OBSERVER TRAJECTORY GENERATION FOR TARGET-MOTION ESTIMATION USING MONOCULAR VISION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Eric W. Frew

August 2003

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Stephen M. Rock (Principal Adviser)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Robert H. Cannon Jr.

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Claire J. Tomlin

Approved for the University Committee on Graduate Studies:

Abstract

The creation of autonomous robotic vehicles capable of performing complex tasks with little to no human intervention has been a goal of researchers and engineers for many years. These machines promise to accomplish tasks that humans cannot or should not perform themselves. Current technology has advanced to the point where such robotic systems are now being built and operated. Example systems include remotely operated vehicles (ROVs) used mainly for underwater science missions, unmanned aerial vehicles (UAVs), such as the U.S. Navy Predator, used mainly for surveillance, and planetary explorers such as the Sojourner Mars Rover. The development of autonomous capabilities for these robotic vehicles presents many current challenges.

One important objective of many autonomous missions is target-motion estimation. This task requires an autonomous observer vehicle to determine the global position and velocity of a target object in the world, given only measurements of its own global state (position and velocity) and components of the relative position (either bearing or range) between it and the target object. Example applications that utilize target-motion estimation include aerial surveillance, where the sensing is the final goal, or search and rescue, where the target-motion estimation enables more complex interactions with the world. One specific configuration of sensors uses a single camera in conjunction with observer-vehicle navigation sensors such as GPS. The resultant sensor fusion problem which yields an estimate of the global target position and velocity is known as *target-motion estimation using monocular vision.*

This dissertation addresses one major complication with target-motion estimation using monocular vision: the dependence of the estimation performance on the specific observer-vehicle path. Because the monocular vision system provides only a bearing measurement to the target, the camera must move to multiple locations in order to triangulate the measurements and obtain a solution of the global target state. The accuracy of the resulting target-motion estimate is a function of the specific camera path, and therefore some trajectories lead to better performance than others. Typical current systems rely on ad-hoc observer-vehicle paths or motion resulting from the satisfaction of other objectives to enable estimation. The goal of the work presented here is to enhance target-motion estimation using monocular vision by generating near-optimal observer trajectories.

The central thesis of this work is that trajectory design to improve target-motion estimation using monocular vision can be achieved and integrated into an autonomous robotic observer with fast, near-optimal algorithms. To that end, this dissertation presents the details of a novel observer-trajectory generator that focuses on three important issues. First, the limited field of view of the monocular vision system must be addressed. Second, a new optimization objective is desired for use on operational vehicles. Third, the uncertain nature of the target-motion estimate leads to a difficult conundrum -- an optimal trajectory can be determined if the target motion is known; however the whole point of the trajectory generation is to enable the state estimation of the target's unknown motion.

Important features of the new observer-trajectory generator include the identification of a quality metric used to evaluate candidate trajectories, a novel observer-trajectory-generation algorithm for known initial target motion capable of quickly finding near-optimal vehicle paths, and integration of the trajectory-design algorithm as the core of a new observer-trajectory generator for the general case. The success of the trajectory generator and characteristics of the new trajectory design algorithm are verified by experimental results obtained using the Aerospace Robotics Laboratory's micro autonomous rover platform.

To Audrey

You and Me, Baby

Acknowledgements

Although the name on the front of this dissertation is mine, the efforts of countless colleagues, friends, and family have contributed to its final completion. I would not be who I am, and as a result this work would not be as it is, without the friendship, love, and support of them all. For this I am forever grateful.

In particular, I would like to first thank Professor Steve Rock for leading the Aerospace Robotics Laboratory and being my adviser here at Stanford. Professor Rock has taught me that while technical detail is critical to performing quality research, the ability to explain the who, what, where, when, and why in simple terms, to every audience, is the true sign of mastery of a subject and the ultimate goal of the Ph.D process. Special thanks to Professor Robert Cannon for creating the Aerospace Robotics Laboratory. The ARL is a unique environment in which openness, sharing, collaboration, daring, and learning are all encouraged. Also, thanks to Professor Claire Tomlin for being the final member of my dissertation reading committee.

I would like to extend a second round of special thanks to the support staff here at the Department of Aeronautics and Astronautics and the Aerospace Robotics Laboratory. Thanks to Jane Lintott, Sherann Ellsworth, and Dana Parga for helping with all of the many details that go into conducting research, and allowing the students to concentrate on our precious work. Thanks to Godwin Zhang, Gad Shelef, and Aldo Rossi for their help designing and creating the many robots and machines that have I have had such fun playing with. Finally, thanks to Sally Gressens for making sure I knew all the steps needed along the way to completing this dissertation and for helping me sort out the mess every time I missed one.

I have already mentioned the wonderful environment created by Professor Cannon -- the Aerospace Robotics Laboratory. The ARL is a tribute to Professor Cannon, who created it, as well as the many students who have passed through its doors throughout the years. I had the wonderful opportunity of working with many of them, including such esteemed alumni as H.D. Stevens, David Miles, Steve Ims, Bijan Sayyar-Rodsari, Stef Sonck, Gordon Hunt, Andrew Robertson, Bob Kindel, Heidi Schubert, Tobe Corrazini, Stephen Fleischer, Kortney Leabourne, and Eric Prigge. Thanks to all for making me feel welcome in this lab and for always helping me out, even at the expense of time on their own work. In addition, I would like to thank the helicopter group -- Steve Morris, Bruce Woodley, Hank Jones, and Ed Lemaster -- for letting me see how fun the ARL and robotics research could be. For an engineer like me, there are few thrills as great as seeing something like the Hummingbird helicopter, which we created ourselves, performing out there in the world. Finally, thanks to the current members of the ARL; Jeff Ota, Chris Clark, Jorge Moraleda, Jason Rife, Masa Matsuoka, Tim Bretl, Kristof Richmond, Jack Langelaan, Teresa Miller, and Jinwhan Kim. The spirit of the lab continues in their hands.

While my time at the ARL has seemed long, I have had the good fortune of sharing it with two close friends. Hank Jones, Andreas Huster, and I began our journeys toward the Ph.D. together. Since I was technically the last to join the lab, it is fitting that I am the last to complete his dissertation. Along the way, Hank and Andreas have been a continual source of inspiration, encouragement, and support. I value the many afternoons we spent discussing one another's work, shaping the course of our research programs, and creating a deep, lasting, unforgettable friendship.

One of my most important experiences during my career at Stanford has been my experience coaching the Junipero Serra High School Varsity Men's Rowing Team. I would like to thank the men of my team as well as all of the athletes of the Serra - Notre Dame - Mercy Crew. Those three years of early-morning practices and weekend-long regattas provided the perfect balance to the up and down emotions of the Ph.D. process. Thanks also to the coaching staff of the team -- Robin Heggum, Monica Hilcu, Joe O'Connor, Meave Fallon, and Angela Hall.

My time at Stanford would not have been the same if not for the many friends who have supported me along the way. I need to add another round of thanks to the many great friends I have made while here and the many other friends who have embraced me while on this path. Thanks to my roommates over the years: Sam Yen (and Eve), Martin Snyder, Chris Michaelis, Matt McMullen, Mike Steiger, and Ellie Kim. Thanks also to many other great friends, including: Laurie Carville, London Lawson, Christine Harada, Tom Pace, Nancy Adleman, Jonathan Chow, Allison Nordt, Glen Sapilewski, and Allison Coy.

My parents, William and Patricia Frew, deserve any extra special thanks, not for continually asking “Are you done yet?”, but rather, for instilling in me a life-long commitment to education and learning. Both are teachers, so it is easy to see why I have chosen to continue down the path of academia and pursue a career as a university professor. I have always looked up to my parents and cannot overstate the importance of their love throughout my life. I also want to thank my siblings -- Mark, Tracy, and Sean -- for being such great companions growing up. I look forward to seeing how our lives grow together.

Finally, thanks to my lovely wife Audrey. I have dedicated this dissertation to her because she has made my life complete. I am forever grateful at the way she has made my goals and dreams her goals and dreams too. This dissertation is a tribute to the love and strength she has given me. You and me (and the baby makes three)!

Contents

List of Figures

CHAPTER 1 Introduction

The problem addressed in this dissertation is the generation, for autonomous robotic observer vehicles, of trajectories that improve their ability to estimate a target's position and velocity using monocular vision. Estimation performance is a strong function of the observer vehicle's path, and can therefore be optimized by the generation of trajectories that trade final-target-estimate uncertainty for total trajectory duration. Trajectories are desired that minimize the estimate uncertainty in a given time or that minimize the time needed to achieve a specified error bound. In order to generate the vehicle trajectories, several key issues must be addressed. These include (i) the identification of a quality metric to capture the utility of a given trajectory, (ii) the development of a computationally-fast trajectory-design algorithm that includes computer vision field-of-view limitations and dynamic vehicle constraints, and (iii) integration of the design method with an operational system.

1.1 Motivation

The creation of autonomous robotic vehicles capable of performing complex tasks with little to no human intervention has been an elusive goal of researchers and engineers for many years. These machines promise to accomplish tasks that humans cannot or should not perform themselves. They will enable the study of life in the deep oceans, survey of remote or dangerous areas, and exploration of other planets. Current technology has advanced to the point where such robotic systems are now being built and operated. Example systems (Figure 1.1) include remotely operated vehicles (ROVs) used mainly for underwater

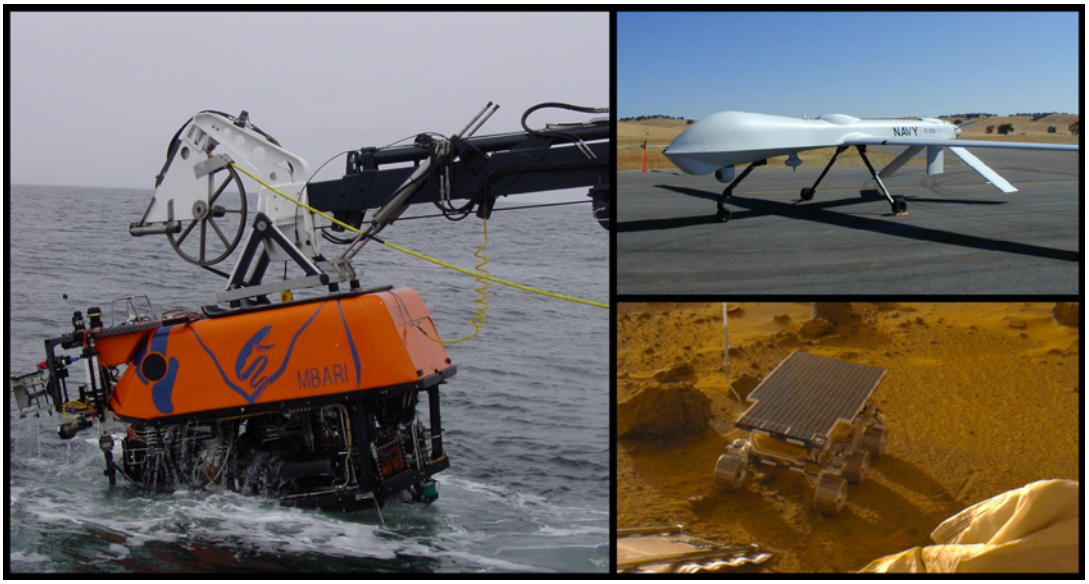

Figure 1.1 Robotic vehicles: The Ventana ROV, the Predator UAV, and the Sojourner Rover.

science missions [68], unmanned aerial vehicles (UAVs), such as the U.S. Navy Predator, used mainly for surveillance [70], and planetary explorers such as the Sojourner Mars Rover [69]. The development of autonomous capabilities for these robotic vehicles presents many current challenges.

One core capability for many autonomous missions is determining the global position and velocity -- the state -- of target objects in the world. This object-state information allows autonomous vehicles to operate safely in an environment populated by terrain, obstacles, buildings, other vehicles, people, etc. For some missions, such as aerial surveillance, the sensing of these targets is the final goal while for others, such as search and rescue, it enables more complex interactions.

In general, an autonomous vehicle is capable of measuring components of its own position and velocity as well as components of the position and velocity of other objects in the world. Depending on the content and quality of these sensor measurements, different sensor-fusion strategies have evolved to combine the measurements so as to estimate all of the vehicle and object states. Specifically, the desired states are the global position and velocity of the autonomous observer vehicle and the global position and velocity of target objects, which may be any object of interest, including people, buildings, or other vehicles:

$$\boldsymbol{x}_{desired} = \begin{bmatrix} \boldsymbol{x}_{target} \\ \dot{\boldsymbol{x}}_{target} \\ \boldsymbol{x}_{observer} \\ \dot{\boldsymbol{x}}_{observer} \end{bmatrix}. \tag{1.1}$$

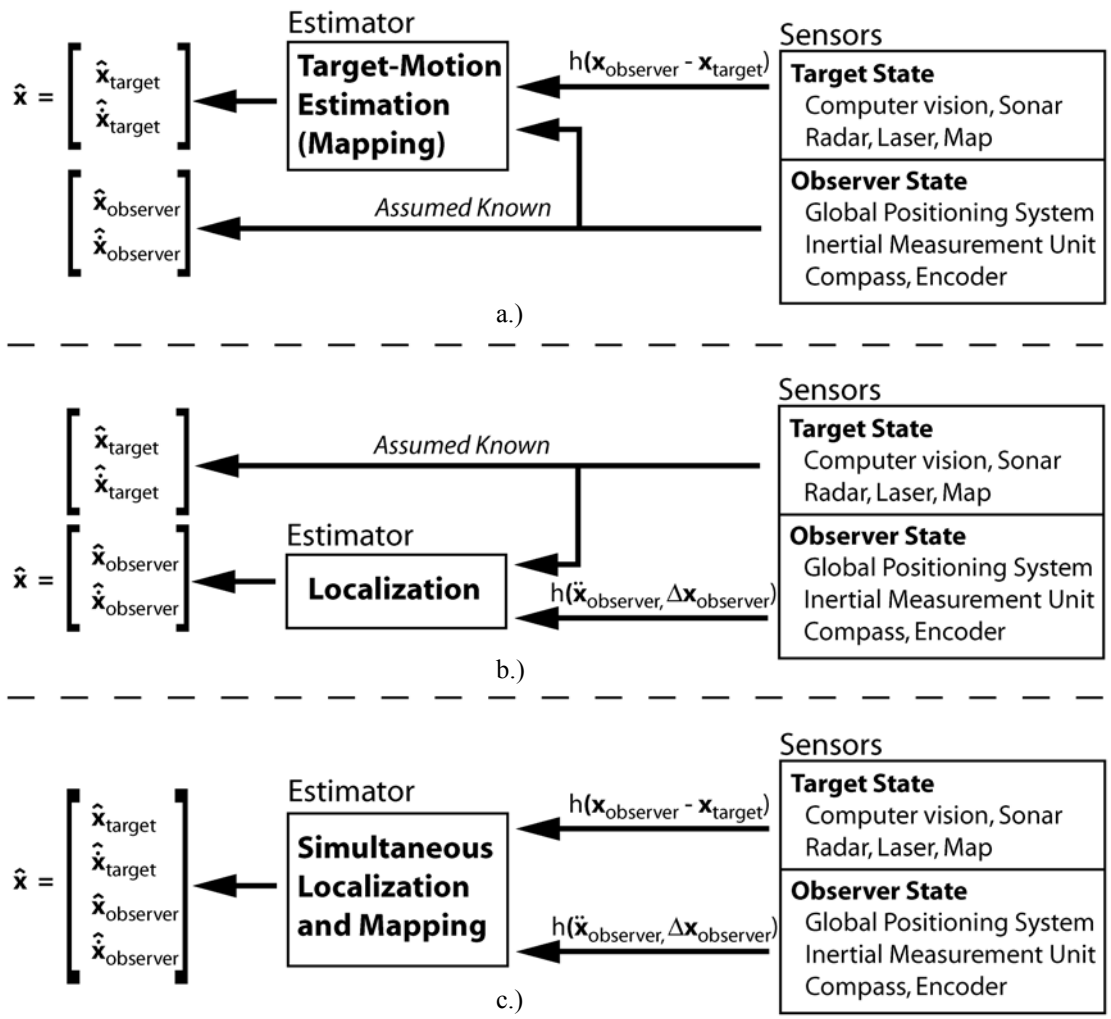

Figure 1.2 Block diagrams of several different sensor-fusion problems. a.) Target-Motion Estimation. b.) Localization. c.) Simultaneous Localization and Mapping.

Ongoing research efforts into the estimation of observer-vehicle and target-object states can be broken down into three main categories (Figure 1.2): target-motion estimation, observer localization, and simultaneous localization and mapping. Given measurements of the position of an object relative to the observer vehicle (e.g. bearing measurements from computer vision), and knowing the observer vehicle position and velocity (e.g. using GPS), the problem of calculating the position and velocity of the object is called *target-motion estimation* or *target-motion analysis* (Figure 1.2a). *Mapping* refers to the same problem when the object is stationary. Conversely, when a map of landmark locations is known in advance (e.g. from satellite imagery) and the observer vehicle is capable of measuring a component of its own position relative to the landmarks (e.g. range from radar), the problem of estimating the position and velocity of the observer vehicle is known as *localization* or *navigation* (Figure 1.2b). Finally, given a partial measurement of the autonomous

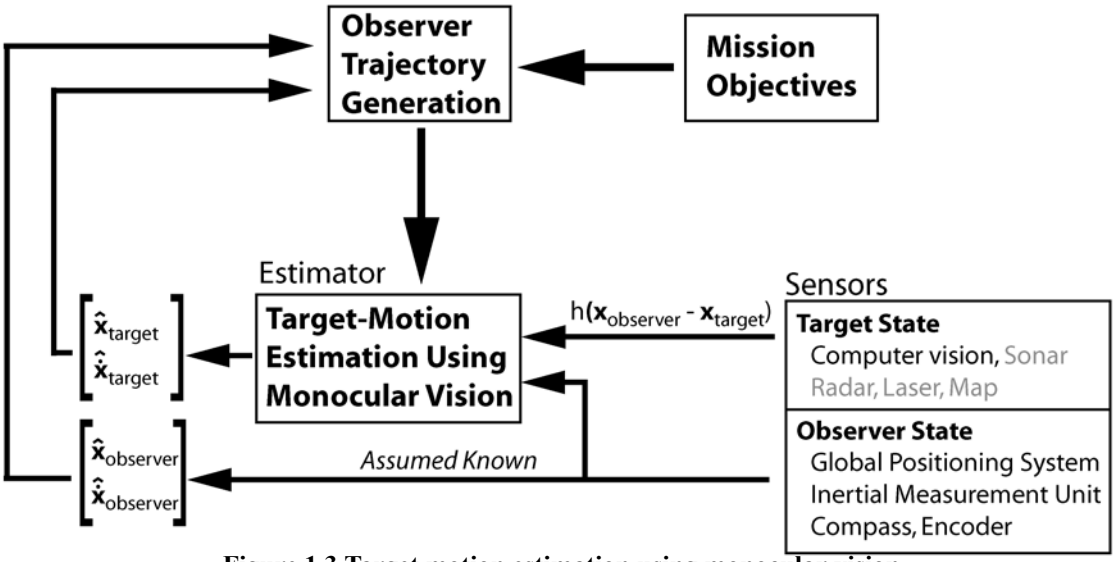

Figure 1.3 Target motion estimation using monocular vision

observer-vehicle state (e.g. integrated acceleration from an inertial measurement unit) and measurements of the position of target objects (or landmarks) relative to the observer vehicle (e.g. range from a laser range finder), the problem of calculating a map of the targets and the position and velocity of the observer vehicle within that map is referred to as *simultaneous localization and mapping* (Figure 1.2c). More details of each sensor-fusion problem are presented in Section 1.3.1.

This dissertation addresses trajectory design for the improvement of *target-motion estimation using monocular vision*, a specific example of target-motion estimation in which the vehicle position and velocity are assumed known, and a single camera is used as the only relative-target-position sensor (Figure 1.3). The development of GPS and other similar navigation technology has spawned the production of many robotic vehicles capable of directly sensing their own global position and velocity. The addition of a single camera then allows these vehicles to calculate the state of other target objects in the world.

This research is motivated by the importance of tailoring observer-trajectory design to the collection and processing of sensory information for target-motion estimation by autonomous observer vehicles. By investigating the link between sensor-fusion strategies and high-level planning and control, mission-level success for autonomous vehicles can be enhanced. The ability to extract and use information from autonomous robots enables the development of systems that yield improved performance and require less accurate sensors, less time, and less fuel, leading to cheaper systems with many future applications.

Figure 1.4 Image from the Monterey Bay Aquarium Research Institute's Ventana ROV. The human pilot or the autonomous controller uses the onboard image sequence to control the vehicle.

Benefits of Using Monocular Vision

Computer vision is a popular relative-position sensor for natural environments [22, 30, 40, 41, 43, 68, 69, 71]. A single digital image contains a large amount of information which can be used to perform many of the important tasks necessary for autonomous operation, including object identification and classification, recognition, tracking, and modeling [58, 59]. Additionally, vision provides information and situational awareness to human operators in an intuitive manner [71]. Human operators often control robotic vehicles remotely using onboard video. For example, Figure 1.4 shows an image from the Monterey Bay Aquarium Research Institute's (MBARI) Ventana ROV used by the human pilot to service the underwater seismometer shown. Furthermore, cameras provide large amounts of scientific data and are often included on robotic vehicles as payload. Utilizing existing cameras for relative-position sensing adds no additional physical burden to these vehicles.

As a relative-position sensor, computer vision provides a basic bearing measurement from the camera to the sensed object. An object's full state (its position and velocity) can be determined only by triangulating multiple measurements taken from different locations. One approach to generating these displacements among the measurements uses multiple cameras fixed on the same vehicle, a system referred to as stereo vision [51, 68, 69, 71]. Given the known positions of the cameras relative to one another and the assurance that they are all looking at the same object, these systems are able to triangulate the object's position using measurements taken at a single instant. In contrast, an alternative approach uses a single camera that moves

in time. By tracking the object from frame to frame, a time series of displaced measurements is generated from which the object's state can be calculated.

While stereo-vision solutions are good for some applications (for example in an industrial setting where the workspace is mostly structured), there are situations where issues with stereo arise, and single, moving-camera solutions offer potential advantages. First, automatic initialization of the correspondence problem, which is the problem of determining the location of the same object in every image, can be problematic with stereo for some scenes. Tracking an object or object feature across a time sequence of images is a key component of both monocular and stereo vision systems. Small displacements and distortions in the object between frames ease the calculation of correspondence once the object has been identified. For stereo systems, correspondence must first be established across the different cameras. Large baselines between the cameras can lead to large displacements of the object within the image as well as perspective changes that distort the object's appearance. In turn, these large distortions and displacements can limit the ability of the system to identify an object in each camera and establish the initial correspondence.

.Second, and more important, the fixed baseline between cameras in a stereo system defines an optimal target depth, while a moving-camera system can achieve any relative baseline between measurements and thus effectively sense objects at any depth. For a given amount of image processing noise, the baseline

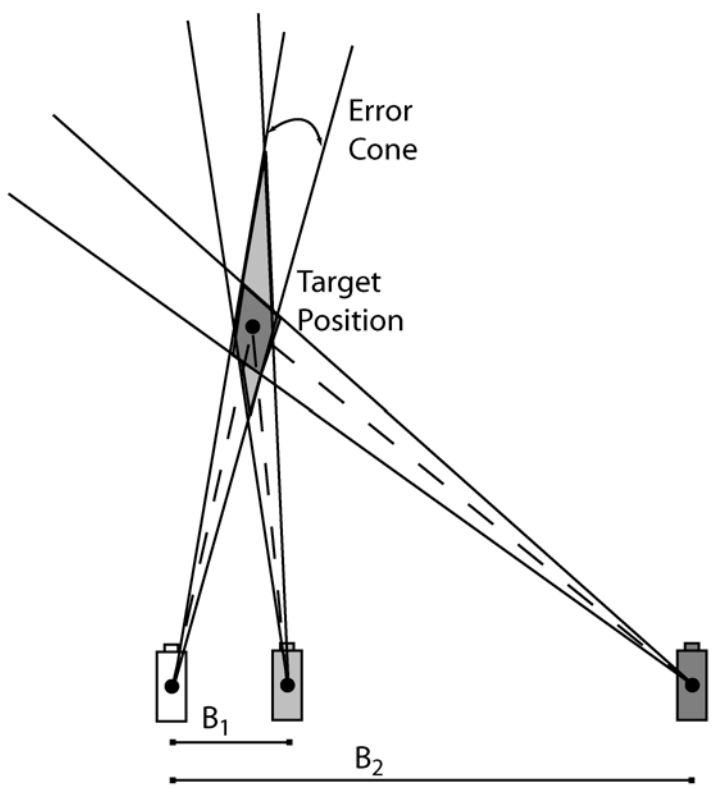

Figure 1.5 The error region around the object is smaller for the camera pair with the larger baseline.

between camera locations defines the overall level of triangulation error using vision Figure 1.5 shows a simple example. The region in which the object can be inferred to exist is noticeably smaller for the camera pair with the larger separation. Conversely, as the camera baseline shrinks, the maximum depth at which an object can be measured also decreases. Two example applications where this baseline sensitivity become significant are micro air vehicles (MAVs) and micro rovers (Figure 1.6) [42, 73, 74]. The most likely sensor configuration for an MAV is a set of motion sensors for navigation and a small, fixed camera for vehicle and operator situational awareness. Because the goal is to develop as small a vehicle as possible, size restrictions limit the possible baseline of a stereo system. Therefore, in order to triangulate objects at reasonable distances, solutions must use single cameras on a moving base.

The New Issues for Trajectory Generation

Due to the perspective projection of the computer vision sensor, which includes scaling the target position by the inverse target range and performing a rotated-frame transformation, target-motion estimation using monocular vision is a nonlinear problem. For linear problems, given a linear estimator and a set of sensors with specific noise characteristics, the initial conditions of the estimate error uniquely determine the final estimation performance (in terms of a statistical measure of the final result). In contrast, the final performance of estimators for nonlinear problems can also be functions of other variables such as the actual state estimate, the control input, or other input parameters. Specifically, the performance of target-motion estimation using monocular vision depends on the relative position between the vision sensor and the target object, and therefore certain observer trajectories (which directly determine the sensor trajectories) are better than others for estimation. Compare a camera that moves toward an object at a slight angle away from the line of sight to one that moves perpendicular to it. Both paths yield enough information to calculate the target's position; however, the second one provides a better set of measurements that will yield less overall error in the object-state estimate. This dependence of the estimation performance on camera trajectory can be exploited to generate paths that enhance the estimation process; *and this dependence is the focus of this research*.

Figure 1.6 A micro rover and a micro air vehicle (MAV)

Specifically, this dissertation presents a novel observer-trajectory generator that improves the performance of target-motion estimation using monocular vision. For many autonomous vehicles, accuracy requirements and time constraints define most missions, and camera paths are desired that successfully trade one for the other. For example, consider one typical mission scenario in which the autonomous vehicle has a full tank of fuel and its objective is to map an important target to as high an accuracy level as possible. A trajectory is desired that achieves a minimum level of target-position uncertainty for a given amount of time. In contrast, some systems require only a certain level of accuracy in order to manipulate or intervene with objects in the world. Since resources like time are limited, the robot wants to achieve this workable level as quickly as possible. Thus a second example scenario asks for paths that achieve a specified level of accuracy in minimum time. The first scenario has been addressed by previous methods focused on related passive sonar applications [10-20], while the second scenario has never been addressed before. The trajectory generator described in this dissertation solves both types of problem.

Trajectory optimization for *bearings-only tracking*, a form of target-motion estimation, has been studied extensively in the context of passive sonar applications [10-20] and represents a starting point for the problem of interest here (details are presented in Section 1.3.2). These previous methods are capable of calculating the optimal trajectory that minimizes the estimate uncertainty for a given observer-target geometry. In order to apply this work to autonomous observer vehicles using monocular vision and the trajectory-design problem addressed here, several additional issues must be considered. First, the vision-sensor limitations and autonomous-observer dynamics must be incorporated. Second, the computational cost, in terms of planning time, must be minimized in order to use the design method in real-time. Finally, the uncertainty of the object estimate must be taken into account in the trajectory design process.

Unlike the passive sonar systems described in the previous work [10-20], monocular vision has a restricted field of view that must be considered in the trajectory-generation process. In combination with vehicle dynamic constraints that also must be considered, the limited view of the vision sensor can lead to vehicle paths that bring the target object out of view. The previous work in the sonar community assumes that the target can always be sensed. However, this assumption is not valid with a monocular vision system.

Autonomous vehicles operate in continuously changing environments. When a new object is encountered, an autonomous observer vehicle must be able to respond in a timely manner: if it cannot, a surveillance target may be lost before it can be tracked, or an obstacle may not be avoided in time. Therefore any vehicle planner or trajectory generator must calculate its results quickly. Calculation of the true-optimal observer path for target-motion estimation using monocular vision requires, in principle, searching through an infinitely large candidate-trajectory space. In order to generate usable observer trajectories, it is necessary to develop a real-time strategy that achieves near-optimal results quickly. As a solution, the new observer-

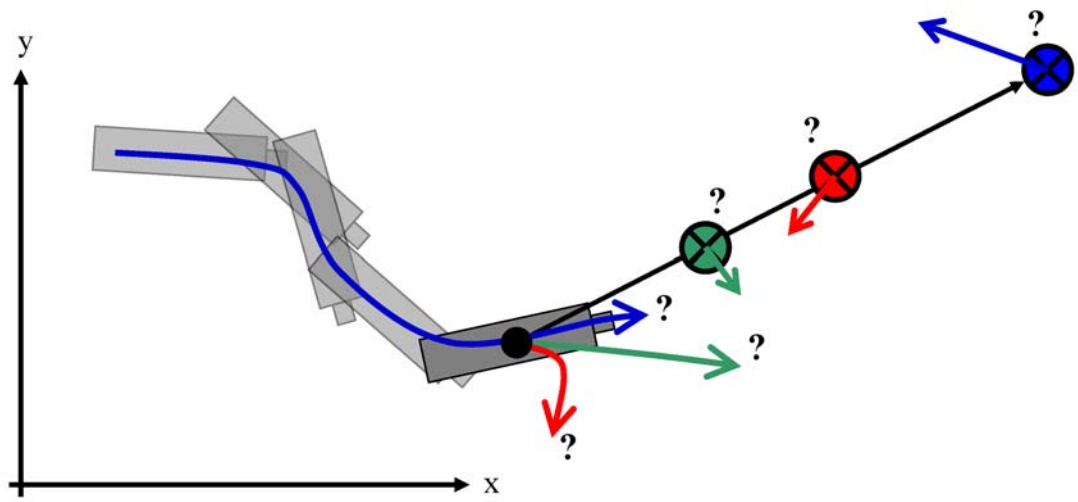

Figure 1.7 The optimal observer-vehicle trajectory depends on the object position and velocity. However, the whole point of designing the trajectory is to estimate that same object state.

trajectory generator presented in this dissertation uses an iterative, breadth-first search method to create a set of candidate trajectories that are then evaluated by a trajectory quality metric. By selecting appropriate values for several parameters, the performance of the method can be balanced by fast planning times.

The third main issue addressed by the new trajectory generator for observer vehicles is the uncertain nature of the object state estimate. Optimal observer-vehicle trajectories can be designed that depend on knowing the position and velocity of the object [10-20]. However, the object state is not known in advance, and the whole point of designing the vehicle trajectory is to determine that unknown position and velocity (Figure 1.7). Therefore the new trajectory generator monitors the state of the object estimate and continually redesigns the vehicle trajectory as new information becomes available.

The central thesis of this work is that computationally-fast observer-vehicle-trajectory generation to improve the performance of target-motion estimation using monocular vision can be achieved and integrated into an operational autonomous vehicle system. Towards this goal, this dissertation presents the details of a novel trajectory generator by focusing on several key steps. These include (i) the identification of a quality metric that explicitly includes the estimate error uncertainty and computer vision limitations to capture the utility of a given trajectory relative to estimation performance, (ii) the development of a computationally-fast trajectory-design algorithm based on this metric that calculates near-optimal solutions in real-time, and (iii) the integration of the trajectory-design algorithm into a new trajectory generator that responds to changes in the target estimate as the estimate uncertainty evolves.

1.2 Important Semantics

This section contains the definitions of major terms that will be used throughout this entire dissertation.

An observer is the total system of a vehicle carrying a camera, which is following a trajectory designed to make its observations of a target most useful in estimating both the position and velocity of the target versus time. In practice the trajectory of the camera must be commanded to improve the estimation. However, the camera is mounted on the vehicle, which has navigation sensors and actuators that are not collocated with the camera. This dissertation assumes a fixed, known transformation between the camera, all navigation sensors, and all actuators. Therefore, knowing and controlling the motion of the camera is equivalent to knowing and controlling the motion of the vehicle. In order to simplify discussion, the term *observer* is used to refer to the total vehicle and camera system and the necessary transformation of frames is implicitly assumed.

The target is the object of interest in the world, whose state -- global position and velocity -- is being estimated by the observer.

Target-motion estimation refers to the general process of determining the position and velocity, in either relative or global coordinates, of an object of interest in the world. Every such system fuses measurements from a relative position sensor of some kind (a sensor capable of measuring one or more components of the position of a target object with respect to that sensor) with other information about that sensing device.

> **Target-motion estimation using monocular vision** refers to the specific class of target-motion estimation using the set of sensors and system configuration shown in Figure 2.1. A single camera measures the relative bearing between the target object and the autonomous vehicle, and the camera motion is measured by additional navigation sensors such as GPS.

Trajectory generation is the computation of a path that the autonomous observer vehicle will be instructed to follow in space and time.

> **The core trajectory design algorithm** calculates the near-optimal observer path for a given observer-target scenario. It is built on presumed complete knowledge of the initial target state. This is the subject of Chapter 4.

> **The full trajectory generator** extends the core design algorithm to the case where the initial target state is in fact unknown. This is the subject of Chapter 5.

A trajectory quality metric is a scalar value that represents the relative potential of a given trajectory to lead to good estimation performance. This is the subject of Chapter 3.

Monocular (computer) vision refers to the use of a single camera to generate the time history of the location of a target object in a digital image. The process of determining the target motion from the image measurements is target-motion estimation.

1.3 Related Work

1.3.1 Estimation

Figure 1.2 shows diagrams of the three main categories of sensor-fusion strategies for estimating the global state of observer vehicles and target objects: target-motion estimation, observer localization, and simultaneous localization and mapping. The distinction between each category and examples are given in this section.

Target-Motion Estimation and Mapping

If the observer-vehicle position and velocity can be determined directly from sensor measurements, for instance using GPS, the problem of calculating the position and velocity of objects by combining the observer information with components of the relative object state (the position and velocity of the object with respect to the vehicle) is referred to as *target-motion estimation* or *target-motion analysis* (Figure 1.2a). When the objects being sensed are all stationary, the estimation problem is called *mapping*. Applications of target-motion estimation and mapping are numerous, and include submarine detection using towed sonar arrays, 3-D scene reconstruction using stereo vision, collision detection and avoidance, and satellite orbit estimation [8, 61, 62, 63].

The example of target-motion estimation most relevant to the monocular-vision-based system used in this work is bearings-only tracking, which refers to the fusion of bearing measurements from passive sonar with sensor-motion data in order to triangulate the range and range rates to target vehicles. Estimation of a stationary vehicle is referred to as *bearings-only localization,* and estimation of a moving target is called *bearings-only tracking (BOT)* or *bearings-only target-motion analysis*. Initial research in the bearings-only tracking community focused on estimator design and estimator performance [1-9]. Because target range is not observable without sensor movement, nonlinear filter techniques such as the Extended Kalman Filter do not perform well when applied with the standard cartesian-coordinate formulation [6]. Several different formulations and coordinate systems have been proposed, and the modified polar coordinate system has

become one of the most useful [7]. Trajectory design for bearings-only tracking has also been studied, and the related work is summarized in Section 1.3.2.

A second related example is *visual point tracking*, which refers to the global-state estimation of objects represented by single points in a visual image. Guanghui et al. [21] presented an estimator for calculating the motion of a point target using a single camera with known motion. Although they did not derive the result, they presented the constraints on the camera motion that are needed in order to generate a unique tracking solution. No attempts at trajectory design were made.

A third class of related algorithms combine observer motion data, in the form of translational and rotational velocities, with monocular image sequences to estimate the depth to sparse features or across the entire image [29, 66]. Because these algorithms are typically used when vehicle position measurements are not available, the resulting depth maps cannot be transformed into global space (the end goal of target-motion estimation using monocular vision), and the complete target state cannot be determined.

Localization

When object positions are known in advance, and a robotic vehicle is capable of measuring its position with respect to those objects, the process of that vehicle thereby determining its own position and velocity is known as *localization* or *navigation* (Figure 1.2b). Mobile robotics has served as the main driver for the development of localization methods. Important applications include the exploration of other planets and navigation through buildings and streets [50-54]. In most cases the known object positions come in the form of landmark or terrain maps, and a major challenge consists of registering sensor readings with the maps in order to determine vehicle location. Localization performance can be improved by choosing the best landmarks to sense and by designing trajectories that increase the number of landmarks that can be seen [50]. Terrain following is a similar form of localization in which the vehicle determines its position relative to a terrain map as it moves along a predefined path [82, 83].

The Global Positioning System provides a powerful example of the solution of a localization problem [57]. Standard GPS technology uses signals from orbiting satellites to determine positions and velocities of the receiving vehicles. The satellites broadcast their positions in space and the time the signal is sent. By measuring the time the signal was received, the range to the transmitting satellite is calculated. Knowing the position and range to several satellites, the vehicle location is determined by triangulation.

Simultaneous Localization and Mapping

As its name implies, *simultaneous localization and mapping* (SLAM) defines the set of problems in which the observer state and static-target positions cannot be calculated separately and are therefore determined

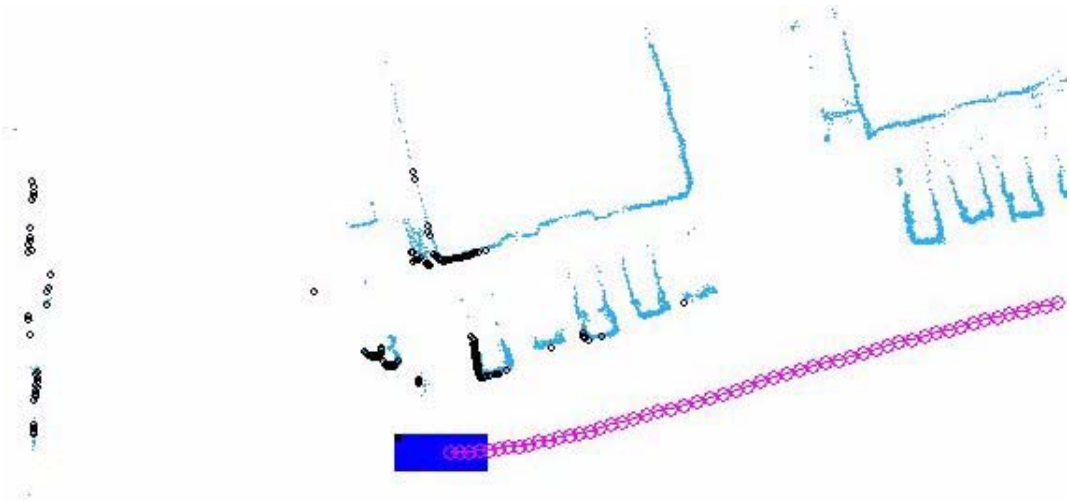

Figure 1.8 Results from a typical SLAM algorithm. The small circles correspond to object measurements while the larger circles represent the estimated positions of the observer[28].

together by a single estimator (Figure 1.2c). In other words, the autonomous observer starts at an unknown location within an unknown environment. As the vehicle moves it creates a map of the environment and determines its position and velocity relative to that map. The problem is also referred to as *concurrent mapping and localization*.

The SLAM problem has many important applications which include the exploration of remote areas and distant planets, travelling through buildings and urban environments, and navigating across open terrain [22-28]. In all cases SLAM solutions combine sensors such as inertial units or wheel encoders that measure partial vehicle state with other sensors that measure components of the position of target objects relative to the observer. The general SLAM algorithms are well understood and consist of tracking multiple objects successfully over multiple time steps. Current research efforts are focused on reducing computational complexity, selecting landmarks and vehicle motion for optimal performance, and including real-time object motion estimation as well [22-28].

While SLAM systems often have range sensors, several applications use monocular vision or other similar bearing-only sensors alone to generate a solution. Deans [22] developed a bearings-only SLAM system that combines monocular-vision measurements to natural landmarks with wheel-encoder measurements from a planetary rover. Structure-from-motion solutions have been developed that use multiple images of a static scene taken from a single moving camera in order to calculate the motion of the camera and the shape of the scene to a scale factor [44, 58, 59, 60]. These algorithms require tracking multiple features in every image.

1.3.2 Trajectory Generation for Vision-based Estimation and Tracking

This section describes work related to the design of observer trajectories to improve the performance of target-motion estimation using monocular vision. Bearings-only tracking using passive sonar is closely related to target-motion estimation using monocular vision, and trajectory design has been studied extensively in that context. Motion planning under visibility constraints focuses on creating observer paths that ensure or maximize coverage using sensors, such as computer vision, with limited visibility.

Bearings-Only Tracking

Trajectory design for bearings-only tracking has its roots in the investigation of estimator performance and the impact of observer motion. It is well known that even given a well-designed estimator, a solution cannot be obtained without appropriate observer motion [6]. For a stationary object, the bearing sensor must have some component of motion perpendicular to the line of sight to the target, otherwise the bearing measurement will not change and the target position cannot be calculated. For a constant-velocity target an observer maneuver (change in heading) is required.

The concept of designing the observer trajectory in order to optimize estimator performance was first introduced by Hammel et al.[12]. They addressed the problem of trajectory design for a stationary target by maximizing the determinant of the Fisher Information Matrix (FIM) which represents an upper bound of the estimation performance (described in detail in Section 3.2.1). They showed that although an observer maneuver (change in heading) is not needed to solve the localization problem, paths comprised of one or more maneuver improve the estimator performance. Hammel et al. used numerical algorithms to maximize the expected estimator performance for a given length of time and they derived optimal deviated pursuit curves where the observer follows a constant aspect angle (i.e. the difference between the observer heading and the target bearing). Lower bounds for the determinant of the FIM were also used to derive analytical expressions for observer paths.

The trajectory design for bearings-only localization of Hammel et al. has been extended by several researchers through the inclusion of system constraints and the application of different optimization methods. The original work by Hammel et al. did not consider dynamic constraints imposed on the observer motion, e.g. non-holonomic constraints of a wheeled vehicle or maximum velocity and acceleration constraints, so subsequent work has enabled the inclusion of additional constraints. Oshman and Davidson incorporated threat functions defined by a target defense system [16]. Other work has included constraints imposed by bearing-sensor limitations and estimation-performance specifications. In the process of including these system constraints, several different optimization methods have been applied to the bearings-only localization problem. Oshman and Davidson compared direct gradient-based methods, orthogonal-function parameterizations, and differential-inclusion solutions. Logothetis et al. [10] and Frew

et al. [65, 67] both used exhaustive search techniques. Huster et al. [66] applied a function parameterization that superimposed a set of motion primitives that optimized the estimator performance while bringing a robotic arm into contact with a stationary object.

For a moving target, observer trajectory design becomes more complicated. Unlike for the stationary-target case, where a maneuver is not required to estimate the target state, for the case of a moving target the observer must make a maneuver in order to determine the target motion. Trajectory design for estimating a moving target was studied by LeCadre [19] and Passerieux and VanCappel [15]. LeCadre linearized the target-motion analysis problem by transforming the measurement equation into a linear pseudomeasurement. He also used a discrete-time formulation in order to reduce the analysis to multilinear algebra. Passerieux and VanCappel used a combination of analytical and numerical techniques to solve for optimal trajectories under a set of basic assumptions such as perfectly omnidirectional sensing and no constraint on observer maneuverability.

The cost function used to formulate the optimal observer-trajectory design problem is a key feature of any solution strategy. Most of the works above use the Fisher Information Matrix (FIM) to describe the observer-trajectory quality and to derive cost functions from it. According to the Cramer-Rao bound, the inverse of the FIM is equivalent to the covariance matrix of the ideal estimator. In other words, the FIM represents the best possible performance achievable by any estimator for a given trajectory. Some work has been done using different cost functions; most notably Logothetis et al. [10] use the mutual information (Section 3.2.2) between the vehicle and the target to define the optimization cost.

Sub-optimal methods have been designed to reduce complexity and computation time. Several different approaches have been taken in order to speed the trajectory-design process. Hammel et al. [12] used function approximations that marked lower bounds of the determinant of the FIM and could be solved analytically. Logothetis et al. [18] proposed several different types of sub-optimal solutions. One type made use of reduced-order dynamic programming to decrease the optimization duration. A second type reformulated the optimization problem and used descent-based techniques to optimize the next time step. Function parameterizations were used by both Huster et al. [66] and LeCadre [19] to reduce the size of the search space, as was enumeration with pruning by Logothetis et al. [10].

Visual Tracking and Motion Planning Under Visibility Constraints

The problem of tracking a moving target among obstacles has received considerable attention in the field of computer science. *Visual tracking* and *visual servoing* applications use visual feedback to control the behavior of the observer directly [84, 85]. Examples include missile interception, mechanical part

manipulation and assembly, and vehicle docking. No attempt is made to calculate the state of the vehicle or target in the world or to create trajectories that avoid potential obstacles.

Becker et. al [55] introduced a complementary problem called *motion planning under visibility constraints*. The main issue addressed in this class of problems is planning observer movement in order to keep a target object in view when it can be lost by leaving the sensor's range or by becoming occluded by obstacles. Applications of these planning methods include telepresence, graphic animation, medical surgery, and target tracking and interception. Initial results assumed knowledge of the target motion and the map of its environment [55, 81]. Recently, Lee [56] has developed an online algorithm that minimizes the risk that the target will escape from view in a finite amount of time. Lee uses a laser range finder to identify the target and other objects in the environment. As in all work in this area, Lee assumes an accurate measurement of the target and the world. Also, his motion planning solution contains no information-gathering component.

1.3.3 Other Related Topics

Optimal Input Design

One difficulty with designing observer trajectories to improve target-motion estimation using monocular vision is the dependence of the solution on the target motion -- the state to be estimated. Computer vision and bearing sensors in general produce measurements that are nonlinear in the target states. As a result, the estimator behavior and the subsequent optimal observer trajectory are dependent on the actual target motion. However, the whole point of the estimation and trajectory design is to determine that motion. Paraphrasing, Cochran describes this dependence as such: "You tell me the target's motion and I promise to tell you the best trajectory for estimating the target's motion [39]."

The field of *optimal input design*, also known as *nonlinear experiment design,* investigates the problem of determining the control input for a nonlinear system (referred to as the experiment) in order to enable calculation of one or more unknown system parameters [34 - 39]. Typical nonlinear experiments include chemical reaction rates, enzyme kinematics, molecular decay and other phenomena that use standard nonlinear regression models [35]. The main issue addressed by nonlinear experiment design is this dependence on the solution of the parameter being estimated [34 - 39]. Ford et al. [34] present an overview of the general input design problem and a survey of solution methods that exist. Nonlinear experiment solutions tend to fall into two categories - static or sequential. Static designs are experiments in which the inputs are all preselected. In contrast, in a sequential design the current input is selected using a well-defined rule that is based on previous inputs and outputs.

Static design methods may be further broken down into three categories -- locally optimal designs, expectation and minimax designs, and Bayesian-motivated designs. Locally optimal designs are designs based on some prior estimate of the parameter of interest. Ford et al. [34] stress their importance for several reasons: they provide a useful reference point for other designs, they make for good batch designs in batch sequential designs, and they may be stable over a range of parameter values. Expectation and minimax designs aim to control the properties of the design over a range of possible parameter values. An example method may seek to design the best experiment for the worst possible true parameter value. Finally, Bayesian-motivated designs incorporate prior information about the parameter into the experiment.

Sequential design methods are differentiated as being either fully sequential or batch sequential. For a batch design, the total number of measurements is divided into a number of batches. In each batch, the measurements are all taken and the parameter is estimated. Based on this estimate, the inputs for the next batch are determined. In general the total number of measurements and the batch size is determined beforehand. Fully sequential designs are equivalent to batch designs with a batch size equal to one. After every measurement, the next input is designed. For fully sequential designs, it is common to design the next input to step in the direction of steepest descent rather than calculate the full optimal input sequence each time.

Several practical solutions to the nonlinear design problem have been suggested. Ford et al. [34] conclude that sequential designs are useful when the initial estimates are poor, but that batch-sequential designs with a few batches are generally sufficient. However, they also conclude that suboptimal designs are often better than optimal ones: they can be more robust to poor initial guesses and to the noise distribution. Chaudhari and Mykland [35, 36] propose a hybrid scheme using an initial static design based on a random selection of inputs followed by a fully sequential stage. They show that in the limit this method leads to efficient estimation of the parameter of interest.

The new trajectory design method presented in this dissertation is structurally similar to the hybrid scheme proposed by Chaudhari and Mykland [35, 36]. An initial static design based on a predefined set of inputs is followed by a batch sequential method that uses a combination locally-optimal and Bayesian-motivated design for each batch. Optimal-input design methods are typically used for nonlinear regression problems, and this dissertation represents the first application of these concepts to target-motion estimation using monocular vision.

1.4 Contributions

The goal of the work presented in this dissertation is to enhance the performance of target-motion estimation using monocular vision for autonomous observer vehicles by generating helpful observer trajectories. Previous work has investigated many problems and issues related to this goal but has not addressed several key features. In particular, this thesis addresses (i) the specific characteristics of the computer vision sensor, (ii) the need for short planning times in order to adjust to new information, and (iii) the inherent dependence of the optimal trajectory on the uncertain target state being estimated necessitate the development of new techniques.

The main contribution of this dissertation is the development of a novel observer-trajectory generator to improve the performance of target-motion estimation using monocular vision. With this method, trajectories can be created that allow an autonomous vehicle capable of sensing its own state (position and velocity) to estimate the state of target objects with a single camera. The trajectories can be designed in real-time in order to allow the observer to adjust and respond to new information as it becomes available. Furthermore, this new generator addresses the inherent dependence of the optimal trajectory on the uncertain target state.

In the course of developing the new trajectory generator the following additional contributions are also made:

- The predicted estimate error covariance is identified as the most useful metric for estimation performance and subsequent trajectory design. The estimate error covariance matrix better predicts estimation performance than the typically used Fisher Information Matrix (FIM) and mutual information under the observer limitations and target geometry present in this work.
- A new core observer-trajectory design algorithm is developed for improving target-motion estimation using monocular vision when the initial target position and velocity are assumed known. The algorithm uses a pyramid, breadth-first enumeration algorithm to generate trajectories in real-time. It includes sensor field-of-view and observer dynamic constraints that do not appear in the related sonar applications. Trajectories can be generated that minimize the estimate uncertainty in a given time or that minimize the time taken to reach a given uncertainty bound. The algorithm serves as the core for the full observer-trajectory generator, needed when the initial state of the target is in fact not known.
- Key parameters are identified that influence the performance of the design algorithm. The proper trade-off between these parameters has a significant impact on the final performance of the method. The important parameters include the number of observer maneuvers and the number of discrete heading intervals used to define the search space.

- A new observer-trajectory generator and high-level monitor are created that integrate the trajectory design algorithm into an operational system that considers the uncertainty of the target estimate. This trajectory generator addresses several of the issues that arise when developing a real-world system, including estimate initialization, response to a changing world model, and unexpected losses of the target vehicle from the field of view.
- Simulation results are generated that demonstrate the successful application of the observer-trajectory generator. Trajectories for the two desired cost functions -- minimum uncertainty in given time and minimum time for desired uncertainty -- are produced, the latter for the first time ever.
- An experimental system is developed. The Stanford University Aerospace Robotics Laboratory's Micro Autonomous Rover platform is augmented to include an observer robot that can visually track colored and infrared targets. A new computer vision system is also designed and integrated into the test bed.
- The core observer-trajectory-design algorithm and the full trajectory generator are experimentally demonstrated on the Micro Autonomous Rover platform. The system is able to track a moving target successfully given no prior information about its initial state. Both the minimum-time and minimum-uncertainty objectives are demonstrated, as well as the system's ability to recover from unexpected changes in the world, including changes in the target state and its disappearance from view.

1.5 Roadmap

The first chapter of this thesis gives the motivation for the research, a description of related work, a statement of the research objectives, and a summary of contributions.

Chapter 2 defines the target-motion estimation problem and includes details of the monocular-vision solution. Details of the computer-vision sensor and the non-holonomic observer kinematics are discussed. Finally, estimator design is discussed in order to show how both the target and observer state enter the estimator equations.

Chapter 3 presents a discussion of nonlinear observability and makes the connection between observer trajectory and estimation performance. Several possible metrics are discussed that describe the quality of the trajectory as it relates to the estimator performance. The advantages and disadvantages of the metrics are discussed and the predicted error covariance is shown to be the best for target-motion estimation.

Chapter 4 presents, for the case of known initial target state, a new core observer-trajectory design algorithm whose quality metric is based on the predicted error covariance. The computer vision limitation and non-holonomic vehicle kinematics of a wheeled rover are taken into account by using a breadth-first enumeration

algorithm. Using this algorithm, the fixed-accuracy, minimum-time solution is made possible for the first time.

Chapter 5 presents the new full observer-trajectory generator that extends the core trajectory-design algorithm to enable successful estimation of the target position and velocity for the case that the initial target state is not known. Important components of the new generator are initialization of the target-state estimate, replanning as new information becomes available, and responding to other dynamic events such as losing the object.

Chapter 6 contains a description of the hardware used to perform the experiments in this thesis. Two micro autonomous rovers are used as the observer and target vehicles. The main system components, including all sensors, are described.

Chapter 7 presents results, both in simulation and on the experimental hardware, that validate the new observer-trajectory generator and the core trajectory-design algorithm. The experimental demonstration shows the successful estimation of target motion under varied conditions and desired costs.

Chapter 8 gives a summary of the results of this dissertation and a discussion of future work.

CHAPTER 2

Target-Motion Estimation Using Monocular Vision

This chapter introduces the details of target-motion estimation using monocular vision. Components described in this chapter include the monocular-vision sensor, the target and observer models, and the target-motion estimator.

2.1 Introduction

Target-motion estimation refers to the general process of determining the position and velocity of an object of interest in the world, in either relative or global coordinates. Every such system fuses measurements from a relative-position sensor (a sensor capable of measuring one or more components of the position of a target object with respect to that sensor) with other information about that sensing device. In the context of this dissertation, the term *target-motion estimation using monocular vision* will refer to the specific set of sensors and system configuration represented in Figure 2.1 -- a single camera measures the relative bearing between the target object and the autonomous observer, and the camera motion is measured by additional navigation sensors such as GPS. The work presented in this dissertation is based on this specific system configuration.

The target-motion-estimation system has several major components. The monocular vision system provides a relative measurement of the direction to the object of interest. The observer vehicle on which the camera

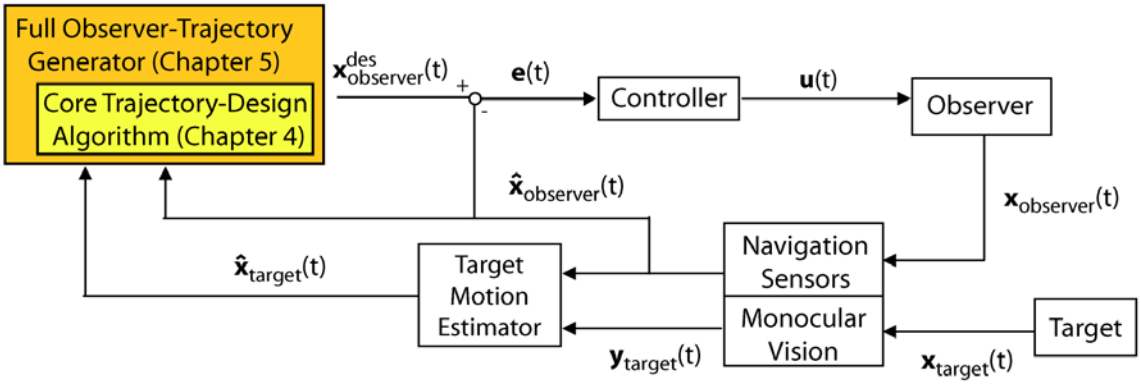

Figure 2.1 Block diagram for target-motion estimation using monocular vision.

sits has navigation sensors which determine its motion. The output of these sensors are then fused by the target-motion estimator to estimate position and velocity of the target object. An observer-trajectory-design algorithm and observer-trajectory generator use the target estimate as well as the navigation data to command the motion of the observer vehicle. This chapter presents the details of the Observer, Target, Monocular Vision, Navigation Sensors, and Target Motion Estimator components. The Controller is discussed briefly in Chapter 6 while Chapter 4 and Chapter 5 are devoted to the Core Trajectory-Design Algorithm and The Full Observer-Trajectory Generator, respectively.

The object of interest, whose motion is being estimated, is called the *target*. The target is sensed by a single camera which provides a bearing measurement to the target with respect to the observer vehicle. Such a system that uses only bearing measurements for relative position is called *bearings only* [1 - 20].

While the camera is the only sensor that provides a relative measurement to the target, the camera motion must also be known in order to determine the target motion. In general, when the camera is mounted on a vehicle that has navigation sensors and actuators, they are not collocated with the camera. This dissertation assumes a fixed, known transformation between the camera, all navigation sensors, and all actuators. Therefore, knowing and controlling the motion of the camera is equivalent to knowing and controlling the motion of the vehicle. The term *observer* is used to refer to the total vehicle and camera system and the necessary transformation of frames is implicitly assumed.

For ease of discussion and presentation, several simplifications are made in this dissertation, and specific configurations of the target, observer, and motion estimator are chosen. They are:

- Motion is restricted to a two-dimensional plane.
- The target is a point object moving with constant velocity.
- The observer robot is a wheeled rover.
- An Extended Kalman Filter using the modified polar coordinate system is used as the target-motion estimator.

The details of the target-motion estimation system are discussed in the sections that follow. The observer-trajectory analysis, trajectory-design algorithm, and full trajectory generator presented in Chapter 3 - Chapter 5 are all general and extend to three dimensions and other configurations. Section 4.6 and Section 8.2 discuss the specific ramifications of the simplifications listed above.

2.1.1 Problem Geometry

The geometry for the target-motion-estimation problem is shown in Figure 2.2. For the work presented in this dissertation, motion is restricted to the two-dimensional plane.

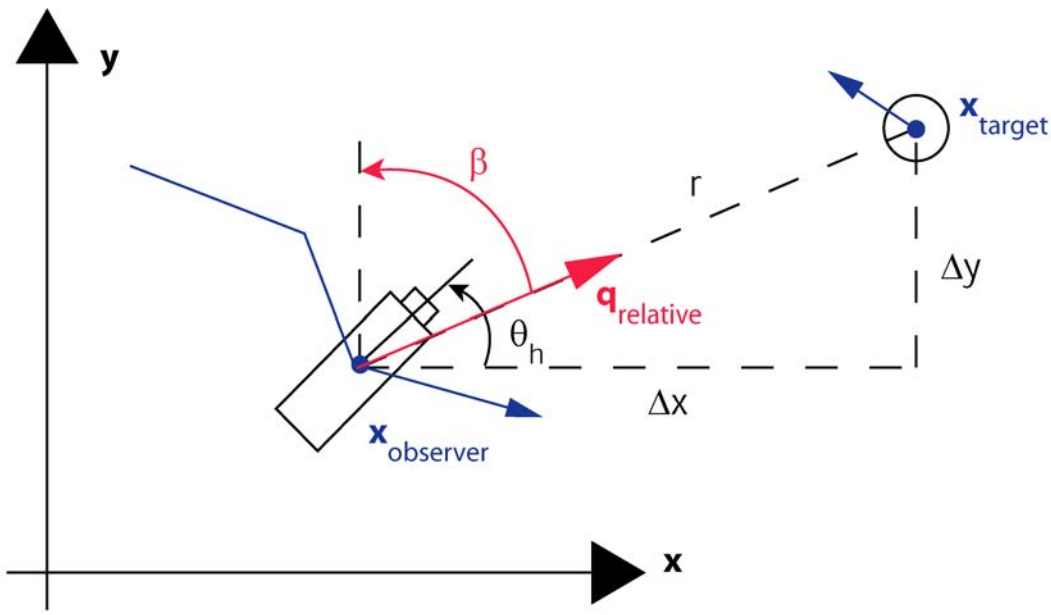

Figure 2.2 System geometry

The goal of the target-motion estimation is to calculate

$$\boldsymbol{x}_{target}(t) = \begin{bmatrix} x_{target}(t) \\ y_{target}(t) \\ \dot{x}_{target}(t) \\ \dot{y}_{target}(t) \end{bmatrix} \tag{2.1}$$

given the observer position and velocity (measured by the navigation sensors)

$$\boldsymbol{x}_{observer}(t) = \begin{bmatrix} x_{observer}(t) \\ y_{observer}(t) \\ \dot{x}_{observer}(t) \\ \dot{y}_{observer}(t) \end{bmatrix} \tag{2.2}$$

and bearing measurement $\boldsymbol{y}_{target}(t)$ in the form of the bearing angle $\beta(t)$ or the unit relative position $\boldsymbol{q}_{relative}(t)$ (measured by the computer vision sensor) where

$$\boldsymbol{x}_{relative} = \boldsymbol{x}_{target} - \boldsymbol{x}_{observer} = \begin{bmatrix} \Delta x \\ \Delta y \\ \dot{\Delta x} \\ \dot{\Delta y} \end{bmatrix} \tag{2.3}$$

$$r = \sqrt{\Delta x^2 + \Delta y^2} \tag{2.4}$$

$$\beta = \operatorname{atan}(\Delta y / \Delta x) \tag{2.5}$$

$$\boldsymbol{q}_{relative} = \begin{bmatrix} \Delta x / r \\ \Delta y / r \end{bmatrix} \tag{2.6}$$

$$\boldsymbol{y}_{target} = \{\beta, \boldsymbol{q}_{relative}\} . \tag{2.7}$$

2.2 Monocular Computer Vision

The general term *computer vision* refers to the extraction of useful information from a digital image. In some cases it is also used to refer to the extrapolation from information about the image to information about the real world, such as the identification of a person or the calculation of the size and shape of an object. In this dissertation, *monocular computer vision* (or just *monocular vision*) refers to the use of a single camera to generate the time history of the location of a target object in a digital image. The process of determining the target motion from the image measurements is target-motion estimation.

Computer vision encompasses several subtasks which are all subjects of current research beyond the scope of this work [58, 59]. Figure 2.3 presents the flow diagram for a typical system using computer vision. The raw processing, segmentation, classification, and tracking steps are all restricted to the image only and comprise the computer vision system as defined above. This dissertation assumes the existence of such a system that can classify and track the target object in the image.

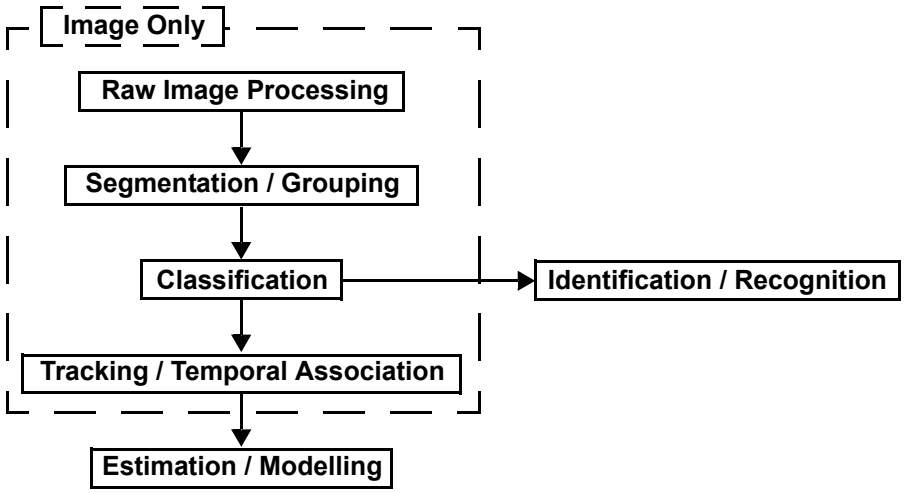

Figure 2.3 Flow diagram for a typical system using computer vision system

Within an image, a set of pixels with interesting characteristics is called a *feature* and is identified by its location within the image. The image of a single, real-world object may contain one or more features. The distances between features in the image and their change in time contain information about the size and shape of the object. For the work developed in this dissertation, objects are assumed to contain a single feature. This situation can occur when an object is far away from the camera and represents only a point on the image plane or when it is very close and only a single feature can be seen.

Assuming the location of an object feature within an image can be determined, this work models monocular computer vision as a bearing sensor with a limited field of view. Figure 2.4 shows the projection geometry of the standard two-dimensional pinhole-camera model [58, 59]. Every image pixel represents the projection of a two-dimensional position onto the one-dimensional image plane. From Figure 2.4 it is clear that every point on an image, or every pixel in the case of a digital image, represents the projection of a ray onto the camera image plane. Conversely, the physical object that is represented by a pixel lies somewhere along the ray that projects outward through the focal point of the camera. In this sense, a computer vision measurement provides the direction to the object that the feature represents.

The following equation relates the position of a point target in the world to the location of its projected feature in the image plane:

$$x_c = f \cdot (x_s / y_s) \tag{2.8}$$

where $\boldsymbol{x}_s = \begin{bmatrix} x_s \\ y_s \end{bmatrix}$ is the location of the target in relative camera coordinates.

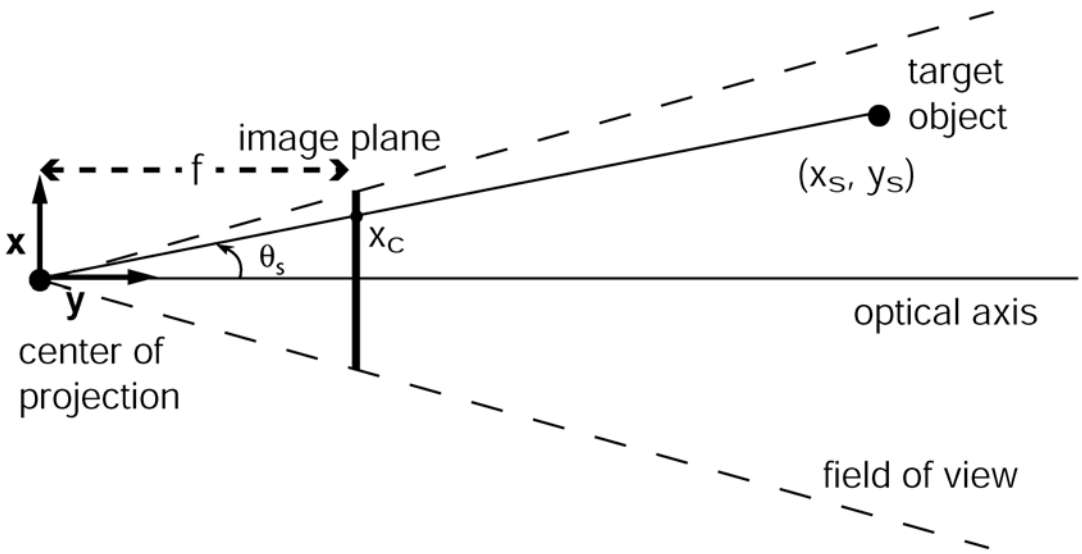

Figure 2.4 Pinhole camera projection geometry

Without loss of generality the focal length can be set to $f = 1$ and the vision measurement x_c can be transformed into the measurements of Equation **(2.5)** and Equation **(2.6)**:

$$\theta_s = \operatorname{atan}(x_c) \tag{2.9}$$

$$\beta = \frac{\pi}{2} - (\theta_s + \theta_h) \tag{2.10}$$

$$\boldsymbol{q}_{relative} = \boldsymbol{T}_{s2w} \cdot \begin{bmatrix} x_c \\ 1 \end{bmatrix} \cdot \frac{1}{\sqrt{x_c^2 + 1}} \tag{2.11}$$

$$\boldsymbol{T}_{s2w} = \begin{bmatrix} -\sin(\theta_h) & \cos(\theta_h) \\ \cos(\theta_h) & \sin(\theta_h) \end{bmatrix} \tag{2.12}$$

where θ_h is the observer's heading (Figure 2.2) and $\boldsymbol{T}_{s2w}$ is the transformation matrix from camera to world coordinates.

2.3 Target Model

In order to design a target-motion estimator, a model of the target behavior is needed. The estimator used in this dissertation assumes a constant-velocity target. On-going research efforts are investigating the design of estimators that can consider more-complex target models [1 - 9]. The assumption of a constant-velocity target directly affects the output of the trajectory generator and the resultant observer behavior. However, the development of the observer-trajectory generator itself is independent of this assumption and the

trajectory generator could calculate paths for more-complex target motion if the corresponding estimator is used.

Since the target-motion estimator is implemented on a digital computer, a discrete formulation of the target kinematics is required and therefore the target motion is fully described by

$$\boldsymbol{x}_{target}[k] = \begin{bmatrix} x_{target}[k] \\ y_{target}[k] \\ \dot{x}_{target}[k] \\ \dot{y}_{target}[k] \end{bmatrix}. \tag{2.13}$$

The well known kinematic equation of the constant-velocity target is:

$$\boldsymbol{x}_{target}[k+1] = \Phi \cdot \boldsymbol{x}_{target}[k] \qquad \Phi = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

where $k = \{0, 1, 2, \ldots\}$ is the discrete time interval and Δt is the sampling period.

2.4 Observer Navigation Sensor and Kinematic Model

An explicit model of the observer is not needed in order to estimate the target state, even though the monocular vision measurement is a function of observer position. It is sufficient that the location and velocity of the observer is known at the instant every vision measurement is taken. The motion of the observer between samples is unimportant as long as the navigation sensors continue to measure it. The algorithms developed in this dissertation assume the existence of an independent navigation system that provides the observer position and velocity as needed.

Nevertheless, in order to design feasible trajectories, the observer kinematic equations must be considered by the trajectory generator. This work uses a two-wheeled rover -- a common example of a kinematically constrained system -- as the observer vehicle. Figure 2.5 shows the velocity geometry of the two-wheeled rover. Each wheel is independently driven to allow for control of both linear velocity and angular rate about the vertical (z) axis. The kinematic equations for the rover are:

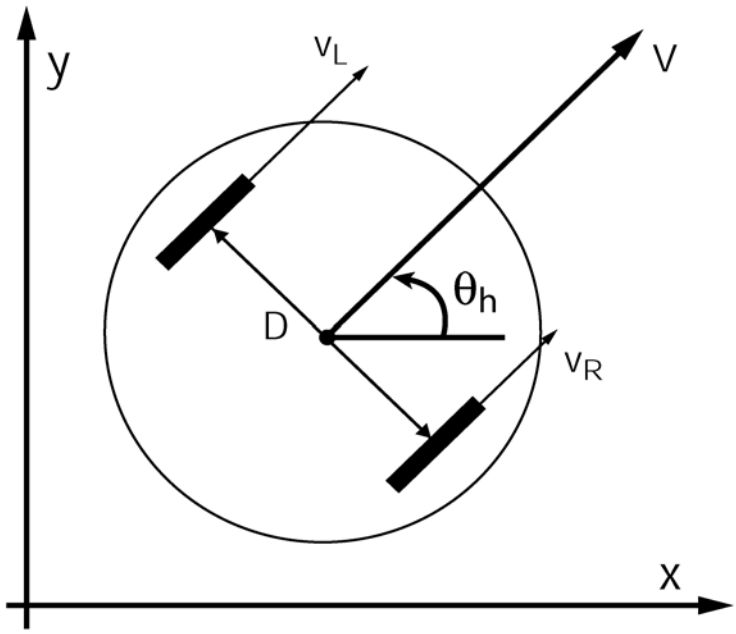

Figure 2.5 Wheeled-robot velocity geometry

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta}_h \end{bmatrix} = (1/2) \cdot \begin{bmatrix} \cos(\theta_h) & \cos(\theta_h) \\ \sin(\theta_h) & \sin(\theta_h) \\ 1/D & -(1/D) \end{bmatrix} \cdot \begin{bmatrix} v_L \\ v_R \end{bmatrix}. \tag{2.15}$$

where D is the spacing between the wheels and $[v_L, v_R]$ are the velocities of the left and right wheels, respectively. In practice, these simple, purely geometric relations are achieved by controlling the speeds of the wheels very tightly, using strong feedback such that the wheel speeds equal the commanded wheel speeds.

From the equations above it can be seen that the forward velocity of the rover $\boldsymbol{V}$ always points in the same direction as its heading.

The kinematics of the wheeled rover have significant impact on the requirements of the observer-trajectory design algorithm (Chapter 4) and the full trajectory generator (Chapter 5). In combination with the vision field-of-view limitation, the rover-motion constraint results in an observer that cannot keep a fixed camera pointed at a single location if it must change direction, and therefore it may become necessary to lose view of the target. This restriction drives several details of the trajectory generator. The full impact of the wheeled-rover observer is discussed in Chapter 4 and Chapter 5.

2.5 Estimator Formulation

Estimator design has been studied extensively for target-motion estimation, particularly in the context of bearings-only tracking applications [1 - 9]. Intrinsic nonlinearities prohibit the use of simple linear solutions, so the Extended Kalman Filter (EKF) framework has received considerable attention [7]. The EKF preserves two important advantages of the standard Kalman filter while incorporating nonlinear sensor and process dynamics. First, the EKF maintains the recursive structure of the Kalman Filter, allowing it to run in real-time and to update continually as new information becomes available. Second, the Extended Kalman Filter calculates the state estimate as well as a measure of the uncertainty of that estimate. Because all sensors have some noise and all estimators are inexact, this uncertainty measure allows for appropriate interpretation or use of the state estimate by other systems.

In order to quantify the effects of the observer motion on estimator performance, a single estimator formulation must be assumed. The specific target-motion estimator used throughout this dissertation is based on the Extended Kalman Filter framework. A general description of this framework is presented in the ensuing section, followed by detailed equations for the target-motion estimation problem.

All discussion in subsequent chapters in this dissertation assumes the modified polar coordinate system is used to solve the target-motion estimation problem. It is well known that the choice of coordinate system has a large impact on the performance of the EKF when applied to target-motion estimation [6]. In practice, estimator performance is improved by the use of the modified polar coordinate system which separates the observable (bearing) and unobservable (range) states [7]. However, the cartesian coordinate system better illustrates the nonlinearities in the system and the dependence of estimator performance on the observer and target motions. Therefore the cartesian coordinate system formulation is first described in Section 2.5.2. The modified polar system is then described in Section 2.5.3.

2.5.1 The Extended Kalman Filter Framework

The Extended Kalman Filter applies to systems with state equations of the form:

$$\boldsymbol{x}[k+1] = f(\boldsymbol{x}[k], \boldsymbol{u}[k]) + \boldsymbol{w} \tag{2.16}$$

$$\boldsymbol{y}[k] = h(\boldsymbol{x}[k], \boldsymbol{u}[k]) + \boldsymbol{v} \tag{2.17}$$

where

$\boldsymbol{x}[k]$ = true state vector at sample k
$\boldsymbol{u}[k]$ = control input at sample k
$f(\boldsymbol{x}[k], \boldsymbol{u}[k])$ = state transition function which relates $\boldsymbol{x}[k+1]$ to $\boldsymbol{x}[k]$
$\boldsymbol{w}$ = zero mean gaussian process noise with covariance matrix $E[\boldsymbol{w}\boldsymbol{w}^T] = \boldsymbol{Q}$

and

$$\begin{aligned} \boldsymbol{y}[k] &= \text{measurement at sample k} \\ h(\boldsymbol{x}[k], \boldsymbol{u}[k]) &= \text{measurement function which relates } \boldsymbol{y}[k] \text{ to } \boldsymbol{x}[k] \\ \boldsymbol{v} &= \text{zero mean gaussian sensor noise with covariance matrix } E[\boldsymbol{v}\boldsymbol{v}^T] = \boldsymbol{R} \end{aligned}$$

The EKF generates an estimate in two steps. The first step is the time update or prediction step in which the dynamics of Equation **(2.16)** are propagated forward in time:

$$\bar{\boldsymbol{x}}[k] = f(\hat{\boldsymbol{x}}[k-1]) \tag{2.18}$$

$$\begin{aligned} \bar{\boldsymbol{P}}[k] &= E[(\bar{\boldsymbol{x}}[k] - \boldsymbol{x}[k])(\bar{\boldsymbol{x}}[k] - \boldsymbol{x}[k])^T] \\ &= \boldsymbol{F}[k-1]\hat{\boldsymbol{P}}[k-1]\boldsymbol{F}^T[k-1] + \boldsymbol{Q}[k-1] \end{aligned} \tag{2.19}$$

$$\boldsymbol{F}[k-1] = \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x} = \hat{\boldsymbol{x}}[k-1]} \tag{2.20}$$

where the overbar ($\bar{x}$) indicates the estimate includes the current measurement, while the carat ($\hat{x}$) indicates the estimate does not.

The second step is the measurement update or innovation step in which the current measurement is used to correct the estimate:

$$\hat{\boldsymbol{x}}[k] = \bar{\boldsymbol{x}}[k] + \boldsymbol{K}[k] \cdot (\boldsymbol{y}[k] - h(\bar{\boldsymbol{x}}[k])) \tag{2.21}$$

$$\begin{aligned} \hat{\boldsymbol{P}}[k] &= E[(\hat{\boldsymbol{x}}[k] - \boldsymbol{x}[k])(\hat{\boldsymbol{x}}[k] - \boldsymbol{x}[k])^T] = E[(\tilde{\boldsymbol{x}}[k])(\tilde{\boldsymbol{x}}[k])^T] \\ &= (\boldsymbol{I} - \boldsymbol{K}[k]\boldsymbol{H}[k])(\bar{\boldsymbol{P}}[k]) \end{aligned} \tag{2.22}$$

$$\boldsymbol{K}[k] = \bar{\boldsymbol{P}}[k]\boldsymbol{H}^T[k](\boldsymbol{H}[k]\bar{\boldsymbol{P}}[k]\boldsymbol{H}^T[k] + \boldsymbol{R}[k])^{-1} \tag{2.23}$$

$$\boldsymbol{H}[k] = \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x} = \hat{\boldsymbol{x}}[k]} \tag{2.24}$$

The only additional requirement to apply the EKF is that the filter is initialized with a prior state estimate and covariance matrix

$$\begin{aligned} \hat{\boldsymbol{x}}[0] &= \boldsymbol{x}_0 \\ \hat{\boldsymbol{P}}[0] &= \boldsymbol{P}_0 . \end{aligned} \tag{2.25}$$

2.5.2 Cartesian Filter

Section 2.2 and Section 2.3 describe the sensor and target models respectively in cartesian coordinates. Therefore, the estimator state is the target state vector as defined by Equation **(2.13)** and the state update behavior is defined by Equation **(2.14)**. Because Equation **(2.14)** is linear, Equation **(2.18)** - Equation **(2.20)** become the standard Kalman update equations with $\boldsymbol{F} = \Phi$. Furthermore, the target is assumed to move with no acceleration or process noise so $\boldsymbol{Q} = \boldsymbol{0}$. (Note, for ease of presentation the sample index has been dropped from the equations. Unless specifically included, assume all variables are indexed at sample k.)

The measurement equation is derived from the vision projection equations, so from Equation **(2.8)**:

$$h(\boldsymbol{x}_{target}) = x_c = [x_s / y_s] \tag{2.26}$$

where

$$\boldsymbol{x}_s = \boldsymbol{T}_{w2s} \cdot (\boldsymbol{x}_{target} - \boldsymbol{x}_{observer}) \tag{2.27}$$

$$\boldsymbol{T}_{w2s} = (\boldsymbol{T}_{s2w})^T = \begin{bmatrix} \cos(\theta_h) & \sin(\theta_h) \\ -\sin(\theta_h) & \cos(\theta_h) \end{bmatrix}. \tag{2.28}$$

The Jacobian of the measurement function is then

$$\boldsymbol{H} = \begin{bmatrix} \begin{bmatrix} \frac{1}{y_s}, \frac{x_s}{y_s^2} \end{bmatrix} \boldsymbol{T}_{w2s} & 0 & 0 \end{bmatrix}. \tag{2.29}$$

It is important to note the dependence of the Jacobian on both the observer and target states. The observer and target states enter the equations in Equation **(2.27)** while only the observer state enters through Equation **(2.28)**. Because the estimate uncertainty, in the form of the covariance matrix P, is a function of these variables, the estimator performance is also a function of the observer and target states. The dependence on the observer state is exploited by the trajectory generator in order to improve the performance of the estimation. The second dependence, on the target state, introduces an issue into the observer-trajectory-design process. As will be discussed further in Chapter 3, the motion of the target must be known in order to generate the optimal observer trajectory. However the whole purpose of designing the trajectory is to enable estimation of that target motion.

In practice, the cartesian-coordinate formulation of the target-motion estimator performs poorly. The feedback of estimate uncertainty through the measurement Jacobian often leads to premature covariance collapse and filter divergence [7]. When the filter does remain stable, estimates tend to be heavily biased.

Good estimator performance can be achieved by using an alternate filter formulation, the most popular of which uses the modified-polar coordinate system.

2.5.3 Modified Polar Filter

An alternative target-motion estimator is presented using the modified polar-coordinate system within the Extended-Kalman-Filter framework. Analysis has shown that this filter formulation results in asymptotically unbiased state estimates [7]. The main advantage of the modified polar-coordinate system is that it decouples the observable and unobservable target states, preventing ill-conditioning of the covariance matrix and its premature collapse. The work presented in the remainder of this dissertation is based on a modified polar target-motion estimator.

Referring to Figure 2.2, the state vector in the modified polar coordinate system is

$$\boldsymbol{x} = \boldsymbol{x}_{mp} = \begin{bmatrix} \beta \\ 1/r \\ \dot{\beta} \\ \dot{r}/r \end{bmatrix} = \begin{bmatrix} \beta \\ \zeta \\ \dot{\beta} \\ \dot{\eta} \end{bmatrix}. \tag{2.30}$$

In contrast to the cartesian filter in which the process dynamics were linear and the measurement was nonlinear, the modified polar (MP) filter has a simple linear measurement equation

$$y = \beta + v = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \cdot \boldsymbol{x} + v \tag{2.31}$$

where v is the zero mean gaussian noise and Equation **(2.10)** is used to transform the monocular vision measurement into the proper form.

The state transition function for the MP filter contains all of the nonlinearities of the target-motion estimation problem. While these equations are complex, they are easily derived from the cartesian equations. First, the target state is converted into cartesian coordinates

$$\boldsymbol{d}[k-1] = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}\Bigg|_{k-1} = \left(\boldsymbol{x}_{observer} + \frac{1}{\zeta} \cdot \begin{bmatrix} \sin(\beta) \\ \cos(\beta) \\ \dot{\beta}\sin(\beta) + \dot{\eta}\cos(\beta) \\ \dot{\beta}\cos(\beta) - \dot{\eta}\sin(\beta) \end{bmatrix} \right)\Bigg|_{k-1} \tag{2.32}$$

Next, the target kinematics are propagated forward using Equation **(2.14)**

$$\boldsymbol{d}[k] = \Phi \cdot \boldsymbol{d}[k-1] \tag{2.33}$$

Finally, the target state vector is converted back into modified polar coordinates

$$\boldsymbol{x}[k] = \left.\begin{bmatrix} \operatorname{atan}(x/y) \\ \dfrac{1}{\sqrt{x^2+y^2}} \\ \dfrac{y\dot{x}-x\dot{y}}{x^2+y^2} \\ \dfrac{x\dot{x}+y\dot{y}}{x^2+y^2} \end{bmatrix}\right|_k . \tag{2.34}$$

The Jacobian of the state transition equation is written [5]

$$\boldsymbol{F}[k-1] = \boldsymbol{M} \cdot \Phi \cdot \boldsymbol{N} \tag{2.35}$$

where

$$\begin{aligned} \boldsymbol{M} &= \frac{\partial \boldsymbol{d}}{\partial \boldsymbol{x}}[k] \\ &= \left.\left(\zeta \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -\zeta & 0 & 0 \\ -\dot{\eta} & -\dot{\beta} & 1 & 0 \\ \dot{\beta} & -\dot{\eta} & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\beta) & -\sin(\beta) & 0 & 0 \\ \sin(\beta) & \cos(\beta) & 0 & 0 \\ 0 & 0 & \cos(\beta) & -\sin(\beta) \\ 0 & 0 & \sin(\beta) & \cos(\beta) \end{bmatrix} \right)\right|_k \end{aligned} \tag{2.36}$$

$$\begin{aligned} \boldsymbol{N} &= \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{d}}[k-1] \\ &= \left.\left(\frac{1}{\zeta} \cdot \begin{bmatrix} \cos(\beta) & \sin(\beta) & 0 & 0 \\ -\sin(\beta) & \cos(\beta) & 0 & 0 \\ 0 & 0 & \cos(\beta) & \sin(\beta) \\ 0 & 0 & -\sin(\beta) & \cos(\beta) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{1}{\zeta} & 0 & 0 \\ \dot{\eta} & -\frac{\dot{\beta}}{\zeta} & 1 & 0 \\ -\dot{\beta} & -\frac{\dot{\eta}}{\zeta} & 0 & 1 \end{bmatrix} \right)\right|_{k-1} . \end{aligned} \tag{2.37}$$

2.6 System Equations

The following sets of equations are used to define the system used throughout the remainder of this dissertation:

Target Kinematics

Equation **(2.32)** - Equation **(2.37)** are the kinematic equations of the constant-velocity target in modified polar coordinates (Equation **(2.30)**). The target-motion estimation is carried out by an Extended Kalman Filter (EKF) using this set of equations. The result is converted into cartesian coordinates for the observer-trajectory-design algorithm using Equation **(2.32)** and Equation **(2.35)**.

Vision Measurement

Equation **(2.31)** gives the measurement equation for the monocular vision system used by the EKF. The output of the monocular vision sensor is transformed from image-plane coordinates into the modified polar coordinates by Equation **(2.10)**.

Observer Kinematics

Equation **(2.15)** represents the kinematic equation of the wheeled observer rover.

CHAPTER 3 *Trajectory Analysis*

This chapter presents the development of an observer-trajectory quality metric that evaluates the impact of the observer path on the performance of the target-motion-estimation system. Target-motion estimation is a nonlinear process and as a result the performance of the estimation depends on the path of the observer. This concept, called *dynamic observability*, is exploited in Chapter 4 and Chapter 5 to generate observer trajectories that enable and improve the target-motion estimation. In order to generate these observer trajectories, a quality metric is needed that defines the notion of a good trajectory by correlating the performance of the estimator output with the choice of observer trajectory.

Such a trajectory-quality metric is developed by identifying the measure of estimator performance most appropriate to the trajectory-design problem studied here. Trajectory quality and the notion of a good trajectory is defined in the context of target-motion-estimation performance. In order to define a good trajectory, vis a vis estimator performance, it is necessary to define the notion of good performance first. Therefore the problem of identifying the best trajectory-quality metric is the problem of determining the best description of estimation performance. Figure 3.1 depicts the relationship between the trajectory quality metric and the estimator performance measure.

The concept of observability provides tools to analyze estimation problems and answer questions of solvability and performance. These tools are well developed for linear dynamic systems and partially extend to nonlinear problems such as target-motion estimation. However, observability analysis makes

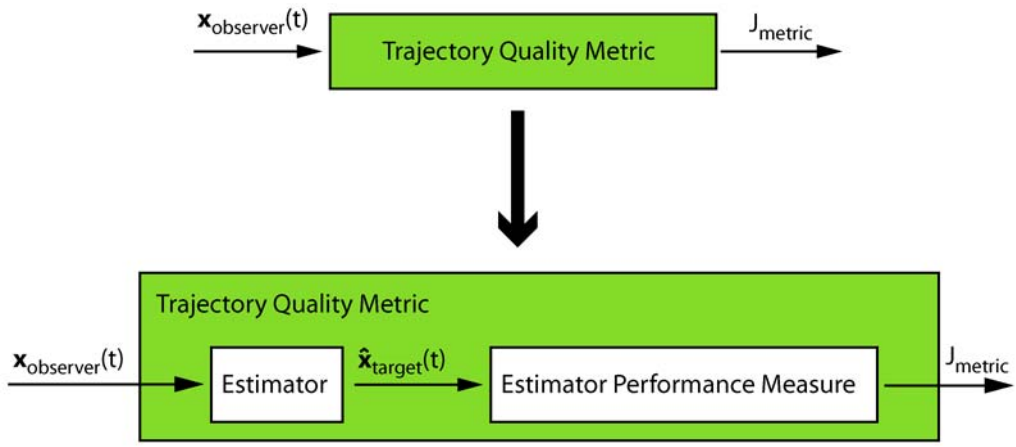

Figure 3.1 Determining the best trajectory quality metric is equivalent to identifying the best estimator performance measure

claims only on the structure of the estimation problem and limits of performance. Additional tools such as mutual information and the error covariance matrix are used to incorporate specific estimator formulations and produce more cogent assessments of estimator performance.

This chapter first presents a brief discussion of nonlinear observability and clarifies the differences between solvability analysis and performance prediction. Several estimator-performance metrics are then introduced and compared, and the best measure of estimate (and therefore trajectory) quality is identified. The major conclusion of this chapter is that the estimate error covariance is a better measure of trajectory quality than the typically used Fisher Information Matrix (FIM) and mutual information for the design of trajectories that improve target-motion estimation using monocular vision.

3.1 Observability Analysis

Observability analysis provides tools for examining an estimation problem and determining properties of the solutions. These tools look at the structure of the estimation problem -- the sensor models used, the geometry of the problem, the process dynamics -- and make claims about the possibility of generating a solution and the quality of that solution. Observability analysis is dependent on the problem, not the specific estimator formulation used to solve the problem.

3.1.1 Observability

The first important property to be determined when attempting to solve an estimation problem is whether there is enough information available to the estimator to solve the problem at all. In other words, can the estimation problem possibly be solved. That is:

Given a state vector $\boldsymbol{x}(t)$ comprised of variables whose values are desired and a measurement $\boldsymbol{z}(t) = h(t, \boldsymbol{x}(t)) + \boldsymbol{v}$ where $\boldsymbol{v}$ is some random noise vector, a system is *observable* at time $t_1 > t_0$ if it is possible to determine the state $\boldsymbol{x}(t_0)$ by sensing $\boldsymbol{z}(t)$ in the interval (t_0, t_1). If all states $\boldsymbol{x}(t)$ corresponding to all $\boldsymbol{z}(t)$ are observable, the system is *completely observable*. [49]

Observability is well understood for linear systems. For a discrete linear system the observability criterion is developed by first creating the *observability grammian*

$$\boldsymbol{N}[k_f, 0] = \sum_{k=0}^{\kappa_f} \Phi^T[k, 0]\boldsymbol{H}^T[k]\boldsymbol{H}[k]\Phi[k, 0] \tag{3.1}$$

where $\Phi[k_2, k_1]$ is the state transition matrix (which may be time-invariant or time-varying) and $\boldsymbol{H}[k]$ is the linear measurement matrix. The system is observable if the observability grammian is nonsingular for some finite value of k_f [48].

Observability analysis becomes more involved for nonlinear systems. There are no general observability criteria for nonlinear systems. However, analysis can be performed for certain types of nonlinear systems or if certain simplifications are made. One class of nonlinear systems that can be analyzed are those systems where the measurements are nonlinear but the dynamics are linear. This is the case for the target-motion estimation problem of interest here, where the linear dynamic model is given by Equation **(2.14)**, and the nonlinear measurement model is given by Equation **(2.26)**. Observability analysis can be carried out for systems of this type in several ways [13, 45 - 47].

The first observability analysis method utilizes a transformation of the nonlinear measurement equation

$$\boldsymbol{z}[k] = h(k, \boldsymbol{x}[k]) + \boldsymbol{v}[k] \tag{3.2}$$

into pseudolinear form [13] through algebraic manipulation such that

$$\boldsymbol{z}_{pl}(k, \boldsymbol{z}[k]) = \boldsymbol{H}_{pl}(k, \boldsymbol{z}[k], \boldsymbol{x}[k]) \cdot \boldsymbol{x}[k] + \boldsymbol{n}[k]\,. \tag{3.3}$$

The pseudolinear measurement matrix $\boldsymbol{H}_{pl}[k]$ is then used in Equation **(3.1)** to generate an observability grammian which is analyzed as if it described a linear system. The main drawback of the pseudolinear form is the transformation of the original noise vector. In many cases the original noise vector is transformed by a nonlinear equation, and since it is assumed gaussian, the transformed noise $\boldsymbol{n}[k]$ will not be gaussian and any estimator based on this equation will be biased [14]. However, this observability analysis method still provides insight into the estimation problem and accurately describes its observability as the sensor noise tends to zero.

For the target-motion estimation problem the pseudolinear form of Equation **(2.26)** is

$$0 = \begin{bmatrix}\cos(\beta[k]) & -\sin(\beta[k]) & 0 & 0\end{bmatrix} \cdot \boldsymbol{x}_{target}[k] + \boldsymbol{n}[k] \tag{3.4}$$

where $\beta[k]$ is a function of the original measurement and is calculated from Equation **(2.10)**.

The second method uses the Fisher Information Matrix (which simplifies to the observability grammian for a linear system) to capture the observability of the system. This matrix is described in more detail in Section 3.2.1, but its equation is

$$\boldsymbol{J}_f = \sum_{k=0}^{\kappa_f} \Phi^T[k,0]\boldsymbol{H}_{lin}^T[k]\boldsymbol{R}[k]\boldsymbol{H}_{lin}[k]\Phi[k,0] \tag{3.5}$$

where $\boldsymbol{H}_{lin}[k]$ is the Jacobian of the nonlinear measurement equation (Equation **(3.2)**)

$$\boldsymbol{H}_{lin}[k] = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}} h(k, \boldsymbol{x}[k]) . \tag{3.6}$$

and $\boldsymbol{R}[k]$ is the sensor noise covariance matrix

$$\boldsymbol{R}[k] = E[(\boldsymbol{v}[k]) \cdot (\boldsymbol{v}[k])^T] . \tag{3.7}$$

The estimation problem is observable if the Fisher Information Matrix is nonsingular for a finite number of samples.

It is important to note that observability is defined at each instant of time. An estimation problem may be unobservable for some time, but then become observable as the system evolves (for example, the location of a stationary target is unobservable until the observer moves perpendicular to the line of sight to the target). Likewise, observability may be a function of other variables that also evolve over time. In particular, the target-motion estimation problem is dependent on the observer path through the nonlinear measurement equations. This dependence is the *dynamic observability* that is exploited by the trajectory generator of Chapter 4 and Chapter 5.

Although observability as defined above is a binary function -- a problem is either observable or it is not -- full interpretation of the observability grammian or Fisher Information Matrix yields additional information regarding the characteristics of the estimation problem. Given observability -- that a solution exists -- performance bounds on the accuracy of the state estimate can be determined. This discussion is left for the ensuing section. When a problem is unobservable, the observability matrices describe which components

of the state vector cannot be determined. Likewise, observability relates to the convergence of recursive estimators used to solve the estimation problem.

In many cases a system is unobservable because a subset of the components of the state vector cannot be determined. If some components of the state vector can be determined while others cannot, the system is considered *partially observable*. For example, the target-motion estimation of interest here is partially observable with no observer motion when cast in the modified polar coordinate system. The computer vision sensor gives a bearing measurement to the target and with more than one measurement the bearing and bearing rate to the target can be calculated while the range and range rate cannot. If the problem is cast without motion in cartesian coordinates neither the position nor velocity of the target can be calculated and the target's state is considered unobservable.

For recursive estimators, such as the Extended Kalman Filters presented in Section 2.5, observability relates to the convergence properties of the estimators. It has already been mentioned, and is shown in Section 3.1.3, that observability of the target-motion estimation requires sufficient observer motion. Before this sufficient motion can occur, the recursive filter begins processing the sensor measurements. While the state estimation remains unobservable or only partially observable, the behavior of the unobservable estimator states is unstable and in some cases the filter diverges [6]. Only when sufficient motion is achieved can all the estimate states begin to converge toward the true values. Incidentally, the modified polar-coordinate filter performs well because the partial observability of the bearing and bearing rate limit the amount of divergence that occurs before full observability is achieved. The cartesian-coordinate filter performs poorly because all states are divergent before observability occurs. Feedback of these divergent estimates within the filter amplifies the estimation error and often leads to filter instability [6].

3.1.2 Performance Bounds

Once observability is established, the second important set of properties of an estimation problem to determine are the performance bounds of the system. Like observability, performance bounds are computed independently from the specific estimator formulation that is eventually used. Depending on the estimator design, these bounds may be highly conservative or they may accurately predict the estimator behavior.

Estimator performance bounds are often expressed as bounds on the state-estimate-error covariance matrix that can be achieved by any estimator. For an unbiased estimator the covariance matrix is

$$\boldsymbol{P} = E[(\boldsymbol{x} - \hat{\boldsymbol{x}}) \cdot (\boldsymbol{x} - \hat{\boldsymbol{x}})^T] \tag{3.8}$$

where $\hat{x}$ is the state estimate.

The most common error bound used to describe an estimator is the Cramer-Rao bound which bounds the error covariance from below by the Fisher Information Matrix ($\boldsymbol{J}_f$) [46]

$$\boldsymbol{P} \geq \boldsymbol{J}_f^{-1} . \tag{3.9}$$

A corollary of this bound relates the variance of each estimate state to the corresponding diagonal elements of the Fisher Information Matrix (FIM). This bound is

$$\sigma_i^2 = E[(x-\hat{x})_i^2] \geq J_{f_{ii}}^{-1} \qquad i = 1, 2, \ldots, n . \tag{3.10}$$

Though the FIM bounds the error covariance from below, it is not always the tightest bound. Estimators for which the bound becomes an equality are said to be *efficient*. For inefficient estimators, tighter bounds which are more difficult to compute exist, including the Bhattacharyya and Barankin bounds[46, 47].

The estimator performance bounds can be interpreted several ways. The simplest interpretation is strictly numerical -- Equation **(3.9)** and Equation **(3.10)** express the lowest achievable values for the estimate-error covariances. A second interpretation relates to the concept of observability described above. The Fisher Information Matrix contains information about the relative observability of the estimate states. The singular values of the FIM describe the relative observability of the corresponding estimate states, showing which states can be estimated with good accuracy and which will have large uncertainties. A singular value of zero implies that the FIM is nonsingular and that the estimation problem is unobservable. However, it also shows which of the individual states is unobservable.

A third interpretation of the performance bounds is geometric. The estimate error covariance defines a confidence region around the state estimate. The shape of this region is an ellipsoid with the equation

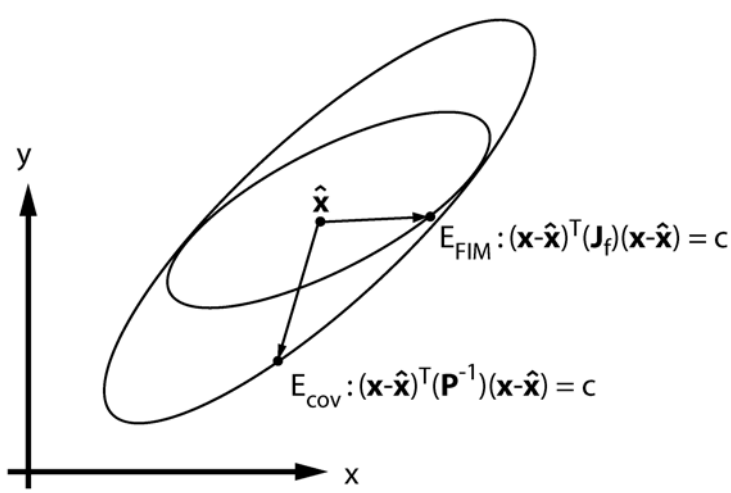

Figure 3.2 Geometric representation of the Cramer-Rao Lower Bound

$$(\boldsymbol{x}-\hat{\boldsymbol{x}})^T \cdot \boldsymbol{P}^{-1} \cdot (\boldsymbol{x}-\hat{\boldsymbol{x}}) = c \tag{3.11}$$

where c is a constant scaling parameter.

The Cramer-Rao bound states that the concentration ellipse [46] defined by Equation **(3.11)** lies either outside or on the bound on the ellipse defined by

$$(\boldsymbol{x}-\hat{\boldsymbol{x}})^T \cdot \boldsymbol{J}_f \cdot (\boldsymbol{x}-\hat{\boldsymbol{x}}) = c\,. \tag{3.12}$$

Figure 3.2 depicts this geometric relationship.

3.1.3 Motion Requirements

Before investigating metrics for trajectory performance, it is beneficial to apply the observability analysis tools to derive sufficient observer motion conditions. The observability requirement is derived as a function of the observer trajectory and then solved for the sufficient motion.

First consider the stationary target depicted in Figure 3.3. Two measurements of the target bearing are taken at times t_0 and t_1. The observability test leads to the requirement that the position of the observer at t_1

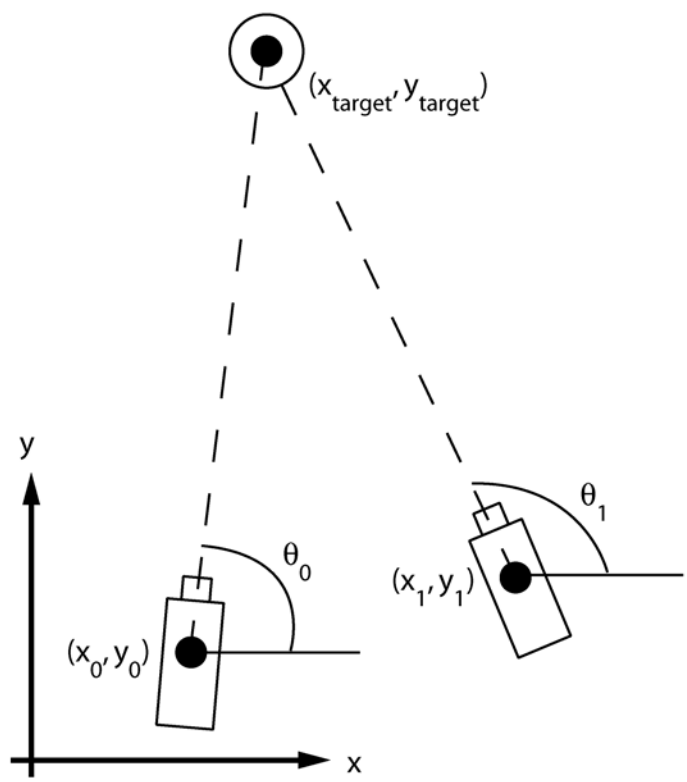

Figure 3.3 Two measurements taken of a stationary target

cannot lie on the line connecting the target and its position at t_0. Assuming no measurement noise, the pseudolinear approach (Equation **(3.3)**) offers the clearest derivation.

Since the target is stationary, the estimation state is the target position

$$\boldsymbol{x} = \begin{bmatrix} x_{target} \\ y_{target} \end{bmatrix} \tag{3.13}$$

and the state transition function is simply the identity matrix. The measurement equation for the target bearing is

$$\theta[k] = \operatorname{atan}\left(\frac{y_t - y[k]}{x_t - x[k]}\right). \tag{3.14}$$

Transformed into pseudolinear form the measurement equation becomes

$$y[k] - \tan(\theta[k]) \cdot x[k] = \begin{bmatrix} -\tan(\theta[k]) & 1 \end{bmatrix} \cdot \boldsymbol{x}[k] \tag{3.15}$$

so

$$\boldsymbol{H}_{pl} = \begin{bmatrix} -\tan(\theta[k]) & 1 \end{bmatrix}. \tag{3.16}$$

Substituting Equation **(3.16)** into Equation **(3.1)** yields

$$\begin{aligned} \boldsymbol{N}[k_f, 0] &= \sum_{k=0}^{k_f} \begin{bmatrix} -\tan(\theta[k]) \\ 1 \end{bmatrix} \cdot \begin{bmatrix} -\tan(\theta[k]) & 1 \end{bmatrix} \\ &= \sum_{k=0}^{k_f} \begin{bmatrix} \tan^2(\theta[k]) & -\tan(\theta[k]) \\ -\tan(\theta[k]) & 1 \end{bmatrix} \end{aligned} \tag{3.17}$$

so, changing notation slightly

$$\boldsymbol{N}[1, 0] = \begin{bmatrix} \tan^2(\theta_0) + \tan^2(\theta_1) & -(\tan(\theta_0) + \tan(\theta_1)) \\ -(\tan(\theta_0) + \tan(\theta_1)) & 2 \end{bmatrix}. \tag{3.18}$$

The stationary target position is observable if $\boldsymbol{N}[1, 0]$ is nonsingular. A matrix is nonsingular if its determinant is non-zero, so the observability requirement is $det(\boldsymbol{N}[1, 0]) \neq 0$. Substituting Equation **(3.18)** into this relation

$$det(\boldsymbol{N}[1, 0]) = 2 \cdot (\tan^2(\theta_0) + \tan^2(\theta_1)) - (\tan(\theta_0) + \tan(\theta_1))^2 \neq 0. \tag{3.19}$$

Expanding and then simplifying terms

$$\tan^2(\theta_0) - 2 \cdot \tan(\theta_0) \cdot \tan(\theta_1) + \tan^2(\theta_1) \neq 0 \tag{3.20}$$

$$((\tan(\theta_0) - \tan(\theta_1)))^2 \neq 0 \,. \tag{3.21}$$

Equation **(3.21)** is satisfied when

$$\tan(\theta_0) \neq \tan(\theta_1) \tag{3.22}$$

or

$$\theta_0 \neq \theta_1 \pm 180° \,. \tag{3.23}$$

Substituting Equation **(3.14)** into Equation **(3.22)** yields the observability requirement for the case of a stationary target

$$y_1 \neq \left(\frac{y_t - y_0}{x_t - x_0}\right) \cdot x_1 + y_t - \left(\frac{y_t - y_0}{x_t - x_0}\right) \cdot x_t \,. \tag{3.24}$$

A similar argument can be made for the case of a constant-velocity target. The details of the proof are developed by Hammel et al. [11] and will not be given here, but the resulting observer motion requirement for a continuous system is

$$\int_{t_0}^{t} (t - \tau) \cdot \boldsymbol{a}(\tau) \cdot d\tau \neq \alpha(t) \cdot [\boldsymbol{r}(t) + (t - t_0) \cdot \boldsymbol{v}(t_0)] \tag{3.25}$$

where $\boldsymbol{r}(t)$, $\boldsymbol{v}(t)$, and $\boldsymbol{a}(t)$ represent the relative position, velocity, and acceleration between the target and the observer and $\alpha(t)$ is an arbitrary scalar.

Equation **(3.25)** provides the mathematical requirement for the observability of constant-velocity-target tracking. This expression has a simple physical interpretation. With $\alpha \equiv 0$, the equation states that there must be some non-zero acceleration between the target and the observer. The simplest maneuver that achieves this acceleration is a single change in observer course. If $\alpha(t) \neq 0$, the motion constraint can still be violated, even if $\boldsymbol{a} \neq \vec{0}$. These maneuvers always result in observer position changes that lie along the instantaneous line-of-sight associated with a constant-velocity trajectory. Under these conditions the measurements are indistinguishable from those produced by an unaccelerated observer.

3.2 Estimator Performance

Trajectory quality is measured by the impact of the observer path on the performance of the target-motion estimation. Thus determination of trajectory quality is in fact calculation of estimator performance given the observer trajectory. Likewise, the identification of the best trajectory quality metric becomes the identification of the best estimator performance measure.

There are several different methods by which estimator performance can be expressed. This section presents three of these methods, compares them, and identifies the most appropriate method for trajectory design. The first two measures (the Fisher Information Matrix and mutual information) are based on information theoretic approaches. The Fisher Information Matrix corresponds to the sensitivity of the estimation process at a given value while the mutual information corresponds to incremental reductions in estimate uncertainty. The third measure, the estimate error covariance, directly describes the probability distribution function of the target estimate error.

3.2.1 Fisher Information Matrix

The Fisher Information Matrix (FIM) provides a summary of the amount of information in a data set relative to quantities of interest [33]. More specifically, the FIM measures the sensitivity of a conditional probability function with respect to the quantity of interest. The larger the sensitivity, the easier it is to distinguish the quantity of interest, and the better the performance of the measurement (or estimation) system. For the target motion estimation problem of interest in this work, the quantity of interest is the target state and the data set is equivalent to the monocular vision sensor measurements. The remainder of this section presents the derivation of the Fisher Information Matrix for target motion estimation using monocular vision.

Consider a vector $\boldsymbol{x}$ of states to be estimated and a measurement z related to $\boldsymbol{x}$ by the measurement equation

$$z = h(\boldsymbol{x}) + \boldsymbol{v} \tag{3.26}$$

where $\boldsymbol{v}$ is a random noise vector. The conditional probability density of z given $\boldsymbol{x}$ is used to define the log-likelihood function

$$L(\boldsymbol{x}|z) = -\log(p_{z|\boldsymbol{x}}(z|\boldsymbol{x})) . \tag{3.27}$$

The Fisher Information Matrix (FIM) is the Jacobian of this log-likelihood and is defined

$$\boldsymbol{J}_f = E\left[\left(\frac{\partial}{\partial \boldsymbol{x}} L(\boldsymbol{x}|z)\right) \cdot \left(\frac{\partial}{\partial \boldsymbol{x}} L(\boldsymbol{x}|z)\right)^T\right] = E\left[\frac{\partial^2}{\partial \boldsymbol{x}^2} L(\boldsymbol{x}|z)\right] \tag{3.28}$$

The FIM describes the relative rate at which the conditional probability function changes with respect to the estimate state [33, 47]. The greater the expectation of change at a given value, say $\hat{\boldsymbol{x}}$, the easier it is to distinguish $\hat{\boldsymbol{x}}$ from neighboring values and hence the more precisely $\boldsymbol{x}$ can be estimated at $\boldsymbol{x} = \hat{\boldsymbol{x}}$.

The general definition of the FIM is independent of the specific probability density function. In many cases the random variables are modelled with normal, gaussian distributions. For the case with linear state dynamics, a nonlinear measurement equation, and gaussian measurement noise $\boldsymbol{v}$ with zero mean and covariance $E[\boldsymbol{v}\boldsymbol{v}^T] = \boldsymbol{R}$, the Fisher Information Matrix becomes

$$\boldsymbol{J}_f[k_f] = \sum_{k=0}^{\kappa_f} \Phi^T[k, 0]\boldsymbol{H}_{lin}^T[k]\boldsymbol{R}[k]\boldsymbol{H}_{lin}[k]\Phi[k, 0] \tag{3.29}$$

with

$$\boldsymbol{H}_{lin}[k] = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}} h(\boldsymbol{x}) . \tag{3.30}$$

For the constant velocity target tracking problem with $\boldsymbol{R} = \sigma^2 \cdot \boldsymbol{I}$ the FIM becomes

$$\boldsymbol{J}_f[k_f] = \sum_{k=0}^{k_f} \left(\frac{\sigma^2}{r^4} \cdot \begin{bmatrix} \Delta y^2 & -\Delta x \Delta y & kT_s \Delta y^2 & -kT_s \Delta x \Delta y \\ -\Delta x \Delta y & \Delta x^2 & -kT_s \Delta x \Delta y & kT_s \Delta x^2 \\ kT_s \Delta y^2 & -kT_s \Delta x \Delta y & k^2 T_s^2 \Delta y^2 & -k^2 T_s^2 \Delta x \Delta y \\ -kT_s \Delta x \Delta y & kT_s \Delta x^2 & -k^2 T_s^2 \Delta x \Delta y & k^2 T_s^2 \Delta x^2 \end{bmatrix} \right) \tag{3.31}$$

where T_s is the time between samples and

$$\begin{aligned} \Delta x &= x_{target} - x_{observer} \\ \Delta y &= y_{target} - y_{observer} . \\ r^2 &= \Delta x^2 + \Delta y^2 \end{aligned} \tag{3.32}$$

The subscripts were dropped from the terms in Equation **(3.31)** for convenience, but it is important to note that these values are also functions of time.

Equation **(3.31)** assumes no prior information. If that is not the case, the FIM becomes

$$\boldsymbol{J}_t[k] = \boldsymbol{J}_f[k] + \boldsymbol{J}_p[0] \tag{3.33}$$

where $\boldsymbol{J}_p[0] = \boldsymbol{P}_0^{-1}$ represents the *a priori* information and is the inverse of the *a priori* estimate error covariance matrix.

The Fisher Information Matrix describes the performance of the system at a specific time sample ($k = k_f$). Another useful estimation measure is the Cumulative Fisher Information Matrix (CFIM)

$$\boldsymbol{J}_{cf}[k_f] = \sum_{k=0}^{\kappa_f} \left(\frac{J_f[k]}{k_f} \right). \tag{3.34}$$

The CFIM represents the average FIM (average performance) over the entire trajectory from time $k = 0$ to $k = k_f$. Because typical target motion estimation solutions carry large amounts of uncertainty before adequate observer motion and estimator performance is achieved, the Cumulative Fisher Information Matrix does not make a good performance metric for target-motion estimation using monocular vision.

3.2.2 Mutual Information

Similar to the Fisher Information Matrix, mutual information concepts focus on the conditional probabilities between a set of measurements and one or more random variables of interest. In particular, the mutual information between a measurement and a state variable (the quantity of interest) describes the amount of information gained about the state variable given the measurement. In other words, mutual information describes the incremental increase in the conditional probability distribution of the state variable. Like the FIM, the greater the mutual information between a variable and a measurement, the greater the increase in estimate certainty and the better the estimator performance. The discussion presented in this section follows that presented in [10, 31, 32] and develops the formal definitions of mutual information for nonlinear state estimation.

The *entropy* of a continuous random variable x with probability density function $p_x(x)$ is defined as

$$H(x) = -E[\ln(p_x(x))] = -\int_{-\infty}^{\infty} p_x(x) \cdot \ln(p_x(x))dx. \tag{3.35}$$

$H(x)$ is a measure of the average uncertainty of x.

The *conditional entropy* of x given measurements z, with the conditional probability function $p_{x|z}(x|z)$ is defined

$$H(x|z) = -E[\ln(p_{x|z}(x|z))] = -\int_{-\infty}^{\infty} p_{x|z}(x|z) \cdot \ln(p_{x|z}(x|z))dx. \tag{3.36}$$

and is a measure of the average uncertainty of x given z.

The *mutual information* between two random variables x and z with joint probability density function $p_{x,z}(x, z)$ is defined

$$\begin{aligned} I(x,z) &= -E\left[\ln\left(\frac{p_{x,z}(x,z)}{p_x(x)\cdot p_z(z)}\right)\right] \\ &= -\int_{-\infty}^{\infty} p_{x,z}(x,z)\cdot \ln\left(\frac{p_{x,z}(x,z)}{p_x(x)\cdot p_z(z)}\right)dx\,dz \end{aligned} \tag{3.37}$$

and

$$\begin{aligned} I(x,z) &= H(x) - H(x|z) \\ &= H(z) - H(z|x) \end{aligned} \tag{3.38}$$

The mutual information represents the average reduction in the uncertainty of x when z is observed.

Mutual information can be calculated for a discrete linear dynamic system. Consider the system

$$\begin{aligned} \boldsymbol{x}[k+1] &= \Phi[k+1,k]\cdot \boldsymbol{x}[k] + \Gamma\cdot \boldsymbol{w}[k] \\ z[k] &= \boldsymbol{H}[k,\boldsymbol{x}[k]]\cdot \boldsymbol{x}[k] + \boldsymbol{v}[k] \end{aligned} \tag{3.39}$$

where $\boldsymbol{w}[k]$ is zero mean, white, Gaussian process noise with covariance matrix $\boldsymbol{Q}$ and $\boldsymbol{v}[k]$ is zero mean, white, Gaussian sensor noise with covariance matrix $\boldsymbol{R}$. Furthermore, $\boldsymbol{x}[0]$ is gaussian with mean $\boldsymbol{x}_0$ and covariance $\boldsymbol{P}_0$.

The mutual information between the final state $\boldsymbol{x}[k_f]$ and the measurement sequence, denoted z_K, is given by

$$I(\boldsymbol{x}[k_f], z_K) = \frac{1}{2}\cdot \ln\left(\frac{\left|\tilde{\boldsymbol{P}}[k_f]\right|}{\left|\boldsymbol{P}[k_f|k_f]\right|}\right) \tag{3.40}$$

$$\begin{aligned} \tilde{\boldsymbol{P}}[k_f] &= E[(\boldsymbol{x}[k_f] - E\{\boldsymbol{x}[k_f]\}),(\boldsymbol{x}[k_f] - E\{\boldsymbol{x}[k_f]\})^T] \\ \boldsymbol{P}[k_f|k_f] &= E[(\boldsymbol{x}[k_f] - E\{\boldsymbol{x}[k_f]|z_K\}),(\boldsymbol{x}[k_f] - E\{\boldsymbol{x}[k_f]|z_K\})^T|z_K]\,. \end{aligned} \tag{3.41}$$

The numerator in Equation **(3.40)** is the determinant of the final estimate error covariance matrix given no measurements and the denominator is the determinant of the error covariance given the measurement sequence.

The mutual information between the state sequence $\boldsymbol{x}_K$ and the measurement sequence z_K is given by

$$I(\boldsymbol{x}_K, z_K) = \frac{1}{2}\cdot \sum_{k=0}^{\kappa_f} \ln\left(\frac{|\boldsymbol{S}[k]|}{|\boldsymbol{R}[k]|}\right) \tag{3.42}$$

$$\begin{aligned} \boldsymbol{S}[k] &= E[(z[k] - \boldsymbol{H}[k]\cdot \boldsymbol{x}[k|k-1]),(z[k] - \boldsymbol{H}[k]\cdot \boldsymbol{x}[k|k-1])^T] \\ \boldsymbol{x}[k|k-1] &= E[\boldsymbol{x}[k]|z_{k-1}] \end{aligned} \tag{3.43}$$

Equation **(3.42)** states that the mutual information between the state sequence and the measurement sequence is equivalent to the sum of the ratio of the determinant of the predicted measurement error covariance matrix over the determinant of the measurement error covariance matrix. If the measurement error covariance matrix is constant, maximizing the mutual information is equivalent to maximizing terms involving the predicted measurement covariance matrix. This fact may seem counterintuitive, however, if the predicted measurement error is large, the corresponding increase in information gained by the measurement is also large and thus the mutual information between the state and measurement sequences is also large. Furthermore, similar to the Cumulative Fisher Information Matrix, the mutual information function $I(\boldsymbol{x}_K, \boldsymbol{z}_K)$ represents an average over the entire trajectory and is not well suited to the problem addressed in this work.

The expressions for mutual information above were derived for linear systems. Unfortunately the target motion estimation problem has a nonlinear measurement equation. Logothetis et al. [10] suggest using a nominal state trajectory $\bar{\boldsymbol{x}}[k]$ to linearize the system dynamics

$$\begin{aligned} \bar{\boldsymbol{x}}[k+1] &= \Phi \cdot \bar{\boldsymbol{x}}[k] \\ \bar{\boldsymbol{x}}[0] &= \boldsymbol{x}[0] \end{aligned} \tag{3.44}$$

and to calculate a linearized measurement matrix

$$\boldsymbol{H}[k] = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}} h(\boldsymbol{x}) \bigg|_{\boldsymbol{x} = \bar{\boldsymbol{x}}} \tag{3.45}$$

that can be used in Equation **(3.40)** and Equation **(3.42)**.

3.2.3 Estimator Error Covariance Matrix

This section presents a description of the final estimator performance measure -- the estimate error covariance matrix. The error covariance matrix directly describes the probability distribution of the target-state estimate. It represents the expected mean-squared error of the state estimate and can be used to describe statistical confidence regions around the state estimate. Formal definitions and interpretations of this matrix are given below.

The *variance* of a random variable x is the mean squared deviation of the random variable from its mean $\bar{x}$. It is denoted by σ_x^2 where

$$\sigma_x^2 = E[(x - \bar{x}) \cdot (x - \bar{x})] \,. \tag{3.46}$$

The variance of a random variable describes the spread of the distribution around the mean value.

The *covariance* of two random variables is the expectation of the product of the deviations of the random variables from their means and is denoted σ_{xy} where

$$\sigma_{xy} = E[(x-\bar{x})\cdot(y-\bar{y})]\,. \tag{3.47}$$

The covariance is a measure of the linear dependence between two random variables. A covariance value of zero indicates no dependence.

Given a vector $\boldsymbol{x}$ of n random variables, the variances and covariances between the components can be represented by a covariance matrix

$$\boldsymbol{P} = E[(\boldsymbol{x}-\bar{\boldsymbol{x}})\cdot(\boldsymbol{x}-\bar{\boldsymbol{x}})^T] = \begin{bmatrix} \sigma_{x_1}^2 & \sigma_{x_1x_2} & \cdots & \cdots \\ \sigma_{x_2x_1} & \sigma_{x_2}^2 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \sigma_{x_{n-1}x_n} \\ \cdots & \cdots & \sigma_{x_nx_{n-1}} & \sigma_{x_n}^2 \end{bmatrix}\,. \tag{3.48}$$

If the random vector possesses a multivariate normal distribution, the vector mean and covariance matrix fully describe the vector's statistics.

Many estimator formulations, including the Kalman Filter and other nonlinear extensions, assume normal sensor and process noise distributions as well as normal error distributions. For an unbiased estimator, in which the mean of the estimate error is zero, the covariance matrix is used to describe the uncertainty of the state estimate and enables discussion of the estimate properties independently of the mean value of the state variables. The *estimate-error covariance matrix* is denoted by $\boldsymbol{P}$ where

$$\boldsymbol{P} = E[(\boldsymbol{x}-\hat{\boldsymbol{x}})\cdot(\boldsymbol{x}-\hat{\boldsymbol{x}})^T] = E[\tilde{\boldsymbol{x}}\cdot\tilde{\boldsymbol{x}}^T]\,, \tag{3.49}$$

$\hat{\boldsymbol{x}}$ denotes the state estimate, and $\tilde{\boldsymbol{x}}$ denotes the unbiased estimate error.

The estimate-error covariance matrix may be interpreted geometrically by considering the confidence ellipsoid defined by the following equation [46]

$$(\boldsymbol{x}-\hat{\boldsymbol{x}})^T\cdot\boldsymbol{P}^{-1}\cdot(\boldsymbol{x}-\hat{\boldsymbol{x}}) = n_\sigma\,. \tag{3.50}$$

This equation describes an ellipsoid centered around the state estimate $\hat{\boldsymbol{x}}$. The probability that the actual state vector lies within this ellipsoid is a function of the sole parameter n_σ. For $n_\sigma = 2$ the probability is approximately two-thirds. Conversely, for a fixed confidence level $n_\sigma = c$, the size of the ellipsoid described by Equation **(3.50)** indicates the uncertainty of the state estimate. Figure 3.4 shows the two-

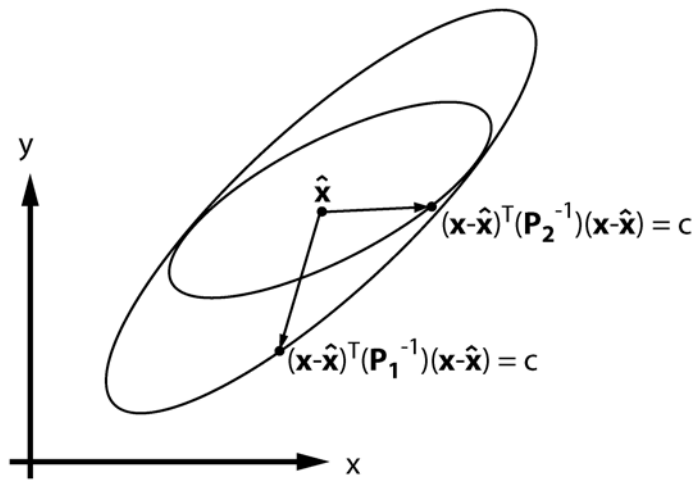

Figure 3.4 Geometric interpretation of the estimate error covariance matrix

dimensional confidence ellipses for two different estimate error covariance matrices. In this example, the figure demonstrates that $\boldsymbol{P}_2$ describes an estimate that is more accurate than $\boldsymbol{P}_1$.

3.2.4 Comparison

The estimate-error covariance matrix is the best estimator performance measure to use for trajectory design. The covariance matrix describes the expected output of the target-state estimator and by accounting for process noise and error propagation, provides the least-conservative performance bound, and thus least-conservative uncertainty region around the target state. This region is critical for designing observer paths that optimally reach a desired level of accuracy. A conservative uncertainty bound, i.e. a bound that overstates the estimate uncertainty, causes the observer to travel farther than needed in order to reach the desired accuracy level. This excess observer motion in turn leads to wasted resources and less optimal behavior.

There are two main disadvantages of the error covariance. The first is its computational complexity. Unlike with a linear system for which the behavior of the covariance is independent of the estimate state, the full nonlinear target-motion estimator must be propagated forward in order to calculate the error covariance. Second, calculation of the error covariance matrix is based on simplifications and linearizations in the modelling and estimation processes respectively. There is no guarantee that the covariance matrix returned by the target-state estimator reflects the actual statistics of the state probability distribution. However, the other measures (the FIM and the mutual information) would be subject to the same assumptions.

Through Equation **(3.40)**, the mutual information of the final target state relates to the ratio of the determinants of the error covariance matrix given the observer trajectory and its measurements to the error covariance matrix of the target given no observer motion or measurements. Since this second covariance is

independent of the observer path, optimizing the mutual information function is equivalent to optimizing the determinant of the estimate-error covariance. The estimate-error covariance matrix is therefore preferred to the mutual information because the covariance matrix requires slightly less computational cost and (as will be discussed in Section 4.3.3) because the determinant is not the only measure of the covariance matrix that can be used to describe it.

The Fisher Information Matrix and the Error Covariance Matrix are related via the Cramer-Rao bound (Equation **(3.9)**). The significance of the Cramer-Rao bound was already discussed in the previous sections. Estimators for which the inverse of the FIM is equivalent to the error covariance matrix are called *efficient*. Although the target-motion-estimation problem is efficient for some conditions, particularly for long target ranges and medium error noise levels [20], for the problem as stated in this dissertation it is not. Thus the lower bound described by the FIM will be a conservative prediction of the estimate-error covariance matrix achieved by the target state estimation process.

The apparent strengths of the Fisher Information Matrix are that it is computationally simpler than the estimate error covariance and that it is independent of the specific estimator used. The computational simplicity of the FIM relative to the error covariance is an important advantage, particularly since real-time performance is one goal of the trajectory design process. However, this strength is not counterbalanced by the disadvantages of the FIM. The second apparent strength of the FIM, its independence from the specific estimator used, is actually a weakness. Calculation of the estimate-error covariance is critical for nonlinear problems, such as target-motion estimation. The specific formulation of the state estimator has a large effect on the estimation performance for these nonlinear problems. For example, the Extended Kalman Filter uses linearized measurement equations that have the effect of feeding state errors back into the estimation process, often producing biased results [6]. Deriving a performance measure that is independent of the actual estimator used to solve the problem neglects many such effects.

The inefficiency of the FIM makes it less suitable than the error covariance for quantifying trajectory quality. Although the FIM is a measure of the ideal estimator performance, it does not predict the results of any particular estimator. Using the estimate-error covariance matrix includes more nonlinear effects of the specific estimator and produces a result that will be closer to the actual covariance than the Fisher Information Matrix. The closest covariance measure leads to a less conservative estimator performance measure and thus to better overall optimization results.

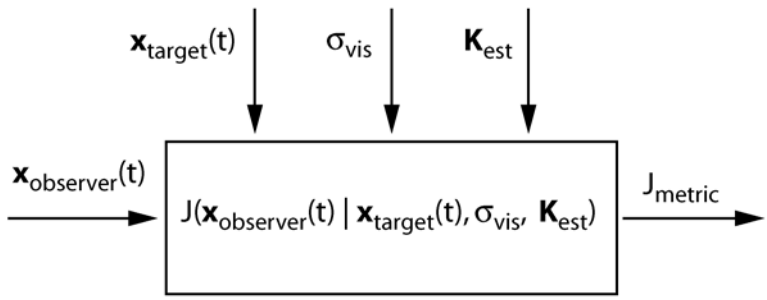

Figure 3.5 Trajectory quality function

3.3 Trajectory Quality Function

Trajectory quality describes the performance of the target-motion estimation that results from the observer following the given path. For this reason the development of trajectory analysis has focused on estimator observability tools and estimator performance measures. In order to create a quality assessment function that uses the estimate-error covariance matrix to evaluate and compare observer trajectories, several implicit assumptions must be made clear.

Estimator performance, and thus trajectory quality, is a function of many other factors in addition to the observer trajectory. These other factors include the vision-sensor characteristics (σ_{vis}), the relative path between the observer and target ($\mathbf{x}_{target}(t)$), and the specific estimator algorithm ($\mathbf{K}_{est}$). Therefore, in order to compare the quality of different trajectories, these additional factors must be held constant. Figure 3.5 depicts a graphical representation of computing the trajectory-quality function -- the observer trajectory serves as the sole input while the target path, sensor characteristics, and estimator formulation are kept constant.

In order to compare trajectories, it is convenient to calculate a single scalar value (J_{metric})to represent the quality of each path. However, the estimate-error covariance matrix which is proposed to describe the quality of the estimator performance is not a single value. Several functions can describe a matrix by a scalar value. The choice of function used to condense the estimate-error covariance matrix into a single value has a large effect on the computational cost of the trajectory design process and on the behavior of the resulting observer trajectory. Choices for this function and their effects on the design process are discussed in Section 4.3.3.

CHAPTER 4

Core Observer-Trajectory Design Algorithm

This chapter describes a new observer-trajectory-design algorithm that enhances the performance of target-motion estimation using monocular vision for a given observer-target geometry, based on the estimate-error covariance identified in Chapter 3 as the best trajectory-quality metric. Typical autonomous observer operation calls for the rapid generation of paths that either minimize estimate uncertainty for a given path duration or that minimize the time needed to achieve a desired uncertainty bound. The second desired objective, along with monocular vision limitations and observer kinematic constraints, are not considered in any previous trajectory generation algorithm. Furthermore, the algorithm presented here serves as the core of the full trajectory generator for the case of unknown initial target state presented in Chapter 5.

4.1 Introduction

Observer-trajectory design helps improve the performance of target-motion estimation using monocular vision by optimizing the amount of information presented to the target-motion estimator. Autonomous observer vehicles that perform this estimation operate under many competing objectives. In particular, resources such as fuel limit the amount of time an observer can devote to a task, while performance specifications such as accuracy bounds require sufficient movement. Typical autonomous observer operation dictates two types of trajectory-design objectives. For the first, resource limits define the amount of time that can be devoted to the estimation problem, so a path is desired that minimizes estimate

uncertainty in a specified amount of time. For the second, performance specifications dictate the accuracy level that must be achieved in minimum time in order to preserve vehicle resources.

All previous trajectory-design algorithms have satisfied only the former objective -- minimizing the estimator uncertainty while fixing the amount of time allowed to do so [10 - 20]. For these algorithms, the fixed-time constraint is explicitly accounted for in the formulation of the search space, and the estimate uncertainty is minimized by standard descent-based optimization techniques. The latter objective -- the desired performance level is specified and the time to achieve it is minimized -- is not suited to similar solution methods. The performance specification is now a complex function of the trajectory path and is not easily incorporated into the optimization as a constraint function. Likewise, this latter optimization cost (the trajectory duration) cannot be described by the standard state-space representation used to parameterize the trajectories. By extending previous trajectory design algorithms and developing a new optimization approach, both the original fixed-time, minimum-uncertainty trajectory and the fixed-uncertainty, minimum-time trajectory can be calculated.

Another key feature of trajectory design that has been overlooked in the past is the need for real-time performance [18]. The autonomous vehicles that benefit from trajectory design operate in the real world with moving objects. In order to react to the world in a timely manner, plans must be generated while the information they are based on is still reasonable. Previous trajectory-design algorithms have focused on the calculation of optimal solutions without regard for computation time [10 - 20]. The new trajectory-design algorithm presented here balances near-optimal performance with total planning time.

This chapter presents the development of a new observer-trajectory-design algorithm that uses an iterative, breadth-first search technique to generate a set of candidate observer trajectories. Each candidate trajectory is evaluated by the metric based on the estimate error covariance matrix developed in Chapter 3, and the best result is selected. Important elements of the design algorithm include the parameterization of the observer path, initialization of the target-motion estimate, system constraints, and the choice of cost function.

Among other things, optimal observer motion is a function of the observer-target geometry encountered, the noise characteristics of the monocular-vision sensor, and the specific target-motion-estimator formulation. The trajectory-design algorithm takes as input the location of the observer and the position and velocity of the target. When available, *a priori* information about the initial target state, in the form of an error covariance matrix, can also be treated as an input and used to calculate the final target-motion-estimation performance. Likewise, the trajectory-design algorithm presented here does not assume specific

formulations of the vision- sensor model or the target-motion estimator. They are also treated as inputs into the design process and must be specified in advance.

Because the design algorithm calculates the future trajectory for the observer, it must predict the behavior of the target into the future. In order to do so it uses the initial target state to generate a predicted target path. In this chapter the time at which the design algorithm is called is referred to as the *current* time. Because the target state includes position and velocity information, a single state also defines a single possible future target path. The target path that is defined by the current target state is called the *predicted target path*. The trajectory-design algorithm uses the predicted target path to simulate the estimator and calculate the predicted error covariance matrix, which is then used to calculate the measure of estimator accuracy.

One important benefit of the breadth-first search optimization is that it takes advantage of sensor and observer constraints to reduce the size of the search space. It also allows for the inclusion of other constraints that have not been used in previous work. Most importantly, the simulated error covariance matrix defines a safety region around the predicted target state into which the observer is prohibited from entering.

Although the new trajectory-design algorithm was developed to address a new problem that old methods could not, it is still capable of solving those simpler cases as well. The trajectory parameterization and the optimization method can be modified and additional constraints can easily be added or deleted in order to provide the specific formulation required to solve the related bearings-only tracking problems [10 - 20].

The remainder of this chapter presents the details of the new observer-trajectory-design algorithm for assumed known initial target state.

4.2 Predicted Error Covariance

Chapter 3 provided a discussion of metrics used to assess the observability of target-motion estimation using monocular vision and the effects of the observer trajectories on the performance of the estimator. The tools presented were used to analyze the effects of an observer trajectory for a given estimator and target path. Chapter 3 identified the estimate error covariance matrix as the most desirable metric to quantify the effects of an observer trajectory on estimator performance. Because the final accuracy level of the target-state estimate is important, only the error covariance matrix of the final estimate is used and not a weighted sum across the entire trajectory.

In order to use the quality metric to design trajectories, it is extended to make predictions of estimator performance. The estimate-error covariance matrix, identified in Chapter 3 as the metric of estimation performance and thus trajectory quality, describes the final performance of the estimation process -- after it has been carried out. However, candidate observer trajectories must be evaluated by a quality metric that assesses the future performance of the target-motion estimation. Therefore a prediction of the estimate-error covariance matrix must be determined. In order to generate this prediction, the initial target state is used to calculate the future target path and simulate the target-motion estimator.

The current target state includes position and velocity and therefore predicts a future path. Given this predicted target path, a candidate trajectory is evaluated by simulating the estimator as if the predicted target path were truth. At each instance along the candidate trajectory the predicted target path is used to generate a simulated measurement. The simulated measurement and candidate observer position are then fed into an estimator. Because there is no noise in this simulated system, the innovation is always zero and the simulated target estimate stays in agreement with the predicted path. However, the error covariance matrix, which is the representation of estimator uncertainty that is of prime importance, varies and the final covariance matrix is a measure of the final predicted estimate uncertainty.

It is important to stress that all constraints and functions evaluated along the candidate trajectories are applied to the predicted target state and the simulated (or predicted) estimate-error covariance matrix. Any use of the term estimate-error covariance matrix or target-state estimate refers to the simulated target estimate.

4.2.1 Initialization

In order to propagate the simulated estimator, initial estimator states must be chosen. The natural choice for initial states are the target-state inputs at the time of the design. Additionally, if an *a priori* covariance matrix exists for the initial target state, it is used to initialize the simulated estimator covariance matrix. As a result, the simulated estimator uses all of the information available to it at the current time. In this sense, the trajectory-design algorithm takes full advantage of all known information at the current time and the resultant trajectory represents the best possible choice of observer paths.

4.2.2 Computer Vision Simulation

The monocular-vision constraints explicitly enter the trajectory-design algorithm through the estimate covariance matrix prediction. The vision system imposes two main limitations that are included in the simulated estimation process: the camera has a limited field of view and has a maximum useful range (Figure 4.1). These vision constraints affect the propagation of the simulated estimator. If the predicted

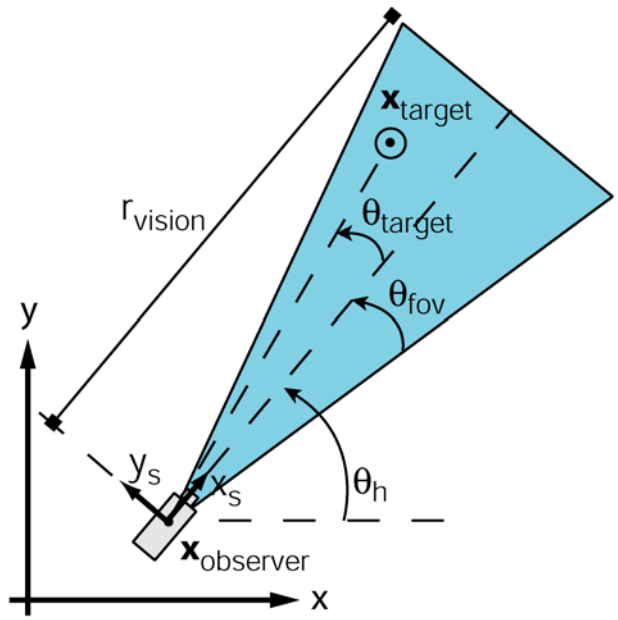

Figure 4.1 Vision sensor limitations

target and candidate observer positions do not meet the vision constraints at a given sample time, no simulated measurement is calculated and the measurement update step of the simulated filter is skipped.

In terms of the observer and target state vectors, the maximum range constraint is:

$$0 \le x_s \le r_{vision} \tag{4.1}$$

where

$$\boldsymbol{x}_s = \begin{bmatrix} x_s \\ y_s \end{bmatrix} = \boldsymbol{T}_{w2s} \cdot (\boldsymbol{x}_{target} - \boldsymbol{x}_{observer}) \tag{4.2}$$

$$\boldsymbol{T}_{w2s} = \begin{bmatrix} \cos(\theta_h) & \sin(\theta_h) \\ -\sin(\theta_h) & \cos(\theta_h) \end{bmatrix}. \tag{4.3}$$

Likewise, the field-of-view constraint can be written:

$$\theta_{target} = \operatorname{atan}(y_s / x_s) \tag{4.4}$$

$$-\theta_{fov} \le \theta_{target} \le \theta_{fov} \tag{4.5}$$

The field-of-view constraint is one of the main differences between the monocular-vision estimation problem and other bearing-only problems. The main ramification of this constraint is to create circumstances when the observer may lose the target while completing a leg of its trajectory. This behavior is one of the issues addressed by the full observer-trajectory generator described in Chapter 5.

4.3 Problem Statement

The following section presents the formal description of the observer-trajectory-design problem. The choice of trajectory parameterization, the specifications of the imposed constraints, and the optimization cost function are all defined. These three components shape the optimization method then used to solve the problem.

4.3.1 Path Parameterization

The representation of the observer trajectory is a key component of the trajectory-design algorithm and must meet several competing requirements. First, the trajectories must meet, or be capable of meeting, the kinematic and dynamic constraints of the observer vehicle. This is often accomplished by choosing a path representation that explicitly accounts for them. In other words, choosing actuator commands or control inputs as the search-state variables as opposed to velocities or positions. The second important criteria is to choose a state representation that has a sufficiently large feasible-trajectory set. This criterion ensures that all useful motions are considered in the optimization process and that the final choice can indeed achieve the stated performance objectives. Third, it is also important, in contrast, to pick a representation that is compact enough to allow for the optimization to occur in reasonable time. Finally, the goal of the trajectory-design algorithm is to enable paths of both fixed time and fixed accuracy. These two goals require different considerations, but it is desirable to have a trajectory representation that can be used for both objectives.

In order to satisfy the competing requirements listed above, the observer trajectory is represented by a finite number of constant velocity maneuvers of fixed duration. In order to satisfy the kinematics of the autonomous observer rover used in this work, each maneuver consists of a zero-radius turn followed by a straight line traverse. The duration of each maneuver is specified and is distributed between the two maneuver phases -- the more time needed to make the turn, the less time, and therefore less distance, allowed for the traverse. In Chapter 3 it was shown that at least one observer maneuver is required to obtain a unique estimate of a constant-velocity target. By parameterizing the trajectory as a series of maneuvers, the possibility of feasible motion for a unique solution is guaranteed [11].

The full-trajectory parameterization is illustrated in Figure 4.2. The observer speed V, the observer turning rate ω, and the duration of each maneuver T_{man} are kept constant in order to reduce the size of the parameter search space. A trajectory is then fully defined by the initial observer position X_0 and the heading of each maneuver. Since the initial position will be defined by the observer at the time it requires a new trajectory, the remainder of the path is defined by the headings. The state vector used for the trajectory-design optimization is then the list of maneuver headings, given the initial observer location, the observer speed and turning rate, and the maneuver duration:

$$\Theta = \left[\theta_1 \ \theta_2 \ \ldots \ \theta_n \middle| \ X_0 \ V \ \omega \ T_{man}\right]^T \tag{4.6}$$

Because Equation **(4.6)** fully represents the observer trajectory, the remainder of this chapter will use the term *trajectory* to refer to a vector of successive heading angles.

The equations describing the transformation from trajectory to observer states at the beginning of each maneuver are:

$$T_{turn} = (\theta_{k+1} - \theta_k)/\omega \tag{4.7}$$

$$T_{leg} = T_{man} - T_{turn} \tag{4.8}$$

$$\Delta L = V \cdot T_{leg} \tag{4.9}$$

$$\boldsymbol{x}_{observer}[k+1] = \begin{bmatrix} x_{obs}[k+1] \\ y_{obs}[k+1] \\ \dot{x}_{obs}[k+1] \\ \dot{y}_{obs}[k+1] \end{bmatrix} = \begin{bmatrix} x_{obs}[k] \\ y_{obs}[k] \\ \dot{x}_{obs}[k] \\ \dot{y}_{obs}[k] \end{bmatrix} + \begin{bmatrix} \Delta L \cdot \cos(\theta_{k+1}) \\ \Delta L \cdot \sin(\theta_{k+1}) \\ V \cdot \cos(\theta_{k+1}) \\ V \cdot \sin(\theta_{k+1}) \end{bmatrix} \tag{4.10}$$

where T_{turn} is the duration of the zero-radius turn, and T_{leg} is the duration of the straight-line traverse.

Depending on the trajectory-design objective, the maneuver duration is specified in one of two ways. For the case when minimum uncertainty is desired in fixed time, the total trajectory duration and number of maneuvers are specified. In this case the maneuver duration is just $T_{man} = T_{total}/n$ where n is the number of maneuvers. For the fixed-accuracy scenario, the maneuver duration is fixed and the number of

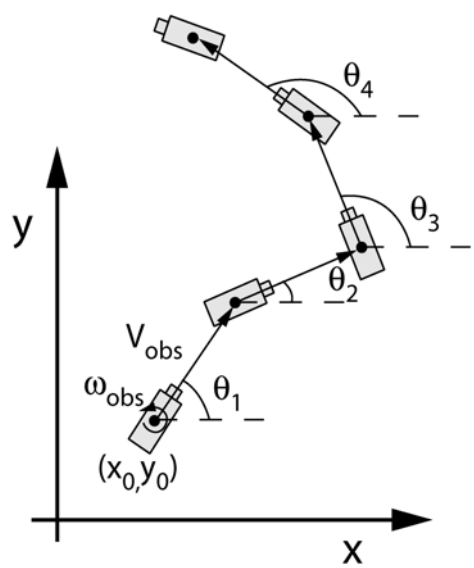

Figure 4.2 Trajectory parameterization

maneuvers becomes the cost. The choice of maneuver duration and number of maneuvers are important because they determine the smoothness of the resultant trajectory. Many short maneuvers lead to a smooth trajectory at the expense of a large parameter search space. In contrast, for the same total time fewer, longer maneuvers will reduce the size of the parameter space in exchange for more jagged paths.

Because the observer speed is held constant, there is a relationship between maneuver length and time. If infinite observer turning rate is assumed, the minimum-time path is also the minimum-length path, and vice versa. If the length of each maneuver, ΔL, is chosen directly, the resulting paths would still be minimum length and the speed could vary over the course of the trajectory. This formulation could be useful to an observer that has path-length-dependent resources, like fuel, that must be conserved. The main drawback is that the total trajectory time may not be known in advance and therefore motion after the completion of the trajectory could not be calculated until that point is reached.

There are several advantages and disadvantages to the trajectory parameterization presented above. The main advantage of this formulation is that the path duration can either be specified beforehand or used as the cost function of the optimization without changing the underlying search-space representation. Furthermore, sufficient observer motion is guaranteed to exist in the admissible set of observer trajectories. The main disadvantages are that the state vector is not always of fixed length and that smoothness comes at the expense of an increased search space.

Other trajectory parameterizations are possible and can also be used to solve this problem. For example, Huster et al. parameterize the trajectory as the superposition of a set of motion primitives that meet additional task-specific constraints [66]. With this parameterization, the trajectory-design concepts presented in this work are used to improve the manipulation of a fixed object by a robot arm that fuses computer vision with inertial sensors.

4.3.2 Observer Motion Constraints

Characteristics of the observer's environment and knowledge about that environment impose constraints on the observer's motion. These types of constraints apply to possible observer positions at each instant of time and are functions of several other variables at that time. The following section describes several constraints that limit the locations to which the observer can move. Additional constraints could also be defined to include obstacles or other vehicles in the environment.

Error Ellipse

As described in Section 3.2.3, the estimate-error covariance matrix defines an uncertainty ellipse around the target-state estimate. The estimate, along with the uncertainty ellipse, represent a region in which the target-

state estimate is located with some statistical probability. Since the target could occupy any position in this space with some probability, it is unwise to allow the observer to move into this ellipse.

The geometry of this error ellipse is shown in Figure 4.3. Given the two-dimensional position components of the target-state estimate,

$$\hat{\boldsymbol{x}}_{2D} = \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} \tag{4.11}$$

and the error covariance matrix of these two components,

$$\hat{\boldsymbol{P}}_{2D} = \begin{bmatrix} P_{xx} & P_{xy} \\ P_{yx} & P_{yy} \end{bmatrix} \tag{4.12}$$

where

$$\begin{gathered} P_{xx} = E[(x-\hat{x})\cdot(x-\hat{x})^T] \qquad P_{yy} = E[(y-\hat{y})\cdot(y-\hat{y})^T] \\ P_{yy} = P_{yx} = E[(x-\hat{x})\cdot(y-\hat{y})^T] \end{gathered} \tag{4.13}$$

the following equation represents ellipses of constant probability density,

$$(\boldsymbol{x}-\hat{\boldsymbol{x}}_{2D})^T \cdot \hat{\boldsymbol{P}}_{2D} \cdot (\boldsymbol{x}-\hat{\boldsymbol{x}}_{2D}) = c\,. \tag{4.14}$$

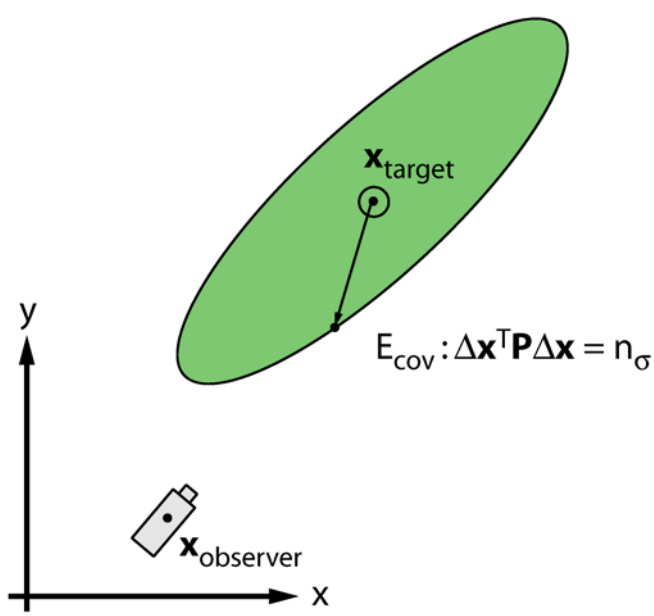

Figure 4.3 Uncertainty ellipse defined by the estimated error covariance matrix P

The $1-\sigma$ ellipse, ($c = 1$), has a significant probability of circumscribing the target position and designates the uncertainty of the target-state estimate. The constraint on the observer position is then

$$(\boldsymbol{x}_{observer2D} - \tilde{\boldsymbol{x}}_{2D})^T \cdot \hat{\boldsymbol{P}}_{2D} \cdot (\boldsymbol{x}_{observer2D} - \tilde{\boldsymbol{x}}_{2D}) \geq 1 \,. \tag{4.15}$$

Minimum Approach Distance

In order to maintain a safe distance between the target and the observer, a minimum approach distance is imposed. The equation for this constraint is:

$$0 \leq \left\| (\boldsymbol{x}_{observer2D} - \hat{\boldsymbol{x}}_{2D}) \right\| \leq r_{approach} \,. \tag{4.16}$$

4.3.3 Cost Functions

A scalar function describing the predicted error covariance matrix and the corresponding uncertainty ellipsoid is needed in order to define an optimization cost for the trajectory-design algorithm. Section 4.2 described the predicted error covariance matrix used to predict estimation performance. The covariance matrix defines an uncertainty ellipsoid around the target estimate. There are several different measures of matrix size that can be used to describe the covariance matrix and all have slightly different, but related, meaning. Commonly used functions are the matrix determinant and the maximum singular value of the matrix.

Although the maximum singular value is the ideal matrix measure, the determinant is used for the experimental results presented in Chapter 7. The determinant is significantly faster to compute than the

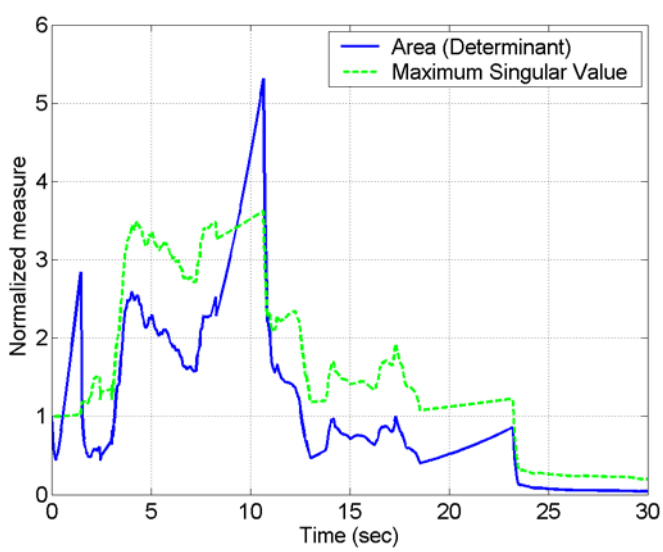

Figure 4.4 Results of the two different error covariance matrix measures for the experimental run described in Section 7.4.1. Each curve is normalized by its initial value.

maximum singular value and both measures (the determinant and the maximum singular value) show the same general trend as the observer moves along its trajectory (Figure 4.4)

Maximum Singular Value

The maximum singular value (σ_{max}) of the estimate error covariance matrix is the preferred estimate error uncertainty measure because it describes the largest error in any dimension

$$J_{msv} = \sigma_{max}(\hat{\boldsymbol{P}}) . \tag{4.17}$$

The maximum singular value denotes the length of the major axis of the error ellipse. Typically the performance specifications of estimation or intervention systems are described in similar terms (i.e. maximum errors), making σ_{max} the most desirable and intuitive measure from a physical standpoint. Unfortunately, the major disadvantage of the maximum singular value is its large computational cost compared to other measures such as the matrix determinant or trace. This computation constitutes a large percentage of the overall computation, and hence planning time, of the trajectory design algorithm.

Uncertainty Ellipse Area (Determinant)

The determinant of the estimate error covariance matrix is directly related to the volume of the uncertainty ellipsoid described by the matrix. Minimizing the determinant is therefore equivalent to minimizing the volume of the uncertainty ellipsoid. For the two-dimensional position covariance matrix (defined by Equation **(4.12)** and Equation **(4.13)**) the relationship between the determinant and the area of the 1-sigma ellipse is

$$J_{area} = A_{ellipse} = \pi\sqrt{det(\hat{\boldsymbol{P}}_{2D})} . \tag{4.18}$$

Use of the Ellipse Area in place of the Maximum Singular Value

Although the maximum singular value of the estimate error covariance matrix is the preferred uncertainty measure, the determinant can be used in its place with little change in optimization performance and large gains in computational cost (planning speeds). For target-motion estimation using monocular vision, the maximum singular value of the estimate-error covariance matrix corresponds closely to errors in range to the target object. Error in the estimate of the target bearing (corresponding to the minor axis of the error ellipse) is mainly a function of the vision-sensor noise and remains approximately constant throughout the estimation. Assuming the minor axis of the error ellipse is a constant, the area of the ellipse (related to the determinant of the covariance matrix) is directly related to the length of the major axis of the ellipse (described by the maximum singular value). Therefore the area of the uncertainty ellipsoid and maximum singular value only differ by a (approximately) constant scale factor (Figure 4.4). Since the determinant is

easier to compute than the maximum singular value, the uncertainty ellipse area is used in the trajectory-design algorithm as the uncertainty measure.

4.3.4 Trajectory-Design Problem

Two typical observer objectives are solved by the trajectory-design algorithm. Formal definitions of each follow in the sections below. The first, referred to as fixed-time, minimum-uncertainty (FTMU), refers to the scenario in which the observer has a set amount of time with which to achieve the best possible estimate accuracy. The second, referred to as fixed-uncertainty, minimum-time (FUMT), describes the complementary scenario in which performance specifications must be achieved as quickly as possible. Previous trajectory design work has focused exclusively on the FTMU formulation of the trajectory-design problem [10 - 20]. This dissertation presents the first observer-trajectory generator to address the FUMT scenario.

Fixed-time, minimum-uncertainty

For the fixed-time, minimum-uncertainty (FTMU) case, the duration of the trajectory is fixed, and a trajectory is generated that minimizes the cost function of the predicted error covariance. In the context of the parameterization defined in Section 4.3.1, the total trajectory duration is fixed by specifying the duration of each maneuver and the total number of maneuvers. The state vector remains a constant length and the goal of the optimization is to select the trajectory that minimizes the trajectory quality metric:

$$\min_{\Theta} J_{ftmu}(\Theta | T_{man}, n) \tag{4.19}$$

where the cost function is the area of the uncertainty ellipse, defined by Equation **(4.18)**

$$J_{ftmu} = J_{area}. \tag{4.20}$$

and the observer trajectory Θ is restricted by the constraints of Equation **(4.15)** - Equation **(4.16)**.

Fixed-uncertainty, minimum-time

The fixed-uncertainty, minimum-time (FUMT) optimization is now possible with the new trajectory design algorithm presented in this dissertation. In this case the desired uncertainty measure is specified and the time taken to achieve it is minimized. The maneuver duration is still specified, but in contrast to the fixed-time case, the size of the state vector (the number of maneuvers) is allowed to vary. This is described in more detail in Section 4.4.1. With the fixed maneuver duration, minimizing the total time is equivalent to minimizing the number of maneuvers (n) needed, so the goal of the optimization is

$$\min_{\Theta}(n) \tag{4.21}$$

subject to

$$J_{fumt}(\Theta | T_{man}, n) = J_{desired} \tag{4.22}$$

where the metric function is the area of the uncertainty ellipse, defined by Equation **(4.18)**

$$J_{fumt} = J_{area}. \tag{4.23}$$

and the observer trajectory Θ is restricted by the constraints of Equation **(4.15)** - Equation **(4.16)**.

4.4 Optimization Method

The trajectory-design problem is solved by performing a pyramid, breadth-first search. Although the ideal result of the optimization is the global minimum, this optimum cannot be found in a reasonable amount of time. The search space has local minima unsuitable for gradient-based methods and is too large for an exhaustive search. Instead, a sub-optimal result is obtained in real-time using an iterative pyramid scheme.

The goal of the optimization is to achieve minimum uncertainty for a given duration or to obtain a fixed uncertainty level in minimum time. In each case a breadth-first enumeration algorithm is used to generate the list of candidate trajectories that satisfy all the given constraints and to calculate the predicted error covariance matrices that quantify those trajectories. The best candidate trajectory is identified and the process is repeated using the previous best trajectory as an initial guess.

4.4.1 Breadth-First Search Tree

Candidate observer trajectories are generated by creating a breadth-first search tree. The initial observer position serves as the root of the tree, and nodes represent the position of the observer after each maneuver. The nodes in the tree are expanded by choosing a heading value and propagating the parameterized observer dynamics (Equation **(4.7)** - Equation **(4.10)**) forward in that direction. In order to maintain a manageable search space, the range of possible heading values is discretized. The tree is expanded breadth-first, meaning every possible heading value for a given node is chosen and expanded before a new node is selected. In this manner, the candidate trajectories are grown uniformly outward from the initial observer position.

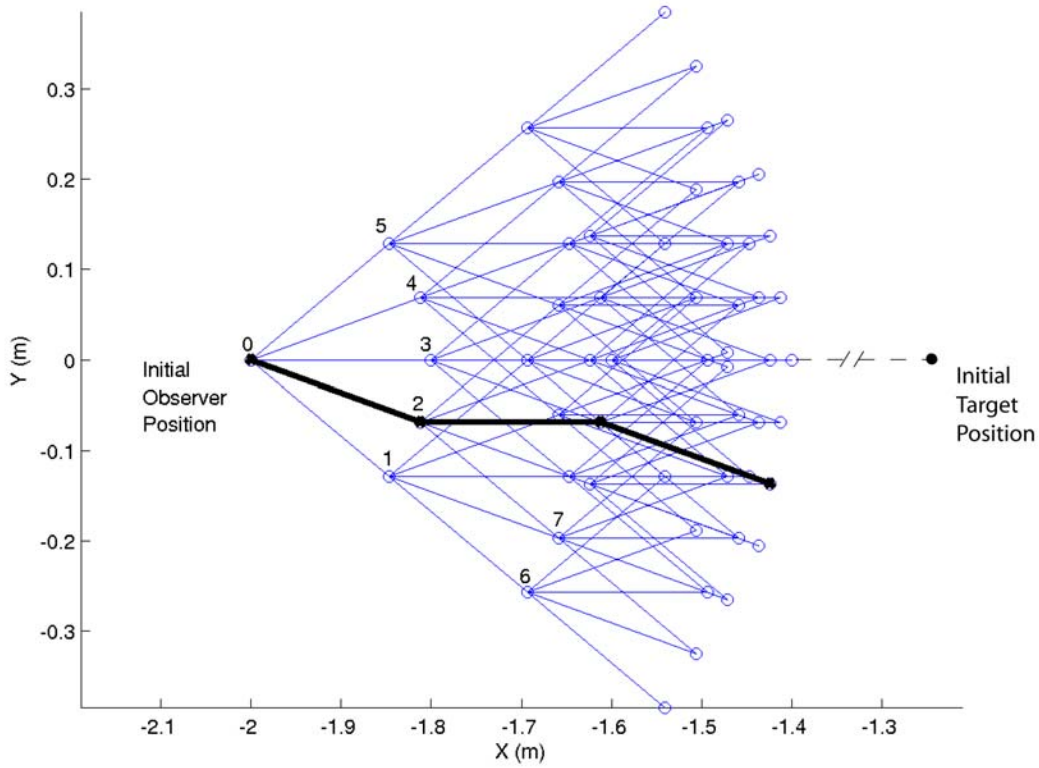

Figure 4.5 Breadth first expansion from initial observer position using five possible heading values

The breadth-first expansion is shown in Figure 4.5. In this example the range of possible heading values is restricted to

$$\boldsymbol{S}^{i}_{heading} = \{-40°, -20°, 0°, 20°, 40°\} \tag{4.24}$$

where the superscript i indicates the set applies to the ith maneuver.

The expansion is breadth first, so the points labelled 1-5 are all generated from the initial position. Once they are generated, point 1 is selected and points 6 and 7 are generated. This process is continued for all directions from point 1, then from point 2, then 3, etc. The lines that originated at the initial observer position are said to be at level or depth 1 and represent the possible positions of the observer after one maneuver. The lines originating from the endpoints of the level 1 maneuvers are referred to as level 2 and the convention continues for all levels. All maneuvers at a given level occur the same time away from the original node. Because the target motion is predicted based only on the current target estimate, the predicted target location that corresponds with the observer position at a given node is the same for every node at a given level. In other words, the target is predicted to move to the same location regardless of whether the

observer is at nodes 1-5. By fully expanding to a certain level, all trajectories of a given duration are generated.

Each node in the tree represents a point in space. The path from the initial node to any node in the search tree defines a trajectory in space. Furthermore, the path from the root node to any given node is unique, and the corresponding real-world trajectory is also unique. Thus, every node in the search tree also represents a unique candidate trajectory. Looking at point 7 in Figure 4.5 it is clear that more than one node can reside at a given position in space. However, the paths in the tree, and thus the trajectories in space, are different.

The constraints described in Section 4.3.2 apply to the position of the observer and the target at each instant along the trajectory. Because the search nodes are the trajectory positions, the constraints are imposed at every node. Nodes that do not satisfy the constraints are pruned from the tree. The elimination of nodes, especially ones at the earliest levels, reduces the size of the search space and the subsequent length of the optimization process.

The predicted error covariance is calculated as the search tree is expanded by simulating the estimator during each maneuver. Using the simulated target state and covariance matrix for a given node, the state and covariance are calculated for the next node. This is accomplished by using the predicted target position that corresponds to the level of the new node to simulate the estimator. The resulting covariance matrix is then stored as the cost of moving to that node. By calculating the predicted error covariance matrix as the tree is expanded, the error ellipse constraint is readily applied and used to reduce the size of the search space.

By expanding the search tree to a given depth, all trajectories up to a specified duration are generated. For the case where the time is fixed and the uncertainty level is to be minimized, the candidate trajectories are all of the nodes from the highest (deepest) level. The cost of each node is monitored as the tree is expanded and the minimum cost node is always updated. Once the tree has been fully expanded, the best trajectory is immediately determined.

The FUMT problem is also easily solved using the breadth-first expansion. In this case the tree is continually expanded until one of two conditions occurs. The first condition is that the predicted estimator performance achieves the desired value. Because the expansion is breadth first, the first trajectory to achieve the goal performance is guaranteed to be in minimum time. When a given node is being expanded, every trajectory of a lower level, and therefore shorter duration, has already been calculated. Furthermore, there is no need to generate more candidate trajectories because there is no way they can improve the time. The second condition is that a specified search depth is achieved without meeting the performance requirement. This condition is needed to prevent the tree from expanding forever.

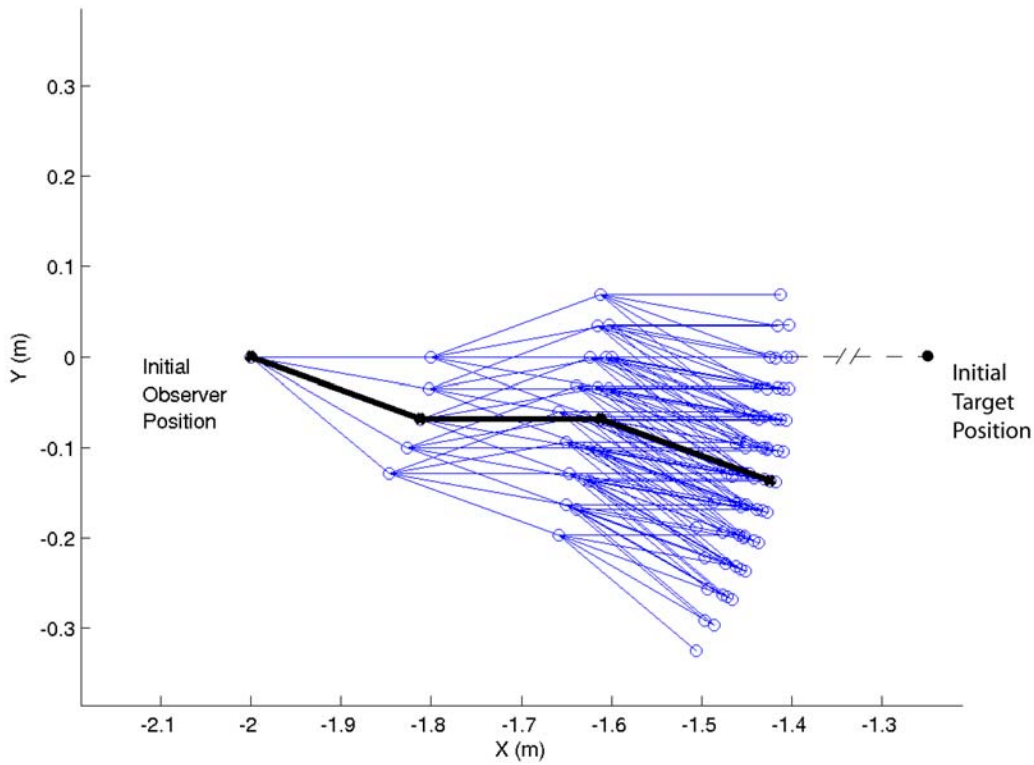

Figure 4.6 Search tree generated from refined heading space

The initial breadth-first tree is always centered on the line connecting the observer to the initial target location. For the example given in Figure 4.5 the target lies somewhere on the line $y = 0$. If the target were located at a different location with a relative bearing of β_0 then the discretized heading values $\boldsymbol{S}^i_{heading}$ would be shifted by that amount. Because the candidate trajectory tree is based on this line of sight, the center trajectory is also viewed as an initial seed which the design algorithm improves.

4.4.2 Pyramid Search

A pyramid iteration is used to refine the resolution of the heading discretization. In order to span a significant range of headings and maintain a small search space, the initial allowable heading set is small and coarse (the example of Equation **(4.24)** used five angles spaced twenty degrees apart). While this set yields a solution, it terminates with a large heading resolution and ignores a large set of possible paths. One obvious solution appears to be increasing the number and decreasing the size of the heading intervals. However, the complexity of the optimization is $O(n_{ang}^d)$, where n_{ang} is the number of angles in the allowable heading set and d is the depth of the search tree, so increasing n_{ang} quickly leads to slow planning times. It is therefore advantageous to develop an alternative method to improve the final heading resolution.

An iterative pyramid scheme is used to rediscretize the heading space and refine its resolution after successive iterations. This is accomplished by first generating the candidate trajectory tree and finding the best trajectory. At each node in the resulting trajectory, the heading space is rediscretized about the heading value at that point. The rediscretization uses the same number of intervals as the previous iteration to span one of those intervals on either side. For example, given the thick line in Figure 4.5 as the best path, the heading space for the first maneuver is rediscretized from Equation **(4.24)** to

$$\boldsymbol{S}^{1}_{heading} = \{-40°, -30°, -20°, -10°, 0°\} \tag{4.25}$$

The same process is carried out for each heading value in the trajectory. Using this new heading space, a new search tree is created. Figure 4.6 shows the new search space based on the optimal trajectory from Figure 4.5. The best path from this new tree is determined and the iteration process is repeated until the desired level of heading resolution is achieved.

The overall speed of the optimization is determined by the number of iterations required and the number of discrete angles allowed each time. The pyramid iteration is able to achieve small heading resolution without exponential growth in the time to complete it. In fact, the pyramid approach is of order $O(n_{iter} \cdot n^{d}_{ang})$ where n_{iter} is the number of iterations. The duration of the optimization increases only linearly with the number of iterations.

Because the best trajectory from the previous iteration is preserved in the new search tree, each successive refinement of the heading space is guaranteed to maintain or improve the overall performance of the optimization. The new search tree creates a new set of candidate trajectories and the best trajectory among them is selected. Because the previous best trajectory is included in the new set, the best trajectory from the new set must be as good as or better than the result of the previous iteration.

4.5 Parameter Choices

The trajectory-design algorithm requires the specification of several design parameters that influence the performance of the planning process as well as the behavior of the final observer trajectory. This section presents these parameters and discusses their effect on the outcome of the trajectory-design algorithm.

4.5.1 Non-dimensional Trajectory Length

The observer trajectory can be described by the non-dimensional parameter K that relates the total trajectory length possible to the initial target geometry [16]:

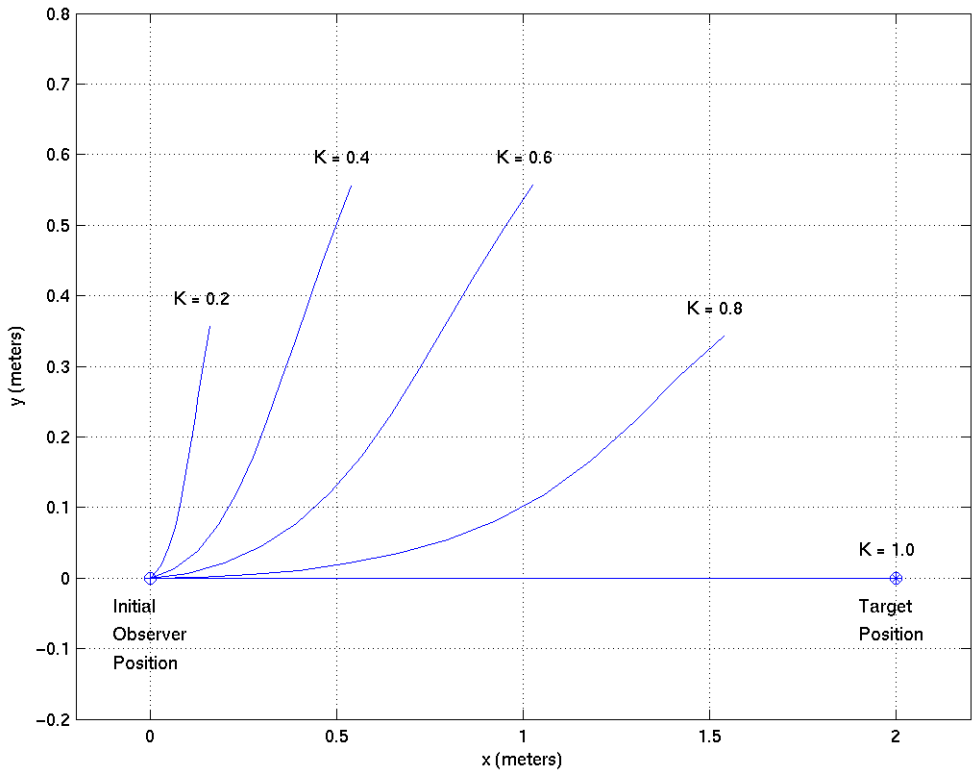

Figure 4.7 BOT solutions for different values of the non-dimensional trajectory-length K

$$K = \frac{V \cdot n \cdot T_{man}}{R_0} \tag{4.26}$$

where V is the observer speed, n is the number of observer maneuvers, T_{man} is the maneuver duration, and R_0 is the initial target range. A value of $K = 1$ would signify that given no constraints on its motion, the observer could exactly close the distance to the initial target location. The numerator in Equation **(4.26)** corresponds to the total possible length, not the total achieved length, because in most cases part of the maneuver duration is utilized making the zero-radius turn before the observer is able to traverse forward. If the observer turning rate were infinite, the numerator would correspond to the total trajectory length.

The performance of the trajectory-design algorithm depends on the non-dimensional trajectory length. For the bearings-only localization problem (in which the target is stationary, the sensor has an unlimited field of view, and the initial target error covariance is infinite) the optimal trajectory is a function of only this parameter [16]. The larger the value of K, the larger the relative bearing and bearing rates the observer is able to achieve. Figure 4.7 shows observer trajectories for the bearings-only localization problem as a function of the non-dimensional length parameter. For small values of K the trajectories are predominantly transverse to the line of sight to the target. This creates the bearing change that enables triangulation. As K

increases, the trajectories are able to move closer towards the target before moving transversely. They achieve a larger final bearing rate and a larger total bearing change. For the bearings-only localization problem the maximum value of the length parameter is $K = 1$. At this value the optimal solution is to move to the target location and additional observer motion is unnecessary because the target-state estimate becomes exact.

For the more general case in which the target moves, and the vision sensor has field-of-view limitations, and other constraints are imposed, and the target estimate has finite initial uncertainty, the observer trajectory depends on the non-dimensional length as well as additional parameters. However, the general behavior of the observer remains -- small values of K lead to more-transverse paths while larger values lead to more motion toward the target. In addition, values of $K > 1$ are possible, since the target moves from its initial location and constraints prohibit the observer from approaching too closely.

Empirical results show that choosing K from the range $0.6 \leq K \leq 0.8$ provides good performance. This range allows for motions long enough to provide the required observability while maintaining a reasonable total duration. In practice, the observer speed and initial target range are predefined, and the non-dimensional length is used to select the total trajectory length $T_{total} = n \cdot T_{man}$. The number of maneuvers and maneuver duration are then balanced to achieve the best performance.

4.5.2 Number of maneuvers

Together with the maneuver duration, the number of observer maneuvers determines the total duration, and total length, of the observer trajectory. The total duration is typically defined using the non-dimensional trajectory length described in the previous section or from other mission requirements. In either case, once the total duration is specified, a trade-off between the number of maneuvers and their duration must occur.

Increasing the maneuver number has two main benefits. First, every additional maneuver provides an opportunity for acquiring a good measurement with more information content. As discussed in Section 3.1.3, a maneuver is always needed in order to provide observability for the target-motion estimation problem. Multiple maneuvers increase the observability of the problem and the expected overall estimation performance. Second, the more maneuvers the observer makes in a fixed amount of time, the smoother its path may become. Smoother paths are often easier to follow, and they allow the observer to operate in more-complex environments.

In contrast, the main drawback of increasing the number of maneuvers is that every additional maneuver increases the dimension of the optimization space. As discussed in Section 4.4.1, the size of the candidate trajectory tree, and the corresponding time it takes to perform a search, increases exponentially with the

number of maneuvers. Therefore the increased information content is balanced by the increased time it takes to calculate the trajectory. This time is important because the system operates in real-time and must react to events as they happen.

In order to provide the best balance between increased information and optimization cost, 5 maneuvers are used to design the observer trajectories presented in this dissertation. With this number it takes approximately 1.5 seconds to generate an observer trajectory using a 750 MHz Intel Pentium III-based PC running RedHat Linux 7.2 (Chapter 6). This is quick enough to allow the observer to respond to changes in its knowledge of the world. Full numerical calculation of the trade-off between estimation performance and planning time is presented in Section 7.2.3.

4.5.3 Maneuver duration

For the fixed-uncertainty, minimum-time design problem the maneuver duration, not the number of maneuvers, is the key parameter used to determine the observer's behavior. Unlike the fixed-time, minimum-uncertainty case, the number of maneuvers, and thus total time, is the optimization cost, so only the maneuver duration can be specified ahead of time. As with the number of maneuvers, the maneuver duration is chosen with a trade-off of positive and negative effects in mind.

As the maneuver duration increases and the observer traverse grows longer, the achievable bearing change and bearing rates also increase. In turn, the information presented to the estimator grows and a more accurate estimate results. Thus it is advantageous to have long maneuvers that provide significant parallax. However, the downside is that the longer maneuvers do not allow for trajectories that are as detailed, or smooth, as ones comprised of small maneuvers. Furthermore, because the observer completes the entire traverse of every maneuver, the desired estimate uncertainty may be reached mid-traverse and extra time is wasted completing that maneuver.

One important distinction of the maneuver duration compared to the number of maneuvers is that it does not directly affect the size of the candidate trajectory space or the time taken to perform a search. For the FTMU case, the search depth is defined by the number of maneuvers. In practice, the number of maneuvers is limited by the computation speed of the trajectory design and is specified first. The maneuver duration is then calculated from the total trajectory duration. For the FUMT scenario, the maneuver duration will indirectly effect the final search depth and total computational cost. There is a lower limit to the amount of motion needed to achieve a given level of accuracy. If the maneuver duration is low, more maneuvers will be required and the search space will be large. However, due to the discrete manner with which the observer performs each maneuver, changing the maneuver duration may not change the number of maneuvers

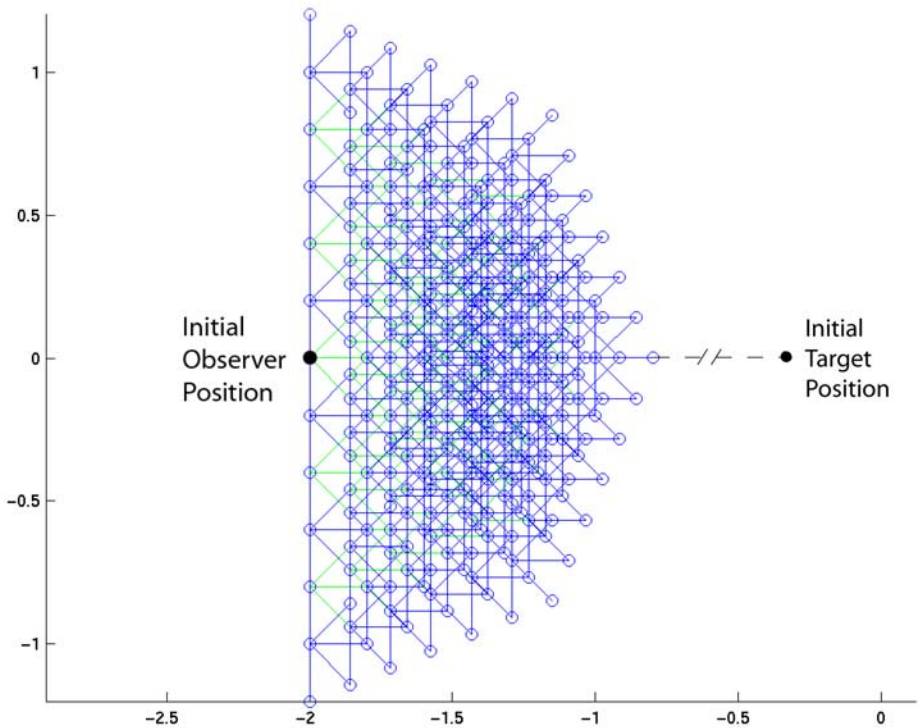

Figure 4.8 Reachable world region for a 180 degree heading space

needed, but will instead change the conservatism of the resulting estimate performance, i.e. the final uncertainty will approach the desired uncertainty from below.

For the results presented in this dissertation (Chapter 7), a maneuver duration of 6.0 seconds is used for both the fixed-uncertainty, minimum-time and fixed-time, minimum-uncertainty experiments. This duration corresponds to a value of non-dimensional trajectory length K equal 0.75 and total maneuver time T_{total} equal 30.0 seconds for a FTMU design with 5 maneuvers. This duration allows for paths complex enough to apply all of the observer position constraints and still achieve desired levels of estimate uncertainty.

4.5.4 Heading discretization

The range of the heading space used to generate the candidate trajectory tree determines the area of the world that the observer is able to reach. Because the candidate trajectories are generated by specifying the headings of straight-line maneuvers, and because each new maneuver can occur over a range of fixed heading values, the reachable region of the world is determined by a series of cones that expand from the original observer position. Figure 4.5 shows one example of the set of candidate trajectories for the observer motion. The cone that bounds these trajectories represents the total region in the environment that the

observer can reach. Because the region grows out from the original observer position, there are many points near the target where the observer can go and there are few allowable points close to the original point. Increasing the size of the heading range will enable a larger total region, and more importantly a larger initial region, for the observer to traverse.

The observer heading range is discretized in order to create a finite search space. The size of the search space depends on the number of intervals into which the heading space is divided. More intervals allow for more candidate trajectories, but increase the size of the search space and the time it takes to generate a solution. As discussed in Section 4.4.2, a pyramid iteration scheme is used to allow for detailed trajectory resolution at reasonable computational speeds. However, the size of the heading space must still be specified.

Because of the many constraints applied to the observer trajectories, it is important to allow for as much motion as possible. This is accomplished by using an initial heading range of $180°$. The observer can move towards the object or transverse to it with equal ease. In order to provide a reasonable search space while still spanning the large heading space, the heading range is discretized into five equal intervals. Figure 4.8 shows the resulting set of reachable trajectories. The number of pyramid iterations then determines the final resolution of the heading space. Using four iterations a resolution of 2.9 degrees is achieved.

4.6 Scalability

Although the trajectory-design algorithm presented here was developed for an autonomous observer rover estimating a constant-velocity target moving in a two-dimensional plane, the technique extends to different observer types, higher degrees of freedom, and more-complex target motions. The observer parameterization developed in Section 4.3.1 can easily be replaced by one that describes different dynamics. For example, Huster et al. used an algorithm similar to the one developed here to plan trajectories for an underwater robotic manipulator [66]. Likewise, the modified polar filter presented in Section 2.5.3 can be replaced by other filters that estimate more-complex target behavior. Furthermore, the observer parameterization and estimator formulations can be expanded to include multiple observers sensing multiple targets, and the trajectory-design algorithm can be extended to include them.

One important characteristic of the new trajectory-design algorithm is its ability to produce near-optimal solutions in real-time. The current algorithm is capable of calculating a result in approximately 1.5 seconds for the typical scenario and design parameters (Section 7.2). As the observer and target motions become more complex, the computational cost and planning times will grow. The two components of the trajectory

design algorithm that will have the largest direct effects are the creation of the candidate trajectory space and the simulation of the target estimator.

The size of the candidate trajectory set increases exponentially as the number of physical dimensions increases (i.e. moving from two dimensions to three). Therefore planning time for an enumerative algorithm, such as the breadth-first method used here, will also increase exponentially in time when applied to three dimensions, if all other parameters are kept constant. However, the computational time is also a function of the trajectory parameterization, the discretization size, and the search method used. Each can be changed in order to achieve reasonable computational time. Randomized motion planning produces fast, feasible results and is a promising method for this problem as the physical dimension increases.

The majority of computational cost and planning time is devoted to simulating the target-motion estimator (i.e. calculating the predicted estimate-error covariance matrix). As the estimated target motion becomes more complex, either by estimating accelerated movements or by estimating multiple targets, the number of states needed to describe the motion will also increase. The complexity of the estimation is of the order $O(n_{states}^2)$ where n_{states} is the number of total target-motion states. Furthermore, the nonlinear estimator needed for target-motion estimation will have several matrix inversions and other linearization steps which require significant computational cost. Unlike the candidate trajectory set, the computation and time requirements of the simulated estimator cannot be reduced easily. Future extensions to multiple or more-complex targets will therefore require faster computers with advanced estimation techniques in order to perform in real-time.

CHAPTER 5

Full Observer-Trajectory Generator

In Chapter 5 the core observer-trajectory-design algorithm described in Chapter 4 is extended to create a new observer-trajectory generator for target-motion estimation when the initial target state is unknown. Trajectory design for estimating this unknown initial target motion requires the consideration of several new issues including the application of observer-motion constraints, target-motion estimator initialization, trajectory redesign as the estimate converges, and reacquisition when a target is lost. This chapter presents the details of the new observer-trajectory generator that addresses these issues while using the trajectory-design algorithm of Chapter 4 for known target state as its core.

5.1 Introduction

For an operational autonomous observer, trajectory generation for target-motion estimation using monocular vision is complicated if the initial target state is unknown or uncertain. When the target is first encountered, information about its state is sparse and its motion is highly uncertain. However, as discussed in Section 3.1.3, the observer must move in order to gain more information and estimate the target location and velocity. The trajectory generator that controls the observer movement must therefore incorporate the highly uncertain target-state information as best as possible in order to generate a response that best enhances target-motion estimation performance.

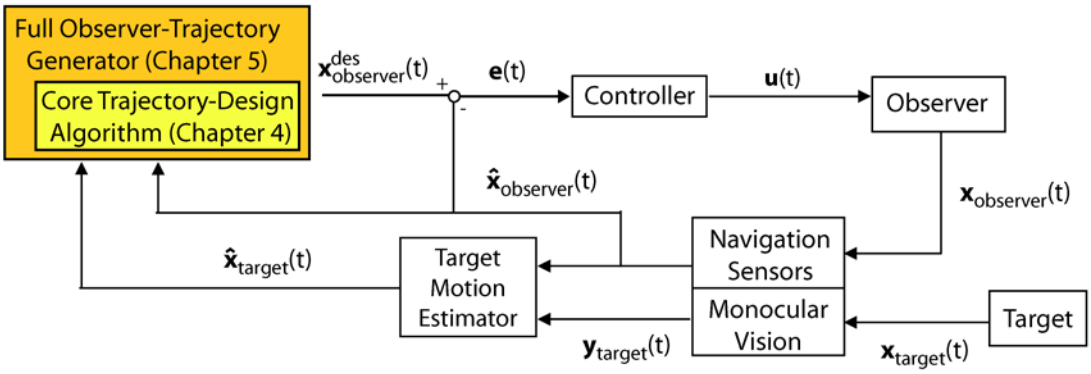

Figure 5.1 New observer-trajectory generator integrated with the target-motion estimation system using monocular-vision.

Given vision-sensor characteristics, a specific estimator formulation, and known initial observer-target geometry, the algorithm developed in Chapter 4 generates a near-optimal observer trajectory in real-time. However, that algorithm does not apply to the case of unknown initial target state, so a new observer-trajectory generator is needed. This chapter presents a new observer-trajectory generator that extends the core trajectory-design algorithm developed in the previous chapter. Figure 5.1 shows the relationship of the new trajectory generator to the rest of the target-motion estimation system. The new generator uses the observer trajectory-design algorithm as its core, and addresses several additional issues.

The uncertainty in the initial target state leads to a fundamental conundrum for any trajectory generator that is to improve target-motion estimation, regardless of the specific relative-position sensors used. The optimal observer trajectory can be determined as a function of the position and velocity of the target, yet the whole point of designing and following the trajectory is to determine those unknown target states. In other words, "You tell me the target's motion and I promise to tell you the best trajectory for estimating the target's motion"[39]. Previous trajectory design work has focused exclusively on known initial target scenarios [10 - 20]. The new generator presented here addresses for the first time the issue of unknown initial target states.

A second key issue arises from the fact that the presence of the target is initially unknown, and subsequent estimates are highly uncertain. Because the best observer trajectory is a function of the target state, the trajectory must be updated as the estimate improves. Upon first sensing the target, the observer begins the target-motion estimator by specifying an initial target-state estimate and error covariance matrix. Because the estimation process is nonlinear, the resulting performance depends on the initial values. Additionally, the estimate represents the observer's best information regarding the target, and the results of the observer-trajectory generator also depend on the initial values. Furthermore, as the observer moves along its trajectory the accuracy of the target estimate improves, and knowledge of the world improves.

It is important to note the difference between a dynamic world and a changing world model. The work presented in this dissertation assumes all targets move with a constant speed and that the observer knows this about the target. Therefore the world in which the observer operates is not dynamic -- the objects do not change their behavior. However, what does change is the information at the observer's disposal. As it moves, the observer's sensors take new measurements and more information becomes available. The world model, which includes the target-state estimates, changes as a result of this new information. For example, it is highly unlikely that the position of a stationary object is initialized correctly, so as the estimate converges the world model changes. The observer perceives only what the world model tells it, so from the observer's perspective, convergence in the model is indistinguishable from dynamic target behavior. This fact has the benefit of allowing the trajectory generator to extend easily to a dynamic world. That extension, however, is beyond the scope of this dissertation.

The final key issue addressed by the new trajectory generator results from the combination of target-estimate uncertainty with the limited vision field of view and the constrained observer motion. The main ramifications of the vision field-of-view limitation and observer kinematic constraints are that the observer cannot always keep the target in view and that trajectories will be generated that allow this to occur. When the target state is known, the observer can confidently turn away, lose sight of the target, and then turn back to reacquire the target at a new, but predicted, location. If the target motion is uncertain, the observer can no longer expect the target to reappear at a predicted location. Therefore the new trajectory generator must either eliminate trajectories that lose sight of the target or include additional motion primitives that enable reacquisition if the target becomes lost.

The new observer-trajectory generator described here creates trajectories using at every calculation the best available target-state information at that time. The target-motion estimator recursively fuses the known observer motion with the monocular-vision sensor measurements in order to calculate a state estimate and an estimate-error covariance matrix. These two variables define a probability distribution that represents the sum of all previous measurements and describes the observer's current knowledge of the target state. The trajectory generator therefore bases all observer paths on the target estimate, which is the best available description of the target state. Furthermore, the core trajectory-design algorithm described in the previous chapter is invoked by using the target-state estimate in place of the known target state. Because the state estimate converges as the observer moves, the trajectory generator continually replans as the target-state estimate changes. In this manner, the trajectory generator always uses the current, best available estimate and takes advantage of the trajectory design algorithm developed in Chapter 4.

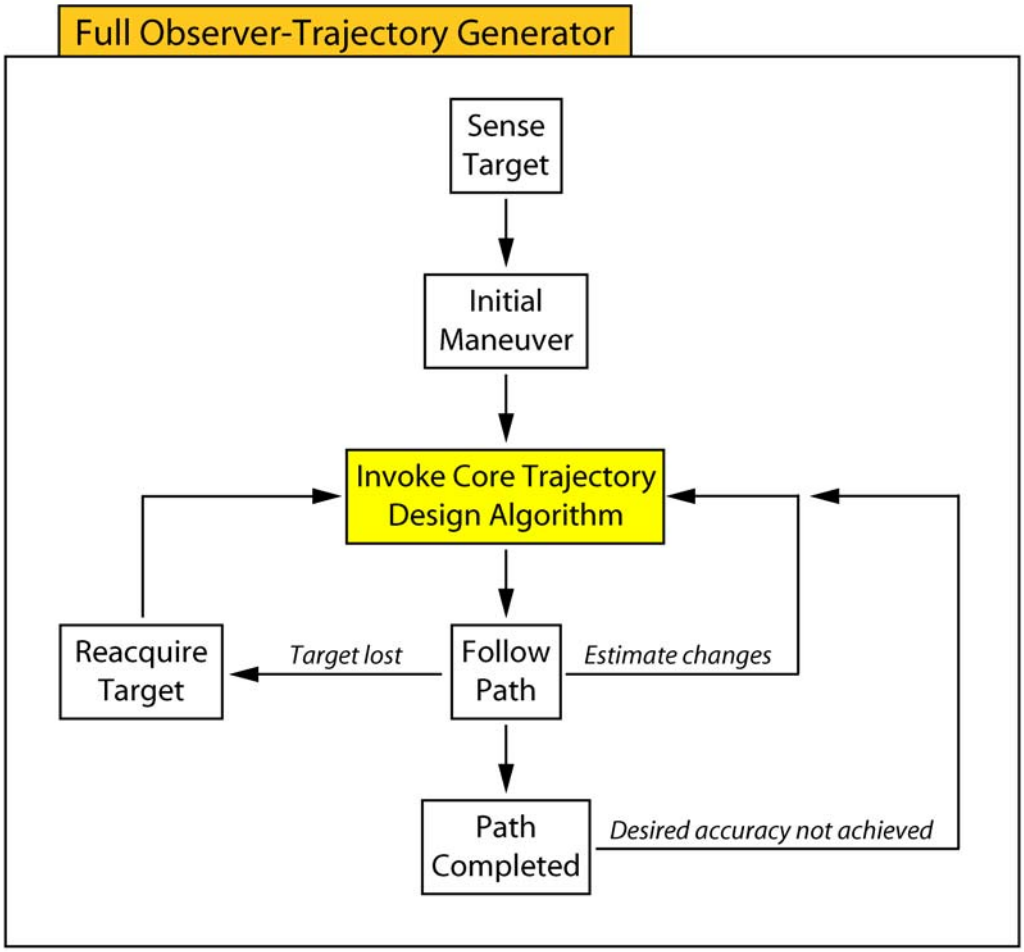

Figure 5.2 Flow diagram of the full observer-trajectory generator

As part of the trajectory-generation process, the observer monitors the state of the world and initiates responses as information changes. Important changes to the world include the presence of a target seen for the first time, changes in the estimated target motion relative to its predicted motion, and the loss of the target from view. The process of full observer-trajectory generation is outlined as a flow diagram in Figure 5.2. When the target is first seen, the observer initializes the target-motion estimator and then performs a short, predefined maneuver (different from the maneuvers that comprise the trajectory parameterization of Section 4.3.1) before invoking the core design algorithm with the target-state estimate. As the observer follows the resulting trajectory, the generator monitors the target-state estimate and compares it to the predicted target path. If the target becomes lost, the observer stops following the planned trajectory and invokes a reacquisition maneuver. Finally, the mission objective, i.e. the estimate uncertainty or total trajectory duration, is also monitored and trajectories are either replanned or halted depending on the performance of the estimation as the observer moves.

The remainder of this chapter describes the details of the new observer-trajectory generator. The components of the flow diagram are discussed, including the Sense Target, Initial Maneuver, and Reacquire Target components. In addition, extensions to the trajectory-design algorithm developed in the previous chapter are presented.

5.2 Sense Target: Estimator Initialization

The first significant change in the perceived state of the world occurs when the observer sees a target object for the first time. Until this point the observer is operating in what it thinks is a target-free environment. Once the target is sensed for the first time by the computer vision system, the observer creates an estimator to fuse additional bearing measurements with the observer navigation data and then the observer calls the trajectory generator. The first measurement the observer receives comes in the form of a single bearing angle to the target; yet initial values for the full target-state estimate and the estimate-error covariance matrix are needed at once. The ability of the trajectory generator to determine an optimal result and the overall performance of the target-motion estimation depend greatly on this initial target-state estimate.

Upon receiving the first vision measurement, the observer-trajectory generator calculates an initial target-state estimate and error covariance matrix in order to start the target-motion estimator. The first vision measurement determines the bearing to the initial target-state estimate. By also setting the initial relative range, the initial target position estimate is specified. Although the vision system has a maximum range, initializing the estimate at a shorter range (i.e. assuming the target is close) encourages the observer to generate conservative trajectories that avoid hitting the target. If the target is assumed to be farther away than it actually is, there is a greater likelihood of the observer striking it before sufficient observability is achieved. For the system used in this dissertation the initial range r_0 is always set to 0.5 meters. Given the initial bearing and range, the target-state estimate is initialized as follows:

$$\hat{\boldsymbol{x}}_{target}[0] = \begin{bmatrix} x_{observer}[0] + r_0 \sin\beta_0 \\ y_{observer}[0] + r_0 \cos\beta_0 \\ \dot{x}_{observer}[0] \\ \dot{y}_{observer}[0] \end{bmatrix} \tag{5.1}$$

or equivalently in modified polar coordinates

$$\hat{\boldsymbol{x}}_{mp}[0] = \begin{bmatrix} \beta_0 \\ 1/r_0 \\ 0 \\ 0 \end{bmatrix}. \tag{5.2}$$

Initializing the target with the observer velocity is the same as initializing it with zero relative velocity.

More important than the initial target-state estimate, the initial error covariance matrix must also be specified. This matrix represents the uncertainty of the initial estimate and must reflect as best as possible the likely error in that target-state estimate. The initial computer vision measurement provides only a bearing angle to the target object. However, because the target is in view, it must be in front of the observer. The range uncertainty is therefore initialized to extend from the target-state estimate to the observer. The cross-range error is a function of the noise on the bearing measurement. For a measurement error of β_{err} (in radians) the corresponding cross-track estimate error is $r_0 \cdot \beta_{err}$. Figure 5.3 depicts the initial target-state estimate and the error ellipse defined by the initial error covariance matrix

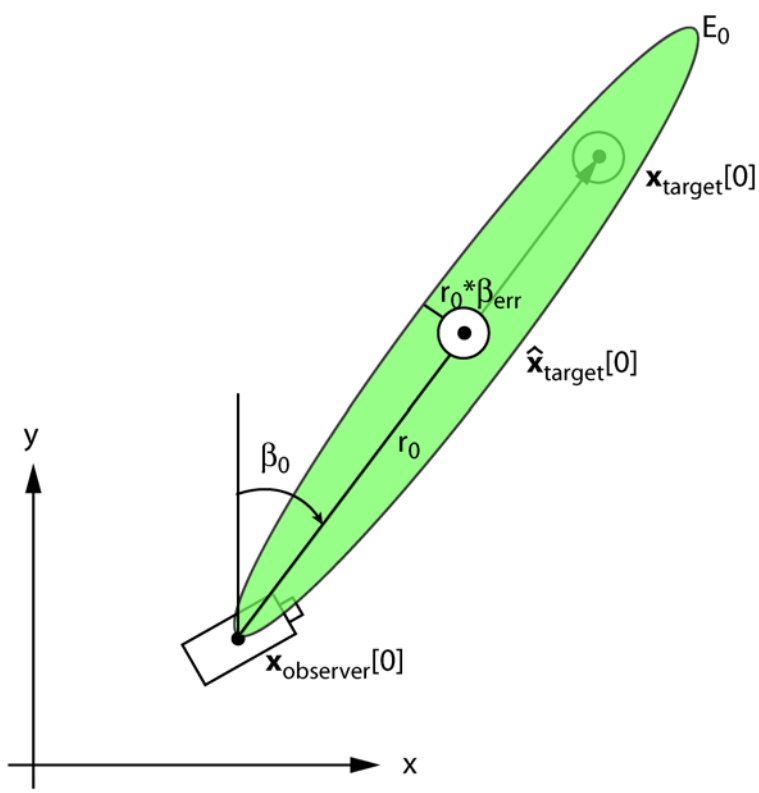

Figure 5.3 Initial target estimate

$$\hat{\boldsymbol{P}}_{target}[0] = \begin{bmatrix} \boldsymbol{T}_\beta \cdot \begin{bmatrix} r_0^2 & 0 \\ 0 & (r_0 \cdot \beta_{err})^2 \end{bmatrix} \cdot \boldsymbol{T}_\beta^T & \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} & \boldsymbol{T}_\beta \cdot \begin{bmatrix} V & 0 \\ 0 & V \end{bmatrix} \cdot \boldsymbol{T}_\beta^T \end{bmatrix} \tag{5.3}$$

where $\boldsymbol{T}_\beta$ is a transformation matrix defined by the target bearing β_0

$$\boldsymbol{T}_\beta = \begin{bmatrix} \cos(\beta_0) & \sin(\beta_0) \\ -\sin(\beta_0) & \cos(\beta_0) \end{bmatrix} \tag{5.4}$$

and V is the speed of the observer. In modified polar coordinates

$$\hat{\boldsymbol{P}}_{mp}[0] = \begin{bmatrix} \beta_{err}^2 & 0 & 0 & 0 \\ 0 & 1/r_0^2 & 0 & 0 \\ 0 & 0 & (V/r_0)^2 & 0 \\ 0 & 0 & 0 & (V/r_0)^2 \end{bmatrix}. \tag{5.5}$$

The initial target-state estimate used to generate the observer trajectory has a large effect on the overall behavior of the trajectory design process. The convergence of the target-state estimate is a strong function of the initial state estimate. If the initial estimate is accurate, convergence will be rapid and the predicted values will always be close to the actual estimate values. This reduces the need to institute replans as will be discussed in Section 5.5. Furthermore, the shape of the optimal observer trajectory is a strong function of the initial range to the target. Large initial errors in the estimate result in trajectories that are not close to the optimal paths.

5.3 Initial Maneuver

The initial target-state estimate has large uncertainty, and any observer trajectory that attempts to estimate the target motion based on it will perform poorly. Although the target bearing and range are initialized with reasonable, conservative values, the target velocity cannot be initialized well based on a single bearing measurement. Without a good velocity estimate, the observer cannot determine the proper direction to move and can easily lose the target. It is therefore beneficial to allow the estimate to converge before invoking the observer-trajectory generator. However the whole point of planning the trajectory is to enable the estimation in the first place.

In order to generate a reasonable initial target-state estimate to use with the core trajectory-design algorithm, a short, predefined zig-zag motion is performed first. The zig-zag contains two maneuvers (as described in Section 4.3.1, but with shorter durations) which enable observability, albeit poor, of all of the target states [13]. The heading changes (β_{zz}) of each maneuver in this zig-zag motion are kept small, approximately 15 degrees off the target bearing, in order to keep the target in view and insure that new information is obtained (Figure 5.4). After this initial trajectory primitive is performed, the trajectory-design algorithm is invoked to continue the estimation process.

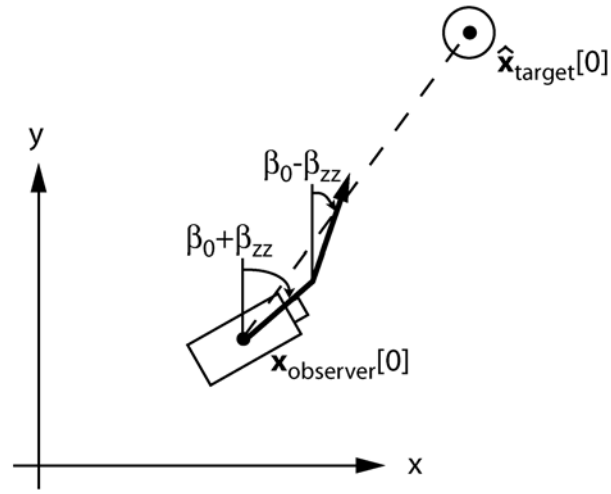

Figure 5.4 Initial zig-zag maneuver

5.4 Invoke Core Trajectory Design Algorithm

The observer-trajectory design algorithm for known initial target state presented in Chapter 4 serves as the core of the new observer-trajectory generator. The algorithm is invoked after the initial observer maneuver, after a lost target is reacquired, and whenever the estimate changes significantly. Although the algorithm was developed for known initial target motion, it can be extended to the case of an uncertain target. In order to make the extension, several issues are addressed including the algorithm input, the transition from an old path to a new one, design parameters for the algorithm, and the activation of observer motion constraints.

5.4.1 Input

The core observer-trajectory design algorithm presented in Chapter 4 takes as input the observer position, the initial target state, the sensor model, and the estimator formulation. However, the target motion is now being calculated by an estimator which fuses the known observer motion with the monocular-vision sensor measurements and can provide only an uncertain, probabilistic measure of the target state. The actual initial target state therefore cannot be given to the algorithm. Although the actual target state is not known, the target-motion estimator still provides a state measurement that can be used by the trajectory design algorithm. Instead of passing the algorithm the actual target state, the current target-state estimate is used. The remaining inputs, however, are unchanged and are passed directly to the core design algorithm.

By using the target-state estimate, the trajectory-design algorithm takes advantage of the best available target state information. The target-state estimate and estimate-error covariance matrix represent a sum of all prior measurements and embody the observer's current knowledge of the target motion. While the actual target motion varies from this estimate, the observer has knowledge only of the estimate and therefore must use that knowledge in order to generate a trajectory. The trajectory-design algorithm also takes as input *a priori* information, in the form of the covariance matrix, of the probability distribution of the initial target state. By including the estimate-error covariance matrix, the trajectory generator returns the best possible observer path given all the available information.

5.4.2 Transition

The combination of observer-motion primitives (i.e. the initial zig-zag maneuver and the reacquisition phase) and multiple observer-trajectory redesigns require a smooth transition from old to new paths. One simple option is to stop the observer while a new path is created leading from that location and then to continue along the new path once it has been generated. However, stopping to wait for a new plan wastes time that could be used gathering information. Plus, not all observers are capable of easily stopping their motion. For example, an unmanned air vehicle must continue moving in order to stay aloft and therefore cannot stop to wait for a new plan. A second option is to call for a new plan based on the expected observer state some time in the future and to switch from old to new trajectories once the observer reaches that state.

In order to provide a smooth transition that wastes no time, the trajectory generator calls the core trajectory-design algorithm with future expected observer and target states as inputs. An observer trajectory is specified by a time-parameterized function that gives future desired observer states. This parameterization is used to determine the current desired position, based on the current time. However, the trajectory can also be used to determine the position of the observer at some future time. Likewise, the current target-state estimate defines a predicted target path that is used to determine the expected future target state.

The amount of lead time given to the trajectory generator is determined by the typical planning time. In order to ensure a smooth transition, the new trajectory must be ready when the observer arrives at the future state. For the system presented in this dissertation, trajectory generation takes approximately 1.5 seconds to complete. Therefore the trajectory generator projects the observer and target motions 2.0 seconds into the future and invokes the trajectory-design algorithm with those values, allowing the generator ample time to design the result before it is expected.

5.4.3 Parameter Values

Because the core observer-trajectory design algorithm is called repeatedly by the new trajectory generator, the balance between planning time and performance is critical. Short planning times enable the generator to respond quickly to estimate changes and lost targets. Likewise, good performance enables the successful estimation of a previously unknown and then highly uncertain target-state. Fortunately the effects of the design parameters are the same regardless of the nature of the target-state input (i.e. the known initial state assumed in Chapter 4 or the estimated target-state used in this chapter). Therefore the trade-offs described in Section 4.5 still apply and no new discussion is needed.

5.4.4 Constraint Activation

The observer position constraints can have a large effect on the trajectory-design output by reducing the size of the search space and by limiting the number of maneuvers that provide observable measurements. While some of the constraints described in Chapter 4 are imposed by the kinematics of the observer and the nature of the vision sensor, others can be activated at the discretion of a higher-level operator in order to influence the behavior of the observer-trajectory generator.

Two main constraints limit the ability of the observer to move towards the target and to move perpendicular to the line of sight to it. Both of these observer-motion components are important to the achievement of the time and uncertainty goals of the observer-trajectory generator. The field-of-view limitation of the computer vision sensor leads to a possible motion constraint: to keep the target in view at all times. This constraint limits the transverse motion of the observer and requires it to move towards the target. Conversely, the error ellipse constraint defines a safety region around the target which tends to push the trajectories to the sides of the line-of-sight vector. The next two sections discuss the effects of these two constraints.

Maintain view

Because the computer vision sensor has a limited field of view and because the observer rover can only move forward in the direction it is pointed, there are a large number of candidate trajectories that take the target out of the field of view of the observer. If the target-state estimate were well known, this action would not cause significant problems. However, the estimate is not well known initially, and it is therefore possible for the observer to lose the target when it turns away. In order to prevent this loss from occurring, the observer can be constrained to keep the predicted target-state estimate in view at all times. The result is that the observer will have a reduced candidate trajectory set and a greatly reduced reachable region of the environment. Rather than having the 180 degrees cone shown in Figure 4.8 the observer motion would be restricted to a cone of angle equal to the camera field of view, approximately 40 degrees.

The effect of the restricted trajectory set is to reduce the amount of transverse motion the observer is able to achieve. This restriction severely limits the amount of parallax -- the total bearing change -- of the observer relative to the target and thus severely limits the performance of any target-motion estimator. The observability of the target-state estimate is increased as this bearing change is increased. Requiring the observer to keep the target in view therefore leads to less accurate results, or conversely, longer duration trajectories to reach a specified level of accuracy.

For the system presented in this work, the constraint that the target remain in view is not applied. The loss in observability and planning flexibility does not outweigh the possibility of losing the target. The Reacquire Target component of the trajectory generator described in Section 5.6 is capable of recognizing and responding to the target loss should it occur.

Error Ellipse

The target estimate-error covariance matrix defines an uncertainty ellipse which can be used as an observer position constraint. Section 4.3.2 explains how this ellipse can be used as a safety constraint to prevent the observer from colliding with the target. In most cases the target uncertainty is in the relative range and the error bound is a thin ellipse pointing from the target to the observer. The ellipse constraint then prevents the observer from moving directly towards the target object without first moving to the side.

Motion toward the object is vital because it keeps the target in sight, provides the actual measurements, and enables better bearing rates. It has already been discussed above how the limited camera view can cause the observer to lose track of the target. Only by moving towards the target can it remain sensed. Therefore, motions toward the target are also the ones that lead to new information about the target. Finally, the estimator performance improves as the bearing rate between the target and observer increases. For a fixed speed, the only way to increase the bearing rate is to decrease the range -- head towards the target.

In the context of this dissertation, the safety margins of the error ellipse outweigh the drawbacks caused by turning the vision sensor away from the target. The uncertainty in the target estimate is initially high and leads to competing factors. This uncertainty makes the act of turning away from the target more likely to lose track. However, it also means there is a large region of uncertainty and the target could effectively be anywhere in front of the observer. In order to increase the speed and improve the end result of the estimator performance it makes more sense to move to the side first to improve the early estimate than it does to move forward.

5.5 Follow Path - Estimate Change

A very important, often overlooked characteristic of observer-trajectory generation is the fact that there is a dilemma inherent in this type of nonlinear design problem [34 - 39]. The purpose of the trajectory generation is to provide the estimator with suitable observability. The observability is a function of the relative geometry between the observer and the target. The optimal observer path is therefore dependent on the target position and velocity. However, the whole point of designing the path is to enable estimation of the target state in the first place. This conundrum is partially resolved by using the current target-state estimate, which represents the sum of all the observer's knowledge to the current point, to predict the target motion and design the optimal trajectory *for that* target state. Because it is not likely at the beginning of the design process that the current target-state estimate matches the true target motion, the resulting trajectory will necessarily perform suboptimally.

The target-estimate conundrum manifests itself over the course of the observer trajectory by the drift of the current estimate away from the predicted target path. As the observer follows its trajectory, more measurements are taken and the target-state estimate converges. Since the estimate is initially highly uncertain, the estimate may vary greatly from the predicted target path used to generate the observer

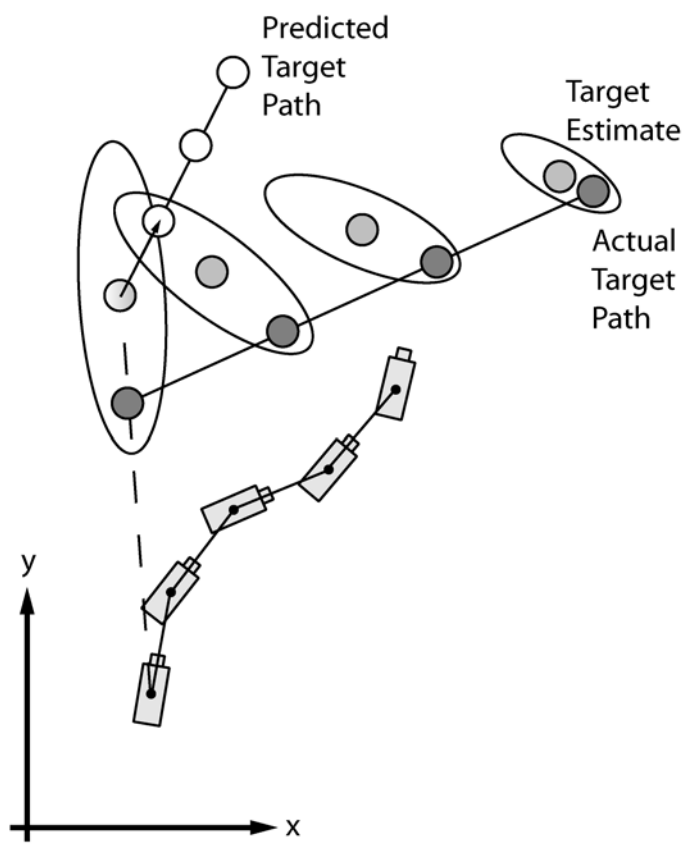

Figure 5.5 As the observer follows its trajectory the target estimate drifts away from the motion it predicted.

trajectory. Figure 5.5 shows an example of the target-state estimate converging away from the predicted target path. At the time of the trajectory design, the predicted path and the target-state estimate are the same, but both are quite different from the truth. With each successive step, the observer gains new data and the estimate converges towards the true state, away from the original predicted target path. Since the observer trajectory was designed for the original predicted target path, there is no reason to expect the design goals to be achieved with the same observer path and the new perceived target motion. At some point the trajectory generator must step in and respond to this changing world.

The observer-trajectory generator addresses the target-state-estimate conundrum by monitoring the target-state estimate and the observer trajectory and invoking a replan when appropriate. Several different strategies can be used to determine when the replan should occur. These strategies depend on the complexity of the trajectory design and the cost, in terms of time and other resources, of invoking the core design algorithm. Two simple strategies are presented in this dissertation. The first calls for the observer to replan continually, invoking the design algorithm as soon as it returns a solution. The second strategy calls for a new plan whenever the current target-state estimate varies significantly from the predicted estimate. In either case, like the initial maneuver, the trajectory generator projects the observer's motion forward in time and always calls for a new plan based on the future prediction of the observer state at the time the design algorithm will return. In this way the plan is ready once the observer arrives at the future point, and the transition to the new trajectory is smooth.

5.5.1 Replan Continually

Continually invoking the core observer-trajectory-design algorithm insures that the observer always follows trajectories based on the most recent information. As the observer moves, the estimate is constantly updated and the current estimate diverges from the predicted target path. Following the trajectory based on the old predicted path ignores the information gained by the observer since it has begun moving. Always replanning with the current estimate insures that all information is used and the resulting trajectory is the best possible observer path.

The continuous-replanning strategy works as long as the cost of calling the trajectory-design algorithm remains low. For the specific problem of interest in this dissertation, the algorithm takes approximately 1.5 seconds to return a solution. Therefore the trajectory generator cannot be called more frequently or else the observer would never move. For the experiments presented in Section 7.3 and Section 7.4, the algorithm is called every 2.0 seconds. This interval gives the method sufficient time to generate a solution by the time the observer expects it. For other systems, the computational time may be longer or other resources, such as power, may limit the ability of the observer to invoke the trajectory generator. In this case, the continuous

strategy would not be recommended and instead the observer would use the second strategy described in the next section.

For the fixed-time, minimum-uncertainty design objective the total trajectory duration is specified, so parameters for the trajectory generator must change as the replans occur. When the trajectory design algorithm is invoked for a replan, the new total time is just the original duration minus the elapsed time. In order to maintain the same general observer behavior, the original maneuver duration is used and the number of maneuvers is reduced. This act has the side effect of reducing the planning time as the target estimation unfolds. Once the remaining time falls below the amount required for two full maneuvers, two maneuvers are always taken, each with half the remaining time.

For the fixed-uncertainty, minimum-time scenario the maneuver duration and desired uncertainty are originally specified. These values are not changed with each successive replan.

5.5.2 Replan when Estimate Varies from Prediction

When the cost of invoking the trajectory-design algorithm is high, a new plan is called for only when the current target-state estimate varies significantly from the predicted target path. Although the target-state estimate continually improves, the convergence of the estimate depends largely on the initial state estimate. If this initial estimate is good, the target-state estimate will remain close to the true target motion and will not change substantially. Therefore a new observer trajectory will not significantly improve the estimation performance. However, if the current estimate diverges from the predicted target path, a new plan will be needed to improve the estimation performance. Therefore the observer-trajectory generator monitors the current target-state estimate and the predicted target path, and calls for a replan if they vary significantly.

The position, speed, and heading of the target are considered when deciding to replan. The change in position between the current target-state estimate and the predicted target location is the most obvious state to check. If the target is not where it is supposed to be, the observer responds. However, the difference in heading between the two is less obvious and more important. While the bearing to the initial position is known and only the range is uncertain, the velocities are not initialized with as much insight. Additionally, the heading differences lead to growing discrepancies between the positions. Even if the current and predicted estimates are identical at a given time, if their headings differ they will not have identical positions in the future. Replanning sooner, before the separation gets large, therefore leads to better results. Finally, the speed is the least important variable to monitor. Nonetheless, it also indicates a future divergence in positions and is therefore monitored. The system developed for this work uses the following values to initiate a replan:

$$\begin{aligned} \Delta r &= \sqrt{(x_{pred} - x_{est})^2 + (y_{pred} - y_{est})^2} \leq 0.8\text{m} \\ \Delta\theta &= \theta_{pred} - \theta_{est} \leq 90° \\ \Delta v &= \sqrt{(\dot{x}_{pred} - \dot{x}_{est})^2 + (\dot{y}_{pred} - \dot{y}_{est})^2} \leq 0.05 \text{ m/s} \end{aligned} \tag{5.6}$$

Although the target-state estimate and the error covariance matrix completely describe the output of the estimator and the knowledge of the target, the replans are triggered only by the estimate. Regardless of the values of the target-state estimate, the error covariance matrix represents the certainty of that estimate and always decreases with more information. Consider the case where the estimate is unknowingly initialized to the true target state. The initial error covariance matrix will still be large and the observer will not have faith in the estimate. As the observer follows its trajectory and gains new information, the target-state estimate will not change. However, the error covariance matrix will shrink and the observer's confidence in its estimate will grow.

An additional strategy not developed in this dissertation compares the final predicted error covariance matrix for the predicted target path to that expected for the current target-state estimate. Rather than initiating a plan when the current estimate and predicted path vary, the observer trajectory generator would initiate a new plan when the final predicted performance would suffer significantly. The development of this strategy is not addressed in this work.

5.6 Target Lost - Reacquire Target

The target object can become lost when the observer trajectory causes the target to leave the vision sensor field of view. The predicted target path is initialized with uncertainties that are reduced by the estimation process. Observer trajectories that turn away from the target provide good observability by creating a large baseline between measurements. At some point the observer turns back toward the predicted target path, expecting the target to come back in view. However, if the observer does not have sufficient motion before the turn, it will not have gathered enough information to calculate an accurate target-state estimate. Furthermore, if the predicted target motion varies significantly from the actual motion, the observer will not see the target when it turns back and the target will be lost.

The target is considered lost when the current target-state estimate is in view of the observer but the vision sensor has no measurement, implying that the actual target is not in view. This situation may occur because the initial target-state estimate is poor or because the estimator itself performed poorly and the estimate has diverged. In either case, the observer is in trouble. It has two conflicting pieces of information, one that

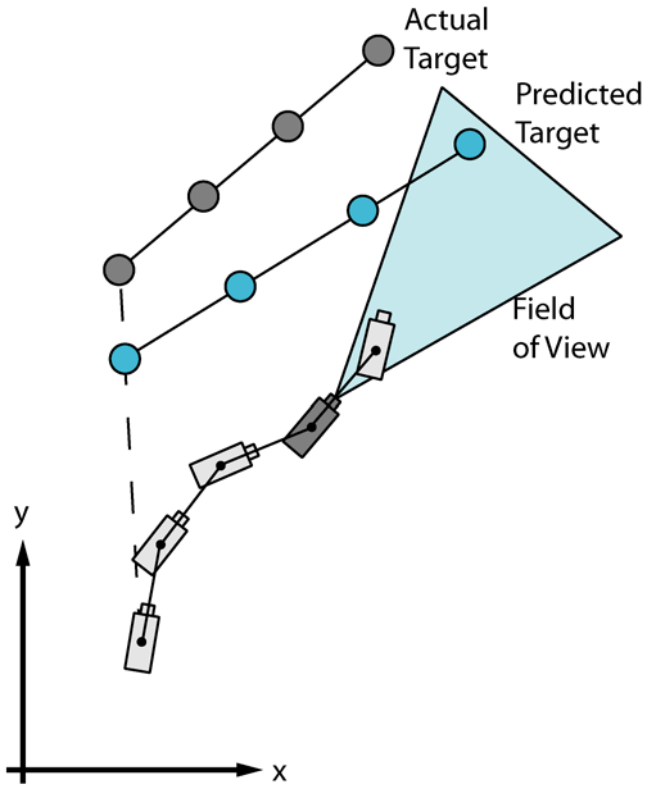

Figure 5.6 The target is lost because it is not in the observer's field of view even though the predicted estimate is in view.

says the target should be in sight and the other that says it clearly is not. In order to resolve the situation the best response is to reacquire sight of the target.

The inverse of the previous case -- the target is sensed by the observer but the current estimate is not in view -- is another example of a lost target. However, this situation does not occur often. As long as the target is behaving as modelled with constant velocity, a well-designed estimator should always have the current estimate in view if there is a current measurement. The observer-trajectory generator expects a well-designed estimator and does not check for this situation.

The final definition of a lost target occurs when the predicted target path (as opposed to the current target-state estimate) is in view but the actual target is not. Figure 5.6 depicts an example scenario. The trajectory has the observer turn away from the target after the first maneuver. Although the velocity estimate is close to the actual velocity, there is a significant difference between the estimated and actual range. When the observer turns back towards the predicted target, it expects the target to come back into view. However, the actual target does not fall in the observer's viewing cone and the target is now lost. The core trajectory-design algorithm expects the observer to gain information and improve its estimate when the predicted target

is in view. If the target is lost according to this final definition, the expected information gain is not taking place and the final performance of the observer trajectory will be significantly reduced.

Once the target is lost, the observer has no reason to believe that it is following a useful path and the observer-trajectory generator must respond. The generator monitors the current estimate and the predicted target path and initiates the reacquisition phase if the target is lost. Because the geometry of the vision sensor is used to design the observer trajectories, the monitor can easily determine if it expects to see the target or not. Once the monitor recognizes that the target is lost, it commands an immediate halt. The observer then enters a reacquisition phase in which it sweeps a specified angle to either side of the line of sight to the current target-state estimate. If the target is reacquired, the estimator incorporates the new measurements and the generator invokes a new trajectory design. If the target is not found, the observation run ends.

5.7 Path Completed - Desired Accuracy Not Achieved

It is possible that the predicted error covariance matrix is inaccurate and the observer trajectory does not achieve the desired level of accuracy. The predicted error covariance matrix is calculated based on the current target-state estimate at the time the observer-trajectory generator is invoked. The estimate at this time is likely to be uncertain and it is therefore likely that the final target covariance matrix at the end of the trajectory does not match the predicted error covariance matrix and does not achieve the desired uncertainty bounds.

If the final error covariance matrix does not achieve the desired uncertainty bounds, the generator initiates a new trajectory design. For the fixed-time, minimum-uncertainty case the allotted time has elapsed so no new plan is invoked. While not achieving the predicted performance, the observer still minimized the uncertainty under the given time constraints, thus achieving its objective. For the fixed-uncertainty, minimum-time scenario, the design algorithm is invoked as if it were occurring for the first time. The target-state estimate is not at the desired accuracy level so the observer-trajectory generator will generate a plan to get it there. The generator continues to call for new plans until the goal uncertainty bound is finally realized.

CHAPTER 6 Experimental System

The theoretical results derived in this dissertation are experimentally demonstrated on the Micro-Autonomous Rover platform developed by the Aerospace Robotics Laboratory and described in this chapter. A small wireless camera is mounted on one robot which serves as the observer vehicle and a second rover is used as the target. Overhead vision sensing provides position and velocity information for both robots. It is used as the navigation sensor for the observer and as a truth measurement for the target. Target tracking is achieved using vision-processing software developed in house as well. The other subsystems required to run the experiments include the estimator, the trajectory generator, the graphical user interface, and the network backbone. Figure 6.1 shows a diagram of the entire experimental system.

6.1 Micro Rover

6.1.1 Physical vehicle

The micro autonomous rover is a 9.8 cm tall, 10.2 cm diameter, wheeled cylindrical robot. Two wheels are mounted underneath the robot and are independently driven by standard radio-controlled (RC) servomotors that have been modified to achieve speed, rather than position, commands. The wheels are mounted adjacent to one another, along the robot diameter, with a separation of 6.8 cm. Two small posts with teflon caps are mounted perpendicular to the wheel base in order to balance the vehicle.

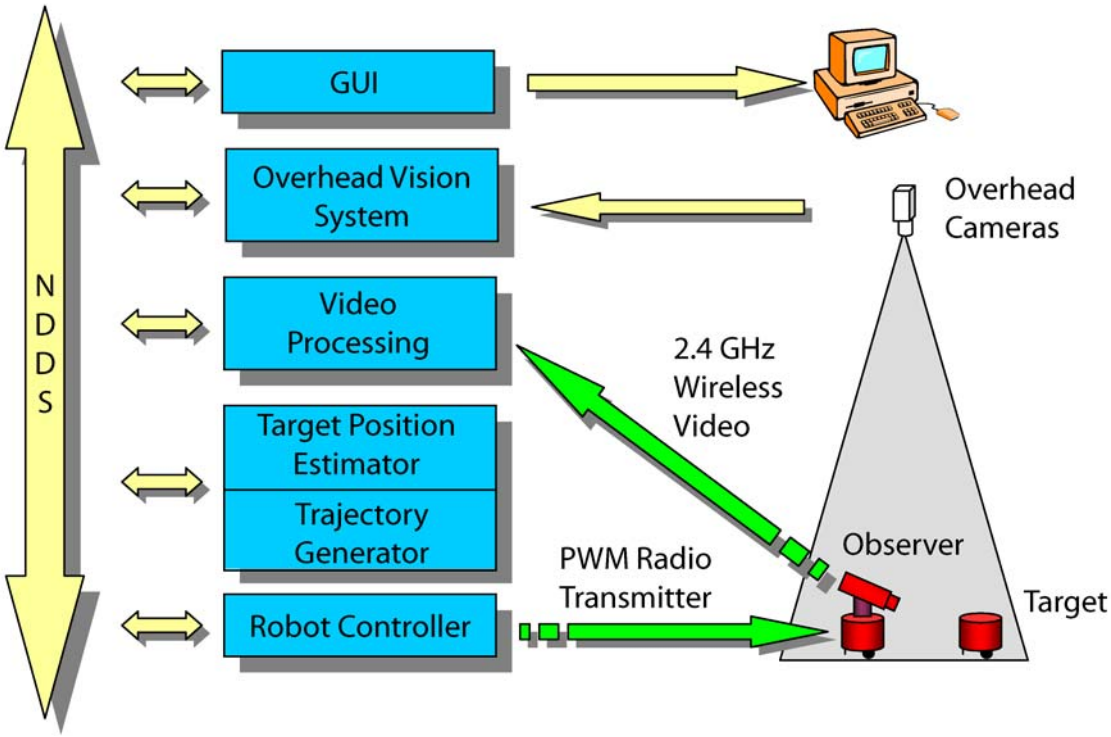

Figure 6.1 Experimental System

The robot interior is covered by a thin teflon frame. It hides a 4 cell, 4.8 volt NiCd battery and the RC receiver whose antenna exits the vehicle at the top and hangs by its side. The robot is topped with a plexiglass plate that holds three infrared LEDs that are used by the overhead vision system described in Section 6.1.2 to sense the robot's position and velocity. All signal processing and control is done off board and is described in Section 6.1.3.

Figure 6.2 The Micro Autonomous Rovers

The robot workspace is a 2 by 3 meter granite table. Multiple robots and additional obstacles may operate on the table depending on the experiment being performed. In this work, the observer and target robots are the only objects in the workspace.

Observer Robot

The observer robot is a micro rover with an additional wireless camera mounted on top such that it is looking slightly downward in the direction of positive robot motion (Figure 6.3). The camera is an X10-Xcam2 color camera integrated with a 2.4GHz wireless transmitter [77]. The Xcam2 has a 1/3 inch CMOS sensing area, a 3.0 millimeter focal length, and a field of view of approximately 50 degrees. The movable Xcam2 mount is glued in place in order to ensure a fixed camera-to-robot transformation. An infrared filter (not shown) is placed over the front of the camera in order to simplify the target-tracking software described in Section 6.2.2.

Target Robot

A second micro rover is used as the target object for the experimental demonstrations (Figure 6.3). The only modification to this robot is the addition of an infrared LED that the computer vision system is able to track. The LED is mounted on the top of the target but faces the side. A small shield (not shown) is placed over the LED in order to prevent interference with the overhead vision system.

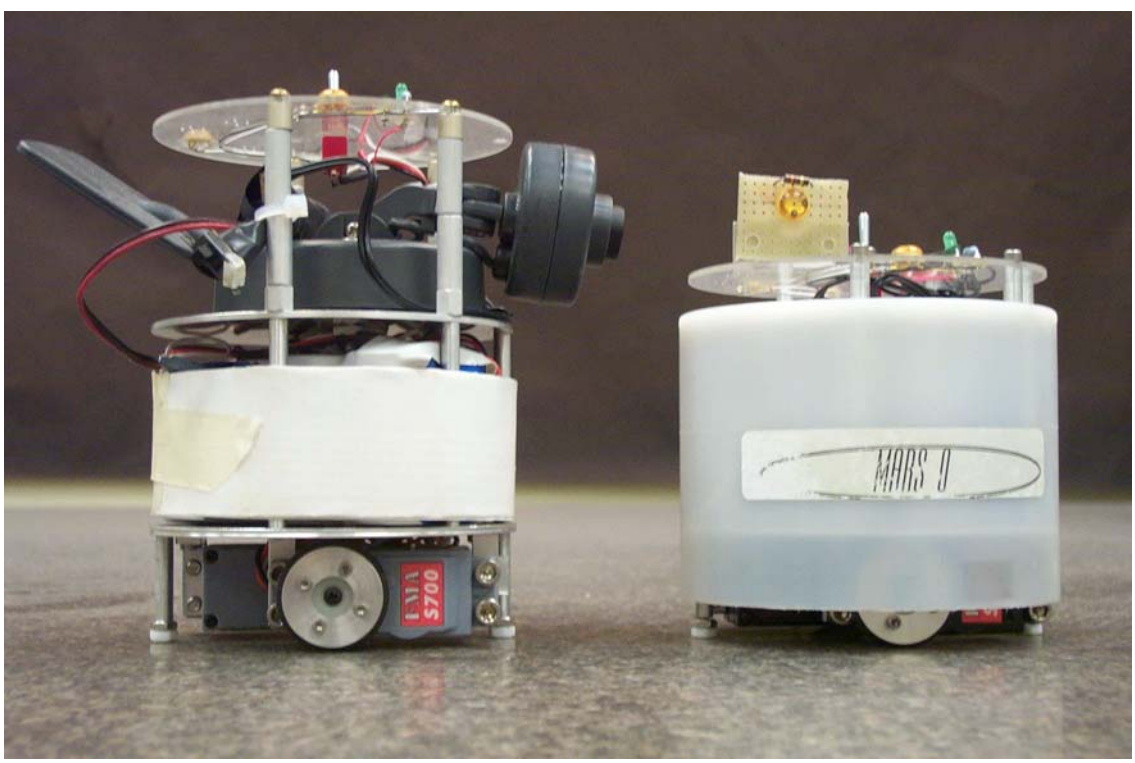

Figure 6.3 Observer and target rovers

6.1.2 Overhead Vision

Fixed atop each micro rover is a unique pattern of three LEDs that are sensed by an overhead vision system mounted above their workspace [78, 79]. Three Pulnix 440 CCD cameras are mounted directly overhead on the ceiling, looking downward on the table. Their views of the table overlap and provide full sensing of the rover workspace. They each have an infrared filter that allows them to see only the LEDs mounted on the robots.

The output of the Pulnix cameras are fed into a Matrox Meteor 1 board on a PC computer. The three black and white signals from the cameras are input into the vision processing computer on the RGB channels of the Meteor board.

The vision processing software consists of several subsystems. The first splits the digitized image into RGB components, which correspond to the black and white channels of the three different cameras. The LEDs are located within each image, the unique LED groups are identified, and finally the position of the object corresponding to each LED group is determined.

Because the three Pulnix cameras have infrared filters mounted on them, the only objects in the images are the infrared LEDs on the rovers. Once the image is digitized, the LED blobs are segmented and their locations within the image are passed to the next processing section.

The individual LED arrangements are next grouped and associated with specific robots based on their unique patterns. These patterns are known by the vision software which is able to recognize and distinguish between them. Once the LEDs have been grouped, the position and velocity of the associated vehicle is determined.

The overhead vision system outputs robot position and velocity information through the network backbone at 30 Hz. The LEDs can be located within the image with an accuracy of approximately one fourth of a pixel. The main source of error comes from the lens distortion and camera calibration used to project the vision measurements back into world coordinates. Using a grid of LEDs, a third-order polynomial fit is calculated to correct for lens distortion, and the intrinsic and extrinsic parameters of the cameras are determined. The system is able to achieve global accuracy of approximately 5 mm across the entire table.

6.1.3 Control system

All control processing for the micro rovers is done off-board on a PC computer running the Redhat Linux 7.3 operating system. Each robot is controlled by a separate thread that runs on the control computer. The control signals for each robot are output as voltages that are converted to pulse-width-modulated signals (PWM) by electronics developed in house (Figure 6.4) and then sent via a radio link to the individual robots.

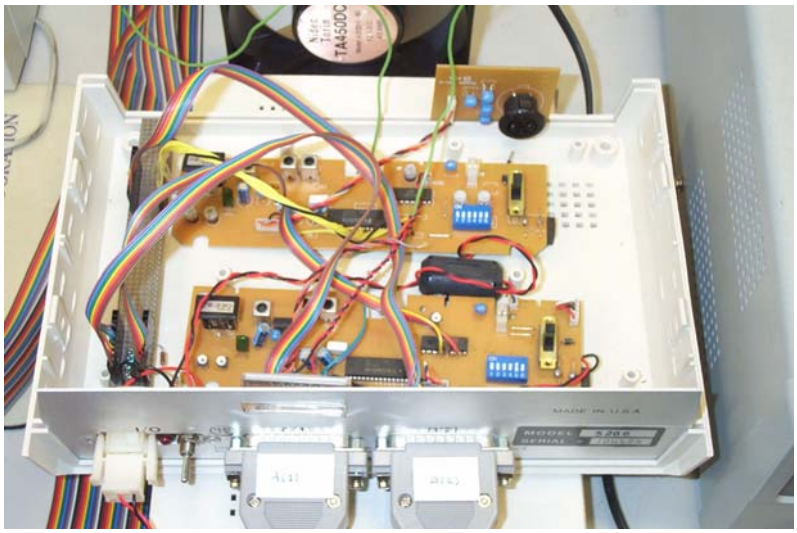

Figure 6.4 Pulse width modulation (PWM) conversion and transmission electronics

Each robot uses two of the 12 total channels broadcast by the controller transmitter. By using commercial RC components, the micro rovers can also be driven by hand using standard RC transmitters.

The RC servomotors on each robot have been converted from position to speed control. By driving each wheel independently, the speed and (yaw) angular velocity of the vehicle are controlled. System identification was performed in order to derive a map between servo voltage and wheel speed. The effects of battery voltage were not included in this analysis so the performance does degrade as the battery is drained.

Several different modes have been developed to control the rovers. Each mode consists of a feedforward control input calculated from the voltage map described above added to a feedback term. In the first mode, a simple LQR regulator is used to keep the vehicle stationary or to turn in place. In the second mode, a nonlinear control algorithm is used to follow smooth paths.

One level above the controller, a trajectory generator creates smooth, constant-radius arcs for the vehicle to follow. All commands to the robots come in the form of time-parameterized waypoints containing a starting position and heading. The trajectory generator converts the waypoints into time parameterized curves and then converts the curves into the desired position and velocity inputs at the current time. In this formulation, straight lines are considered arcs with an infinite radius.

6.2 Monocular Vision

6.2.1 Wireless Video

The monocular vision system consists of a single X10-Xcam2 color camera integrated with a 2.4GHz wireless video transmitter [77]. The video is transmitted roughly 4 meters to the receiver stationed at the image processing computer. While the transmitter has a range of approximately 30 meters, the patch antenna is not omni-directional, and dropouts occasionally occur if it is turned away from the receiver. Furthermore, a wireless ethernet unit in the laboratory was found to produce interference with the video transmission. This unit is not a part of the system described here and is therefore disabled when experiments are run.

The video receiver has 4 channels and the camera may be set to any one of them. Currently only a single camera, and thus one channel, is used as a part of this system. Three additional observer vehicles could easily be accommodated by the current system. The output of the video receiver is sent to a video splitter which sends the signal to a television for displaying and recording the video and to the image processing computer.

Figure 6.5 Target robot as seen by observer: without and with IR filter

6.2.2 Image Processing

Image processing is performed on a 550 MHz Intel Pentium III-based PC running RedHat Linux 6.2 with a Matrox Meteor I digitizer board. The system was designed in house using the Video4Linux2 drivers found on the Internet. The color video is digitized and processed at 30 Hz.

The identification, recognition, and tracking of target objects in a video stream is a challenging area of ongoing research [58, 59]. In an effort to maintain the focus of the research presented in this dissertation, the vision processing is simplified in order to create the most basic, workable system. The infrared filter on the

observer camera blocks all visible light. The only object in the workspace that passes through the filter is the infrared LED attached to the target.

The observer performs simple segmentation of the filtered image in order to track the target. The infrared LED appears as a bright white dot in the image. The centroid of this dot defines the location of the target in the image plane. This image location is then transformed via a lens distortion and camera model into a unit vector in relative world coordinates.

The object tracking may be initialized in one of two ways. First, a human user may click on the image and the vision software will segment the image in that immediate area. This method is useful if there are multiple objects in the scene. The second method calls for segmentation of the entire image and uses the largest segment as the target object. This is the method used in this work.

Once the object has been identified, it must be tracked over subsequent frames. Because the target will exit and enter the camera field of view over the course of an experiment, the image processing must be capable of reacquiring the target. If the target was in view in the previous frame, the vision software searches for the object in a small region around the previous location. If the target was not in view in the last frame, the software scans the entire image for the infrared blob. In this sense, the object is actually not tracked, but simply reinitialized.

For the experiments described in this dissertation, the target is the only object in the workspace, and therefore target identification and recognition are straightforward. In order to accommodate multiple objects in the workspace, colored markers can be used to uniquely represent different objects. Color segmentation can be performed similarly to the simple segmentation described above. Background clutter and image noise introduced by the wireless connection become important limitations for such a system.

6.3 Estimator

The target-motion estimator runs at 30 Hz on a 750 MHz Intel Pentium III-based PC running RedHat Linux 7.2. Inputs to the program are the position, orientation, and velocity of the observer rover and the calibrated target track in relative cartesian coordinates. The estimator assumes there is a single constant-velocity target and that all incoming vision measurements belong to that target.

The estimator is initialized by using the first valid vision measurement. The initial bearing is calculated from this single measurement and the initial range is set to 0.5 meters. The range is intentionally initialized close to the observer since this will result in safer behavior. The target velocities are initially assumed to be

zero. The variance of the bearing estimate is initialized at the noise level of the vision system. Since the object must be in front of the camera, the range variance is initialized as the square root of the range estimate. The velocity variances are set by the velocity of the observer robot. In other words, it is initially assumed that the target could be moving as fast as the observer.

6.4 Trajectory Generator

The observer trajectory generator also runs at 30 Hz on a 750 MHz Intel Pentium III-based PC running RedHat Linux 7.2. It receives the estimated target state from the estimator and the observer data from the overhead vision system. It assumes a single, constant-velocity target. The current version of the trajectory-generation software takes approximately 1.5 seconds to compute a path comprised of six maneuvers. Trajectories are created by the generator and sent directly to the robot controller.

The planner and estimator currently run as separate threads on the same machine. Communication between them occurs over the same network backbone as the other processes. If necessary, the estimator and planner can be run on different computers.

6.5 Graphical User Interface

Two graphical user interfaces (GUIs) were developed for use with the micro-rover platform. These GUIs allow for drag-and-drop mouse commands of the observer and target robots. They display the robot trajectories, including both the past and future legs of the path, the true robot positions as measured by the overhead vision system, the estimated target position, and the error ellipse defined by the estimate error covariance.

The first GUI was developed using the Java programming language. It displays a top-down view of the workspace and uses simple graphical icons to represent the robots. Several buttons on the side of the main display window make system-wide commands. These commands include synchronizing system time, initializing the experiment, beginning the experiment, and returning the rovers to a home location. The Java GUI is shown in Figure 6.6.

The second GUI was developed as part of a separate research effort to design interfaces that allow a single user to control multiple robots [76]. The GUI uses a dialogue-based interaction between the robots and the GUI computer in order to display context-sensitive affordances -- in other words, options are presented to the human user only if the robot knows it is capable of performing them. The GUI was designed using

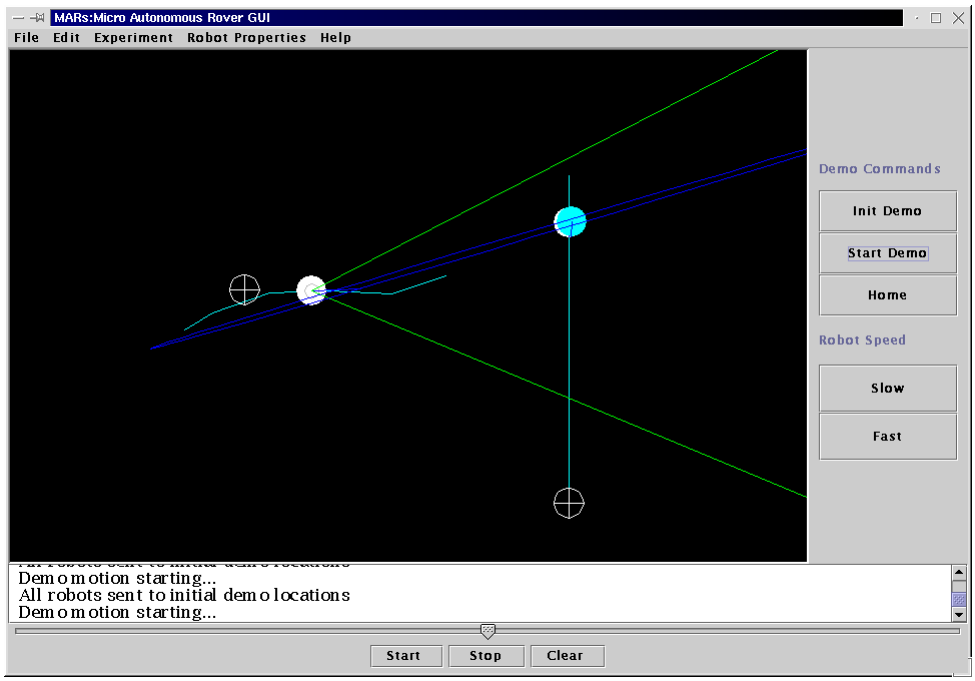

Figure 6.6 Java-based graphical user interface

OpenGL, and provides rendered graphics and multiple views of the workspace. In the context of this research, an overhead view and a 3-D view are both used.

6.6 Computing Environment and Network Backbone

The various software components of the experimental system described in this chapter were all developed using the ControlShell real-time programming environment invented at the ARL and developed by Real-Time Innovations (RTI). This package is a modular C++ based coding environment that enables programming of the low-level control loops and the higher-level strategic components of the system. The algorithms are designed through a graphical interface, and software modules are easily shared between different programmers.

The network backbone for this system is based on the Network Data Delivery System (NDDS) produced by RTI. This package uses a publish-subscribe architecture to create a virtual data bus. Each of the software components in this system publish data packets to the NDDS daemon which is responsible for redistributing the information back to the subscribing components.

6.7 Simulation

A rover simulation program was developed in order to allow hardware-in-the-loop testing of the various software components. The simulation program replaces the actual robots, the control computer, and the overhead vision system with differential equations that model their behavior. The robot-control simulation assumes perfect trajectory following, and outputs simulated overhead vision measurements for every robot.

A second program was developed in order to simulate the monocular-vision system. This program runs independently from the rover simulation and may be used with the actual robots. A pinhole camera model is used to transform the observer and target positions into image-plane measurements. Sensor noise is then added to the simulated vision measurement.

CHAPTER 7

Results -- Simulations and Experiments

The results of hardware-in-the-loop (HIL) simulations (the simulations are run using the actual experimental-system computers while the target and observer rovers are being simulated) and physical experiments are presented in this chapter to verify the performance of the new observer-trajectory generator. First, a series of HIL simulations illustrate the characteristics of the core trajectory-design algorithm for assumed-known initial target state, and example trajectories are presented for a typical observer-target scenario. In addition, the performance of the algorithm is verified by developing the relationship between planning time and estimation performance. The resulting trade-off curves show that near-optimal performance can be achieved quickly, and that to get significant additional improvement would require large amounts of additional planning time. The performance of the core trajectory-design algorithm is also compared against a naive, random approach.

Two additional sets of results demonstrate the success of the new observer-trajectory generator for unknown initial target motion. The first set of results are acquired by applying the trajectory generator to another HIL simulation of the target-motion estimation problem. Both the fixed-time, minimum-uncertainty and the fixed uncertainty, minimum-time objectives are demonstrated. The simulations show that the core trajectory-design algorithm for known target motion fails if applied directly to the problem of unknown target motion. However, the additional issues and extensions developed in Chapter 5 enable the successful estimation of the target motion with no *a priori* information regarding its location and velocity. Finally, a set of physical experiments using the micro autonomous rover platform described in Chapter 6 demonstrate the success of the new trajectory generator for a real target/observer system in experimental hardware.

7.1 Demonstration Scenario

Unless mentioned, the same initial observer-target scenario is used for all simulations and physical experiments presented in this chapter. The observer sits at the position (-1.10, 0.0) meters when it first encounters the target, positioned at (0.75, 0.70) meters with a constant velocity of (0.0, -0.015) meters per second. Figure 7.1 shows the geometry for this typical observer-target encounter.

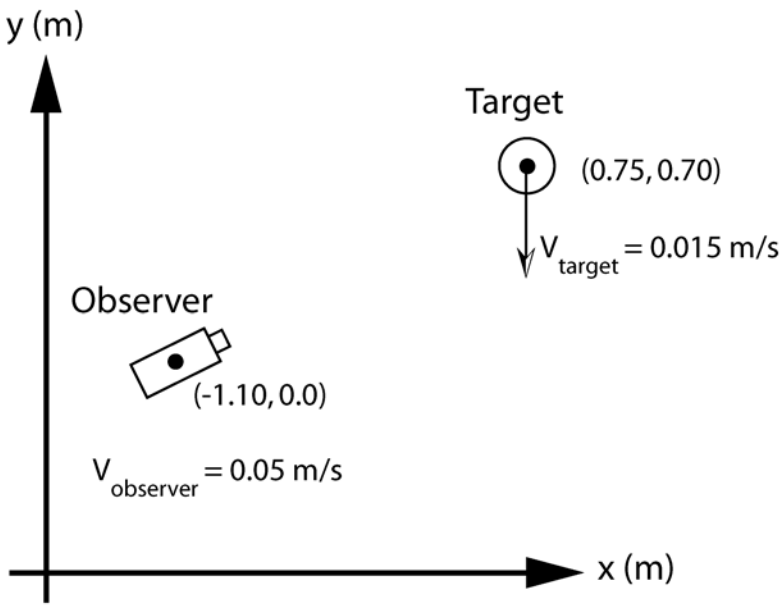

Figure 7.1 A typical observer-target encounter

Table 1 lists the important settings used by the trajectory-design algorithm. The observer speed, camera field of view, and noise variances characterize the capabilities of the experimental hardware described previously. The search depth, number of iterations, and size of the heading space are the default settings derived from the trade-offs which will be discussed in detail in Section 7.2.3. Finally, the maneuver time is selected such that the non-dimensional trajectory-length parameter K (Equation **(4.26)**) has a value of 0.75 for an initial range of 2.0 meters. With 5 maneuvers, this equals 30 seconds total trajectory duration.

Observer Speed (m/s)	0.05
Camera Field of View (rad)	0.35
Minimum Range (meters)	0.5
Sensor noise (rad rms)	0.0087
Process noise (m/s rms)	0.001
Number of Maneuvers	5
Number of Iterations	4
Number of Intervals in Heading Space	5
Maneuver Duration (secs)	6.0
Desired Area (m^2)	0.02

TABLE 1. Default Settings for Trajectory Generator

7.2 Core Observer-Trajectory-Design Algorithm Assuming Known Initial Target State -- Simulation

The trajectory-design algorithm assuming known initial target motion (Chapter 4) serves as the core of the new observer-trajectory generator (Chapter 5), producing near-optimal results in real-time. This section presents the results of a series of simulation runs using the trajectory-design algorithm, conducted using the same computer system used for the hardware experiments of Section 7.4. The first two simulation runs apply the trajectory-design algorithm to the standard demonstration scenario described above. Trajectories for the fixed-time, minimum-uncertainty and the fixed-uncertainty, minimum-time objective are generated and the features of these paths are described.

The third set of simulation results verify the near-optimal performance of the trajectory-design algorithm by developing trade-off curves for the three important design parameters: (i) the number of iterations, (ii) the number of maneuvers, and (iii) the number of discrete heading intervals. As each parameter increases, the planning time also increases while the final estimation performance improves. Each curve shows a distinct knee, after which increased estimation improvement comes at the expense of much longer planning times.

The final set of simulation runs compare the trajectory-design algorithm to a random approach. Again using the standard scenario, a set of 5000 random trajectories is generated and evaluated. These paths are compared against the results of the trajectory-design algorithm, which performs as well as the best randomly attained result. Furthermore, the 5000 random trajectories are used as the initial seed trajectories for the algorithm. In every case the algorithm calculates an improved trajectory (compared to the random initial seed trajectories). However, the default initial seed - heading straight toward the target - leads to the best result of them all.

7.2.1 Fixed time, minimum uncertainty

The trajectory-design algorithm is applied in simulation to the standard scenario described above using the default settings (Table 1). The fixed-time, minimum-uncertainty (FTMU) cost function is used to generate the observer path shown in Figure 7.2. The two curves correspond to the observer and target trajectories. The “X”s marked along the observer path denote the starting point of each maneuver, while the marks on the target path correspond to its location at the time of these maneuvers. The dashed line denotes the field of view of the camera at the final observer location. The small gray ellipse at the end of the target path represents the final uncertainty bound of the target’s position. For this run the initial ellipse area was reduced from 2.2 m^2 at the initial observer location to a final cost of 0.0012 m^2. The observer trajectory was created in 1.25 seconds.

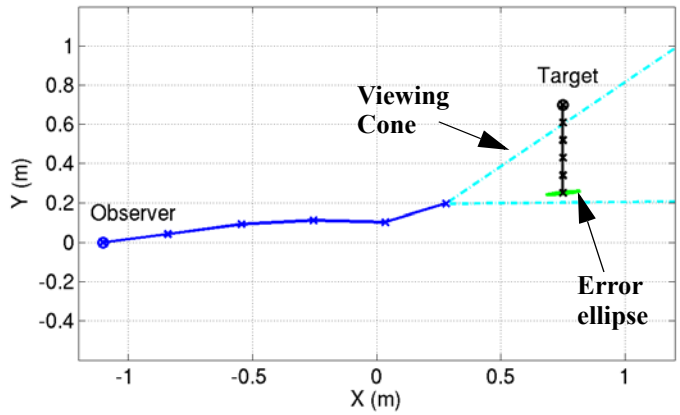

Figure 7.2 Results of trajectory design with known initial target state using the FTMU objective

There are three main features of the observer path. First, the observer moves toward the target throughout the path. As the range to the target decreases, the achievable bearing rate increases, allowing for wider triangulation and thus smaller error. Second, while moving toward the target, the observer also moves tangential to the line of sight to the target. This transverse motion creates the bearing change needed to observe the target properly. Third, in this example the observer did not turn away from the target. Decreasing the range to the target was more important than creating a large bearing change.

7.2.2 Fixed uncertainty, minimum time

The fixed-uncertainty, minimum-time cost is applied to the same scenario, using the same default settings, to produce Figure 7.3. The desired error ellipse area is 0.02 m^2. The trajectory design algorithm took approximately 0.04 seconds to produce a path that achieves an area of 0.019 m^2 in a time of 18.0 seconds. Compared to the FTMU result, this path bends towards the target much sooner. In the previous example the observer delays the benefits of a lateral move until the end of the path when it is closest to the target, and thus achieves a larger bearing rate. In this example the goal is to reduce the uncertainty as soon as possible, so a large change in bearing is desirable immediately.

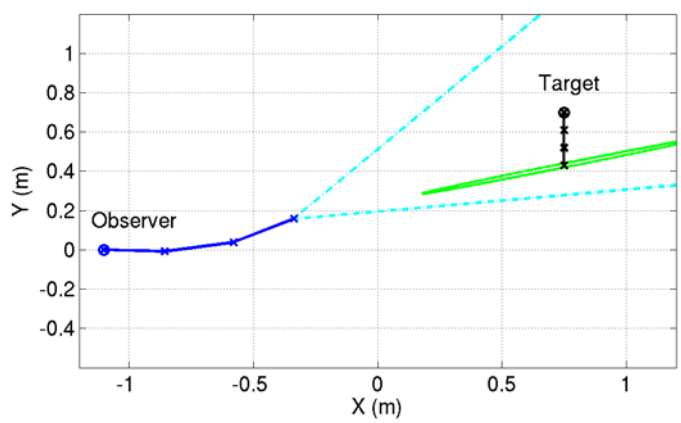

Figure 7.3 Results of trajectory design with known initial target state using the FUMT objective

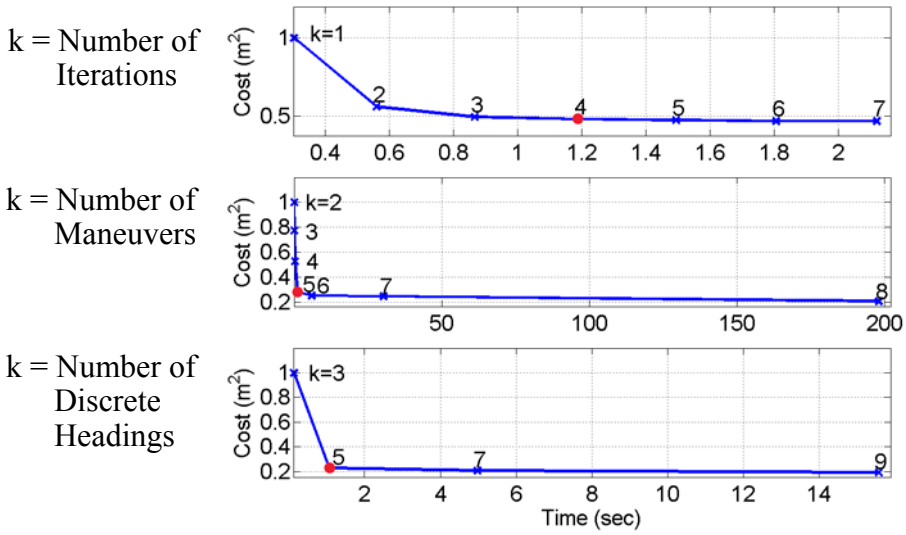

Figure 7.4 Optimization cost versus planning time for three key parameters: number of iterations, number of maneuvers, and number of discrete headings

7.2.3 Parameter Values

The performance of the trajectory-design algorithm depends on three important parameters. These parameters are the number of iterations performed by the pyramid approach, the number of maneuvers used to build the trajectory, and the number of discrete heading values used to define the search space. For each parameter, as the value of the number increases there is a trade-off between planning time and achievable estimation cost.

The relationship between final estimation cost and planning time for the trajectory-design algorithm is calculated by applying the algorithm to twenty randomly selected observer starting points while keeping the standard initial target position and velocity. For each starting point, three separate simulations are run keeping two of the design parameters fixed while the third is varied over a reasonable range. For example, the first run uses fixed values for the number of maneuvers and the number of heading angles, and varies the number of iterations from 1 to 7. The fixed parameters are always held at the default values specified in Table 1. The time taken to complete the trajectory design and the final optimization cost (the area of the final estimate uncertainty ellipse) are recorded, producing data curves for each of the 3 parameters. The costs are normalized by the value corresponding to the lowest parameter in order to compare the data across the random runs. After all twenty runs are performed, the curves for each parameter are averaged together to produce the final results.

Figure 7.4 shows plots of the averaged optimization cost versus planning time as the design parameters are varied. The x-axis of each plot measures the planning time while the y-axis measures the normalized

optimization cost (the area of the final estimate uncertainty ellipse divided by the area corresponding to the lowest parameter value). Note that the scales of each axis are different for each plot. The marks on each curve correspond to the data points for each particular parameter value, and this parameter value is labelled. The circular marks correspond to the default values derived from these plots and listed in Table 1. As expected, the general trend of each trade-off curve is for the optimization cost to decrease as the parameter and corresponding planning time increase. In each case, a larger parameter value leads to a larger candidate trajectory set. In turn, the larger candidate set requires increased computational time but produces a lower final optimization cost.

The distinct knee in each trade-off curve has important implications for the overall performance of the trajectory-design algorithm. The curves show that significant performance is achieved at low parameter values with low planning times, and that beyond a certain point planning time is wasted for little additional reduction in optimization cost. This behavior is most strikingly illustrated by the middle plot which is derived by varying the number of observer maneuvers. The normalized optimization cost decreases from 1.0 to approximately 0.3 as the parameter value increases from $n_{man} = 2$ to $n_{man} = 5$ while the planning time increases to 1.5 seconds. Increasing the number further from 5 to 7 yields no perceptible change in cost but increases the planning time more than an order of magnitude. Increasing further to $n_{man} = 8$ improves the cost to 0.2 m^2 but now requires more than 3 minutes of planning time. The other two curves demonstrate similar behavior. Increasing the number of iterations has the smallest increase in planning time, but also leads to very little increased performance. The number of discrete headings can reduce the cost only from 0.25 m^2 to 0.2 m^2 at the expense of at least several seconds.

By choosing parameter values at the knee of each curve, the trajectory-design algorithm produces near-optimal trajectories in real-time. The true optimal results cannot be computed for target-motion estimation using monocular vision with the methods presented in this work. Instead, the trajectory-design algorithm described in this dissertation could be applied with a large number of maneuvers and a larger number of discrete headings. However, computing requirements in terms of speed and memory become increasingly prohibitive for maneuver numbers larger than eight and discrete heading numbers larger than nine, and such a test could not be performed. Nevertheless, the results displayed in Figure 7.4 suggest that the true optimal will be only incrementally better than the trajectories generated by the algorithm developed here.

7.2.4 Random Trajectories

An alternative method for judging the optimality of the new trajectory-design algorithm is to compare it to other algorithms. To that end, as one example the results of the trajectory-design algorithm are compared to a naive, random approach that uses a set of 5000 randomly created paths (Figure 7.5). For the standard

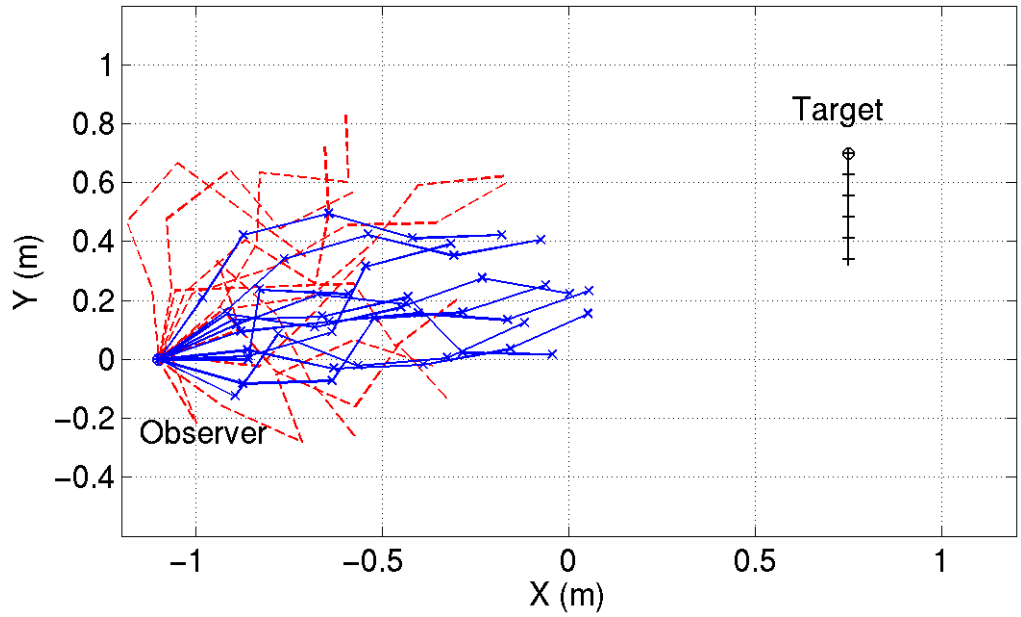

Figure 7.5 Random initial trajectories and improved paths that result from using them as initial seeds

scenario, the paths are generated by randomly selecting the heading of each maneuver from the uniform distribution

$$\theta_{target} - 90° < \theta_i < \theta_{target} + 90° \tag{7.1}$$

where θ_i is the heading of the ith maneuver and θ_{target} is the heading to the initial target location. The FTMU cost of each path is calculated, and the resulting cost distribution ranges from 0.0043 m^2 to 30.09 m^2 with a mean of 11.92 m^2 and a standard deviation of 13.69 m^2. Recall for comparison that the result of the trajectory-design algorithm was a path that yielded a final cost of 0.0012 m^2.

The performance of the new trajectory-design algorithm can also be compared to results using different initial trajectories to seed the heading discretization. The default initial seed path follows the line of sight from the observer position to the initial target location (Section 4.4.1). For comparison, the trajectory-design algorithm is called using each of the random paths described above as the initial seed trajectory, and for every starting condition, the algorithm produces an improved result. The new cost distribution has a range of 0.0012 m^2 to 6.01 m^2 with a mean of 0.0207 m^2 and a standard deviation of 0.24 m^2. Figure 7.5 shows a small sample of the random paths (the dashed lines) and the improved paths (the solid lines). While the final paths all vary greatly because of their initial conditions, they all exhibit the same general pull towards the target.

The results of the two sets of simulation runs presented in this section reinforce the near-optimality of the trajectory-design algorithm. None of the 5000 randomly created trajectories had lower costs than the

trajectory-design algorithm. Furthermore, of the paths that resulted from the 5000 different initial seeds (which corresponds to evaluating approximately 60 million possible trajectories), none produced significantly better costs than using the default seed. The fact that there are multiple improved paths in Figure 7.5, and not one single optimal path for every initial condition, indicates that the trajectory-design algorithm is capable only of finding a locally optimal path. However, the local optima all show great improvement over the random initial trajectories, and the local optimum that results from the default seed approaches the global optimum

7.3 Full Observer-Trajectory Generator for Unknown Initial Target Motion -- Simulation

The full observer-trajectory generator developed in Chapter 5 extends the trajectory-design algorithm presented in Chapter 4 to address unknown initial target motion. The success of this new generator is demonstrated through a set of runs conducted in hardware-in-the-loop simulation of the micro autonomous rover platform described in Chapter 6. Section 2.4 describes the observer-vehicle kinematics and Section 2.2 describes the pinhole camera model used in the simulation. The simulation runs all begin with the same standard observer-target scenario of the previous section; however the true initial target state is now unknown to the observer. Instead, the observer begins with a single vision measurement of the target and must use the target-state estimate as it moves.

Two different simulation runs demonstrate the success of the new observer-trajectory generator. The first applies the fixed-time, minimum-uncertainty objective to the trajectory generator in order to generate the observer trajectories, while the second run applies the fixed-uncertainty, minimum-time objective. The first several steps in the trajectory-design process are common to both runs -- i.e. the estimator initialization and the initial zig-zag maneuver -- and are therefore discussed in detail only in the context of the first simulation run. Furthermore, the generator maintains a periodic replan rate of 0.5 Hz for both runs.

Two aspects of the new trajectory generator discussed in Chapter 5 are not shown in any of the simulation runs presented in this chapter. First, the runs are all conducted with a fixed, periodic update rate rather than monitoring the differences in the states between the predicted and current estimates. Future simulations or experiments can illustrate the performance of the trajectory generator under this alternative scheme. Furthermore, additional research is needed to develop the trade-offs between planning costs and performance using this other approach and to understand what approach is best suited for various situations. Second, the observer never loses the target in any of the simulations or experiments so the reacquisition

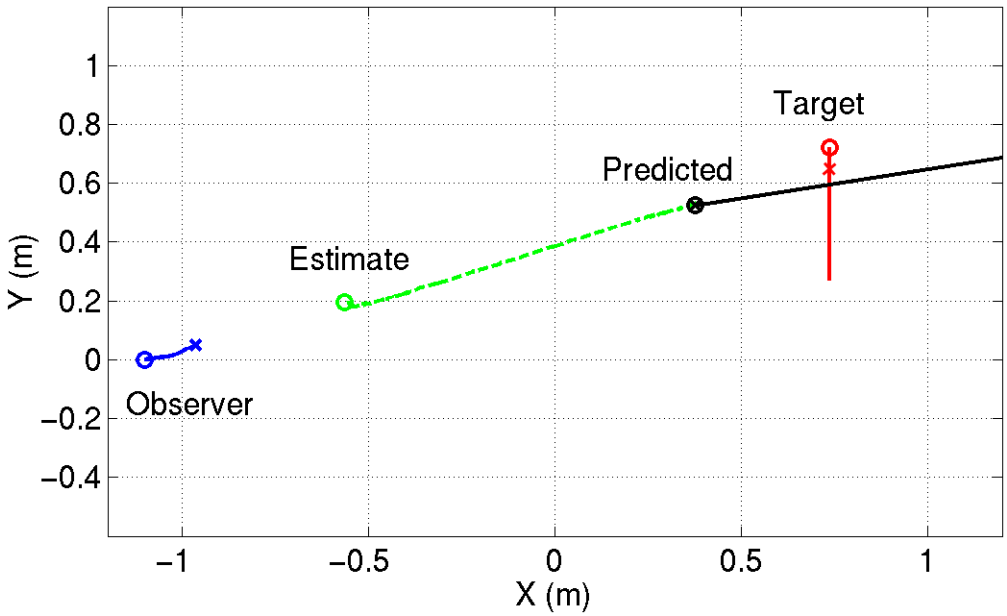

Figure 7.6 State of the world after the initial maneuver

component never activates. More complex observer-target scenarios, less accurate sensors, or worse estimator design should probably be used in order to test the reacquisition components of the system.

7.3.1 Fixed time, minimum uncertainty

The first simulation run uses the fixed-time, minimum-uncertainty objective to generate the observer trajectory. The run begins when the computer vision system first senses the target. The target-motion estimator is initialized, the initial zig-zag maneuver is performed, and the core trajectory-design algorithm is then called at two-second intervals as the observer continues along its path (Figure 7.7). After the specified total duration of thirty seconds the experiment concludes, and the performance of the target-motion estimation is judged.

Initial Maneuver

Upon receiving the first computer-vision measurement, the trajectory generator activates its first component. A new target-motion estimator is initialized using the bearing measurement and an initial range (as described in Section 5.2) of 0.5 meter, although the actual initial range is about 2.0 meters. The location of the initial target estimate is shown on Figure 7.6 by the circle labelled “Estimate”. After initializing the estimator, the generator commands the initial zig-zag maneuver denoted by the solid line labelled “Observer” in the figure. The dashed line depicts the evolution of the target position estimate as the observer moves. Over the course of the initial maneuver the position error is reduced from 1.5 meters to 0.35 meters. The estimate at this point is marked by the “X” within the circle at the end of the dashed line. The true target location that corresponds to this instant is also marked with an “X”. The velocity error at the

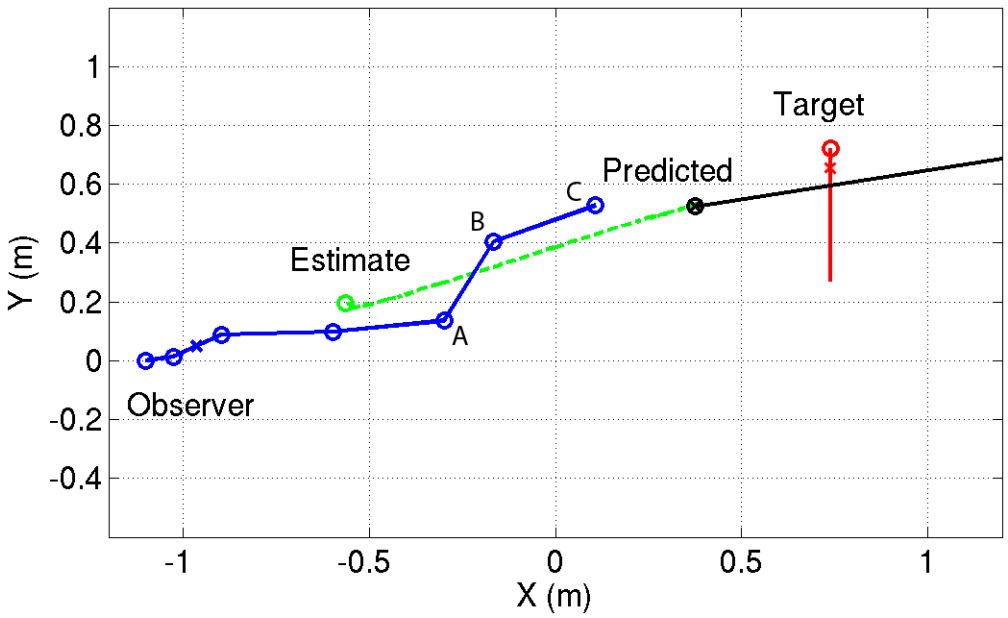

Figure 7.7 First commanded observer trajectory after the initial maneuver

end of the maneuver, however, remains high with an estimate error of [0.042 m/s, 0.023 m/s]. The solid black line labelled "Predicted" represents the predicted target path calculated from the target estimate at the end of the initial maneuver, and the trajectory-design algorithm is now called using this predicted target path as input

Invoke Trajectory-Design Algorithm

After the initial maneuver, the observer-trajectory generator invokes the trajectory-design algorithm for the first time. An estimate of the state of the originally unknown target has been created, and the initial maneuver has provided information to reduce the estimate uncertainty. Since the estimate describes the sum total of the observer's knowledge, the trajectory-design algorithm uses this estimate in order to produce the first complete commanded observer trajectory. For the simulated run presented here, the target estimate has a value of [0.37 m, 0.52 m, 0.042 m/s, 0.008 m/s] and the observer is located at [-0.96 m, 0.049 m] when the trajectory-design algorithm is invoked. Figure 7.7 shows a plot of the resulting trajectory overlaid on the previous figure showing the actual target path, the predicted target path used to generate the trajectory, and the evolution of the target estimate from the beginning of the run to the instant the trajectory-design algorithm is first called.

The first observer trajectory produced by the design algorithm demonstrates the need for repeated replanning. Even before the trajectory was produced, the difference between the actual and predicted target paths suggest improving the trajectory as the observer moves. The predicted path, which represents the only knowledge the observer has about the target, has the target quickly travelling away from the observer in the

positive x-direction. By comparison, the actual target moves in front of the observer in the negative y-direction. The expected optimal observer behavior varies greatly for either situation, and basing the observer's trajectory on the predicted rather than actual target path does not seem promising. This intuition is confirmed by the trajectory that results for the predicted path. The first two maneuvers for this path are similar to the early maneuvers for the trajectory based on the actual path (Section 7.2.1). However, the final two maneuvers indicate trouble. The trajectory segment AB turns the observer away from both the predicted and actual target locations in order to move upward and create a large change in bearing. Segment BC is intended to bring the target back into view. Unfortunately the predicted target path differs from the actual target path to such a degree that the observer would lose the target. Furthermore, because the target would be out of view, the estimator would not gain any information in the final stages of the trajectory when the observer is closest to the target and the information content should be greatest. As a result, the final estimation performance would be poor. As the next runs will show, the replanning and reacquisition components of the trajectory generator solve this dilemma imposed by the initial predicted path and lead to greatly improved final performance.

Replanning

After the trajectory-design algorithm is invoked for the first time, it is periodically recalled every 2 seconds throughout the remainder of the simulation run. Because the target estimate now converges as the observer moves, the predicted target path changes every time the algorithm is called, and the observer trajectory continually changes as well. The final observer trajectory is shown in Figure 7.8 as the solid line with the

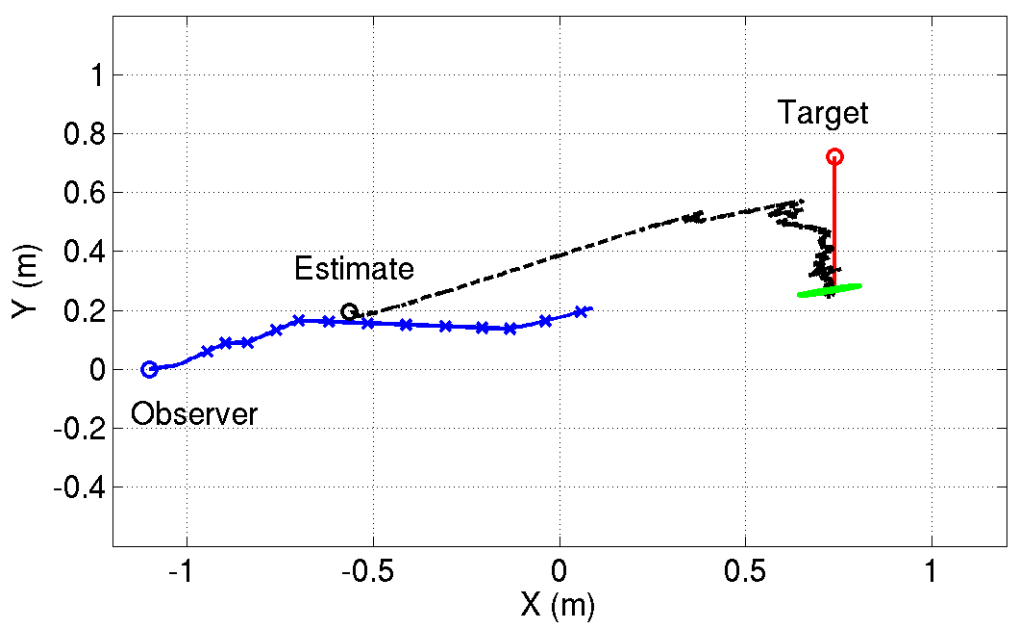

Figure 7.8 Observer trajectory with unknown initial target state using the fixed-time, minimum-uncertainty objective

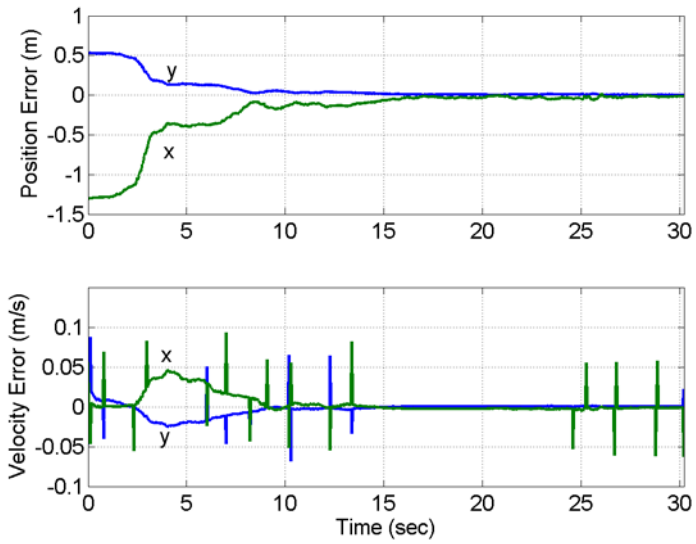

Figure 7.9 Estimate error versus time

"X" markings. These marks indicate the location of the observer each time the trajectory generator is called. The dashed line depicts the target position estimate as it converges over time, the lighter solid line depicts the actual target path, and the gray ellipse at the end of this line represents the final error covariance ellipse. Figure 7.8 shows the convergence of the target-state estimate to the true target path, shows the error ellipse depicting low final uncertainty, and verifies the ability of the trajectory generator -- with periodic replanning -- to enable successful target-motion estimation

The final uncertainty in the target-motion estimate represents the performance of the target-motion estimation and the new observer-trajectory generator. Figure 7.9 contains plots of the target-state estimate error versus time. The final target-state estimate is [0.72 m, 0.27 m, -0.0012 m/s, -0.015 m/s] for an error of [0.014 m, 0.0024 m, 0.0012 m/s, 0.0003 m/s]. The final estimate error ellipse area is 0.0015 m^2, and Figure 7.10 shows a plot of the error ellipse versus time. Unlike the initial observer trajectory, the final trajectory kept the target in view throughout the entire path, and the plot in Figure 7.10 illustrates the large reduction in estimate uncertainty achieved in the final stages.

The observer trajectory produced during this simulation run compares well with the trajectory designed for the known, true target motion. Figure 7.11 shows the observer trajectories for both known and unknown initial target states. The line with the triangular marks represents the trajectory generated for the true target. Both trajectories exhibit the same general shape -- they progress toward the target with two slight bends. The trajectory for the unknown target does not reach as far as the path for the true target. The reason for this behavior is due to lost distance whenever the observer turns to start a redesigned path. Typically a redesigned trajectory continues in a different direction from the preceding path. Every time the observer performs a maneuver or adjusts its course, time that could be spent traversing toward the target is spent

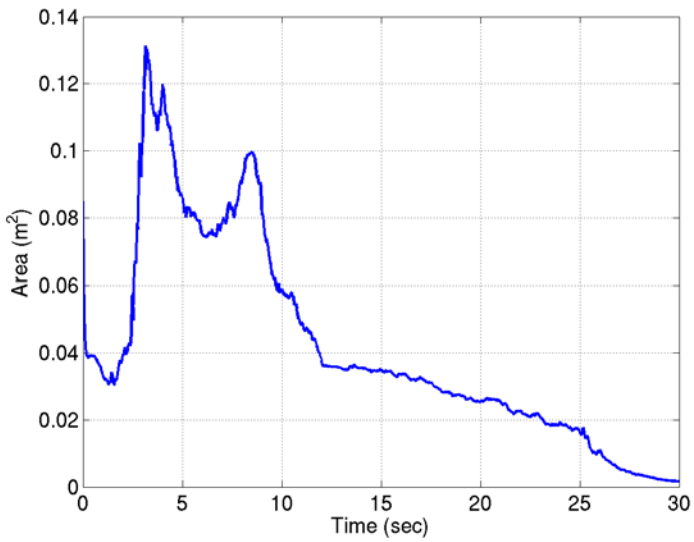

Figure 7.10 Error ellipse area versus time

changing course. For the fixed-time, minimum-uncertainty objectives, the observer trajectory for a known initial target state performs five maneuvers while the trajectory for unknown initial target state changes course with every redesign, i.e. thirteen times.

Path Completed

The results of the simulation run presented in this section demonstrate the success of the new observer-trajectory generator using the fixed-time, minimum-uncertainty objective. The initial maneuver (which is

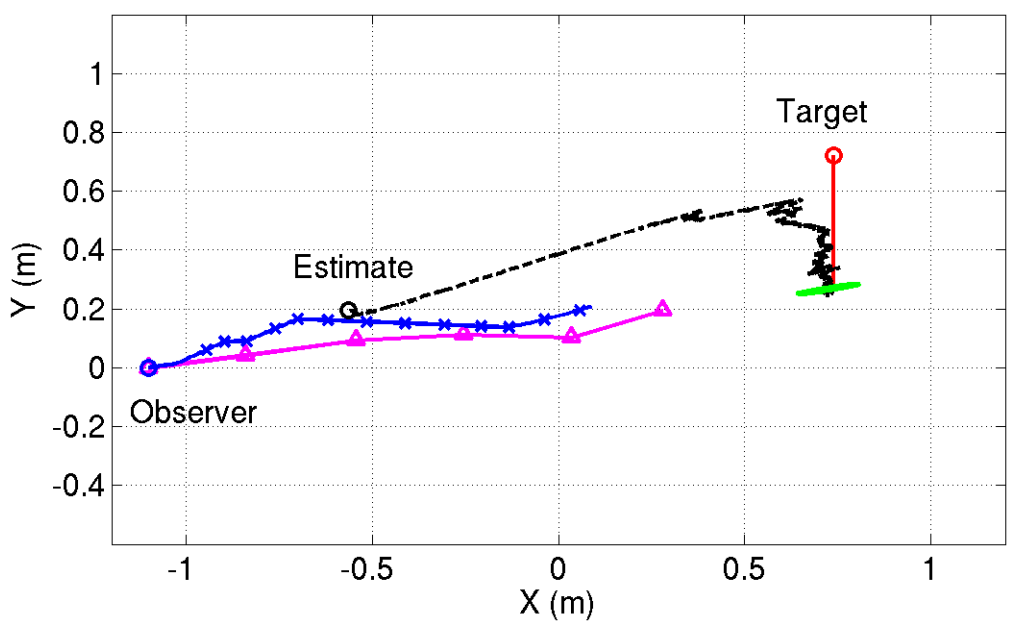

Figure 7.11 Observer trajectory for known ('Δ') initial target state overlaid on results for unknown initial state using the FTMU cost objective.

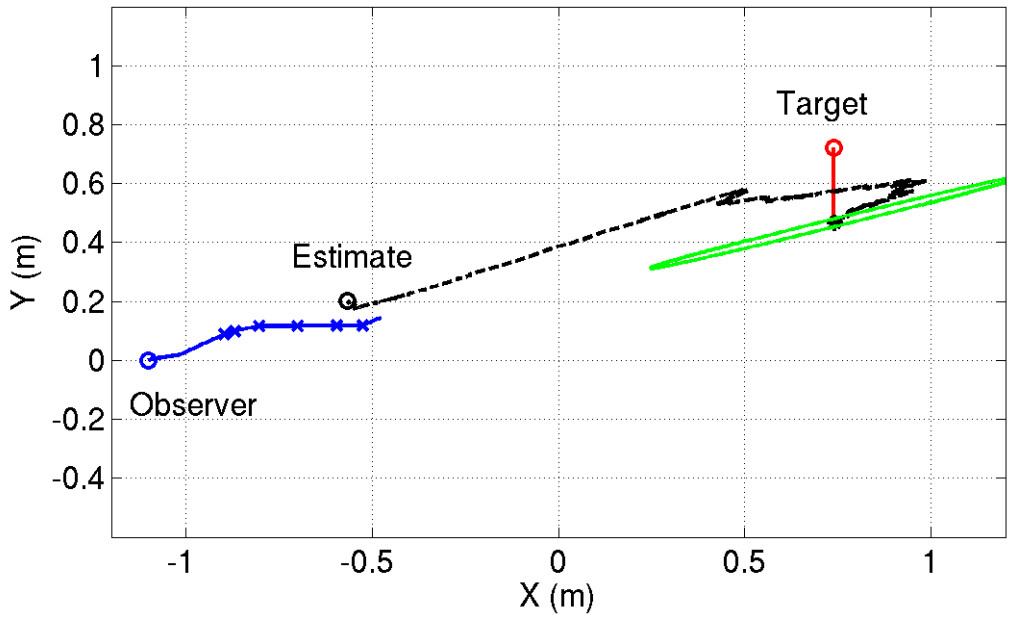

Figure 7.12 Observer trajectory for $A_{des} = 0.02$ m^2

identical for the fixed-uncertainty, minimum-time objective as well) significantly improves the target-state estimate before the core trajectory-design algorithm is called. However, uncertainties in the target-state estimate would still lead to poor performance without the additional components of the new trajectory generator, particularly replanning based on the new estimate of target motion.

7.3.2 Fixed uncertainty, minimum time

Trajectory design using the fixed-uncertainty, minimum-time objective is demonstrated in this dissertation for the first time ever. This second simulation run uses the FUMT objective to design trajectories that achieve a desired estimate-error-ellipse area of 0.02 m^2. Like the design with the fixed-time, minimum-uncertainty objective, the first steps in the trajectory design process are to initialize the estimator and perform the initial zig-zag maneuver. The results of these steps are identical using either objective, and are described in Section 7.3.1. After the initial maneuver completes, the trajectory-design algorithm is called using the FUMT objective and the method continually replans -- also at two second intervals -- until the desired area is reached.

Figure 7.12 shows a plot of the resulting observer trajectory and the evolution of the target-state estimate. The solid line denotes the final observer trajectory and the "X"s mark the observer location at the time of each redesign. The dashed line displays the target-position estimate, the gray ellipse around the final estimate represents the error uncertainty bound, and the light solid line indicates the true target path.

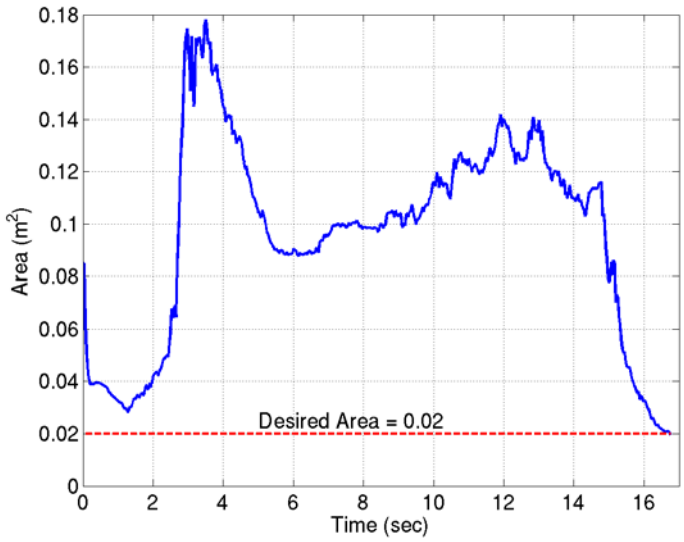

Figure 7.13 Ellipse error versus time

In this run the observer takes 16.8 seconds to achieve the final error ellipse area of 0.02 m^2. Figure 7.13 shows the area of the error ellipse as a function of time. The jump in uncertainty near the three-second mark occurs when the observer rotates in order to change course. Large uncertainty in the velocity estimate of the target at this point feeds into the position uncertainty as well. The slower increase in uncertainty from 6 to 12 seconds corresponds to the straight segment of the trajectory during which velocity observability is low and the velocity error again drives the position uncertainty. The large decrease in uncertainty at the end of the trajectory results from the final maneuver which provides sufficient observability in all the states to reduce the error ellipse size to the desired level.

Figure 7.14 depicts plots of the estimate errors versus time. The large velocity errors near the three-second mark are clearly visible as well as several large spikes. These error spikes correspond to the turning phase of the observer maneuvers. The target-motion estimator does not properly fuse the observer motion at these times. The final target-state estimate at the end of this run is [0.74 m, 0.46 m, 0.054 m/s, 0.042 m/s] for a final estimate error of [-0.0050 m, 0.0026 m, -0.054 m/s, -0.057]. The velocity errors are extremely high for this run, even though the desired ellipse area was achieved. The reason for this discrepancy is that the error ellipse describes only the position components of the target-state estimate. Because the target is kept in view throughout the run, the position errors are reduced at a higher rate than the velocity errors. The use of the four-dimensional error ellipsoid or other covariance matrix measures could resolve this dilemma in future experiments. These large velocity errors do not arise in the hardware demonstrations presented in the final section of this chapter.

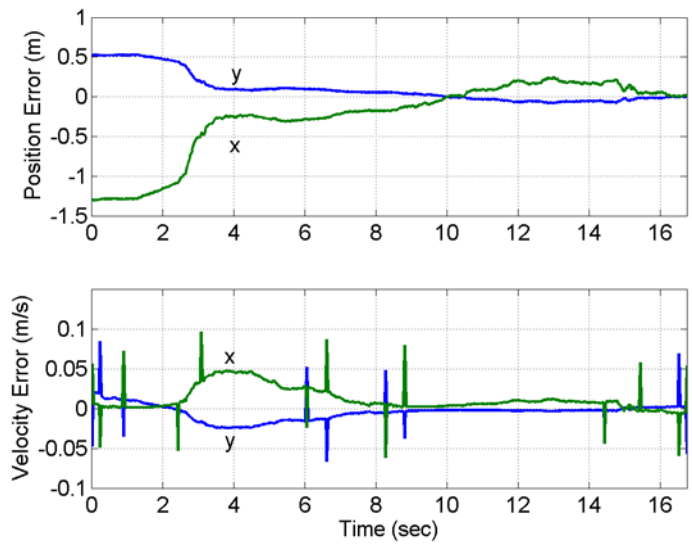

Figure 7.14 Estimate error versus time for the fixed uncertainty, minimum time objective

The observer trajectory designed in this trial can also be compared to the trajectory generated when there is known initial target state. Figure 7.15 shows the resulting trajectories for the cases of both known and unknown initial target state. Unlike the results of the previous simulation run, the trajectories with known and unknown initial target states do not look similar. The observer trajectory for the unknown initial target state performs the initial maneuver which draws the observer towards the target. By comparison, the observer trajectory for the known initial target state immediately begins to move laterally to the target line of sight in order to achieve the bearing change needed for quick triangulation.

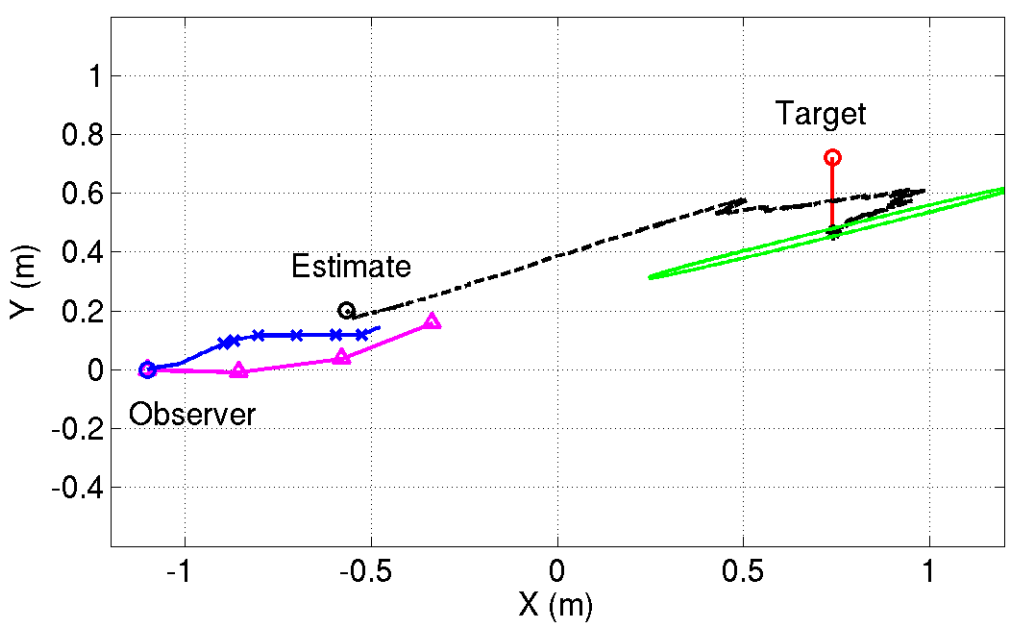

Figure 7.15 Observer trajectory for known ('Δ') initial target state overlaid on results for unknown initial state using the FUMT cost objective.

The second striking difference between the paths is their overall length and duration. The observer path with known initial target motion takes 18.0 seconds and takes full advantage of this knowledge in order to move as close to the target as possible. On the other hand, the observer path for the unknown initial target state requires only 16.8 seconds and spends some of the time while turning. At first glance this result seems contradictory, the unknown-target-motion result performs better than if the target motion is known in advance. However, there are two important facts that resolve this apparent contradiction. First, the observer path for known initial target motion fully completes each maneuver and achieves a better than requested ellipse area of 0.019 m^2 while the observer trajectory for the unknown initial target-state terminates as soon as the desired area is reached. Second, the trajectory-design algorithm assumes measurements are taken only immediately before each maneuver while the estimator ran continuously during the unknown-target-motion simulation. If the observer followed the observer path for known target motion while running the target-motion estimator, the final estimate uncertainty area would be smaller than expected and the longer path would appear better.

7.4 Micro Autonomous Rover Experiments

In this section a set of physical experiments demonstrate the success of the trajectory generator on the micro autonomous rover platform described in Chapter 6. As in the preceding sets of simulations, the initial target motion is unknown and the observer-trajectory generator is invoked once the computer vision system first senses the target. The results for the experiments on the rover platform closely follow those of the simulated system presented above. However, sensor noise in the overhead vision system used to calculate the observer navigation data, errors in the real observer control system, and noise in the computer vision sensor cause the hardware experiments to differ slightly from the simulated results.

The first experiment uses the fixed-time, minimum-uncertainty objective to generate the observer trajectory, while the second uses the fixed-uncertainty, minimum-time objective. In both cases the observer successfully performs the target-motion estimation by reducing the estimate errors and the final estimate uncertainty. The results of these experiments demonstrate for the first time the success of an observer-trajectory generator with unknown initial target state on an operational system. Furthermore, they also demonstrate the first example of trajectory design using the fixed-uncertainty, minimum-time objective.

7.4.1 Fixed Time, Minimum Uncertainty

Using the fixed-time, minimum-uncertainty objective, the observer-trajectory generator guides the observer rover to estimate successfully the target with unknown initial state. Figure 7.16 contains a composite image

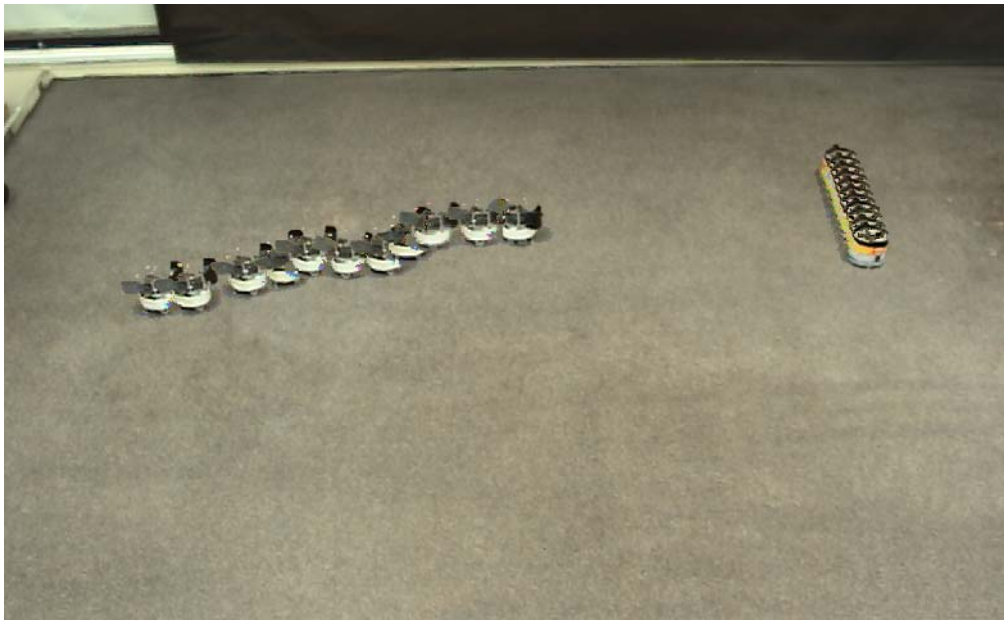

Figure 7.16 Composite image of the observer and target rovers during the experiment using the fixed-time, minimum-uncertainty objective

of the observer and target rovers over the course of the experiment. The image shows the rover positions at three-second intervals throughout the experiment. Figure 7.17 depicts plots of the final observer trajectory, the true target path, and the target-position estimate as it converges to the true path. The "X" markings indicate the observer position for each redesign of the trajectory. Figure 7.18 contains a sequence of plots showing the observer, target, and estimate of the target positions as well as the current observer trajectory, the vision sensor field of view, and the error ellipse.

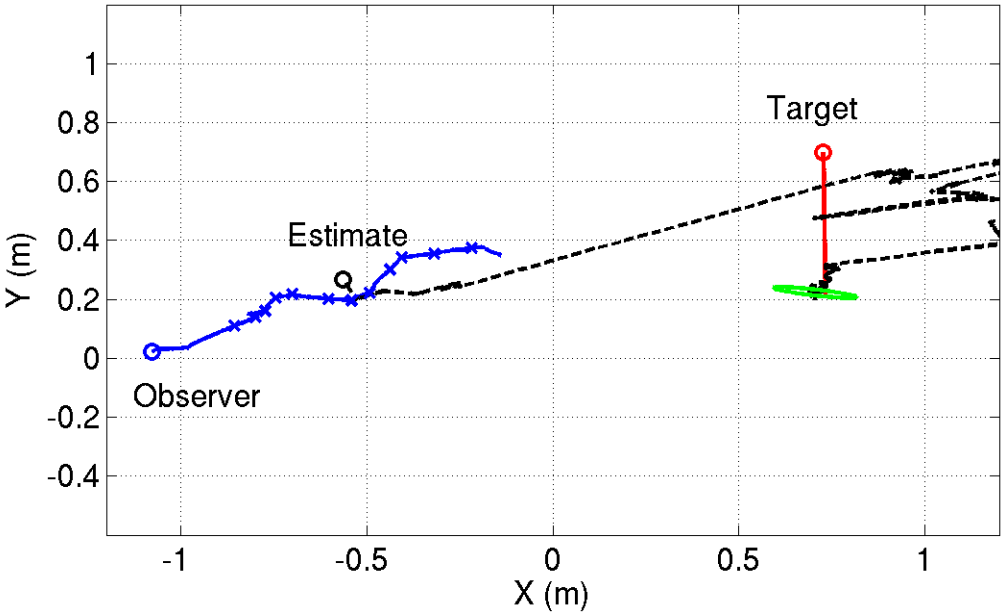

Figure 7.17 Observer trajectory using the fixed time, minimum uncertainty objective

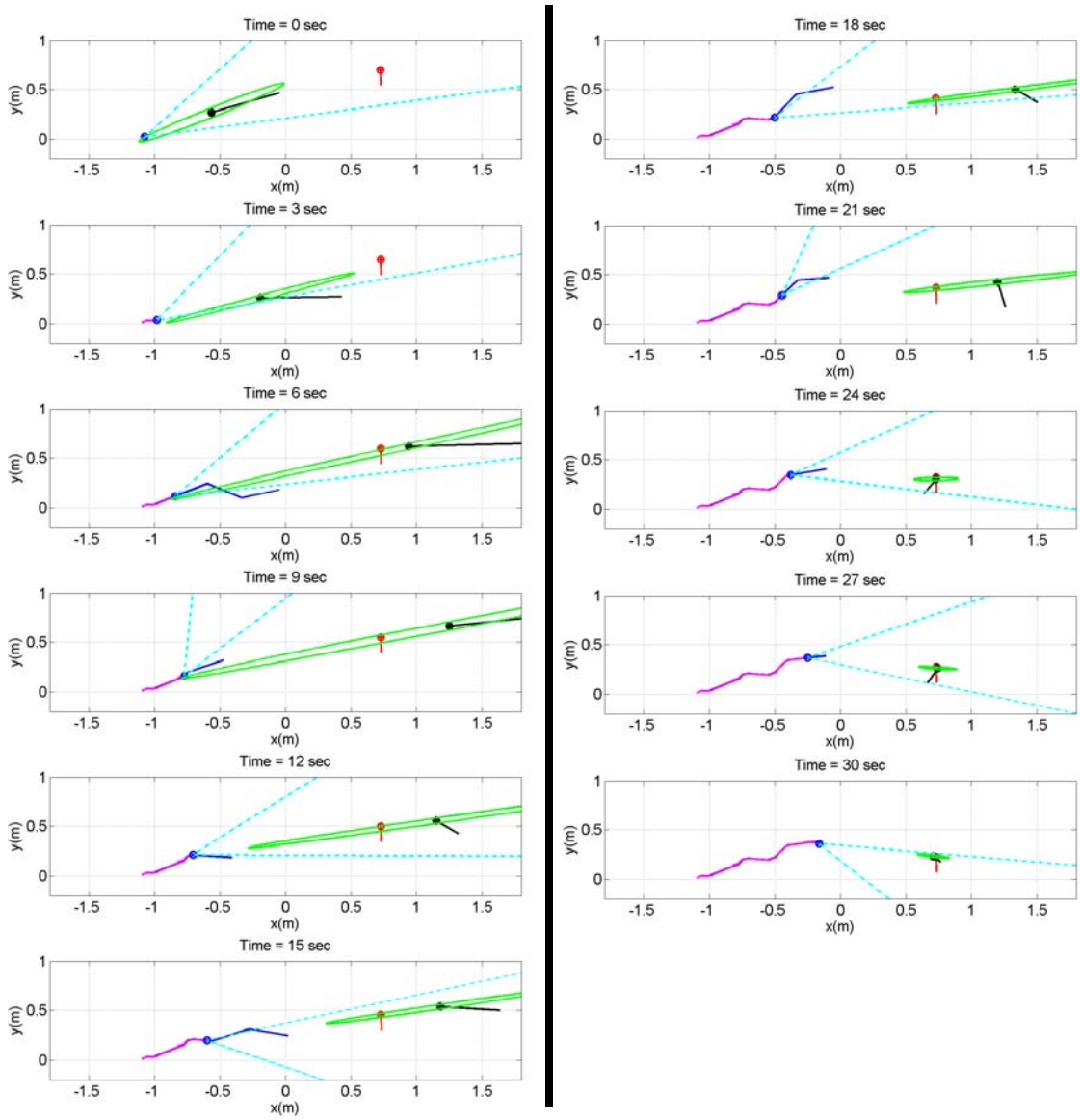

Figure 7.18 Time sequence of the micro rover experiment using the FTMU objective

Compared to the earlier simulation (Section 7.3.1), in the physical experiment the target estimate overshoots the true target path more. This large transient is due to two factors. First, the sensor and control noise mentioned above cause the target estimate to behave differently than the simulated case with perfect sensing. Second, and more dominant, the observer actually turns away from the target twice during the trajectory. With the target out of view, the velocity errors dominate and the position estimate drifts away from truth. Once the observer turns back and brings the target back into view (at 12 seconds and again at 24 seconds) the target-state estimate quickly converges. This convergence illustrates the increased benefit gained by the change in bearing combined with the detriment caused by letting the target out of sight.

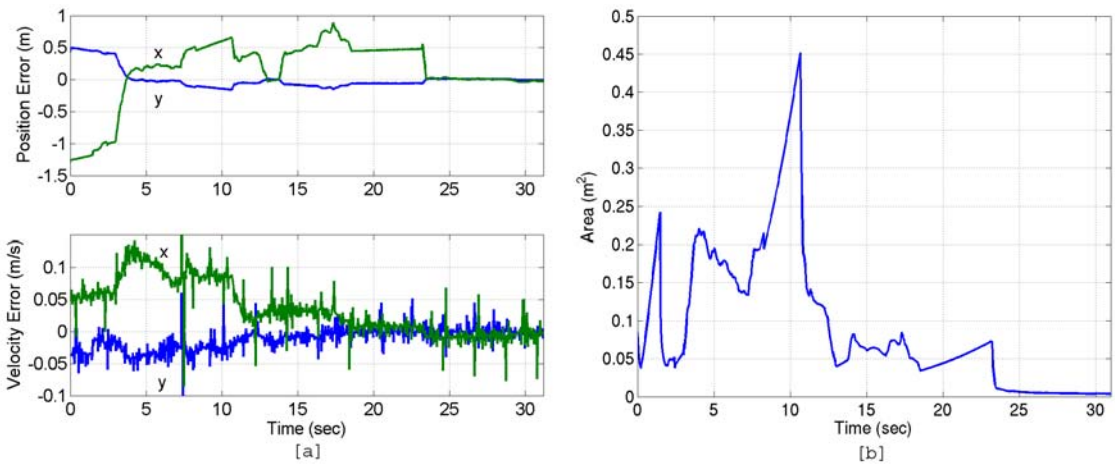

Figure 7.19 Target estimate performance for the micro rover experiment using the FTMU objective: [a] Estimate error versus time. [b] Error ellipse area versus time.

At the end of the thirty-second trajectory, the observer achieves a final error ellipse area of 0.0033 m^2. The final target estimate is [0.70 m ,0.22 m, -0.0030 m/s, -0.013 m/s] and the final estimate error is therefore [0.028 m, -0.0076 m, 0.0030 m/s, -0.0020 m/s]. Figure 7.19 depicts plots of the estimate errors and the error ellipse area versus time. The quick final convergence of the target estimate is evident near the 24 second mark of each plot.

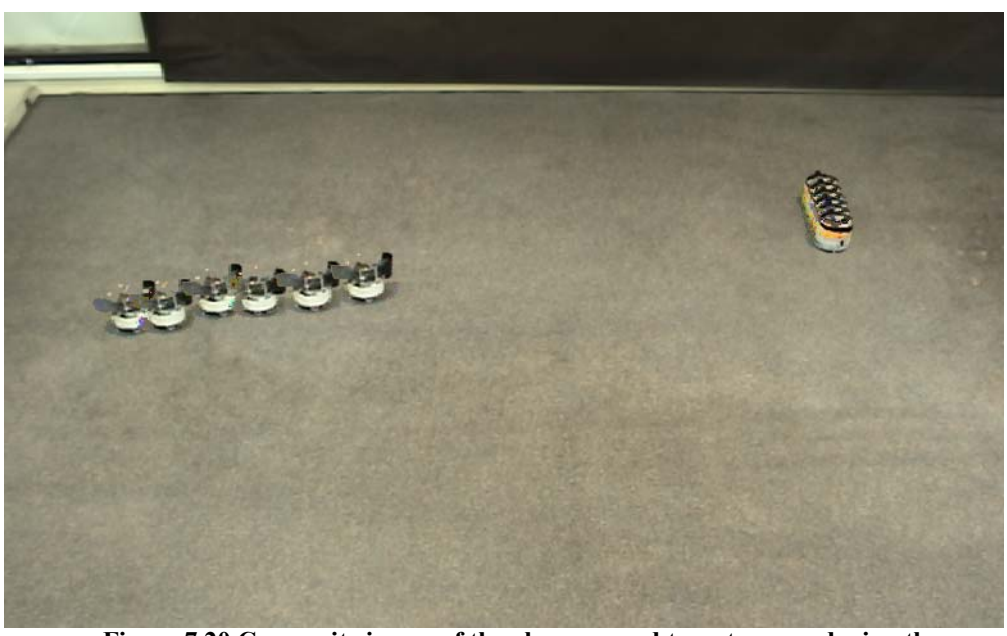

Figure 7.20 Composite image of the observer and target rovers during the experiment using the fixed-uncertainty, minimum-time objective

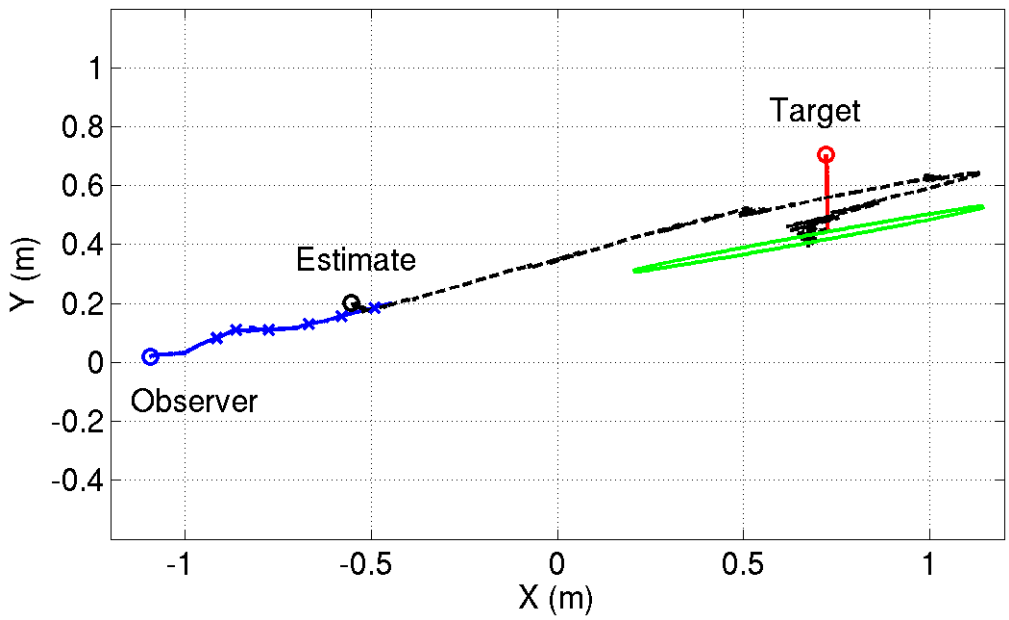

Figure 7.21 Observer trajectory using the fixed uncertainty, minimum time objective

7.4.2 Fixed Uncertainty, Minimum Time

The second experiment with the micro autonomous rover platform demonstrates the success of the observer-trajectory generator using the fixed-uncertainty, minimum-time objective. Figure 7.20 contains the composite image of this experiment, showing the observer and target positions every two seconds. Starting from the same standard scenario of the previous simulations and experiments, the initially unknown target location is estimated to a desired uncertainty level of 0.02 m^2. Figure 7.21 shows plots of the observer trajectory, the true target path, and the target-position estimate. As usual, the "X" marks indicate the observer location at the instant of every trajectory redesign. Also, Figure 7.22 contains the time sequence of plots taken every 2.0 seconds.

The observer achieves a final uncertainty-ellipse area of 0.018 m^2 in 16.0 seconds. The final target-state estimate is [0.67 m, 0.42 m, -0.0098 m/s, -0.024 m/s] and the final estimate error is [0.051 m, 0.032 m, 0.0098 m/s, 0.0088 m/s]. Compared to the simulation run using the same objective (Section 7.3.2), the target-state estimate again overshoots the true target path more. However, the target stays in view the entire time, and the overshoot is not nearly as bad as the previous example. Figure 7.23 contains the plots of the estimate error and uncertainty-ellipse area versus time.

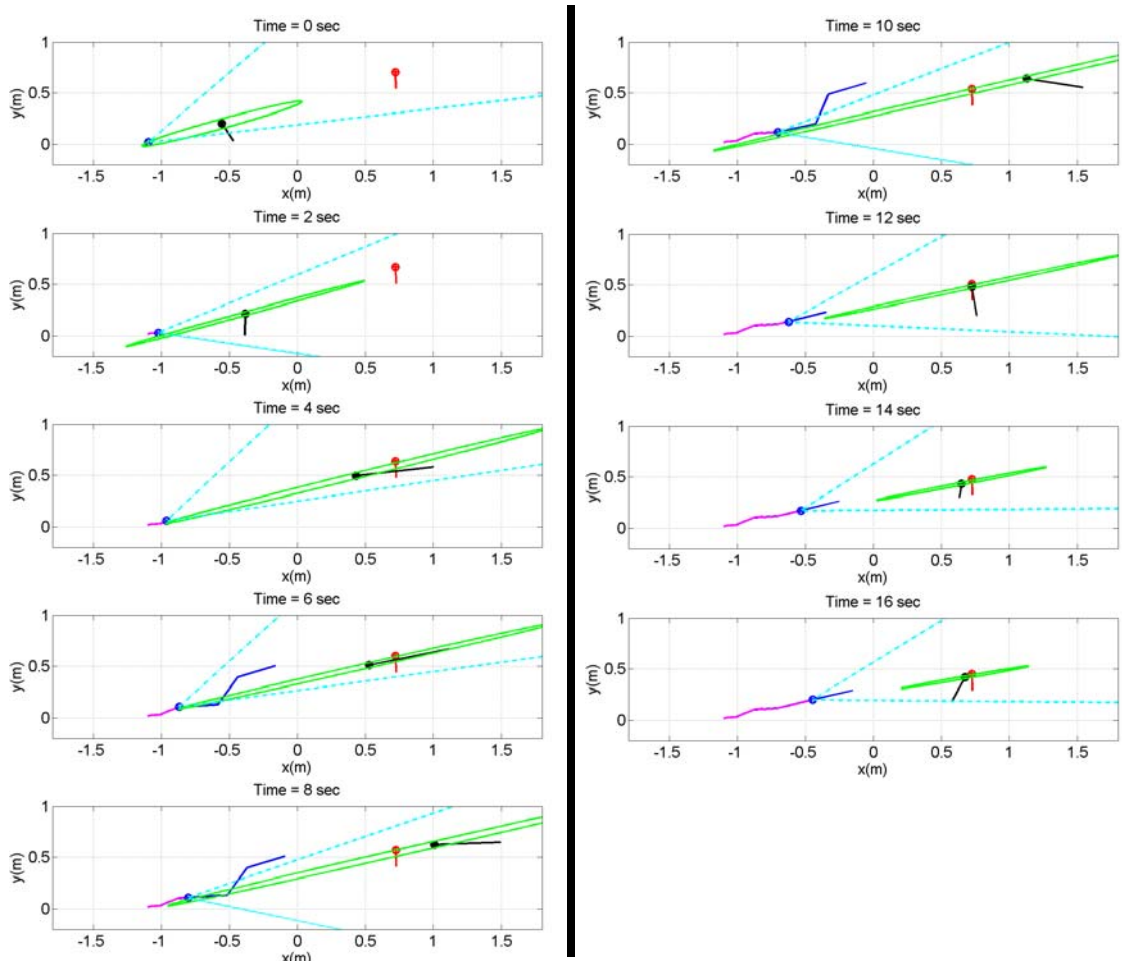

Figure 7.22 Time sequence of the micro rover experiment using the FUMT objective

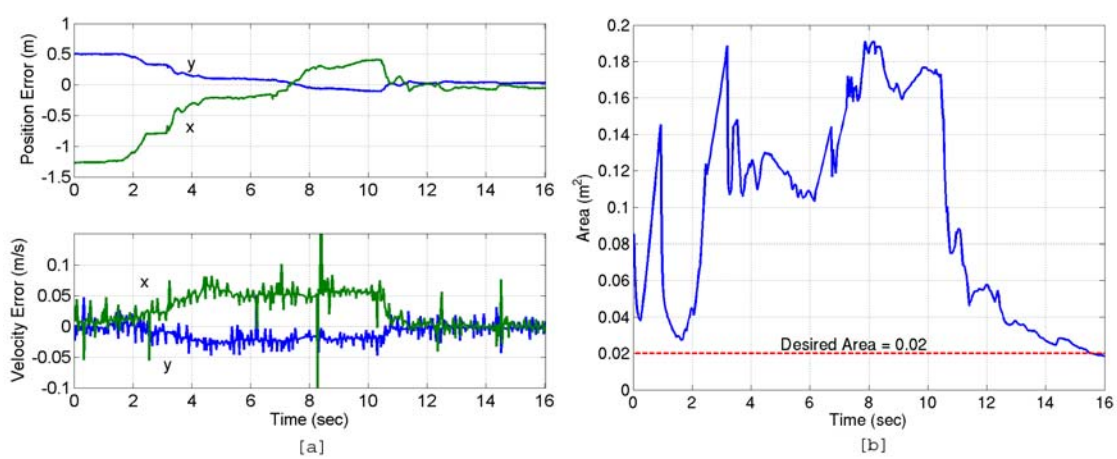

Figure 7.23 Target estimate performance for the micro rover experiment using the FUMT objective: [a] Estimate error versus time. [b] Error ellipse area versus time.

7.5 Conclusion

The simulations and experiments presented in this chapter verify the performance of the new observer-trajectory generator for target-motion estimation using monocular vision and the new trajectory-design algorithm that serves as its core. The first set of HIL simulation runs (Section 7.2) focused on the core trajectory-design algorithm, applying it to a known, standard observer-target encounter using both the fixed-uncertainty, minimum-time and the fixed-time, minimum-uncertainty objectives. Results verified the fast, near-optimal performance of the algorithm and also created benchmark trajectories for comparison to experiments with unknown initial target state. Furthermore, the fixed-uncertainty, minimum-time objective was demonstrated for the first time as the basis of a trajectory-design algorithm.

The second set of HIL simulations (Section 7.3) demonstrated the successful design of observer trajectories with unknown initial target state. These runs used a simulation of an autonomous rover platform to test the new observer trajectory-generator with both the FTMU and the FUMT objective functions. The utility of the initial observer maneuver was shown as well as the need for continual replanning in order to insure ultimate estimation success.

The final set of physical experiments (Section 7.4) applied the new trajectory generator to an operational micro autonomous observer rover platform. The experiments represented the first-ever demonstration of a trajectory generator for target-motion estimation using monocular vision. Additionally, the experiments presented the first-ever demonstration of trajectory generation for unknown initial target motion on an operational system.

CHAPTER 8 Conclusion

This chapter presents a summary of the contributions of this dissertation and recommendations for future work.

8.1 Summary

A new trajectory generator has been developed to give autonomous observer vehicles the ability to estimate an unknown target's motion using a single camera. Using this new method, an observer robot is capable of quickly generating near-optimal trajectories that enable it to fuse computer-vision measurements with navigation and control sensors to calculate the position and velocity of a previously unknown target object. New planning strategies that address the fundamental issues of the trajectory-design problem -- specifically the limitations of the computer vision sensor, the inclusion of a new optimization objective, and the unknown initial target state -- bridge the gaps between optimal trajectory generation and autonomous vehicle control, between bearings-only tracking and vision-based estimation, and between structured, known environments and unstructured, unknown ones. Key elements of the development of the new observer-trajectory generator include the identification of a quality metric to capture the utility of a given trajectory, the development of a fast trajectory-design algorithm that includes computer vision field-of-view limitations and dynamic vehicle constraints, and the integration of this algorithm into a full observer-trajectory generator for an autonomous vehicle interacting with unknown or uncertain targets.

The following sections summarize the contributions made in this dissertation and contain concluding remarks. A detailed list of the contributions may be found in Section 1.4.

8.1.1 Observer Trajectory Analysis

The first step in developing the observer-trajectory generator is the identification of a quality metric that reflects a trajectory's ability to improve the estimation performance (Chapter 3). In order to design optimal observer paths, the quality metric is needed to evaluate candidate trajectories. The end result of the observer's motion is an estimate of the target position and velocity, and the trajectory quality metric must therefore capture this goal. Previous research has utilized the Fisher Information Matrix (FIM) as the basis for trajectory design because it describes the effect of the trajectory on the performance of an ideal estimator. However, the target-motion estimation problem studied here is not efficient and does not reach the ideal bound. By using a metric that describes the final estimator performance, the trajectory generator is able to determine solutions that best correspond to the actual robotic system.

The predicted estimate-error covariance matrix has been identified as the best quality metric for trajectory design, and consequently trajectory evaluation has become estimation performance prediction. The estimate-error covariance matrix describes the second-order moments of the probability distribution representing the estimate error. In turn, bounding ellipsoids described by this matrix represent confidence regions around the target estimate. Although most typical estimators, such as the Extended Kalman Filter, output an estimate-error covariance matrix, the quality metric must predict the final estimator output before the observer has actually followed the path being evaluated. Therefore the predicted estimate-error covariance matrix is calculated by propagating the target-motion estimator forward in time and returning the resultant covariance matrix.

The related concepts of observability, performance, information, and quality were discussed in this work in order to clarify their importance to the trajectory-design problem. It was shown that target-motion estimation using monocular vision is unobservable without sufficient observer motion. Even if sufficient motion occurs, the resultant performance of the target estimation, defined by the uncertainty bounds on the estimation error, is still greatly affected by the observer trajectory. Basing the trajectory-quality metric on observability or information concepts is equivalent to measuring ideal performance bounds independent of the actual estimation scheme used. In contrast, using estimator performance criteria for the metric accounts for the specifics of the estimation process.

8.1.2 Core Trajectory Design Algorithm

At the core of the new observer-trajectory generator presented in this dissertation sits a novel trajectory-design algorithm capable of quickly creating near-optimal paths for an encounter with a constant-velocity

target with assumed known initial state (Chapter 4). The key advancements achieved by this algorithm are the ability to quickly create near-optimal trajectories, the extension of trajectory-design concepts from passive sonar applications to monocular vision, and the inclusion of the new fixed-uncertainty, minimum-time optimization objective. All three advancements are central to the application of the new trajectory generator on an autonomous observer vehicle estimating the motion of an unknown target.

An iterative, breadth-first search algorithm was developed to enable the minimization of estimate uncertainty in fixed time and to enable the minimization of the time needed to reach a fixed uncertainty level -- a new optimization objective never addressed by previous work. Observer dynamic constraints and vision sensor limitations create a search space with many local minima that is not easily maneuvered by traditional descent-based optimization techniques. In response, the new algorithm uses an enumerative approach to create a set of candidate trajectories by propagating the observer dynamics, and then includes the vision sensor limitations in the evaluation of the quality metric for each candidate trajectory. Likewise, the standard descent-based methods cannot easily incorporate the new fixed-uncertainty, minimum-time objective because the uncertainty is a complex function of the trajectory states and is not easily described by a state constraint equation. By using the enumerative process to generate the candidate trajectory set and evaluating the uncertainty metric as each candidate path is produced, what is a challenging optimization for traditional methods becomes the faster case for the new trajectory-design algorithm.

Although the core trajectory-design algorithm and the full trajectory generator for unknown initial target motion were developed to incorporate computer vision sensor limitations and non-holonomic observer dynamics, the methodology can be applied to more general bearings-only tracking problems which also require observer motion to enable successful target estimation. The benefits of the predicted error covariance matrix over the FIM still exist for inefficient versions of the BOT problem -- in particular, scenarios that are not considered long range. Additionally, the fixed-uncertainty, minimum-time objective developed here cannot be incorporated into any of the standard BOT design techniques.

The new trajectory-design algorithm enables the fast creation of observer paths in exchange for sub-optimal solutions, with several design parameters effecting the overall performance of this method. The two most important parameters are the number of observer maneuvers and the number of discrete heading intervals used to define the search space. The first parameter controls the smoothness of the observer path while the second controls the amount of the free space spanned by the candidate trajectory set. As both parameters increase, the size of the candidate trajectory set increases and the achievable cost is improved. However, as both parameters increase, the time to search through the entire candidate set and determine the final solution also increases. For real experiments with the hardware described in Chapter 6, the appropriate balance between planning time and trajectory cost was found (Figure 8.1) so that the method could be used as part

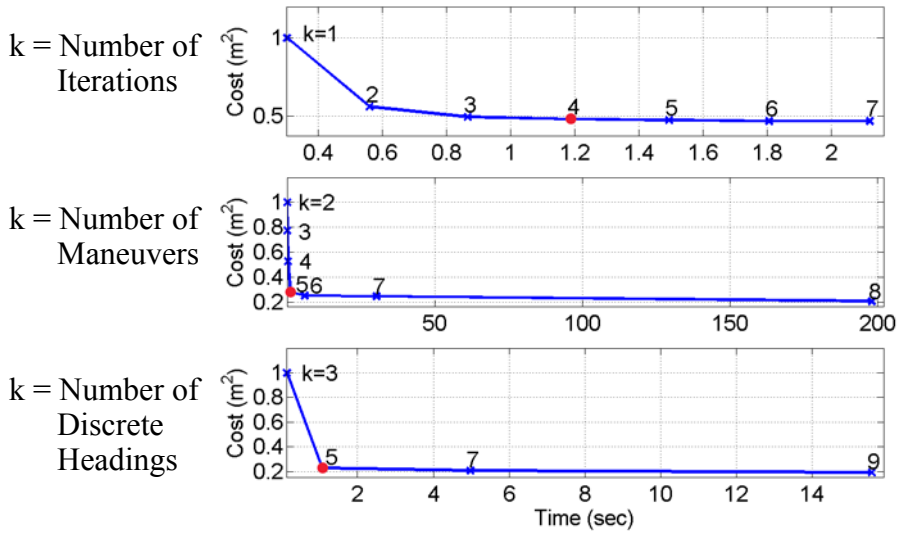

Figure 8.1 Optimization cost versus planning time

of a real-time system. However, scaling the trajectory generator to higher dimensions or more complex target behaviors will require additional performance trade-offs as well as faster computers with advanced estimation techniques in order to perform in real-time.

8.1.3 Full Observer-Trajectory Generator with Unknown Initial Target State

The main contribution of this dissertation is the development of a new observer-trajectory generator that enables autonomous observer vehicles to accomplish target-motion estimation using monocular vision (Chapter 5). This new method integrates the trajectory-quality metric and the core trajectory-design algorithm with other new components that address the fundamental issues of the full design problem. In particular, the unknown initial target state and the subsequent uncertain evolution of the target-state estimate complicate the trajectory-design process. Important details addressed by the generator include initialization of the target-state estimate and the observer path, responses to changes in the observer world model, i.e., the target-state estimate, and recovery when the target object is lost from camera view.

Upon first sensing a target object, the observer activates the trajectory generator by initializing the target-state estimate. The initial values of this estimate have a large effect on the subsequent trajectory design and final estimate convergence. Because the target motion is highly uncertain upon discovery, the generator implements an initial zig-zag maneuver that provides initial, but low, observability to the estimator. After the zig-zag maneuver, the core trajectory-design algorithm is called using the observer state and the target-state estimate as inputs.

In order to utilize all measurements, in the form of the current target-state estimate, a high-level supervision component was developed that monitors the states of the observer and the target and that initiates new plans

whenever necessary and possible. Knowing that the target-state estimate converges as the observer moves along its trajectory, two replanning strategies were devised in order to insure that the observer maintains a reasonable expectation of final performance. The first strategy applies to systems with little concern for the costs of replanning a trajectory, for example, the experimental system presented in this dissertation. For such systems, the trajectory generator simply invokes a new plan whenever possible. Using the experimental system as an example again, the trajectory-design algorithm takes approximately 1.5 seconds to return a new path so the generator calls for new ones every 2.0 seconds (Figure 8.2). In this way the current state estimate is always used to direct the observer's path. The second strategy applies to systems for which replanning consumes significant resources such as computational time or battery power. In this case, the observer-trajectory generator maintains the prediction of the target motion used to create the current observer trajectory and compares it to the current target-state estimate as the observer moves. A new plan is invoked when the predicted path and the current target-state estimate vary significantly. By continually monitoring the expectations of the target-state estimate, both replanning strategies create the additional benefit that the trajectory generator can sense when the target has been lost and can respond by invoking new observer motion to try to rediscover it.

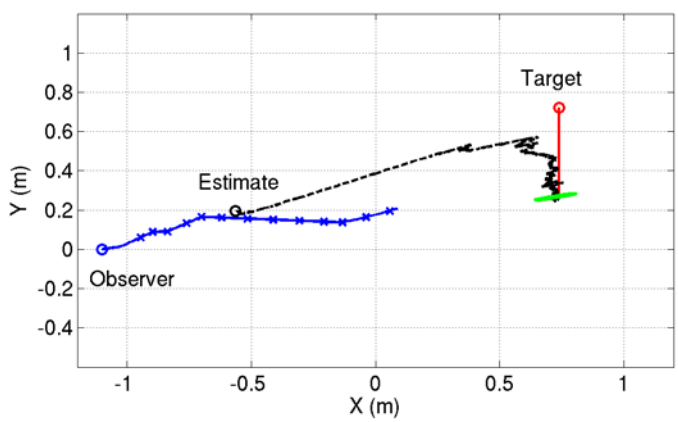

Figure 8.2 Simulation results of the trajectory generator for unknown initial target motion

One key aspect of the real-time performance of the trajectory generator is the need to budget time for the planning process. Upon sensing the need for a new plan, the generator propagates the observer trajectory and target-state estimate forward into the future time when the trajectory generator should return. These states are then passed to the trajectory generator so there will be a seamless transition from one path to the next.

8.1.4 Experimental Validation

The success of the new observer-trajectory generator to enable target-motion estimation using monocular vision was verified through simulation runs and hardware experiments using a micro autonomous rover estimating the motion of a constant-velocity target (Chapter 6 and Chapter 7). The core observer-trajectory design algorithm was first run in simulation in order to demonstrate its performance and to characterize the trade-off between planning time and final estimation performance. In turn, the full trajectory generator was verified in both simulation runs and hardware experiments. The micro autonomous observer rover used for the hardware experiments was specifically designed to model robotic applications that experience the target-motion estimation problem.

Simulation results showed the ability of the core observer-trajectory-design algorithm to generate quickly observer paths that achieved near-optimal final cost. A standard observer-target scenario was described in order to compare results between the observer-trajectory generator with known initial target motion and the full trajectory generator with unknown initial target state, and to compare results between the simulations and the hardware experiments. Using this standard scenario, the trajectory-design algorithm applied the two different optimization objectives to the problem. The results showed the creation of the fixed-time, minimum-uncertainty (FTMU) path and the complementary fixed-uncertainty, minimum-time (FUMT) path. These experiments represent the first-ever inclusion of the computer vision sensor and the new FUMT objective in the observer-trajectory-generation process.

Additionally, for simulations of the core trajectory-design algorithm, plots demonstrating the trade-off between planning speed and final achievable cost identified the appropriate design parameter values to use for the computing resources available to the micro autonomous rover platform. By running the simulations using the same computers that are part of the operational system, the results predict the trajectory-design algorithm's performance during the hardware experiments. The key results of the trade-off analysis are that near-optimal paths can be achieved in approximately 1.5 seconds and that small incremental performance gains require large additional computational times. If different computers are used for the system, the actual results will vary but the general trends should remain.

The fundamental issues addressed by the observer-trajectory generator were demonstrated in simulation and verified on experimental hardware. It was shown in simulation that large initial target-state uncertainty would cause the core trajectory-design algorithm to fail. The adoption of the new trajectory-design strategies developed in this dissertation solved the problem of unknown initial target state and led to the successful estimation of the target motion. The trajectory generator again applied both the typical fixed-time, minimum-uncertainty objective and the new fixed-uncertainty, minimum-time objective.

Figure 8.3 Micro autonomous rover platform

The micro autonomous rover platform (Figure 8.3) created by the Aerospace Robotics Laboratory served as the hardware platform for the operational experiments of the trajectory generator. A new observer rover was designed specifically to demonstrate the results of this dissertation. Using an onboard wireless video camera, the observer rover visually tracked an infra-red LED attached to a second target rover moving at a constant velocity. The new trajectory generator directed the observer rover to successfully estimate the motion of the target rover using both optimization objectives.

8.2 Recommendations for Future Work

The research presented in this thesis can be extended in many future directions as autonomous vehicle development continues. Several interesting and exciting possibilities are discussed below.

8.2.1 Core Observer-Trajectory Design Algorithm

Several basic extensions to the core trajectory-design algorithm immediately spring to mind. The results presented in this thesis focus on constant-velocity, planar, target motion, but the trajectory generator was not developed for a specific estimator, so more-complex target motion, including moving in three dimensions, can be incorporated into the generator easily. Motion estimation for a dynamic target is a difficult task and must therefore be developed further as well. Likewise, if the target vehicle becomes evasive, that is it can react to the observer's motion, new trajectory strategies will need to be created. In addition to more-complex target dynamics, the observer-trajectory-design algorithm can also be extended to allow for more-complex observer dynamics by changing the parameterization of the observer trajectories.

More-efficient optimization methods, especially for the fixed-uncertainty, minimum-time objective function, would improve the performance of the trajectory-design algorithm. Currently the method uses an exhaustive, sub-optimal approach in order to traverse the search space and include the complex objective

constraint. New optimization techniques that can navigate this rough space should speed up planning times and enable the application of the method to longer, more complex scenarios. One possible way to improve the optimization is also to develop new parameterizations of the observer trajectories. Huster et al. present the first attempt at using goal-specific motion primitives to define the observer paths [66].

The issue of target-estimate uncertainty is a driving factor in the overall observer-trajectory generator and can be addressed more directly by the core trajectory-design algorithm. Instead of exclusively using the target-state estimate to develop the observer trajectory, the estimate-error covariance matrix can be added to define a set of target-state estimates which can in turn be used to calculate more robust results. Possible solutions include a minimax approach in which the trajectory is generated to optimize the worst case or a Bayesian approach which optimizes the expected value over the entire set of target-state estimates. Such a design algorithm could reduce the need to monitor the predicted target-state estimate or continually initiate replans.

The inclusion of multiple targets and multiple cooperating observers is another natural extension of the trajectory generator presented in Chapter 4. Most operational vehicles would expect to encounter more than one target in their environment. Inclusion of multiple targets can easily be accomplished by propagating estimator models for all of the objects and using a weighted sum of their performance as the final cost. In contrast, the inclusion of multiple cooperating observers is not nearly as straightforward. The performance of the target-motion estimation can be improved greatly by sharing measurements between the observers. The concept of cooperation would have to be incorporated into a new estimator and a new trajectory-generation methodology for this case.

Finally, the trajectory-design algorithm developed in this work can be applied to other dynamically observable systems. Trajectory design for system observability is not limited to target-motion estimation using monocular vision. The core concept of calculating the predicted error covariance matrix in order to measure trajectory quality can be applied to many similar systems. For example, this method was tested on a 7 degree-of-freedom robotic arm and helped improve tracking performance for underwater manipulation applications [66].

8.2.2 Full Trajectory Generation With Unknown Initial Target State

The current trajectory generator always invokes a new plan when possible in order to incorporate the latest target-state estimate. If the cost of computation is high (which it was not in the experiments presented in Chapter 7) the current estimate and the predicted estimate are compared and a new plan is only invoked if they differ greatly. A useful extension would be to calculate a new predicted error covariance matrix continually for the current target-state estimate and thus determine the final cost for the trajectory the

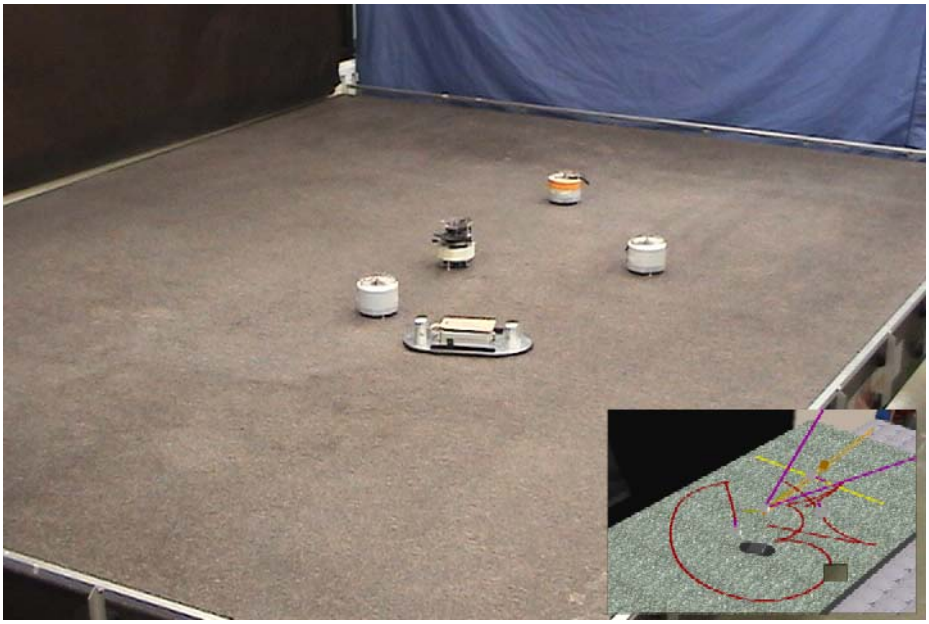

Figure 8.4 Trajectory generator integrated with a dynamic network controlling multiple autonomous rovers [64].

observer is currently following. In doing so, the trajectory generator can avoid calling for a replan if the current trajectory still produces reasonable performance and can make the decision to replan once the expected performance of the current trajectory has degraded substantially.

The monitoring component of the observer-trajectory generator can be broadened to include more criteria and to incorporate multiple planners with competing goals. The current system only focuses on the estimation of target vehicle states and does not concern itself with other tasks such as obstacle avoidance. A higher-level monitor can mediate between different observer objectives and direct one or more planners accordingly. Likewise, the allowable target behavior can be expanded by developing more sophisticated monitors. A dynamic target that has piecewise-constant speed can still be tracked with the method presented here as long as the system knows a target maneuver has taken place and the algorithm responds.

8.2.3 Applications

The trajectory generator for target-motion estimation using monocular vision developed in this dissertation is most applicable to autonomous unmanned aerial vehicles (UAVs) that will have restricted dynamics, limited payload capacity, and large baseline to target ratios [70]. In order to apply the methods presented here, they must first be extended to three-dimensions and incorporate the more-complex vehicle dynamics of the aircraft. The computer vision tracking system used for such a system would likely include a combination of image processing routines and the characteristics of this sensor as a bearing sensor would have to be known. For instance, the performance of shape based tracking is going to be a function of the

range to the target and can be included in the estimation model used to predict the target-state estimate and generate the trajectories. Potential UAV applications, such as surveillance, may also require additional observer motion constraints, such as the inclusion of safe or hostile regions.

The second class of applicable vehicles is wheeled rovers. Near term space exploration plans call for the use of interplanetary rovers to explore the surface of distant planetary objects [69]. These vehicles will also have non-holonomic dynamics, payload restrictions, and video camera equipment. Objects of potential interest, such as rocks, can be considered stationary targets and can be mapped using the new observer-trajectory generator. Once mapped, the rover can return to the object on future traverses or other vehicles can be directed to the same location to perform other measurements or experiments.

Motion planning for mobile robots and autonomous vehicles generally considers two main issues - planning a path that moves the vehicle from a start location to a final, goal location and avoiding obstacles and other hazards along the way. Examples of state of the art research techniques currently being investigated include the use of randomized motion planners and dynamics networks to coordinate the motion of multiple robots (Figure 8.4) in an unstructured, unknown environment [64, 80] and the use of hybrid systems theory to create optimal trajectories for aircraft within the nation's airspace [72]. In both examples one key assumption is the ability of the robots or aircraft to sense each other's location and future motion. Incorporating a trajectory generator with information gathering goals similar to the one developed here would further broaden the applicability of these current planning techniques.

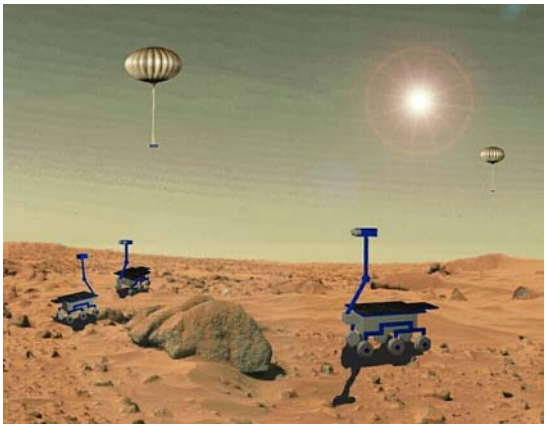

Figure 8.5 Self calibrating pseudolite array on the surface of Mars

Another related area with observability criteria dependent on vehicle or sensor motion are differential GPS applications. Current navigation techniques take advantage of the GPS signal structure to achieve meter- and centimeter-level navigation accuracy over the span of kilometers. The fundamental measurements used for these techniques suffer from an integer ambiguity that is often solved by vehicle motion [57]. One

exciting new application of GPS technology is the creation of self-calibrating pseudolite arrays (Figure 8.5) that may enable centimeter-level accuracy on the surface of Mars [75]. The initialization of such an array requires a moving transceiver to appropriately transverse through the array. To date, the trajectories have been design in an ad-hoc manner. Application of a similar trajectory generator could determine optimal trajectories for the vehicle on Mars, thereby saving resources that can be better used exploring the planet.

Finally, target-motion estimation using monocular vision can serve as a backup for any autonomous vehicle with a multiple-camera computer-vision system. Many autonomous vehicles, such as interplanetary rovers, will cost millions of dollars to create and will be sent into important environments. The reliance on a multiple-camera system could be fatal if one or more of the cameras fails. Using the technique developed in this dissertation, the vehicle could continue on its mission with reduced, but not lost, functionality.

8.3 Conclusion

The central thesis of this work has been that computationally-fast trajectory generation to improve target-motion estimation using monocular vision can be achieved and integrated into an autonomous robotic vehicle. To that end, this dissertation has presented the details of a novel observer-trajectory generator that focuses on several fundamental issues. These issues include the use of the computer vision sensor despite its limited field of view, the inclusion of a new optimization objective that applies to operational missions better than previously studied objectives do, and the unknown and highly uncertain nature of the target-state estimate upon initially encountering the target object.

The experimental results, in the form of hardware demonstrations of a micro autonomous rover platform and simulations of that same system, verify the performance of the new generator and prove the main thesis of this dissertation. With additional analysis and future work, the concepts and methods demonstrated in this dissertation can be expanded and applied to a large range of applications, bringing the initial promise of autonomous vehicles closer to fruition.

Bibliography

[1] Motti Gavish and Eli Fogel. Effect of bias on bearing-only target location. *IEEE Transactions on Aerospace and Electronic Systems*, 26(1):22-25, 1990.

[2] L.G Taff. Target localization from bearings-only observations. *IEEE Transactions on Aerospace and Electronic Systems*, 33(1):2-9, 1997.

[3] Mati Wax. Position location from sensors with position uncertainty. *IEEE Transactions on Aerospace and Electronic Systems*, 19(5):658-662, 1983.

[4] P.L. Bolger. Tracking a maneuvering target using input estimation. *IEEE Transactions on Aerospace and Electronic Systems*, 23(3):298-310, 1987.

[5] N. Peach. Bearings-only tracking using a set of range-parameterized extended Kalman filters. *IEE Proceedings-Control Theory Appl*, 142(1):73-80, 1995.

[6] Vincent J. Aidala. Kalman filter behavior in bearings-only tracking applications. *IEEE Transactions on Aerospace and Electronic Systems*, 15(1):29-39, 1979.

[7] Vincent J. Aidala and Sherry E. Hammel. Utilization of modified polar coordinates for bearings-only tracking. *IEEE Transactions on Automatic Control*, 28(3):283-294, 1983.

[8] M. Karan, J. Wang, and S. E. Hammel. Bearings-only tracking with sea trial sonar data from multiple asynchronous sonobuoys. http://citeseer.nj.nec.com/406714.html. 1-6.

[9] Kraig L. Anderson and Ronald A Iltis. A distributed bearings-only tracking algorithm using reduced sufficient statistics. *IEEE Transactions on Aerospace and Electronic Systems*, 32(1):339-349, 1996.

[10] Andrew Logothetis, Alf Isaksson, and Robin J. Evans. An information theoretic approach to observer design for bearings-only tracking. In *Proceedings of the 36th Conference on Decision and Control*, pages 3132-3137, San Diego CA, December 1997.

[11] Sherry E. Hammel and Vincent J. Aidala. Observability requirements for three-dimensional tracking via angle measurements. *IEEE Transactions on Aerospace and Electronic Systems*, 21(2):200-207, 1985.

[12] S. E. Hammel, P. T. Liu, E. J. Hilliard, and K. F. Gong. Optimal observer motion for localization with bearing measurements. *Computers and Mathematics with Applications*, 18(1-3):171-180, 1989.

[13] J. P. Le Cadre and C. Jauffret. Discrete-time observability and estimability analysis for bearings-only target motion analysis. *IEEE Transactions on Aerospace and Electronic Systems*, 33(1):178-201, 1997.

[14] S. Koteswara Rao. Comments on “Discrete-time observability and estimability analysis for bearings-only target motion analysis”. *IEEE Transactions on Aerospace and Electronic Systems*, 34(4):1361-1367.

[15] J. M. Passerieux and D.Van Cappel. Optimal observer maneuver for bearings-only tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 34(3):777-788, 1998.

[16] Yaakov Oshman and Pavel Davidson. Optimization of observer trajectories for bearings-only target localization. *IEEE Transactions on Aerospace and Electronic Systems*, 35(3):892-902, 1999.

[17] J. P. Le Cadre and C. Jauffret. On the convergence of iterative methods for bearings-only tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 35(3):801-818, 1999.

[18] Andrew Logothetis, Alf Isaksson, and Robin J. Evans. Comparison of suboptimal strategies for optimal own-ship maneuvers in bearings-only tracking. In *Proceedings of the American Control Conference*, pages 3334-3338, Philadelphia PA, June 1998.

[19] J. P. Le Cadre. Optimization of the observer motion for bearings-only target motion analysis. In *Proceedings of the 36th Conference on Decision and Control*, pages 3126-3131, San Diego CA, December 1997.

[20] Steven C. Nardone, Allen G. Lindgren, and Kai F. Gong. Fundamental properties and performance of conventional bearings-only target motion analysis. *IEEE Transactions on Automaitc Control*, 29(9):775-787, 1984.

[21] Ouyang Guanghui, Sun Jixiang, Li Hong, and Wang Wenhui. Estimating 3D motion and position of a point target. In *Proceedings of SPIE* 3137:386-394, 1997.

[22] Matthew C. Deans. *Robust and Efficient Simultaneous Localization and Mapping from Bearings Only Sensing.*

[23] John J. Leonard. Large-scale concurrent mapping and localization. In *Proceedings of SPIE*, 4196:370-376, 2000.

[24] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*. Edmonton, Canada, 2002.

[25] M. W. M. Gamini Dissanayake, Paul Newman, Steven Clark, Hugh F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229-241, 2001.

[26] Jose E. Giuvant and Eduardo Mario Nebot. Optimization of the simultaneuos localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17(3):242-257, 2001.

[27] Howie Choset and Keiji Nagatani. Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17(2):125-136, 2001.

[28] C.-C. Wang and C. Thorpe, Simultaneous Localization and Mapping with Detection and Tracking of Moving Objects. In *Proceedings of 2002 IEEE International Conference on Robotics and Automation*, May 11-15, 2002, Washington D.C.

[29] Larry Mathies, Takeo Kanade, and Richard Szeliski. Kalman Filter-based Algorithms for Estimating Depth from Image Sequences. *International Journal of Computer Vision*. 3: 209-236, 1989.

[30] Shai Avidan and Amnon Shashua. Trajectory triangulation: 3D reconstruction of moving points from a monocular image sequence. *IEEE Transactions on Pattern Analyisi and Machine Intelligence*, 22(4):348-357, 2000.

[31] Ben Grocholsky, Hugh Durrant-Whyte, and Peter Gibbons. An Information-Theoretic Approach to Decentralized Control of Multiple Autonomous Flight Vehicles. In *Proceedings of SPIE*, 4196:348-359, 2000.

[32] Paul Viola and William M. Wells III. Alignment by maximization of mutual information. *International Journal of Computer Vision*, 24(2):137-154, 1997.

[33] James C. Spall. The information matrix in controls: Computation and some applications. In *Proceedings of the 38th Conference on Decision and Control,* pages 2367-2372, Phoenix Arizona, Dec. 1999.

[34] Ian Ford, D. M. Titterington, and Christos P. Kitsos. Recent advances in nonlinear experimental design. *Technometrics*, 31(1):49-60, 1989.

[35] Probal Chaudhuri and Per. A. Mykland. Nonlinear experiments: Optimal design and inference based on likelihood. *Journal of the American Statistical Association*, 88(422):538-546, 1993.

[36] Probal Chaudhuri and Per. A. Mykland. On efficient designing of nonlinear experiments. *Statistica Sinica* 5:421-440, 1995.

[37] R.E. Kalab and K. Spingarn. Optimal inputs for nonlinear process parameter estimation. *IEEE Transactions on Aerospace and Electronic Systems*. 10(3):339-345, 1974.

[38] Evanghelos Zafiriou and Adebola O. Bamgbose. Robust optimal input profile determination for semi-batch processes. Presented at the 1988 AICHE Annual Meeting, Washington D. C. 1988. pages 1-7.

[39] W.G. Cochran. Experiments for nonlinear functions. *Journal of the American Statistical Association*. 68(344):771-781, 1973.

[40] Takeo Kanade, Robert T. Collins, Alan J. Lipton, Peter Burt, and Lambert Wixson. Advances in cooperative multi-sensor video surveillance. DARPA Image Understanding Workshop (IUW), 115-122, Nov. 1998.

[41] Satoshi Yonemoto, Asuka Matsumoto, Daisaku Arita, and Rin-ichiro Taniguchi. A real-time motion capture system with multiple camera fusion. In *Proceedings of 10th International Conference on Image Analysis and Processing*. IEEE. 600-605, 1999.

[42] S.L Laubach and J.W. Burdick. An autonomous sensor-based path-planner for planetary microrovers. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, pages 347-354, Detroit, Michigan, May 1999.

[43] Benedict Wong and Minas Spetsakis. Scene reconstruction and robot navigation using dynamic fields. *Autonomous Robots*. 8:71-86, 2000.

[44] G. Qian, A. Kale, and R. Chellappa. Robust estimation of motion and structure using a discrete H-Infinity filter. IEEE, 616-619, 2000.

[45] Gavin J.S. Ross. *Nonlinear Estimation*. Springer-Verlag, New York, 1990.

[46] Harry L. Van Trees. *Detection, Estimation, and Modulation Theory.* John Wiley and Sons,Inc., New York, 2001.

[47] Jerry M. Mendel. *Lessons in Estimation Theory For Signal Processing, Communications, and Control.* Prentice Hall Ptr, Englewood Cliffs, 1995.

[48] Robert F. Stengel. *Optimal Control and Estimation.* Dover Publications, Inc., New York, 1994.

[49] Arthur Gelb (editor). *Applied Optimal Estimation*. The M.I.T. Press, Cambridge, MA, 1974.

[50] Clark F. Olson. Selecting Landmarks for Localization in Natural Terrains. *Autonomous Robots*. 12:201-210, 2002.

[51] Clark F. Olson and Larry H. Matthies. Maximum likelihood rover localization by matching range maps. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, pages 272-277, Leuven, Belgium, May 1998.

[52] M.C. Martin and H. Moravec. *Robot Evidence Grids*. Tech. report CMU-RI-TR-96-06. Robotics Institute, Carnegie Mellon University, March, 1996.

[53] Richard Volpe, Tara Estlin, Sharon Laubach, Clark Olson, and J. Balaram. Enhanced mars rover navigation techniques. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pages 926-931, San Francisco, CA, April 2000.

[54] J.S. Gutmann and D. Fox. An experimental comparison of localization methods continued. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02)*, 2002.

[55] C. Becker, H. Gonzalez-Banos, J.C. Latombe, and C Tomasi. An intelligent observer. In *Proceedings of the 4th International Symposium on Experimental Robotics*. pp153-160, 1995.

[56] Cheng-Yu Lee. *Real-time Target Tracking in an Indoor Environment*. PhD thesis, Stanford University, Stanford, CA 94305, May 2002.

[57] Bradford W. Parkinson and James J. Spiker Jr. (editors) and Penina Axelrad and Per Enge (associate editors). *Global Positioning System: Theory and Applications.* American Institute of Aeronautics and Astronautics, Inc. Washington D.C., 1996.

[58] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, Upper Saddle River, New Jersey, 1998.

[59] Olivier Faugeras. *Three-dimensional Computer Vision: A Geometric Approach.* The M.I.T. Press, Cambridge, MA, 1993.

[60] T.J. Broida, S. Chandrashekhar, and R. Chellappa. Recursive 3-D Motion Estimation from a Monocular Image Sequence. *IEEE Transactions on Aerospace and Electronic Systems*. 26(4):639-656, July 1990.

[61] M. Han and T. Kanade. Multiple motion scene reconstruction from uncalibrated views. In *Proceedings of the Eighth IEEE International Conference on Computer Vision*. July 7-14, 2001, Vancouver, Canada.

[62] K. Kumar, D. Yadav, and B. V. Srinivas. Adaptive moise models for Extended Kalman Filters. *Journal of Guidance, Control, and Dynamics.* 14(2)475-477, 1991.

[63] C. Wang, Heng Ma, and D. J. Cannon. Human-machine collaboration in robotics integrating virtual tools with a collision avoidance concept using conglomerates of spheres. *Journal of Intelligent and Robotic Systems*. 18(4):367-369, 1997.

[64] Chris Clark, Eric W. Frew, Henry L. Jones, and Stephen M. Rock. An Integrated System for Command and Control of Cooperative Robotic Systems. To appear in *Proceedings of the 11th International Conference on Advanced Robotics*. Portugal, June 2003.

[65] Eric W. Frew and Stephen M. Rock. Trajectory Generation for Monocular Vision-Based Tracking of a Constant Velocity Target. To appear in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*. Taipei, Taiwan, May 2003.

[66] Andreas Huster, Eric W. Frew, and Stephen M. Rock. Relative Position Estimation for AUVs by Fusing Bearing and Inertial Rate Sensor Measurements. In *Proceedings of the Oceans 2002 Conference*, pp. 1857-1864, Biloxi, MS, October 2002.

[67] Eric W. Frew and Stephen M. Rock. Exploratory Motion Generation for Monocular Vision-Based Target Localization. In *Proceedings of the 2002 IEEE Aerospace Conference*, 7:3633-3643, Big Sky, MT, March 2002.

[68] R.L. Marks, S.M. Rock, and M.J. Lee. Real-time video mosaicking of the ocean floor. *IEEE Journal of Oceanic Engineering*. 20(3):229-241, July 1995.

[69] M. P. Golombek. The Mars Pathfinder mission. *Scientific American*. 279(1)24-33, July 1998.

[70] R. Kellett. Train the way you'll fight! [Unmanned aerial vehicle]. *Unmanned Systems*. 17(3):12-14. Fall 1999.

[71] Jason Rife and Stephen M. Rock. A Pilot-aid for ROV based tracking of gelatinous animals in the midwater. In *Proceedings of the OCEANS 2001 Conference*, Honolulu, HI, November 2001.

[72] Claire Tomlin, George J. Pappas, and Shankar Sastry. Conflict Resolution for Air Traffic Management: A Study in Multi-Agent Hybrid Systems. *IEEE Transactions on Automatic Control*. 43(4):509-521, April 1998.

[73] M. Cowley and M. Keenon. Micro air vehicles - a system developers point of view. *Unmanned Systems*. 16(2):14-18, Spring 1998.

[74] J. Hyams, M. W. Powell, and R. Murphy. Cooperative navigation of micro-rovers using color segmentation. *Autonomous Robots*. 9(1):7-16, Aug. 2000.

[75] Edward A. LeMaster and Stephen M. Rock. A Local-Area GPS Pseudolite-Based Navigation System for Mars Rovers. *Journal of Autonomous Robots*. 4(2-3):209-224, Mar-May 2003.

[76] Henry L. Jones II. *An Object-Based Interaction Framework for the Operation of Multiple Field Robots*. PhD thesis, Stanford University, Stanford, CA 94305, February 2003. Also published as SUDAAR 764.

[77] http://www.x10.com/products/vk45a_kit_components.htm.

[78] Vincent W. Chen. *Experiments in Adaptive Control of Multiple Cooperating Manipulators on a Free-Flying Space Robot.* PhD thesis, Stanford University, Stanford, CA 94305, December 1992. Also published as SUDAAR 631.

[79] S. Schneider. *Experiments in the Dynamic and Strategic Control of Cooperating Manipulators.* PhD thesis, Stanford University, Stanford, CA 94305, September 1989. Also published as SUDAAR 586.

[80] Christopher M. Clark, Stephen M. Rock, and Jean-Claude Latombe. Motion Planning for Multiple Mobile Robots using Dynamic Networks. To appear in *Proceedings of the 2003 International Conference on Robotics and Automation*, Taipei, Taiwan, May 2003. IEEE.

[81] H. Gonzalez-Banos, J.-C. Latombe, D. Lin, and P. Fabiani. Tracking an unpredictable target among occluding obstacles under localization uncertainties. *Robotics and Autonomous Systems*. 2002.

[82] P.K.A. Menon, E. Kim and V.H.L Cheng. Optimal trajectory synthesis for terrain-following flight. *Journal of Guidance, Control, and Dynamics*. 14(4):807-813, July-Aug. 1991.

[83] Lu Ping, and B.L. Pierson. Optimal aircraft terrain-following analysis and trajectory generation. *Journal of Guidance, Control, and Dynamics*. 18(3):555-560, May-June 1995.

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

[85] J.P. Hespanha. Single-camera visual servoing. In *Proceedings of the IEEE Conference on Decision and Control*. Volume 3, pp. 2533-2538, 2000.