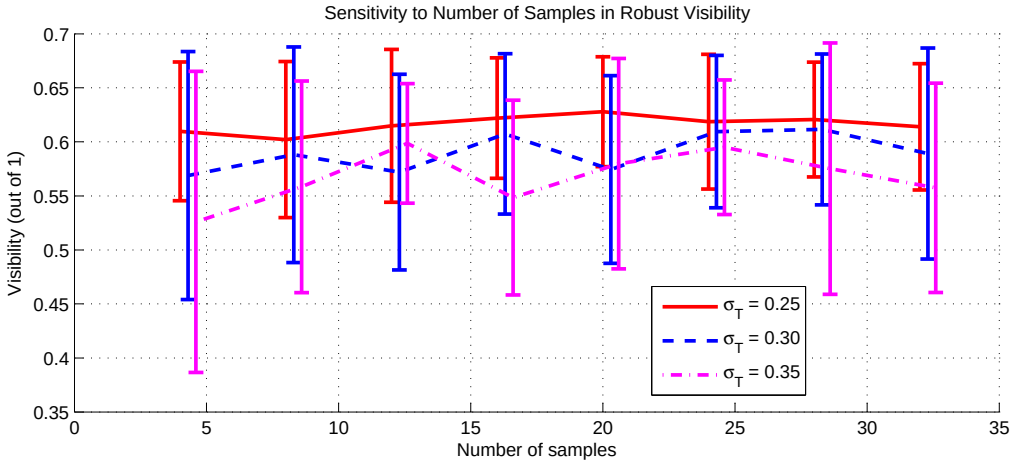


Figure 5-11: Sensitivity to number of samples for robust target approximation.

Chapter 6

Testbed Implementation

This chapter describes the hardware implementation of the visibility maximization algorithms presented in Chapters 3 and 5. Hardware results validate the modeling assumptions applied in a real-world setting. Section 6.1 discusses the hardware and software infrastructure which enables scale experiments in visibility maximization missions. Section 6.2 presents experimental results with multiple moving targets, in the presence of disturbances, for hardware demonstrations of target visibility maximization scenarios.

6.1 Testbed Modules

This section describes the testbed environment and modules. The testbed is a complete hardware and software setup that enables experimentation of algorithms on physical platforms. A testbed implementation and results are necessary to verify the applicability of algorithms in actual hardware.

Figure 6-1 shows the primary modules and data flow for the visibility maximization experimentation setup. There are three major modules: the **VMDP** algorithm, the hardware, and the hardware-software interface.

The three major components are described in the following subsections. Section 6.1.1 describes the RAVEN testbed. Section 6.1.2 describes the environments. Section 6.1.3 describes the target tracking module adopted from the algorithms in this thesis.

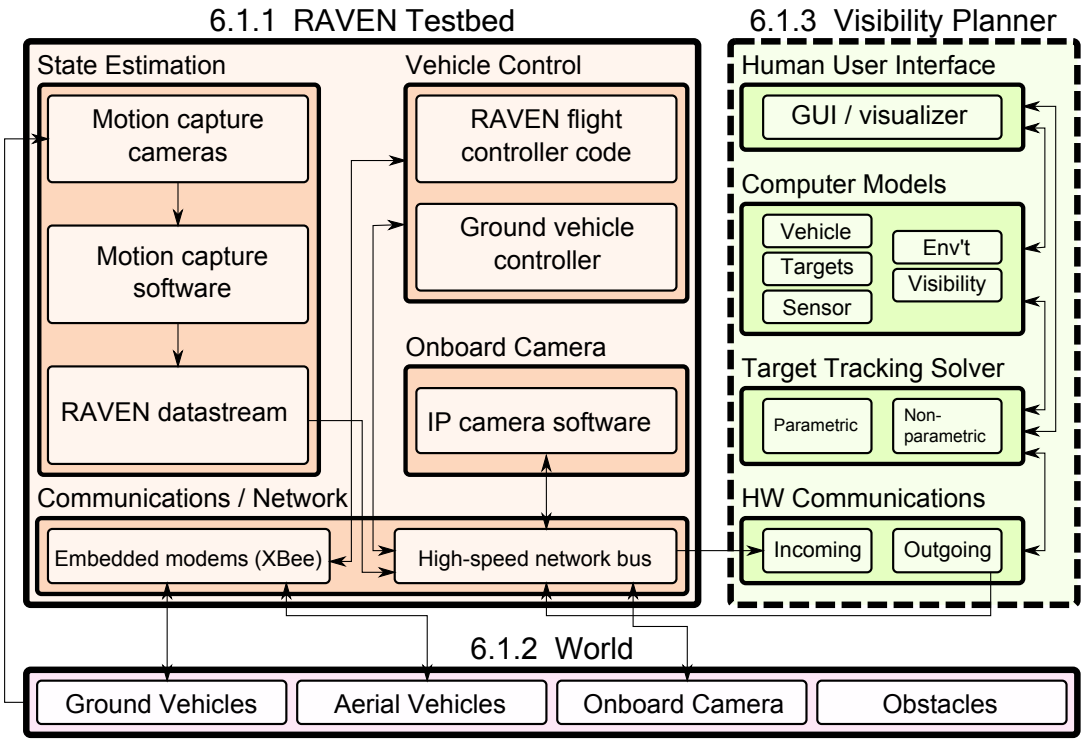


Figure 6-1: Testbed module diagram.

6.1.1 RAVEN Module

The MIT Aerospace Controls Laboratory has developed an indoor testing environment for teams of autonomous aerial and ground vehicles. The *Real-time Indoor Autonomous Vehicle Testbed ENvironment (RAVEN)* [80–82] allows researchers to validate autonomous control and planning algorithms beyond a simulation environment, and provides a subset of some real-world effects such as wind disturbances, communications limitations, and noise in measurement data. RAVEN also provides a standardized, reusable interface between custom software algorithms and common hardware platforms.

The RAVEN architecture can be subdivided into four major components. These include vehicle state estimation, vehicle control, sensor data processing, and networking.

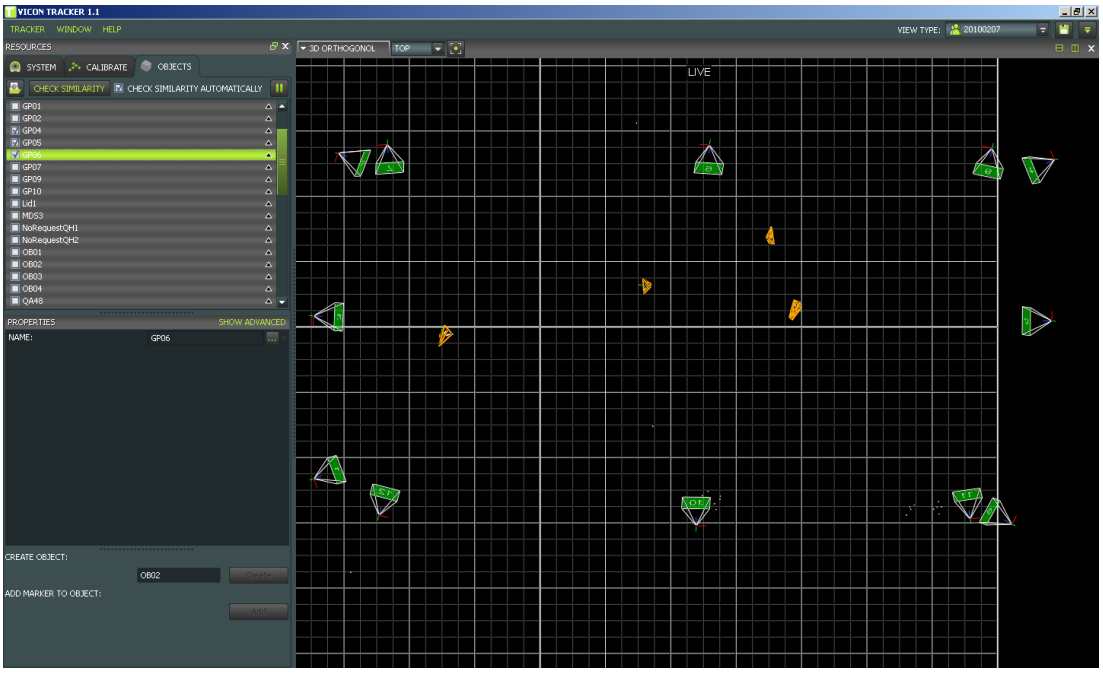


Figure 6-2: State estimation using a motion capture system.

State Estimation

Figure 6-2 shows the RAVEN state estimation system. State estimation provides state data on vehicles and other objects in the environment. Motion capture provides state information about vehicles, targets, and environment features. The motion capture system consists of infrared cameras, image processing software, and a state estimator. This off-board state estimator mimics an indoor GPS, providing accurate state information which can be used for high performance vehicle control, planning, and data collection.

Vehicle Control Computers and Software

RAVEN provides off-board computation for vehicle control. This allows much greater computation power, weight savings, and protection against damage from collisions. Figure 6-3 shows the off-board computers which run the vehicle control algorithms. All vehicles are wireless and radio-controlled. Commands to vehicles are communicated over XBee serial modems. Only low-level actuation is performed on the vehicle microprocessor.

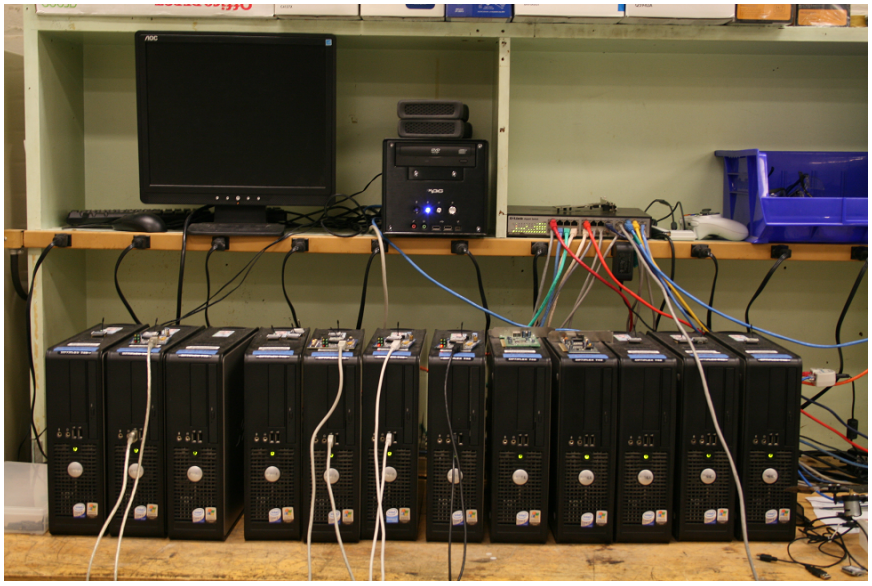


Figure 6-3: Vehicle control computers

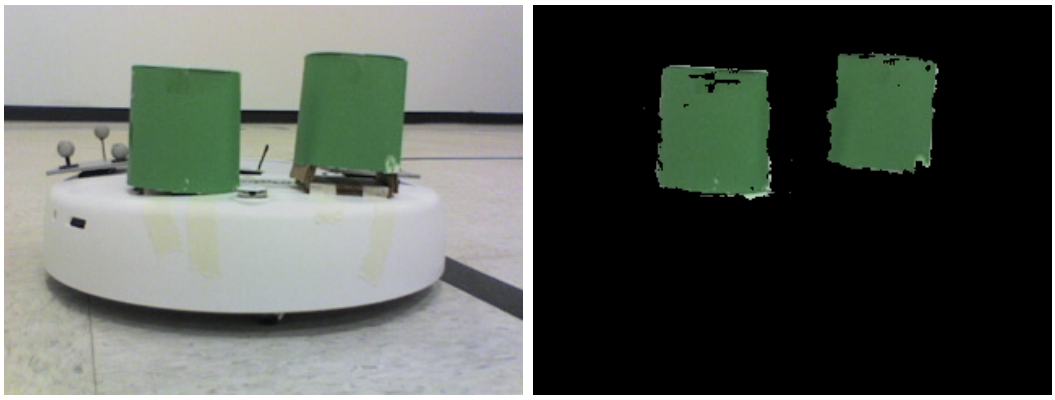
Image Processing Software

Image processing is performed using RAVEN camera code. This software can perform color tracking for target recognition. It uses the OpenCV libraries [83] to access and process the camera images. Figure 6-4 shows a view of the color tracking software performing real-time thresholding computations. Figure 6-5 shows an example of using the histogram of hue, saturation, and value (HSV) from an image to determine the threshold. To achieve fewer missed readings, the thresholds should be increased. To achieve fewer false alarms, the thresholds should be decreased.

Network

A high-speed local area network carries packets containing formatted messages with state information from the estimator. It also carries formatted high-level command packets such as take-off, landing, and fly to waypoint, which are received by the vehicle controllers.

The low-level commands are sent over IEEE 802.15.4 (ZigBee) protocol [84]. XBee modems use the ZigBee protocol to provide low-power serial communications. The lower bandwidth is a necessary tradeoff to keep power consumption low, as required



(a) Actual view from camera (b) Thresholded image

Figure 6-4: Color recognition by image processing software.

in lightweight embedded applications. Figure 6-6 shows the XBee modems.

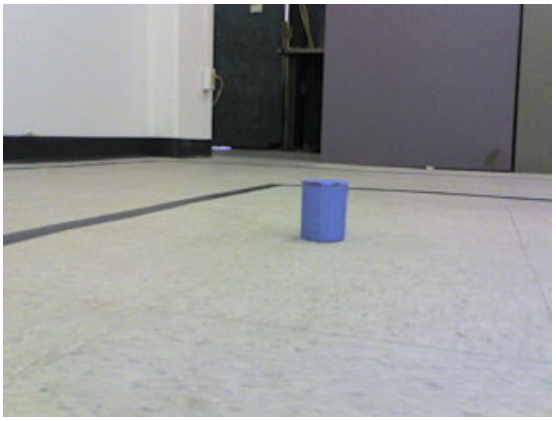
6.1.2 Test Environment

The Aerospace Controls Laboratory's RAVEN environment is situated in an indoor facility. The flight volume[1] is approximately $10 \times 5 \times 3$ [m], which can accommodate over a dozen vehicles simultaneously. Below are a short descriptions of each hardware component, in particular the ground vehicles, aerial vehicles, network cameras, and obstacles.

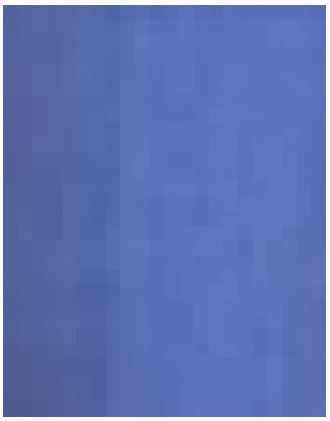
Ground vehicles

The RAVEN ground vehicles are iRobot® Create® [85] platforms, which are modified Roomba robots, shown in Figure 6-8. These are two wheel, differential drive platforms which can travel between 0.05 to 0.5 [m/s]. They can also accommodate about 1 [kg] of payload. With the Create Command Module add-on, onboard processing capability is available for small applications such as path planning and actuating additional motors. Each vehicle is fitted with reflective spheres that can be detected by the motion capture system. Each vehicle has a distinct livery to allow color detection to recognize each vehicle individually.

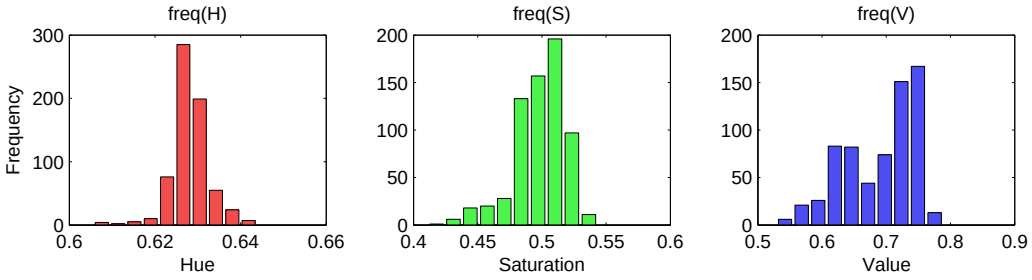
[1]This is a new flight volume completed in 2010.



(a) Example of colored object viewed from camera



(b) Cropped image showing color variance



(c) Histogram of HSV values

Figure 6-5: Hue, saturation, and value (HSV) thresholds for a blue object in one lighting setting. Multiple samples provide an improved estimate of the color thresholds.

Aerial vehicles

The RAVEN quadrotors are unmanned helicopters with four counter-rotating propellers[86]. Onboard stabilization provides stability to the airborne platform. The quadrotors are adapted to accept a desired pose for position and orientation control. The small quadrotors used in the test environment can fly for 6 minutes on a 3200 mAh 20C lithium ion battery with a network camera onboard. These are also equipped with reflective tape for motion capture to provide state feedback. Figure 6-9 shows a quadrotor used in RAVEN.

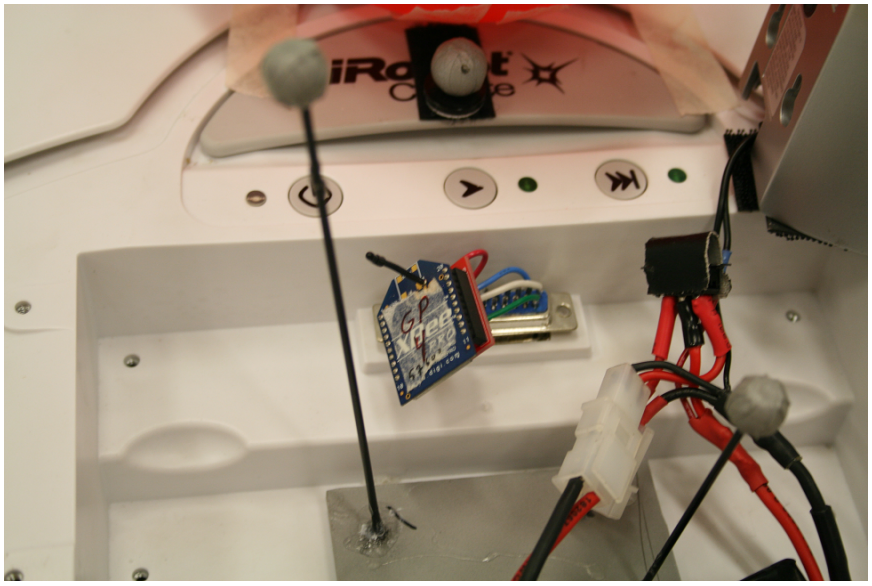


Figure 6-6: RAVEN XBee modem.

Network cameras

Wireless network cameras provide real-time visual feedback. The camera internally hosts the captured images. The image processing software can query the camera server to access the images. A slight delay can be observed between the upload and download process.

The particular camera model is the Panasonic BL-C131a pan-and-tilt wireless network camera [87], with 49 degrees of horizontal field-of-view, 35 degrees of vertical field-of-view, and a CMOS sensor with 640 by 480 pixel resolution. The pan and tilt mechanism is not used. The actual image resolution used is 320 by 240 pixels to speed up image processing calculations and lower network bandwidth. Figure 6-10 shows the camera and an example video capture frame.

The pan and tilt mechanism is disassembled for the quadrotor camera to lower weight and to allow the camera to be placed near the undercarriage between the rotors.

Figure 6-7: RAVEN indoor test environment.

Obstacles

Obstacles in RAVEN are scale models of urban or mountainous environments, such as buildings and cliffs which obstruct movement and occlude sensors. These obstacles are fitted with an arrangement of reflective spheres to determine their pose. CAD models describe the vertices and surfaces of each obstacle.

6.1.3 Visibility Planner Module and Real-Time Visibility Feedback

This section describes the visibility planning software module. The visibility planning software module converts an internal model of the world, including targets, observer, sensor, and environment, into actions for the observer.

The **VMDP** is initialized with RAVEN state information about targets, obstacles, and the observer. Once initialized, the **VMDP** computes the maximum visibility plan. Once the plan is computed, the operator can activates the hardware. When activated, the planner continuously broadcasts commands to RAVEN to control the hardware. The planner also monitors the states of the observer and target using motion capture feedback, and sends updated commands to direct the hardware to

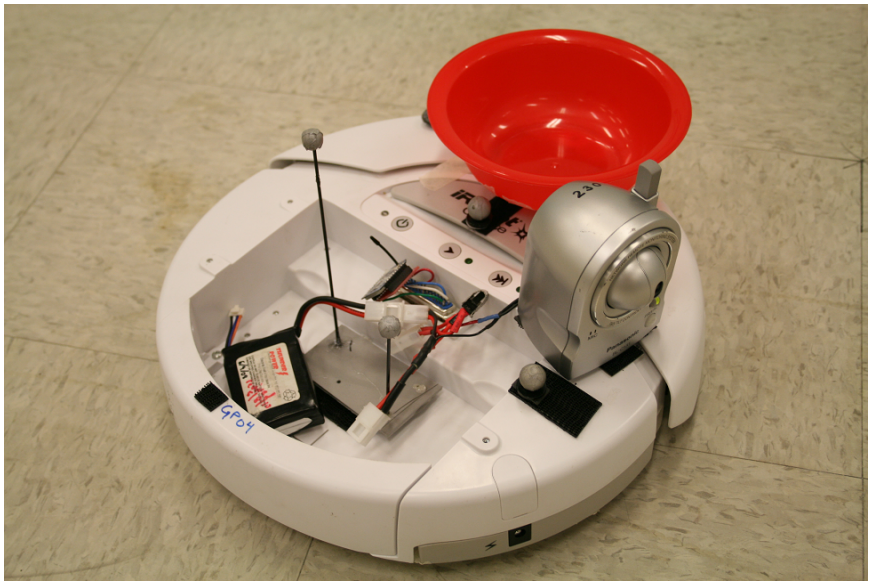


Figure 6-8: Create ground robot.

follow the desired trajectory.

In addition to state feedback, the planner receives feedback on the visibility of targets using the color recognition software. The GUI displays the real-time detection threshold pass of the target. Detection is a binary flag which represents whether the number of detected pixels passes a threshold. The model for the world-to-camera projection is the pinhole camera model, which means objects in the plane normal to the medial axis of the camera appear the same size on the image. Figure 6-11 shows that objects in the plane do not appear distorted, which means that the projection is approximately rectilinear. By this rectilinear assumption, objects in the image appear according to the pinhole camera model projection shown in Figure 6-12. Then, the expected number of pixels seen by the camera can be calculated as follows. Equation 6.1d is the approximate area of the planar surface visible in the camera field of view, subject to any errors from the rectilinear projection process.

$$d = \sqrt{x_{T,A}^2 + y_{T,A}^2 + z_{T,A}^2} \tag{6.1a}$$

$$s_x(d) = 2d \cdot \tan\left(\frac{\theta_{\text{FOV}}}{2}\right) \tag{6.1b}$$

$$s_y(d) = 2d \cdot \tan\left(\frac{\eta_{\text{FOV}}}{2}\right) \tag{6.1c}$$

Figure 6-9: RAVEN quadrotor.

$$
\begin{aligned}
A_{\text{FOV}}(d) &\approx s_x(d) \cdot s_y(d) \\
&= 4d^2 \tan\left(\frac{\theta_{\text{FOV}}}{2}\right) \tan\left(\frac{\eta_{\text{FOV}}}{2}\right)
\end{aligned} \tag{6.1d}
$$

Equation 6.2 is the fraction of the frontal area of the target's livery in view, $A_{T,\text{frontal}}$. This area is roughly approximated by a rectangle with dimensions l_x and l_y which, for the purpose of simplifying the calculation, is always orthogonal and upright in the image plane. Only a fraction of this rectangle can be viewed, and this fraction is approximated using s_{l_x} and s_{l_y}. Alternatively, to improve the accuracy of the projected area calculation, one can approximate the object using a discrete set of points around the surface of a convex object. A cylinder for example can have $2n$ points on its surface, coinciding with the vertices of an n-sided prism approximating the cylinder. The projected area will be the convex hull of those points in the image coordinate frame which overlaps the image area.

$$\Delta\theta = \arctan\left(\frac{y_{T,A}}{x_{T,A}}\right) \tag{6.2}$$

$$\Delta\eta = \arctan\left(\frac{z_{T,A}}{\sqrt{x_{T,A}^2 + y_{T,A}^2}}\right) \tag{6.3}$$

(a) Camera

(b) View from camera onboard moving ground observer

Figure 6-10: Panasonic wireless network camera and onboard view.

$$s_{l_x} \approx \min\left(\max\left(\frac{s_x}{2} - d\tan\left(\Delta\theta\right) + \frac{l_x}{2}, 0\right), l_x\right) \tag{6.4}$$

$$s_{l_y} \approx \min\left(\max\left(\frac{s_y}{2} - d\tan\left(\Delta\eta\right) + \frac{l_y}{2}, 0\right), l_y\right) \tag{6.5}$$

$$A_{T,\text{frontal}} \approx s_{l_x} \cdot s_{l_y} \tag{6.6}$$

Equation 6.7 states that the number of pixels $P(d)$ is the area fraction of the object with respect to the field of view, times the number of pixels of the camera sensor. The total resolution is $P_{\text{sensor}} = 320 \times 240 = 76800$. The area fraction is approximated by the areas calculated above. The target area fraction constant $K_{T,m}$ for target m can be pre-computed given the dimensions of the livery and the horizontal and vertical field of view of the sensor.

$$P(d) = \left(\frac{A_{T,\text{frontal}}}{A_{\text{FOV}}}\right) P_{\text{sensor}} \tag{6.7}$$

$$\approx \left(\frac{A_{T,\text{frontal}}}{4d^2 \tan\left(\frac{\theta_{\text{FOV}}}{2}\right) \tan\left(\frac{\eta_{\text{FOV}}}{2}\right)}\right) P_{\text{sensor}} \tag{6.8}$$

$$= K_{T,m} A_{T,\text{frontal}} \cdot d^{-2} \tag{6.9}$$

$$K_{T,m} = \left(\frac{1}{4\tan\left(\frac{\theta_{\text{FOV}}}{2}\right) \tan\left(\frac{\eta_{\text{FOV}}}{2}\right)}\right) P_{\text{sensor}} \tag{6.10}$$

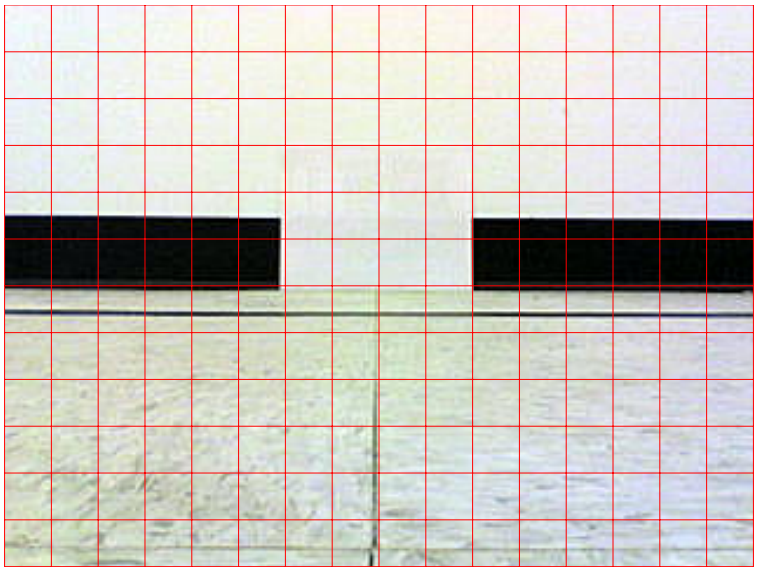


Figure 6-11: Approximate rectilinear projection in camera image. The lower half of the image shows a tiled floor, while the upper half shows a wall 4 feet away and normal to the camera's medial axis. The upper center of the image shows a white letter-sized sheet of paper flush against the wall. The minimal pincushioning or barrel distortion in the image (in other words, straight edges remain straight) can be verified using the overlay. The distortion-free image means that the projection is approximately rectilinear.

6.2 Experimental Results

This section presents experimental results of the **VMDP**. The experiments include ground and aerial observers in 3-D environments.

6.2.1 Ground Observer

This section shows an example ground observer experiment. Figure 6-13 shows a sequence of GUI snapshots and onboard camera video frames from a ground-based observer, with two targets and 2 obstacles for an 80-second run. The environment is constrained between -2 to 6 [m] in x (west-to-east) and -2.25 to 2.25 in y (south-to-north). The observer (GP04) is equipped with a rightward-looking sensor mounted slightly to the right and above the center of the vehicle, and has a speed of 0.15

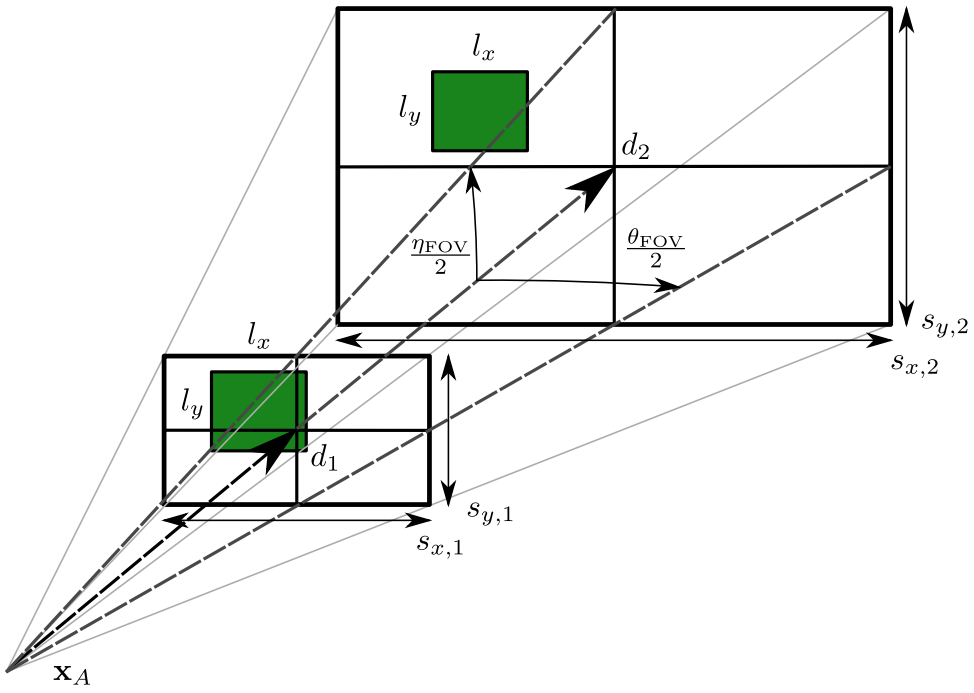
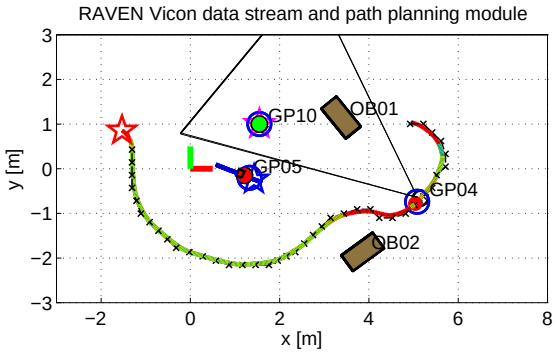


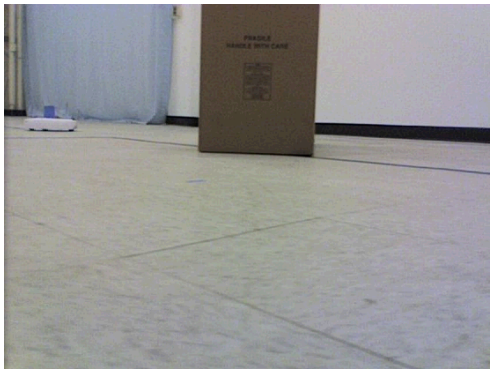
Figure 6-12: Pinhole camera projection. Dimensions are relative to the camera coordinate frame. For simplicity, the frontal area of the target livery is assumed to be a rectangle which always remains upright in the image.

[m/s]. One target (GP05) moves back and forth along a straight line trajectory at a speed of 0.075 [m/s] and the other (GP06) is stationary. The observer begins to the east of the two targets, and is initially oriented eastward away from the targets. The planner determines that the observer should move clockwise to orient the sensor towards the two targets, out of the obstructed view of the north obstacle (OB01), while completing a short 180-degree turn that ends at 20 [s]. It then avoids the obstacle (OB02) by making a gentle S-curve for about 10 [s], then proceeds into a wide loiter pattern around the two targets until the end of the planning horizon at 80 [s]. The sequence of onboard camera views show the target vehicles and their color-tracking livery – green for the first target and blue for the second. The GUI also shows the orientation and position of the sensor in the planner view. Figure 6-14 shows the actual measurement sequence in the form of number of pixels registered by the color threshold program. Figure 6-14(a) shows the pixel threshold (dashed line) as a function of distance, and the planner's prediction applied over the threshold

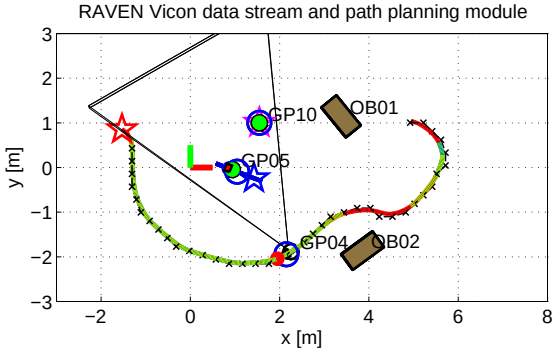
(thick solid line). The actual measurements are shown as noisy dots over the thin line. Figure 6-14(b) shows the result of applying the threshold to achieve binary visibility. The overall trend shows that the predicted and actual pixel counts are similar most of the time. Note the threshold exceeds the prediction to reject low amounts of noise, but also to account for partial visibility that the planner ignores due to the point target assumption. After applying the thresholding, the visibility is 0.55 and 0.59, while the prediction is 0.64 and 0.74 for the first and second targets respectively. There are several sources of measurement error causing the discrepancy between the actual and predicted measurements. There is a small time error between the actual and planned trajectories, causing targets to be out of view during the transition from visible to not visible. Figure 6-15 shows image blur and overlap between targets. Motion blur causes a ghosting effect, where part of the background blends with the foreground, resulting in loss of color uniformity. The motion blurs when the observer makes sudden heading adjustments or when the vehicle encounters a sharp disturbance from an uneven ground surface. Occasional downward spikes in measurement can be attributed to motion blur. Targets can also obstruct one another in view; in the experiment, the first target obstructs the second target between 54 and 57 [s]. To account for occlusion by targets, each target needs to be treated as an occlusion. These occlusions have time-varying positions if the targets are moving. Non-uniformity of lighting and livery materials also adds high-frequency noise to the measurement. Other factors include the camera's automatic adjustments for dynamic range and white balance. The color-based effects mentioned so far ultimately lead to individual pixels falling outside the specified hue, saturation, and value color thresholds. Figure 6-4(b) shows noise speckles scattered in between the green pixels. There can also be false readings. The color thresholds may allow parts of the image which do not belong to the target livery to register as a measurement, causing small upward spikes to appear. This is despite the selection of colors for targets that stand out in the noisy color background. The noise from false readings is relatively little in this environment.



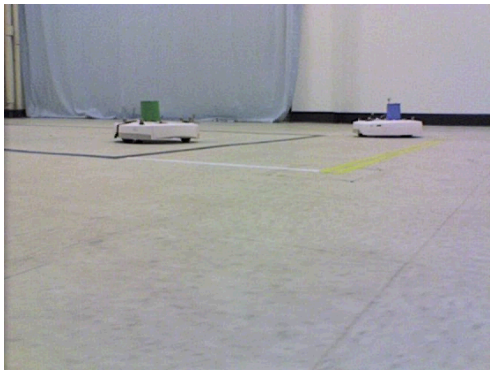
(a) Planner, T = 20



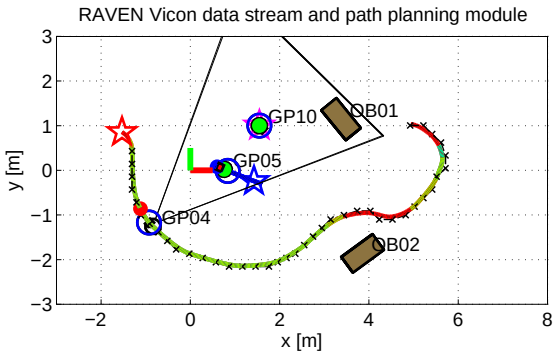
(b) Onboard view, T = 20



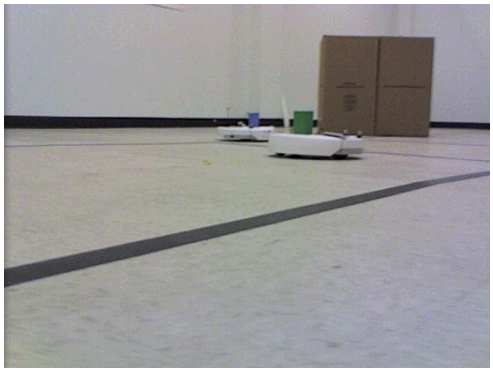
(c) Planner, T = 45



(d) Onboard view, T = 45

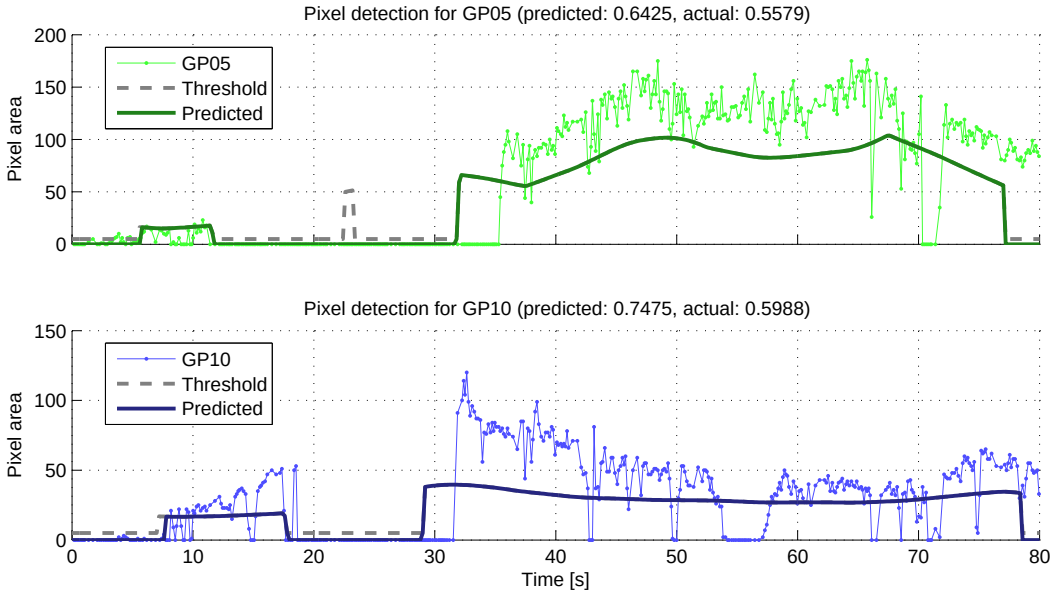


(e) Planner, T = 70

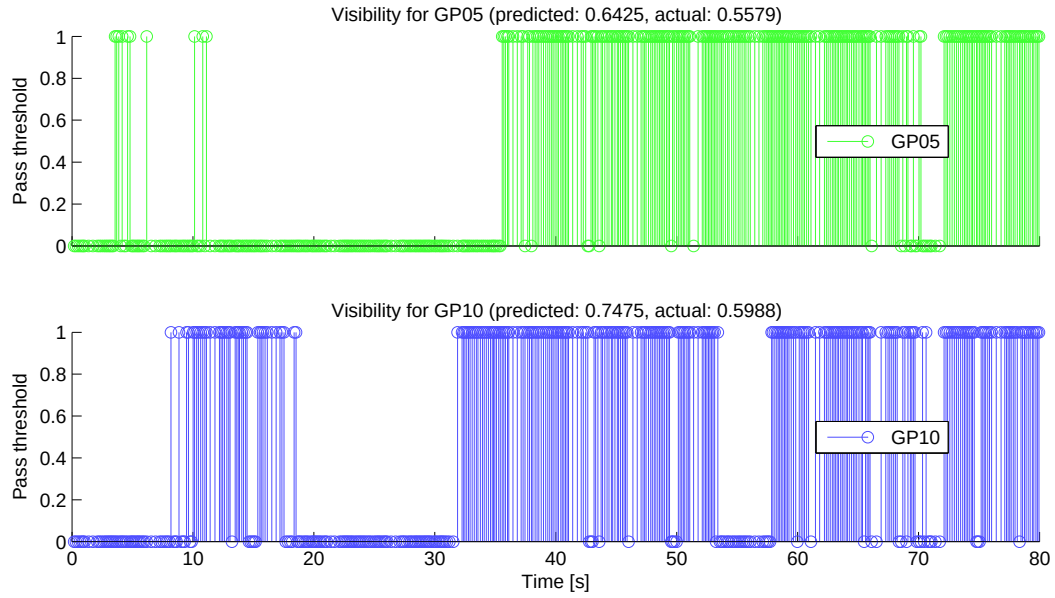


(f) Onboard view, T = 70

Figure 6-13: Moving target with ground observer in RAVEN, 2 obstacles, 2 targets.

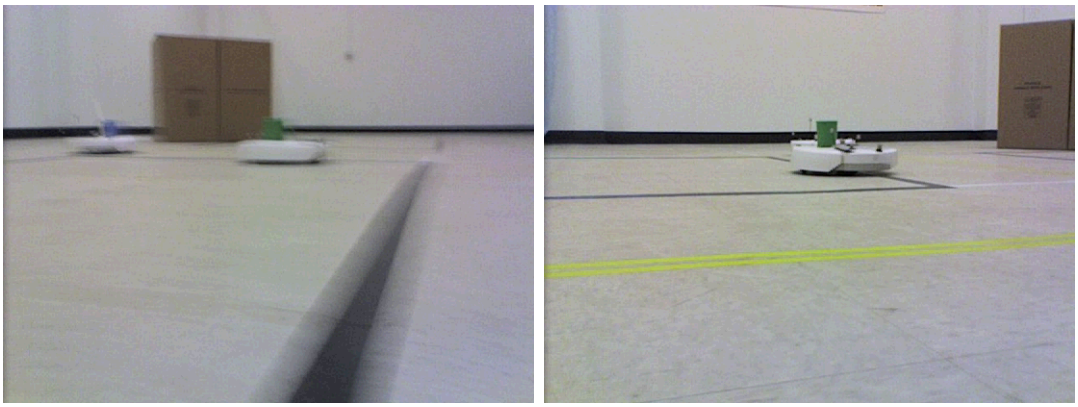


(a) Pixel detection



(b) Threshold pass

Figure 6-14: Color threshold detection, ground observer, 2 obstacles, 2 targets.



(a) Motion blur (b) Target overlap

Figure 6-15: Measurement error sources. (a) shows ghosting or motion blur causing colors to lose uniformity. (b) shows two targets, but the livery of the second target is not visible due to occlusion by the first target.

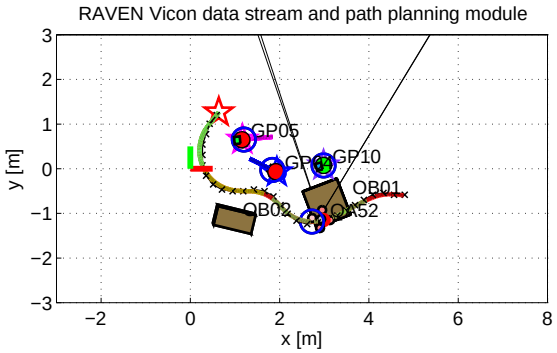
6.2.2 Aerial Observer

This section shows an example aerial observer experiment. Figure 6-16 shows a sequence of snapshots of an aerial observer, three targets, and two obstacles simulating a small urban environment for an 80-second run. The aerial observer moves at 0.1 [m/s], and the sensor is right-facing and angled 35 degrees downward from the horizontal plane. Two targets move at 0.05 [m/s] and a third is stationary. One obstacle can be cleared by the quadrotor and only obstructs visibility. The other obstacle is taller and imposes a path constraint. The targets do not obstruct the aerial observer since they are well beneath the quadrotor's altitude, so the planner can plan trajectories directly over the targets. In this example, the observer starts east of the targets and is facing westward. The plan takes the observer first over the low obstacle (OB01), but close enough to the target such that it can peer over the obstacle. It performs a wide loiter pattern around the blue target (GP10) for about 30 [s], starting at 12 [s] into the plan. After the first loiter, the observer almost immediately enters into a second loiter, this time around the green target (GP05). The target plans its loiter pattern to avoid the tall obstacle (OB02) to obtain line of sight to the green target. As the observer rounds the turn the observer achieves visibility of all three targets from approximately 60 [s] to the end. Figure 6-17 shows the measurement sequence

for the aerial observer. There is generally good agreement between the predicted and actual measurements. The quadrotor uses a waypoint-tracking controller which does not hold a constant speed during yaw (heading) actions, and so the quadrotor starts to fall behind at about 55 [s]. This gives additional visibility of the red target (GP04), but the delay also causes the green target to disappear out of view after 72 [s], resulting in a loss of measurement.

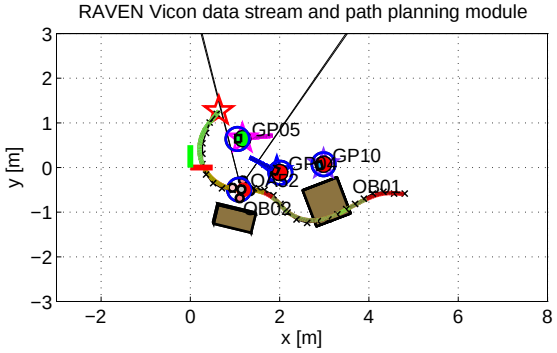
6.3 Chapter Summary

This chapter described the hardware implementation of the visibility maximization algorithm. This chapter described the core hardware and software modules that enable visibility demonstrations for ground and aerial observers, with multiple targets and cluttered environments. This chapter also presented experimental results for visibility maximization in the hardware setting. Two demonstrations, one for a ground observer and the another for an aerial observer with multiple targets and obstacles, show generally good agreement between the predicted and actual measurements from real-time onboard camera feedback. Errors which caused discrepancies between the predicted and actual were primarily position errors in the observer trajectory, as well as noise in the measurement due to motion blur, target occlusions, and various optical effects. The final chapter presents conclusions of the thesis and future work and extensions beyond this thesis.

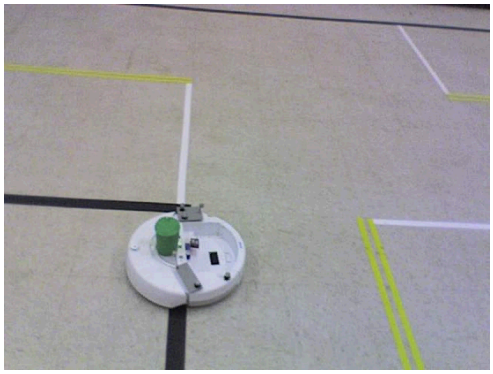


(a) Planner, T = 25

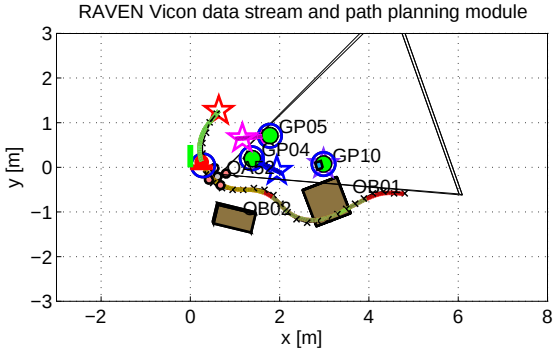
(b) Onboard view, T = 25



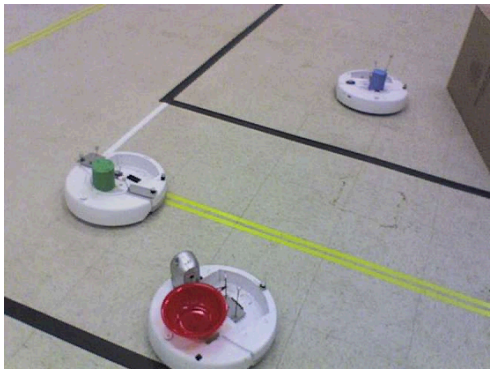
(c) Planner, T = 50



(d) Onboard view, T = 50

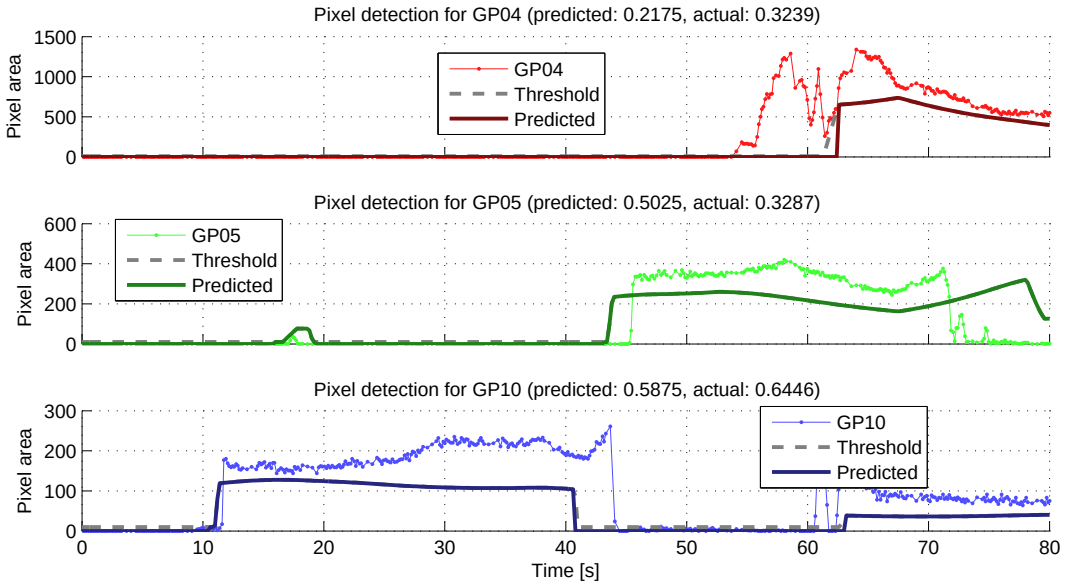


(e) Planner, T = 65

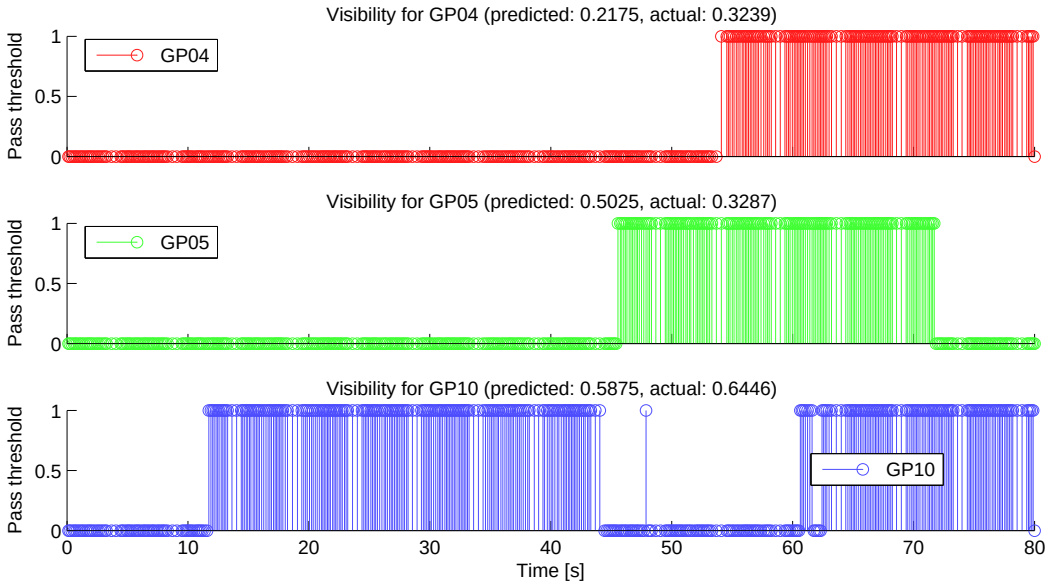


(f) Onboard view, T = 65

Figure 6-16: Moving target with aerial observer in RAVEN, 2 obstacles, 3 targets.



(a) Pixel detection



(b) Threshold pass

Figure 6-17: Color threshold detection, aerial observer, 2 obstacles, 3 targets.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis described the visibility maximization problem, which is relevant to intelligence, surveillance, and reconnaissance (ISR) missions, police and border patrol, teleoperation in robotic surgery, and environment monitoring. In these scenarios, visibility enables information gathering by a dynamically-constrained observer. The observer keeps stationary or moving targets in a constrained sensor's view and in a complex environment. The goal of the visibility maximization problem is to design the observer trajectory to maximize visibility for execution by a human operator or autonomous agent.

Many authors have considered the visibility maximization problem of designing visibility-rich trajectories. However, none have addressed the scope covered in this thesis, which considers dynamic vehicles, view-constrained sensors, 3-D environments and digital elevation models, multiple moving targets, and robustness. Also, the problem remains challenging because visibility and path planning are both computationally intensive. This thesis explains the visibility path planning challenges and provides a novel, approximate solution that addresses the complexities of the problem.

The proposed solution to the visibility maximization problem is a two-step decomposition, which decouples the visibility computation from the path planning optimization. First, visibility is approximated by discrete sampling and interpolation.

Second, the visibility approximation is input to a dynamic programming path planner and trajectory optimizer to determine the maximum visibility path.

Comparisons against a state-of-the-art optimal control solver validate the performance of the new algorithm. The new algorithm is capable of providing computation time as great as an order of magnitude, while retaining near optimality. In general, performance can be improved by increasing discretization resolution at the cost of computation time; the performance asymptotes towards optimality as computation increases. While high resolution searches still can exhibit noisy performance with respect to the choice of resolution (due to grid sensitivity and graph completeness resulting from discrete sampling), this new method provides a good approximation to the optimal solution. In fact, one of the key benefits is that a solution is still returned even for more complex scenarios, whereas the optimal control solver has to be abandoned because it fails to return a result. When visibility is binary, there are non-unique optimal solutions. To reduce the number of non-unique optimal solutions, sensor attenuation can be modeled.

While the optimal control solver can have difficulty with complex constraints, failing to return a feasible solution in these situations, the new solver handles them readily for cluttered 2-D, 3-D and DEM environments. Intuitively the solver produces loiter patterns when the sensor is sideways-looking, and fly-by patterns when the sensor faces forward.

The new solver can also address multiple target visibility. The solver uses a weighted visibility metric which accounts for the importance of individual targets, and the selection of weights and time horizons is studied. At certain weight ratios, a switching behavior occurs, whereupon the observer spends a majority of time focusing on the target with the greatest importance. Increasing the time horizon for multiple targets beyond a transition time horizon leads to periodic trajectories which retrace portions of paths which yield the maximum recurring visibility reward.

However, weighting targets in a weighted sum does not guarantee that all targets are visited, even if difficult-to-see targets are weighted very highly. To ensure visibility of every target, two techniques are used to artificially elevate the importance

of rare sightings. The max-min and diminishing returns metrics both amplify the rare sighting value using either a reward saturation of the minimum target visibility or a reward boost from initial target sighting. However these cost functions cannot be used in a dynamic programming (DP) framework because DP they do not satisfy the requirements of the Bellman equation. Instead, parametric optimization provides an alternative path planning method, enabling a wide range of cost functions such as the rare sighting objectives. Parametric optimization can find visibility maximal paths that are parameterized by a small vector of variables, such as circles, ellipses, and racetrack patterns. This kind of non-convex optimization requires metaheuristic optimization methods such as cross entropy and genetic algorithms. Parametric optimization sacrifices path intricacy and their associated visibility-maximal plans, for objective function complexity and at the same time gaining computation time benefits. Results show that DP paths outperform parametric optimization as expected from a visibility maximization standpoint. However there is no comparison for the max-min and diminishing returns metric. For these cost functions, the parametric optimization captures the rare sighting event very effectively, suggesting that parametric optimization is useful for achieving rare sighting visibility among other reasons.

Realistically, moving targets also need to be considered. The proposed solver can readily handle time-varying problems by a time parameterization extension, and it is shown that single moving targets lead to paths which pursue the target. The path for sideways-facing sensors consists of loiter and pursuit phases. The loiter phase is a tear drop pattern which repeats periodically. The pursuit phase allows the observer to catch up with the target. Targets which travel at a fraction of the observer speed lead to large tear drops corresponding to long loiter phases. Increasing the speed ratio leads to smaller tear drops, representing shorter loiter phases. Once the speed ratio is equal, the tear drops shrink completely and the path only pursues the target. When the speed ratio exceeds 1, the observer performs a rendezvous maneuver which intercepts the target at a point before being outrun by the target. The maneuvers can be influenced by the initial condition and the layout of the environment, for example if the observer starts too far away from a target and the target moves faster, then it

might not possible for the observer to catch up.

When multiple moving targets are considered, the solver reveals that complex behavior emerges, and these depend heavily on the speed ratios and the target importance factors. When targets diverge in position, the observer pursues the target which results in the greatest returns, through a combination of visibility and importance weighting. When targets move in unison, a large fraction of space enclosed by the target cluster is maintained in view. This latter result highlights an important outcome that is used later: if a point target is treated as a cloud of spatially offset targets, robust visibility behavior is observed.

Naturally, robust visibility is an important consideration because there can be uncertainty in modeling, and trajectories can be subject to disturbances. When target motion is uncertain, targets can be represented as regions. However, when the observer trajectory is uncertain, the notion of observer state neighborhoods provides a similar benefit to robust performance. As noise increases, the robust plan maintains consistent performance while the nominal plan underperforms.

This thesis also explores the validity of the developed algorithms by validating their implementation in hardware. Experimental results are obtained using the MIT Aerospace Controls Laboratory testbed for autonomous indoor vehicles. The hardware demonstrations show that visibility-rich paths can be designed while accounting for complex observer dynamics, 3-D environments, and multiple moving targets in the presence of uncertainty. These paths show some correspondence between the predicted and actual measurement, and that target and observer motion uncertainty plays a large role in affecting whether targets are visible to the sensor or not.

7.2 Future Work

Future work in the direction of improving the theoretical understanding of the visibility maximization problem, includes studying the conditions for optimality and sensitivity in the visibility cost function. One interesting theoretical area may be the relation between the gradients in the visibility function and the action sequences of

the observer; smooth loiter patterns appear to follow the direction which minimizes losses in visibility. Another theoretical consideration may be the mathematical characterization of loiter and pursuit patterns in moving target visibility for simple target trajectories.

Future work should also address the current limitations of the proposed solver in solving the visibility maximization problem, by considering extensions to accommodate richer objective functions that are useful for generating behaviors more suited for multiple target and rare target visibility, pursuit-evasion visibility, information maximization. Other visibility-related work involves using multiple observers to increase the amount of visibility, or to reduce the amount of gaps in coverage. Performance and computation improvements may benefit from a deeper theoretical understanding of the visibility maximization problem.

From a modeling perspective, improvements to environment representations can alleviate computation requirements. From a path planning perspective, in both the DP and receding horizon plans, the terminal state should be part of an invariant safe set, which ensures if no new plan is found, the vehicle can remain in the safe set of states under certain assumptions. Also, in the receding horizon framework, the choice of path after completing one window should balance the immediate reward with exploring unvisited parts of the state space; this balance might be formulated in the terminal cost. Learning these weights as the mission progresses may also provide better guidance to the planner.

From a practical perspective, a formal model of uncertainty should be considered in the visibility maximization problem, since targets in real-life situations do not have well-known behaviors. Robustness to disturbances in observer trajectories should also be considered.

Additional improvements to hardware implementations include more accurate color thresholding and pixel area approximations. As well, updated low-level control wrappers will be useful for approximating fixed-speed aircraft using quadrotors.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A

Visibility Formulation

The first component in the target tracking solver black box shown in Figure 2-2 is *visibility*, which is the objective quantity to be maximized and can be calculated as a result of the interaction of the four input models. In this section, the interactions are formalized. Recall the functional dependence of visibility on these models, shown below again mathematically.

$$\mathcal{V} = f[\mathbf{x}_A(t), \mathbf{x}_T(t), \mathcal{T}(t), \mathcal{S}(\mathbf{x}_A(t), \mathbf{x}_T(t))]$$

A.1 Definition of Visibility, and Necessary and Sufficient Conditions

Visibility in the context of a light-based sensor is the line of sight between a sensor $\mathcal{S}$ coupled to a vehicle with state $\mathbf{x}_A(t)$, and one or more point targets with states $\mathbf{x}_{T_m}(t)$, subject to constraints by the environment $\mathcal{T}$. The line of sight regions can be thought as being illuminated by a light source representing either the observer or the target. Figure A-1 shows the line of sight visible region of the target.

The visibility metric $\mathcal{V}$ is the Boolean intersection of **two necessary and sufficient conditions for visibility**:

1. The target state lies inside the visibility region of the observer (*line-of-sight*

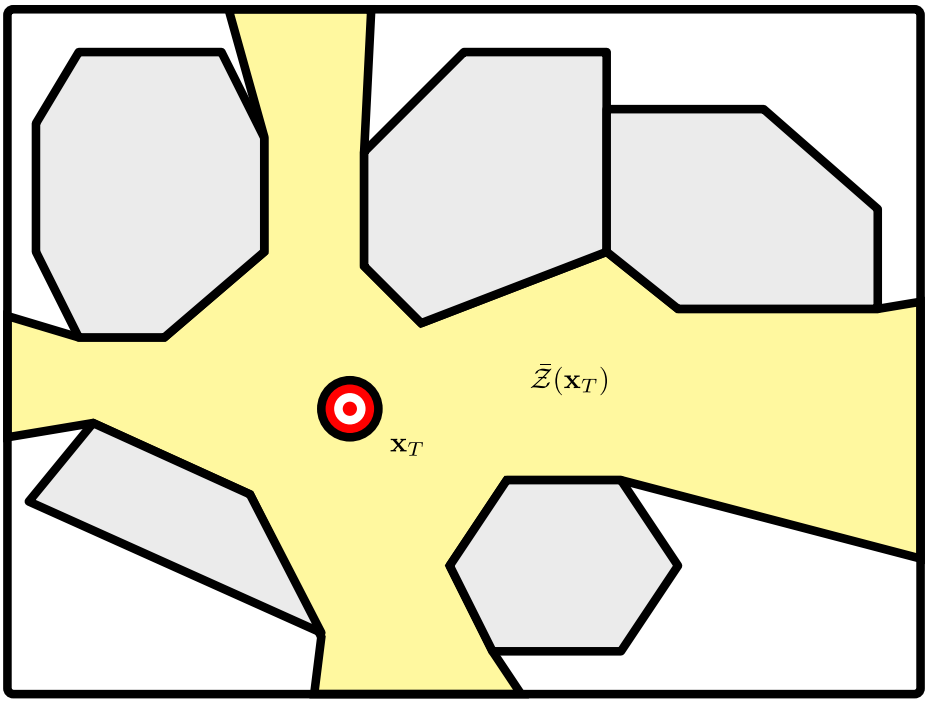


Figure A-1: Line-of-sight visible region of the target.

visibility condition):

$$\mathbf{x}_T \in \bar{\mathcal{Z}}(\mathbf{x}_A) \triangleq \{\mathbf{x} \colon \exists \text{ line of sight from } \mathbf{x} \text{ to } \mathbf{x}_A\} \tag{A.1}$$

An equivalent condition is for the observer's state to lie inside the target's visible region:

$$\mathbf{x}_A \in \bar{\mathcal{Z}}(\mathbf{x}_T) \triangleq \{\mathbf{x} \colon \exists \text{ line of sight from } \mathbf{x} \text{ to } \mathbf{x}_T\} \tag{A.2}$$

2. Target point lies inside the sensor set (*sensor footprint condition*):

$$\mathbf{x}_T \in \mathcal{S}(\mathbf{x}_A) \triangleq \{\mathbf{x} \colon d(\mathbf{x}, \mathbf{x}_A) \in \mathcal{D}, \theta(\mathbf{x}, \mathbf{x}_A) \in \Theta, \eta(\mathbf{x}, \mathbf{x}_A) \in \mathcal{E}\} \tag{A.3}$$

The methods used to calculate these regions are described in Section A.2. Visibility

is thus defined as

$$\mathcal{V}(\mathbf{x}_A, \mathbf{x}_T, \mathcal{T}, \mathcal{S}) = [\mathbf{x}_A \in \bar{\mathcal{Z}}(\mathbf{x}_T)] \cap [\mathbf{x}_T \in \mathcal{S}(\mathbf{x}_A)] \tag{A.4}$$

A careful distinction between *occlusions* and *obstructions* is made here. In the visibility problem, occlusions refer specifically to parts of the environment which block line of sight but *not* vehicle passage; and obstructions exactly the reverse – parts of the environment which obstruct vehicle passage but do not hinder line of sight. *Transparent solids* such as clear glass, and *no-fly zone* boundaries are non-occluding but obstructing; *opaque gases* such as smoke and clouds are occluding but non-obstructing. Most physical objects are simultaneously occluding and obstructing; these will simply be referred to as *obstacles*, if necessary the distinctions *visibility obstacle* and *path obstacle* to represent occlusions and obstructions respectively will be used. Large terrain features such as mountains, tall forests, and buildings are all obstacles to surveillance aircraft.

A.2 Visibility Models

A.2.1 Planar Visibility Model with Point Target

- **Condition 1:** Line-of-sight visibility

  The line of sight visibility condition can be calculated in a number of ways. Several methods are listed below [49]:

  1. *Visibility polygons*

     In Condition A.1, observe that *the line of sight visible region of the observer* $\bar{\mathcal{Z}}(\mathbf{x}_A)$ *is also equivalent to the* visibility polygon $\mathcal{VP}(\mathbf{x})$ *of a light source at the same position* $\mathbf{x}$ *as the observer*, in other words $\bar{\mathcal{Z}}(\mathbf{x}_A) \equiv \mathcal{VP}(\mathbf{x}_A)$. This polygon may be non-convex. Once the visibility polygon is calculated, a check for whether the target state $\mathbf{x}_T$ lies inside this polygon yields the result of the first condition.

Alternatively one can evaluate Condition A.2. In fact, it makes sense to calculate the visibility polygon of the target instead of the observer, in anticipation of the optimization phase of the target tracking problem. A visibility polygon must be computed either for every observer location *or* every target location. Since the optimization is a search over the observer location, with fixed (even time varying) target positions, it is computationally cheaper to evaluate visibility polygons for the targets.

The *VisiLibity* library [88] is an open source C++ toolbox that accepts a list of vertices defining edges of the occluders, and returns the visibility polygon of a light source at $\mathbf{x}$.

2. *Visibility shadow regions*

   The negative region to the visibility polygon is known as the *visibility shadow region*, $\mathcal{Z}(\mathbf{x})$. Line-of-sight visibility exists if $[\mathbf{x}_T \notin \mathcal{Z}(\mathbf{x}_A)] \equiv \left[\mathbf{x}_T \in \bar{\mathcal{Z}}(\mathbf{x}_A)\right]$, or identically $[\mathbf{x}_A \notin \mathcal{Z}(\mathbf{x}_T)] \equiv \left[\mathbf{x}_A \in \bar{\mathcal{Z}}(\mathbf{x}_T)\right]$. This region can be disjoint and unbounded: it is the union of the shadow regions of individual convex occluders. The shadow region of a convex occluder is the interior of a *shadow frontier* and exactly two *extreme rays*.

   The procedure to calculate the visibility shadow region of $\mathbf{x}_T$ is:

   (a) Calculate j shadow regions. First, find the extreme vertices of the obstacle, which have the maximum and minimum angles measured in polar coordinates with the origin at the target location. Then, all vertices between the maximum and minimum extreme vertices, counted in the counterclockwise direction, form the *shadow frontier*. Then, the shadow region is formed by extending rays from the extreme vertices of the shadow frontier out to infinity, with the lines collinear with the rays crossing through the origin.

   (b) Given the j^{th} shadow region, determining whether a point lies in the convex unbounded shadow region requires evaluating cross products against all edges. An edge is defined by the line through the edge

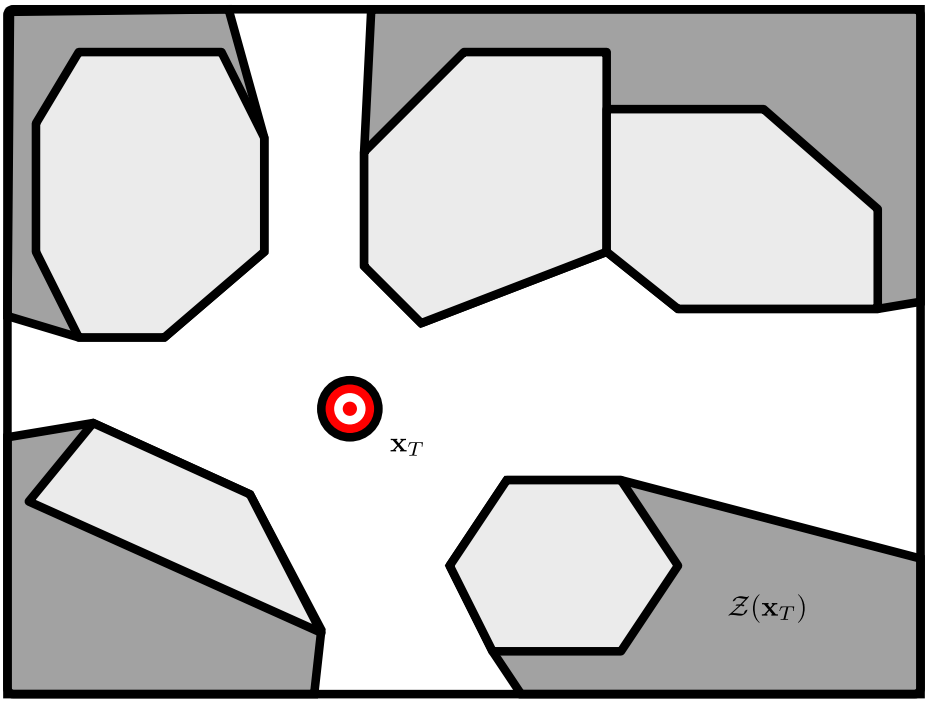


Figure A-2: Visibility shadow region.

vertices, $\overline{P_{j,m}P_{j,m+1}}$. Assume a convention of vertices stored in counterclockwise order. Positive cross products indicate $\mathbf{x}$ lies inside the half-space of the edges. Negative cross products indicates the opposite, and zero corresponds to coinciding with the edge. Optionally, map the negative and positive reals of the cross product into the binary variable {0,1} respectively, denoting whether $\mathbf{x}$ is inside or outside this region. All half-space checks must be true to be inside.

(c) Repeat for all $j = 1, \cdots, N_{occ}$ occluders. Perform a union of all individual shadow regions to create the total shadow region $\mathcal{Z}$.

Invert the total visibility shadow region $\mathcal{Z}$ to obtain the LOS visibility region $\bar{\mathcal{Z}}$.

$$\bar{\mathcal{Z}} = (1 - \mathcal{Z}) = 1 - \bigcup_{j=1}^{N_{occ}} \left\{ \bigcap_{m=1}^{M_j} \left[(\overline{P_{j,m}P_{j,m+1}} \times \overline{P_{j,m}\mathbf{x}_A}) > 0 \right] \right\} \tag{A.5}$$

- **Condition 2:** Sensor footprint

For a simple range-limited, field-of-view-limited sensor as shown in Figure 2-4, the range d and bearing θ between the target and sensor are checked against the limits $[d_{min}, d_{max}]$ and $[\theta_{min}, \theta_{max}]$. Range and bearing are given by the first three elements of the aircraft and target states (namely x, y, z):

$$d(\mathbf{x}_A, \mathbf{x}_T) = \|\mathbf{x}_A - \mathbf{x}_T\|_2 \tag{A.6a}$$

$$\psi_{\text{LOS}}(\mathbf{x}_A) = \psi_A + \theta_{\text{center}} \tag{A.6b}$$

$$\hat{\mathbf{x}}_{\text{LOS}}(\mathbf{x}_A) = [\cos\psi_{\text{LOS}}, \sin\psi_{\text{LOS}}, 0]^T \tag{A.6c}$$

$$\theta(\mathbf{x}_A, \mathbf{x}_T) = \cos^{-1}\left(\frac{(\mathbf{x}_A - \mathbf{x}_T) \times \hat{\mathbf{x}}_{\text{LOS}}}{\|\mathbf{x}_A\|\|\mathbf{x}_T\|}\right) \tag{A.6d}$$

A.2.2 Planar Visibility Model with Target Region

- **Condition 1:** Line-of-sight visibility

  One metric is to evaluate the *line-of-sight visible overlap* of the visibility region with the target region. This represents a situation where the sensor is slightly elevated above the plane of the target region. This is also an example of robustness consideration in the tracking algorithm. To find the area of overlap, the visibility polygon of the observer must be used to intersect with the target region. Note that the visibility polygon for a target region is not used because its definition is not straightforward[1].

  Other models that represent sensors in the plane, such as a planar LIDAR, should involve a frontal area calculation instead of a region overlap.

- **Condition 2:** Sensor footprint Using the first metric (area of overlap), this area must be intersected with the sensor footprint to calculate the *sensor visibility overlap*. The sensor visibility overlap can be disjoint regions depending on the positions of the obstacles and targets.

[1] One would have to specify, for example, level set visibility polygons which define the contour at which a specific fraction of the target region is visible.

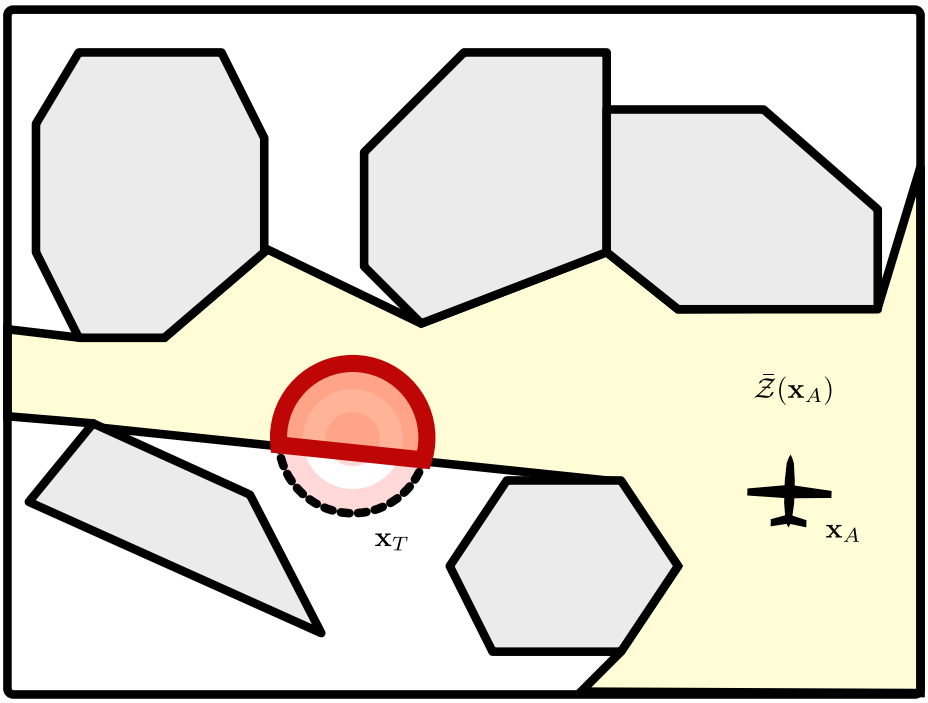


Figure A-3: Line-of-sight visible overlap region.

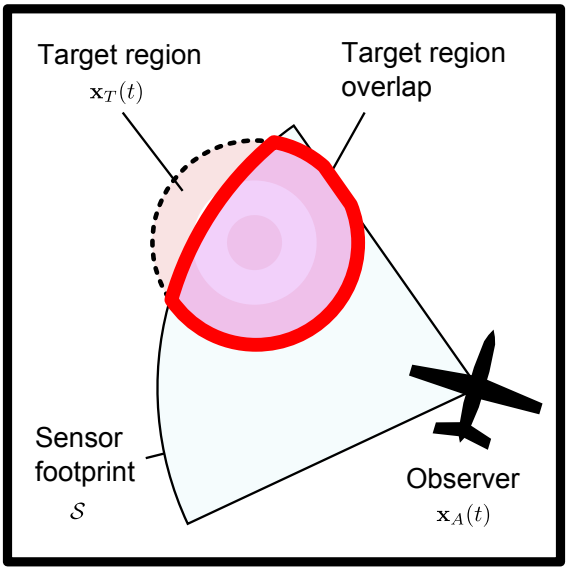


Figure A-4: Sensor visibility overlap region.

A.2.3 3-D Visibility Model

- **Condition 1:** Line-of-sight visibility

  The method in this section is used to construct the visibility shadow region of a target $\mathcal{Z}(\mathbf{x}_T)$ in three dimensions. A visibility shadow region is shown in Figure A-5.

  Line of sight is again evaluated by checking against the occluding surfaces in the environment. A 3-D environment can be represented by triangular patches, where each triangle is specified as three points (P_1, P_2, P_3) in $\mathbb{R}^3$. The shadow region of the triangular occluder is the *shadow volume*.

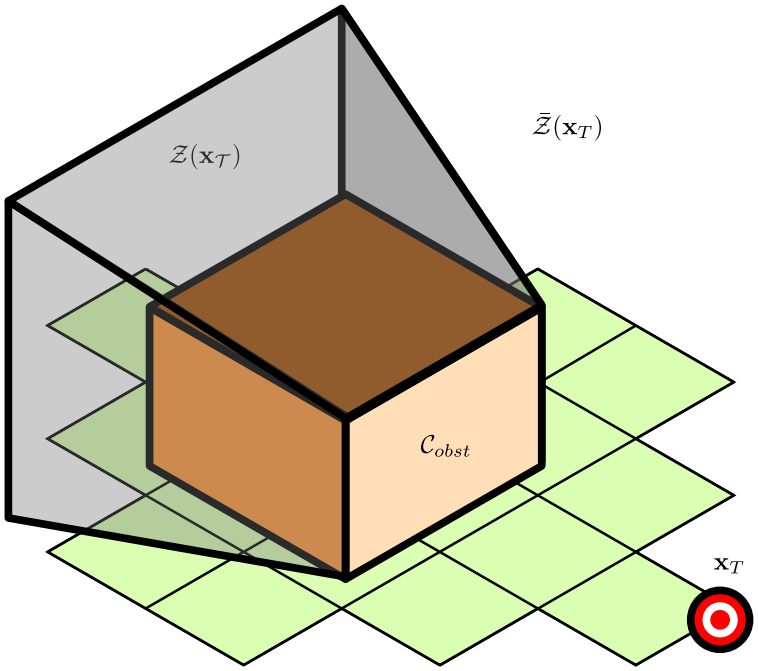


Figure A-5: Visibility and shadow regions in 3-D.

To calculate shadow volumes, light normals and half-space intersections are used:

1. **Light normals:**

   A *light normal* is the normal vector of the plane spanned by the vectors of the triangular patch. It is called a light normal because the direction

points towards the half-space containing the light source. In visibility, the light source is the target.

2. **Shadow volume half-spaces:**

   Each triangular patch creates three additional surfaces which enclose the shadow volume. To form a shadow surface, take two of three triangle points and the target point. The volume is shown in Figure A-6.

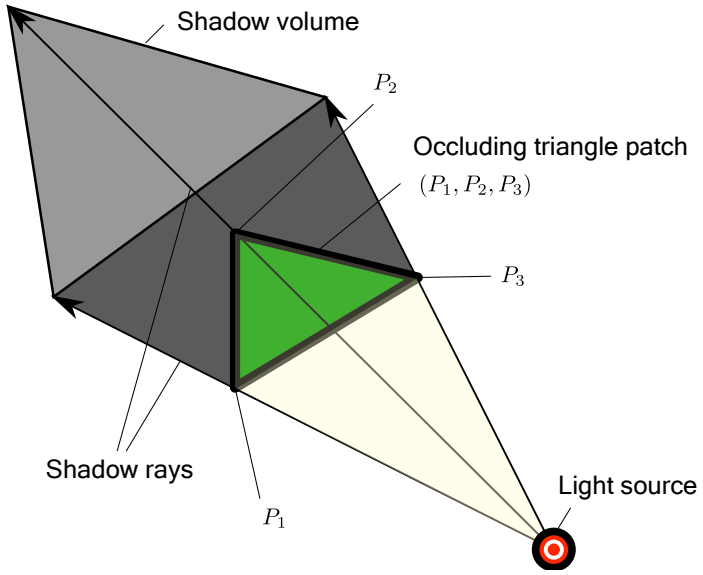


Figure A-6: Light normal and shadow volume.

3. **Visibility shadow region:**

   The visibility shadow is the union of the shadow volumes.

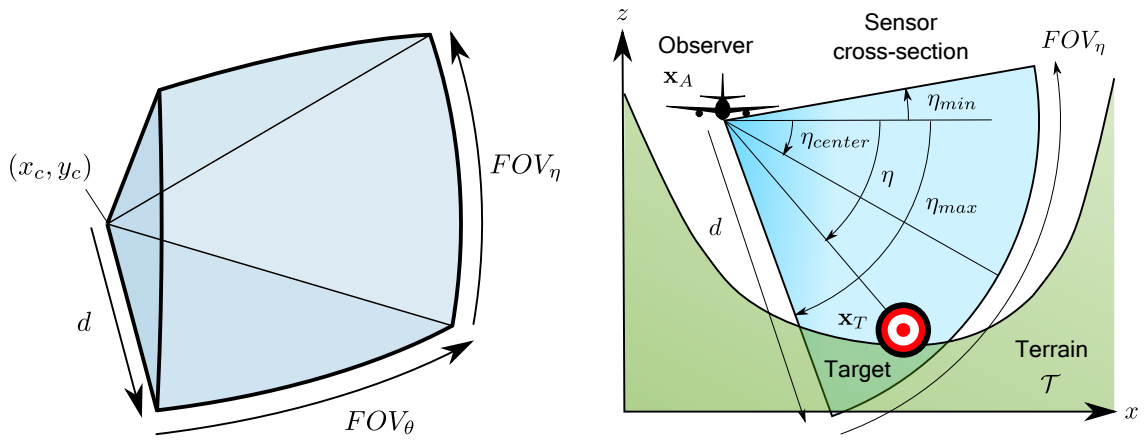
- **Condition 2:** Sensor footprint

  In $\mathbb{R}^3$, a sensor has range $d_{\max}$, horizontal field-of-view θ_{FOV}, and vertical field-of-view η_{FOV}, as shown in Figure A-7. The sensor has a bearing angle ψ_{LOS} offset from the vehicle heading, and an inclination angle η_{LOS} offset from the vehicle bank angle.

$$\psi_{\text{LOS}} = \psi + \theta_{\text{center}} \tag{A.7a}$$

$$\eta_{\text{LOS}} = \gamma + \eta_{\text{center}} \tag{A.7b}$$

The relative azimuth and elevation bearings of the target from the sensor are $\theta(\mathbf{x}_A, \mathbf{x}_T)$ and $\eta(\mathbf{x}_A, \mathbf{x}_T)$.



(a) 3-D sensor model

(b) Vertical (side) cross section of sensor model

Figure A-7: 3-D sensor model with limited range, horizontal FOV, and vertical FOV.

A.2.4 Elevation Model in Visibility

Elevation models are a specific instance of the 3-D environment. The same methods can be applied. However, more efficient routines are available when the model is uniformly gridded.

- **Condition 1:** Line-of-sight visibility

  1. Ray tracing and terrain model

     The altitude map can be treated as a triangular mesh, but this is unnecessary. A linear-time algorithm by [52] can evaluate visibility between two points on the grid efficiently.

- **Condition 2:** Sensor footprint

  The same 3-D sensor footprint model used in the 3-D visibility scenario can be used for elevation models.

A.3 Target Region Intersection Calculation

A.3.1 Translating Target Samples

The robust treatment uses a fuzzier notion of visibility by maximizing the sensor overlap with a target region. Instead of maximizing the time integral of a binary visibility metric, the region overlap optimization is posed according to Eq. 5.3. Figure A-8 shows a target region evolving over time. The intersection area or volume between

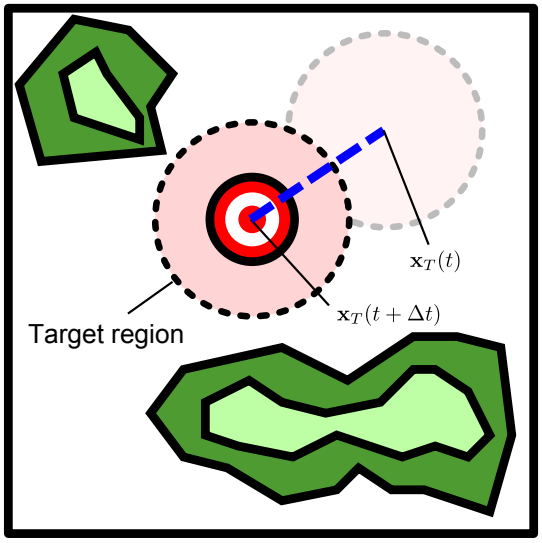


Figure A-8: Moving target region.

sensor and target can be calculated using exact or numerical integration. However, the intersection computation is considerably more expensive compared to calculating the visibility of a single point target.

Instead, a sampling approach can be taken, approximating the 2-D or 3-D target region using a cloud of samples. Then, each sample is treated as a separate point target in the multiple-target visibility maximization formulation. Figure A-9 shows samples taken from the target region.

This approach can be much faster than calculating exact intersection areas and volumes. The number of samples must be small to maintain tractability. The total number of samples is $N_T N_S$, compared to the original moving target formulation which requires N_T visibility table evaluations. Samples can be drawn from a uniform

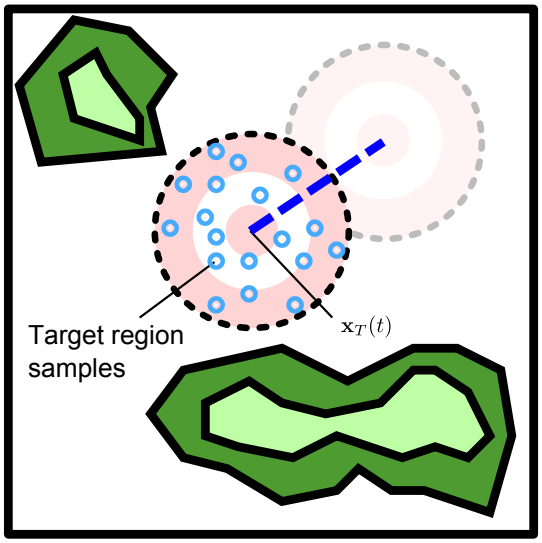


Figure A-9: Sampling over the target region.

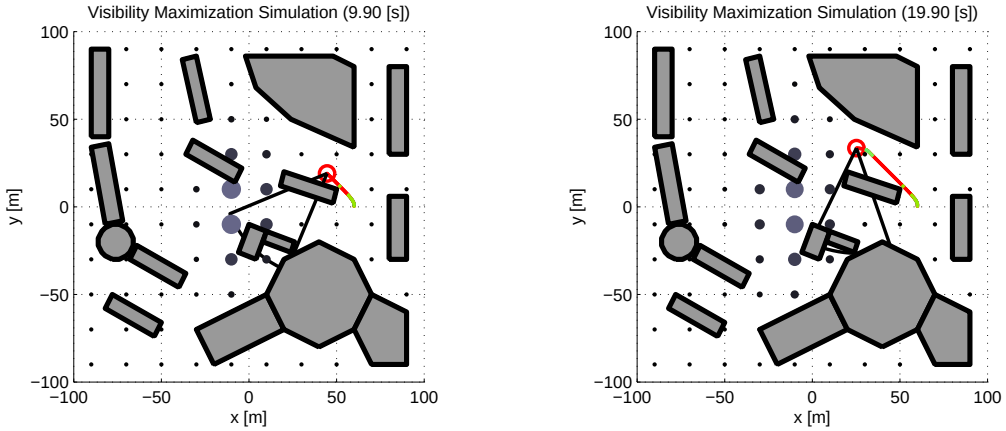
distribution or other density function. They can also be deterministically generated.

A.3.2 Uniformly Spaced Samples

Uniformly spaced samples are analogous to occupancy grids. The visibility maximization solver can also represent the target in an occupancy grid manner. This section presents preliminary results for this implementation.

Figure A-10 shows visibility maximization using static target samples, where the target position uncertainty translation and growth versus time is captured in an importance weighting update process. In this example, the target is moving to the left, and the observer attempts to move in the same direction.

Future work will compare the performance versus computation of the occupancy grid method versus the particle method which was analyzed in Section 5.2.2.



(a) T = 10 s

(b) T = 20 s

(c) T = 30 s

(d) T = 40 s

Figure A-10: Visibility maximization with static targets and weight evolution.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix B

Path Parameterization and Parametric Optimization

B.1 Path Parameterization

If the path is constrained to be a closed contour in the plane, e.g. a circle or ellipse at fixed altitude, and the path is independent of time, the optimization can be posed as a search over the parameter space $\mathbf{x}$ of the path. Assuming a sensor with sufficient vertical field of view FOV_η, the aircraft state can further be simplified so that roll and pitch are constant. Recall that the original objective function is normalized by the total flight time T. The objective function is no longer dependent on time but rather a path parameter $\theta \in [0, 2\pi]$, which represents the polar coordinate of the position on the path, with respect to the path center. The states along the path are $\mathbf{x}_A(\theta) = [x_A(\theta), y_A(\theta), \psi_A(\theta)]^T$. Assuming a vehicle that travels at constant speed, an equivalent normalization is to use distance s instead of time, and the integration takes place over ds instead of dt. To relate ds to $d\theta$, an arc length integral $L(r(\theta), \theta)$ (the polar coordinate transformation of $L(\mathbf{x}_A(\theta))$, where $r(\theta) = \sqrt{x(\theta)^2 + y(\theta)^2}$) is

taken and then differentiated with respect to θ:

$$s = L(\mathbf{x}_A(\theta)) = \int_0^{2\pi} \sqrt{r(\theta)^2 + \left(\frac{dr(\theta)}{d\theta}\right)^2}\, d\theta \tag{B.1a}$$

$$\frac{ds}{d\theta}(\theta) = \dot{s}(\theta) = \sqrt{r(\theta)^2 + \left(\frac{dr(\theta)}{d\theta}\right)^2} \tag{B.1b}$$

$$ds = \dot{s}(\theta) d\theta \tag{B.1c}$$

After the coordinate transformation, the objective function becomes:

$$\max J_V = \frac{1}{s} \sum_{m=1}^{N_T} w_m \int_0^{2\pi} \mathcal{V}_m(\mathbf{x}_A(\theta), \mathbf{x}_T(\theta), \mathcal{T}, \mathcal{S}(\theta)) \dot{s}(\theta) d\theta \tag{B.2}$$

Two approximations to the objective function are used: the first converts the visibility integral into a discrete sum over N_θ segments, and the second uses the visibility function approximation (i.e. interpolation) at each state along the path for each θ_k.

$$\max J_{V,\text{discrete}} = \left(\frac{1}{\sum_{k=1}^{N_\theta} \dot{s}(\theta_k)\Delta\theta_k}\right) \sum_{m=1}^{N_T} w_m \sum_{k=1}^{N_\theta} \mathcal{V}_m(\theta_k)\dot{s}(\theta_k)\Delta\theta_k \tag{B.3}$$

The state equation $\mathbf{x}_A(\theta)$ can be replaced by the parameter vector $\mathbf{x}$. The aircraft state $\mathbf{x}_A$ can be recovered via the parametric transformation

$$\mathbf{x}_A(\theta) = A(\mathbf{x}, \theta) \tag{B.4}$$

where A is a vector of nonlinear transformation equations depending on the type of path. For a circle, the parameter vector $\mathbf{x}_{\text{circle}}$ is $[x_c, y_c, r]^T$ and the transformation vector is

$$A_{\text{circle}}(\mathbf{x}_{\text{circle}}, \theta) = \begin{bmatrix} x_c + r\cos(\theta) \\ y_c + r\sin(\theta) \\ \theta + \pi/2 \end{bmatrix} \tag{B.5}$$

For an ellipse, the parameter vector $\mathbf{x}_{\text{ellipse}}$ is $[x_c, y_c, a, b, \phi]^T$ and the transformation vector is

$$A_{\text{ellipse}}(\mathbf{x}_{\text{ellipse}}, \theta) = \begin{bmatrix} x_c + a\cos(\theta)\cos(\phi) - b\sin(\theta)\sin(\phi) \\ y_c + b\cos(\theta)\sin(\phi) + b\sin(\theta)\cos(\phi) \\ \tan^{-1}\left(-\frac{b^2 x_A(\theta)}{a^2 y_A(\theta)}\right) \end{bmatrix} \tag{B.6}$$

Examples of parametric paths are shown pictorially in Figure 4-6. A special case of the ellipse is the circle, when $a = b$. The domains of each parameter are restricted, so that dynamic constraints are respected. Appendix B.2 outlines the implementation of parametric path optimization.

B.2 Parametric Optimization Methods

For nonlinear and non-convex objective functions, the family of metaheuristic search algorithms including cross entropy, genetic algorithms, and simulated annealing can be applied [89]. Metaheuristic optimization attempts to search a complex objective for the global optimum and almost always returns a local optimum when a limited time budget is given. In this paper, the cross entropy method [74] is chosen because of its straightforward implementation and empirically-observed fast rate of convergence to good local optima, and validated via Monte Carlo simulations. Details of the algorithm are provided below.

B.2.1 Simulated Annealing

An initial candidate solution is randomly selected. A neighborhood of solutions around the candidate is assigned probabilities of selection based on their value. A temperature function determines the probability of selecting candidates with lower value. The temperature "cools" with time, lowering the probability of selecting lower value candidates. The process repeats until to some terminating condition, such as sufficient cooling or after a specified number of iterations.

B.2.2 Genetic Algorithm

An initial population of samples is selected randomly. The value, or "fitness" of each sample is evaluated. Those with the best fitness have increased probability of survival during the selection phase. A reproduction phase inherits properties ("genes") from the entire set of properties ("chromosomes") of the mating samples to generate new samples known as offspring. Termination occurs after a fixed number of generations are bred, or when the fitness reaches an equilibrium.

B.2.3 Cross Entropy

Parameters are represented as probability distributions. Samples are drawn randomly at the start. The best fraction are used to form new distributions. These distributions are used to generate new samples. The process repeats until some convergence tolerance is reached, or after some number of iterations.

B.2.4 Tabu Search

Tabu search evaluates a single candidate solution and chooses its best neighbors; however when no improvement is seen, a restart occurs; if values improve after restart, the lower values are marked as taboo and will not be revisited.

B.2.5 Ant Colony Optimization

An initial population of random samples is chosen. Each sample leaves behind "pheromones" representing a probability that a similar sample will be drawn on a subsequent examination. Better samples leave behind more pheromones, because the same sample will be traversed more often (as in a shorter path towards food sources). The greater the pheromone intensity, the higher the probability of future samples that follow the same path.

B.3 Cross Entropy Implementation

In cross entropy (CE), the parameter vector $\mathbf{x}$ is represented as a vector of probability distributions $\mathbf{f}[k]$ with parameters $(\mathbf{x}_\mu, \mathbf{x}_\sigma)$ at each iteration k. The goal is to draw n random samples $\mathbf{x}_i[k], i \in [1, \ldots, n]$ from these distributions, choose the best samples to update the distributions, and iterate until the distributions converge on a single point in the parameters, i.e. $(\mathbf{x}_\sigma < \mathbf{x}_{\sigma,tol})$.

Figure B-1 illustrates the evolution of the sampling parameters with each iteration. The pseudocode for the CE method is presented in Algorithm 3. A detailed outline of the algorithm is shown in Algorithm 4.

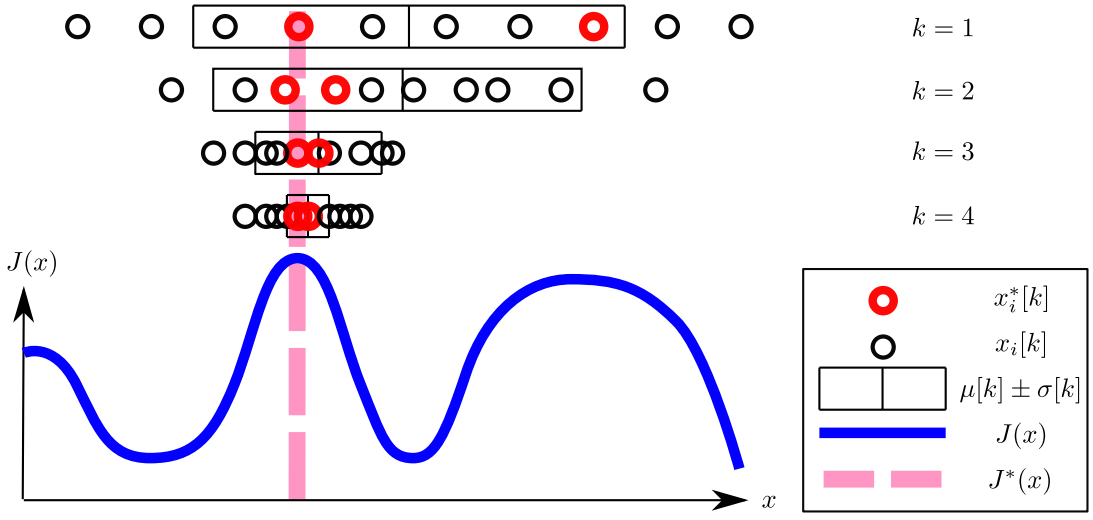


Figure B-1: Illustration of sampling parameter evolution in 1-D cross entropy optimization example.

Algorithm 3 Pseudocode for cross entropy optimization.

```
INITIALIZE f[k = 1](x_μ, x_σ)
while (x_σ > x_σ,tol) and (iteration limit not exceeded) do
    DRAW n random samples from f[k](x_μ, x_σ); reject samples which violate constraints
    EVALUATE J_V,discrete(x_i[k]), i ∈ {1, ..., n}
    SORT J_V(x_i)
    UPDATE f[k + 1](x_μ, x_σ) from best samples
    Update counter: k ← k + 1
end while
return x_best
```

Algorithm 4 Algorithm: using cross entropy for visibility maximization.

1: INITIALIZE $\mathbf{f}[k=1](\mathbf{x}_\mu, \mathbf{x}_\sigma)$

$$\mathbf{x}_\mu[k=1] = \frac{1}{2}(\mathbf{x}_{min} + \mathbf{x}_{max})$$
$$\mathbf{x}_\sigma[k=1] = \frac{1}{6}(\mathbf{x}_{max} - \mathbf{x}_{min})$$

2: **while** $(\mathbf{x}_\sigma > \mathbf{x}_{\sigma,tol})$ and (iteration limit not exceeded) **do**

3: DRAW n random samples from $\mathbf{f}[k](\mathbf{x}_\mu, \mathbf{x}_\sigma)$; reject samples which violate constraints

$$\mathbf{x}[k] = RandomNumberFromGaussian(\mathbf{x}_\mu, \mathbf{x}_\sigma)$$

4: EVALUATE $J_{V,\text{discrete}}(\mathbf{x}_i[k]), i \in \{1, \ldots, n\}$

$$J_{V,\text{discrete}} = \left(\frac{1}{\sum_{k=1}^{N_\theta} \dot{s}(\theta_k)\Delta\theta_k}\right) \sum_{m=1}^{N_T} w_m \sum_{k=1}^{N_\theta} \mathcal{V}_m(\theta_k)\dot{s}(\theta_k)\Delta\theta_k$$

$$\mathbf{x}_A(\theta) = A(\mathbf{x}, \theta)$$

$$V_{\text{interp}} = \{[V_1(1-i_x) + V_2(i_x)](1-i_y) + [V_3(1-i_x) + V_4(i_x)](i_y)\}(1-i_z) + \{[V_5(1-i_x) + V_6(i_x)](1-i_y) + [V_7(1-i_x) + V_8(i_x)](i_y)\}(i_z)$$

5: SORT $J_V(\mathbf{x}_i)$

Call $SortingRoutine(J_V(\mathbf{x}_i))$ to order values from best to worst

6: UPDATE $\mathbf{f}[k+1](\mathbf{x}_\mu, \mathbf{x}_\sigma)$ from best samples

$$\mathbf{x}_\mu[k'] = \frac{\sum_{j=1}^{\rho n} \mathbf{x}_{sort,j}[k]}{\rho n}$$
$$\mathbf{x}_\sigma[k'] = \sqrt{\frac{\sum_{j=1}^{\rho n} (\mathbf{x}_{sort,j}[k] - \mathbf{x}_\mu[k'])^2}{\rho n}}$$
$$\mathbf{x}_\mu[k+1] = \alpha\mathbf{x}_\mu[k'] + (1-\alpha)\mathbf{x}_\mu[k], \quad \alpha \in [0,1]$$
$$\mathbf{x}_\sigma[k+1] = \alpha\mathbf{x}_\sigma[k'] + (1-\alpha)\mathbf{x}_\sigma[k], \quad \alpha \in [0,1]$$

7: Update counter: $k \leftarrow k+1$

8: **end while**

9: **return** $\mathbf{x}_{best}$

- In line 1, the distributions $\mathbf{f}[k=1]$ are instantiated as either uniform distributions or as Gaussian distributions. For Gaussian distributions, the initial mean can be the center of the domain of each parameter. The standard deviation of the Gaussian should be such that 6σ covers the domain of each parameter, to minimize the number of rejected samples.

- In line 3, samples are drawn from the distributions $\mathbf{f}[k]$. A random number generator is used to create samples from either the uniform or Gaussian distributions given the distributions' parameters. The samples are rejected if they lie outside certain bounds, in other words, re-sample if the following does not hold:

$$\mathbf{x}_{min} \leq \mathbf{x}_i[k] \leq \mathbf{x}_{max}, \quad \text{for each } i \tag{B.7}$$

  The samples must also be rejected if the resulting path $\mathbf{x}_A(\theta)$ does not satisfy the constraints in Eq. 2.2.

- In line 4, the objective function $J_{V,\text{discrete}}(\mathbf{x}_i[k])$ from Eq. B.3 is evaluated for every sample of the population.

- In line 5, the samples are sorted by their objective values using a standard sorting routine. Sorting is optional, as described at the end of this section.

- In line 6, the best fraction ρ of samples ($\rho \approx 0.1$–0.2) is used to update the probability distribution vector. The equations for calculating $\mathbf{x}_\mu[k']$ and $\mathbf{x}_\sigma[k']$ are as follows:

$$\mathbf{x}_\mu[k'] = \frac{\sum_{j=1}^{\rho n} \mathbf{x}_{sort,j}[k]}{\rho n} \tag{B.8a}$$

$$\mathbf{x}_\sigma[k'] = \sqrt{\frac{\sum_{j=1}^{\rho n} (\mathbf{x}_{sort,j}[k] - \mathbf{x}_\mu[k'])^2}{\rho n}} \tag{B.8b}$$

  The distribution parameters at $[k+1]$ are updated with a weighting α on the

new values at $[k']$ and $(1-\alpha)$ on the old values at $[k]$, according to the rule:

$$\mathbf{x}_\mu[k+1] = \alpha\mathbf{x}_\mu[k'] + (1-\alpha)\mathbf{x}_\mu[k], \quad \alpha \in [0,1] \tag{B.9a}$$

$$\mathbf{x}_\sigma[k+1] = \alpha\mathbf{x}_\sigma[k'] + (1-\alpha)\mathbf{x}_\sigma[k], \quad \alpha \in [0,1] \tag{B.9b}$$

The value for α is typically between 0.95 and 1.00, with smaller values reducing the rate of convergence, with the benefit of possibly sampling better values. α does not need to be scheduled with the iteration number.

- In line 7, the iterations counter is incremented by one.

  Upon the next iteration $k+1$, the old samples are discarded, new ones are drawn from the new distributions, and the process repeats until $(\mathbf{x}_\sigma < \mathbf{x}_{\sigma,tol})$ which is checked in Line 2. In certain not-well-behaved cases, it may be necessary to terminate if either (i) the distribution standard deviations diverge, or (ii) the time or number of iterations exceeds a limit in Line 2 of the algorithm. This should be incorporated into the optimization loop to ensure termination of the algorithm. On the other hand, if the algorithm terminates before the allotted run time, it is possible to run multiple instances of the algorithm and select the best result from the instances.

- In Line 9, the final path is returned by applying the transformation of the best parameters $\mathbf{x}_{best}$ into the aircraft state.

A variation of the CE method replaces ρ by all samples with an objective value within 10% or some fraction of the best sample. The latter is an alternative to sorting: an indicator function is used instead. Other simple variations involve changing the CE parameters so that the algorithm converges more quickly (for example, a larger $\mathbf{x}_{\sigma,tol}$, smaller ρ, or smaller α), typically resulting in poorer results, or more slowly (changing the parameters in the other direction), resulting in better results at the cost of execution time.

Bibliography

[1] Office of the Secretary of Defense, "FY2009-2034 Unmanned Systems Integrated Roadmap," 2009.

[2] Northrup Grumman, "RQ-4 Block 20 Global Hawk," 2010.

[3] Military Factory, "AAI Corporation RQ-7 Shadow 200 Tactical," 2009.

[4] AAI Corporation, "Aerosonde," 2010.

[5] "FULMAR," 2009.

[6] Parrot S. A., "Parrot AR.Drone," 2010.

[7] Defense Industry Daily, "Rq-4 global hawk uavs prepare for maritime role," 2010.

[8] Flightglobal, "Italian MoD buys AAI Shadow UAVs," 2010.

[9] U. S. Army UAS Center of Excellence, ""Eyes of the Army" U. S. Army Unmanned Aircraft Systems Roadmap 2010-2035," 2010.

[10] M. Iain, A. Mcmanus, M. Clothier, and A. Walker, "AIAC-11 Eleventh Australian International Aerospace Congress Highly Autonomous UAV Mission Planning and Piloting for Civilian Airspace Operations," 2008.

[11] J. Moorthy, R. Higgins, and K. Arthur, "Bringing UAVs to the fight: recent army autonomy research and a vision for the future," in *Proceedings of SPIE*, vol. 6981, p. 69810D, 2008.

[12] S. Rasmussen and T. Shima, *UAV Cooperative Decision and Control: Challenges and Practical Approaches*. Society for Industrial Mathematics, 2009.

[13] J. P. How, C. Fraser, K. C. Kulling, L. F. Bertuccelli, O. Toupet, L. Brunet, A. Bachrach, and N. Roy, "Increasing autonomy of UAVs," *Robotics & Automation Magazine, IEEE*, vol. 16, no. 2, pp. 43–51, June 2009.

[14] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, second ed., 2003.

[15] S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[16] S. LaValle, H. González-Banos, G. Becker, and J. Latombe, "Motion Strategies for Maintaining Visibility of a Moving Target," in *IEEE International Conference on Robotics and Automation*, pp. 731–736, Citeseer, 1997.

[17] J. Kim and Y. Kim, "Moving ground target tracking in dense obstacle areas using UAVs," in *Proceedings of IFAC World Congress*, Citeseer, 2008.

[18] M. Peot, T. Altshuler, A. Breiholz, R. Bueker, K. Fertig, A. Hawkins, and S. Reddy, "Planning sensing actions for UAVs in urban domains," in *Proc. SPIE*, vol. 5986, pp. 143–152, Citeseer, 2005.

[19] F. Morbidi, F. Bullo, and D. Prattichizzo, "Visibility Maintenance via Controlled Invariance for Leader-Follower Dubins-like Vehicles," *Automatica*, Dec. 2009. Submitted.

[20] A. Posch and S. Sukkarieh, "UAV based search for a radio tagged animal using particle filters," in *In Proceedings of 2009 Australasian Conference on Robotics and Automation*, 2009.

[21] L. Parker, "Distributed algorithms for multi-robot observation of multiple moving targets," *Autonomous robots*, vol. 12, no. 3, pp. 231–255, 2002.

[22] G. Hoffmann and C. Tomlin, "Mobile sensor network control using mutual information methods and particle filters," *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 32–47, 2010.

[23] S. Martínez and F. Bullo, "Optimal sensor placement and motion coordination for target tracking," *Automatica*, vol. 42, no. 4, pp. 661–668, 2006.

[24] R. Vidal, O. Shakernia, H. Kim, D. 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, 2002.

[25] Wikipedia, "Bellevue, Washington," 2009.

[26] Robinson Helicopter Company, "City of San Bernardino Gets New R44 Raven II Police Helicopter," 2006.

[27] Y. Bar-Shalom, X. Li, X. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. Wiley-Interscience, 2001.

[28] X. Li and V. Jilkov, "Survey of maneuvering target tracking. Part I: Dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, p. 1333, 2003.

[29] T. Bandyopadhyay, M. Ang Jr., and D. Hsu, "Motion planning for 3-D target tracking among obstacles," in *Proc. Int. Symp. on Robotics Research*, 2007.

[30] G. Xu and Z. Zhang, *Ggeometry in Stereo, Motion, and Object Recognition: A Unified Approach*. Springer, 1996.

[31] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[32] B. Sinopoli, M. Micheli, G. Donato, and T. Koo, "Vision based navigation for an unmanned aerial vehicle," in *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1757–1764, Citeseer, 2001.

[33] E. Frew, T. McGee, Z. Kim, X. Xiao, S. Jackson, M. Morimoto, S. Rathinam, J. Padial, and R. Sengupta, "Vision-Based Road-Following Using a Small Autonomous Aircraft," in *Proceedings of the IEEE Aerospace Conference*, (Big Sky, MT), March 2004.

[34] A. Wu, E. Johnson, and A. Proctor, "Vision-aided inertial navigation for flight control," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, pp. 2005–5998, 2005.

[35] S. Ponda, R. Kolacinski, and E. Frazzoli, "Trajectory Optimization for Target Localization Using Small Unmanned Aerial Vehicles," in *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2009.

[36] H. Helble and S. Cameron, "3-d path planning and target trajectory prediction for the Oxford Aerial Tracking System," in *2007 IEEE International Conference on Robotics and Automation*, pp. 1042–1048, 2007.

[37] B. Bethke, "Persistent vision-based search and track using multiple UAVs," Master's thesis, Massachusetts Institute of Technology, 2007.

[38] C. Hue, J. Le Cadre, and P. Pérez, "Sequential Monte Carlo methods for multiple target tracking and data fusion," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, p. 309, 2002.

[39] Y. Oshman and P. Davidson, "Optimization of observer trajectories for bearings-only targetlocalization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 35, no. 3, pp. 892–902, 1999.

[40] G. Campion, G. Bastin, and B. D'Andréa-Novel, "Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, p. 47, 1996.

[41] L. Dubins, "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[42] B. Stevens and F. Lewis, *Aircraft Control and Simulation*. Wiley-Interscience, 2003.

[43] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[44] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, *et al.*, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727–774, 2008.

[45] A. Briggs and B. Donald, "Automatic sensor configuration for task-directed planning," in *IEEE International Conference on Robotics and Automation*, pp. 1345–1345, 1994.

[46] A. Ganguli, J. Cortés, and F. Bullo, "Visibility-based multi-agent deployment in orthogonal environments," in *American Control Conference (ACC)*, pp. 3426–3431, 2007.

[47] L. Cheng and Y. Tsai, "Visibility optimization using variational approaches," *Communications of Mathematical Sciences*, vol. 3, no. 3, pp. 425–451, 2005.

[48] P. Theodorakopoulos and S. Lacroix, "UAV target tracking using an adversarial iterative prediction," in *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pp. 967–972, Institute of Electrical and Electronics Engineers Inc., The, 2009.

[49] S. Ghosh, *Visibility Algorithms in the Plane*. Cambridge Univ Pr, 2007.

[50] S. Teller and C. Séquin, "Visibility preprocessing for interactive walkthroughs," in *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, p. 70, ACM, 1991.

[51] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*. Springer-Verlag New York Inc, 2008.

[52] A. Shapira, *Fast line-edge intersections on a uniform grid*. San Diego, CA, USA: Academic Press Professional, Inc., 1990.

[53] N. Rowe and D. Lewis, "Vehicle path-planning in three dimensions using optics analogs for optimizing visibility and energy cost," in *JPL, California Inst. of Tech, Proceedings of the NASA Conference on Space Telerobotics,*, vol. 4, 1989.

[54] K. Obermeyer, "Path Planning for a UAV Performing Reconnaissance of Static Ground Targets in Terrain," *AIAA Guidance, Navigation, and Control Conference, Chicago, Illinois, Aug. 10-13, 2009*, p. 5888, 2009.

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

[56] M. Baumann, S. Léonard, E. Croft, and J. Little, "Path planning for improved visibility using a probabilistic road map," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 195–200, 2010.

[57] G. Phillips, *Interpolation and Approximation by Polynomials*. Springer Verlag, 2003.

[58] R. Bellman and S. Dreyfus, "Functional approximations and dynamic programming," *Mathematical Tables and Other Aids to Computation*, vol. 13, no. 68, pp. 247–251, 1959.

[59] S. Dreyfus and A. Law, *The art and theory of dynamic programming*. Academic Pr, 1977.

[60] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[61] G. Franklin, J. Powell, and S. Emami-Naeini, *Feedback Control of Dynamic Systems*. Prentice Hall, 2006.

[62] S. Park, J. Deyst, and J. How, "A new nonlinear guidance logic for trajectory tracking," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2004 (AIAA 2004–4900).

[63] S. Park, J. Deyst, and J. P. How, "Performance and lyapunov stability of a nonlinear path-following guidance method," *Journal of Guidance, Control, and Dynamics*, vol. 30, pp. 1718–1728, November-December 2007.

[64] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. How, "Motion planning in complex environments using closed-loop prediction," in *Proc. of the AIAA Guidance, Navigation, and Control Conference and Exhibit, Honolulu, HI, Aug. 2008*, p. 7166, 2008.

[65] A. Rao, D. Benson, C. Darby, C. Francolin, M. Patterson, I. Sanders, and G. Huntington, "Algorithm 902: GPOPS, a MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method," *ACM Trans. Math. Softw*, 2009.

[66] D. Benson, *A Gauss pseudospectral transcription for optimal control*. PhD thesis, Massachusetts Institute of Technology, 2005.

[67] P. Gill, W. Murray, and M. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Journal on Optimization*, vol. 12, no. 4, pp. 979–1006, 2002.

[68] J. Ren, K. McIsaac, R. Patel, and T. Peters, "A potential field model using generalized sigmoid functions," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 37, no. 2, pp. 477–484, 2007.

[69] C. Kolb, D. Mitchell, and P. Hanrahan, "A realistic camera model for computer graphics," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 317–324, ACM, 1995.

[70] G. S. Aoude, J. P. How, and I. M. Garcia, "Two-stage path planning approach for solving multiple spacecraft reconfiguration maneuvers," *Journal of the Astronautical Sciences*, vol. 56, pp. 515–544, Oct-Dec 2008.

[71] N. Leonard, D. Paley, F. Lekien, R. Sepulchre, D. Fratantoni, and R. Davis, "Collective motion, sensor networks, and ocean sampling," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 48–74, 2007.

[72] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, p. 671, 1983.

[73] D. Goldberg *et al.*, *Genetic algorithms in search, optimization, and machine learning*. Addison-wesley Reading Menlo Park, 1989.

[74] P. De Boer, D. Kroese, S. Mannor, and R. Rubinstein, "A tutorial on the cross-entropy method," *Annals of Operations Research*, vol. 134, no. 1, pp. 19–67, 2005.

[75] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[76] F. Glover and R. Marti, "Tabu search," *Metaheuristic Procedures for Training Neutral Networks*, pp. 53–69, 2006.

[77] T. Mathworks, "Global Optimization Toolbox," 2009.

[78] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-gaussianbayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, 2001.

[79] R. Brown and P. Hwang, *Introduction to Random Signals and Applied Kalman Filtering (3rd ed.)*. John Wiley, 2005.

[80] M. Valenti, B. Bethke, G. Fiore, J. How, and E. Feron, "Indoor Multi-Vehicle Flight Testbed for Fault Detection, Isolation, and Recovery," in *AIAA Guidance, Navigation, and Control Conference (GNC)*, (Keystone, CO), August 2006.

[81] M. Valenti, B. Bethke, D. Dale, A. Frank, J. McGrew, S. Ahrens, J. How, and J. Vian, "The MIT Indoor Multi-Vehicle Flight Testbed," in *IEEE International Conference on Robotics and Automation*, pp. 2758–2759, 10–14 April 2007.

[82] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian, "Real-time indoor autonomous vehicle test environment," *IEEE Control System Magazine*, vol. 28, pp. 51–64, April 2008.

[83] G. Bradski, "The OpenCV Library," *Doctor Dobb's Journal of Software Tools*, vol. 25, no. 11, pp. 120–126, 2000.

[84] P. Kinney *et al.*, "ZigBee Technology: Wireless Control that Simply Works," in *Communications Design Conference*, vol. 2, 2003.

[85] iRobot Corporation, "iRobot Create Programmable Robot," 2010.

[86] D. Gurdan, J. Stumpf, M. Achtelik, K. M. Doth, G. Hirzinger, and D. Rus, "Energy-efficient autonomous four-rotor flying robot controlled at 1 khz," in *IEEE International Conference on Robotics and Automation*, pp. 361–366, 10-14 April 2007.

[87] Panasonic Corporation, "BL-C131A Wireless Pan/Tilt MPEG-4 PetCam Network Camera," 2009.

[88] K. J. Obermeyer and Contributors, "The VisiLibity library," 2008. R-1.

[89] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo, "Model-based search for combinatorial optimization: A critical survey," *Annals of Operations Research*, vol. 131, no. 1, pp. 373–395, 2004.