RELATIVE POSITION SENSING BY FUSING MONOCULAR VISION AND INERTIAL RATE SENSORS

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

Andreas Huster
July 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
Department of Aeronautics and Astronautics
(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.

Stephen P. Boyd
Department of Electrical Engineering

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.

Oussama Khatib
Department of Computer Science

Approved for the University Committee on Graduate Studies.

Abstract

Sensing the relative position between a robot and objects in its environment is a core requirement for many robot tasks and is an area of active research. This dissertation describes the development of a new, robust, relative-position sensing strategy suitable for unstructured and unprepared environments. Underwater manipulation, the use of underwater vehicles to perform object manipulation tasks in the ocean, is the particular application that motivated this research. Although many relative position sensing systems have already been developed, achieving the level of robustness that is required for operation in the underwater environment is very challenging.

The new sensing strategy is based on fusing bearing measurements from computer vision and inertial rate sensor measurements. These measurements are fused to compute the relative position between a moving observer and a stationary object. Inertial rate sensors are composed of accelerometers and rate gyros.

The requirements on the vision system have been chosen to be as simple as possible: tracking a single feature on the object of interest with a single camera. This is equivalent to a bearing measurement. Simplifying the vision system has the potential to create a more robust sensing system. The relative position between a moving observer and a stationary object is observable if these bearing measurements, acquired at different observer positions, are combined with the inertial rate sensor measurements, which describe the motion of the observer.

The main contribution of this research is the development of a new, recursive estimation algorithm which enables the sensing strategy by providing a solution to the inherent sensor fusion problem. Fusing measurements from a single bearing sensor with inertial rate sensor measurements is a nonlinear estimation problem that is difficult to solve with standard recursive estimation techniques, like the Extended Kalman Filter (EKF). A new, successful estimator design—based on the Kalman Filtering approach but adapted to the unique requirements of this sensing strategy—was developed. The new design avoids the linearization of the nonlinear system equations. This has been accomplished by developing a special system representation with a linear sensor model and by incorporating the Unscented Transform to propagate the nonlinear state dynamics.

The dissertation describes how the sensing strategy can be implemented to determine the relative position between a moving observer and a stationary object. A demonstration task is developed that illustrates how a real-time implementation of this sensing strategy can be incorporated into the closed-loop control of an autonomous robot to perform an object manipulation task. The performance of the sensing strategy is evaluated with this hardware experiment and extensive computer simulations. Centimeter-level position sensing for a typical underwater vehicle scenario has been achieved.

To Eileen, with love and appreciation.

Acknowledgments

Being able to pursue my graduate studies at Stanford University and the research that led to this dissertation has been a privilege which I have never taken for granted. Although the opportunity to begin my graduate work came easily to me, finding the effort, dedication, patience, motivation and attitude to complete it was very difficult. Essential to this journey was the support from family, friends, colleagues, professors and staff. Others have provided the material resources to make it happen. For this support, I am very grateful!

I want to thank Eileen for the opportunity to share my life with her. She has become my best friend, partner, and spouse. She has given me the space to be myself, but has taught me the value in being together. She has allowed me to pursue my goals and has insisted that she could do the same. She has blessed me with a family that will, no doubt, continue to grow. She has learned to be patient with me, even though patience does not come easily to her. Because of her hard work, we have had the luxury to be patient, to enjoy the journey, and to dream with confidence. *I dedicate this dissertation to her.*

Julian, our little boy, has been a welcome distraction and an incredible inspiration for the better part of the past three years. He has an amazing desire for life and an enviable capacity to learn. I hope to be an inspiring role model for him, for his new sibling, and for any others that may come after that.

I consider myself to be a very fortunate person because of the beautiful people who are my family and friends. Many of them have been important pillars of support in this process: my parents Roland and Anita, my sister Karin and her family, Eileen's parents Victor and Xochitl, all of our great friends in Vancouver, and our new friends at Stanford. I am honored to be in their company, grateful for their encouragement, and excited about their interest in my work.

A doctorate degree at Stanford University became a serious goal for me during a hike in Switzerland with Brian Hargreaves, where he convinced me that graduate school was what we wanted to do, that California would be the place to do it, and that Stanford was our school of choice. Since I met Brian at the beginning of our undergraduate program, he has been a great friend, my best man, a roommate, a study partner, and above all, an inspiration and a role model for all things academic and otherwise.

Throughout my time at Stanford University, I have been a proud member of the Aerospace Robotics Laboratory. I want to thank Professor Robert Cannon for inviting me to join the ARL and Professor Stephen Rock for helping me to graduate from the lab. When I arrived at Stanford, I quickly found a home in the ARL and never had any reason to wonder whether robotics was really what I wanted to do. The ARL is a very special place because of the students, past and present, that are part of it. I sincerely appreciate their capacity as researchers, their breadth of experience, the aspirations they represent, and the value of their friendship. I respect them all very deeply.

I want to thank Eric Frew and Hank Jones for their special friendship and integral contributions during the most difficult years of this process. We supported each other with weekly meetings, ambitious milestones, clever strategies and inescapable accountability. It was reassuring to know that they were struggling with similar issues; it was helpful to see how they worked through them; and it was gratifying to find opportunities to help them, too. It was especially useful to receive their frequent, candid, and direct feedback. They took responsibility for my progress and ownership of my success, which shows what great friends they have become.

The staff of the ARL as well as the Aeronautics and Astronautics Department are a wonderful group of people, who do so many things for the students and with such a positive attitude. In particular, I want to thank Godwin Zhang, Jane Lintott, Sally Gressens, Aldo Rossi, Shreann Ellsworth and Dana Parga for all their help and for their encouragement.

I want to thank Professor Rock for advising me and funding my research throughout my time at Stanford. He gave me the freedom and the encouragement to pursue my own ideas and to manage my own research. Moreover, he provided an ingredient that nobody else could: however difficult a conversation was, however disappointed he may have been, however distant the Ph.D. degree seemed to me, he always communicated a confidence that these issues are surmountable and that some day, everything would come together into a successful outcome. He was right. Along the way, he never failed to remind me that the research in this dissertation is my own and that it represents a valuable piece of work.

Three other professors have served on my committees: Professor Stephen Boyd, Professor Oussama Khatib and Professor Christian Gerdes. I appreciate their time and effort as well as their insightful comments.

Various organizations have provided resources, financial and otherwise, to support my research. These include the David and Lucile Packard Foundation, the Natural Sciences and Engineering Research Council (NSERC) of Canada, the Canadian Space Agency, the W. Noel Eldred Fellowship, the Monterey Bay Aquarium Research Institute (MBARI), and the NASA Ames Research Center. I sincerely appreciate their support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This dissertation shows that *the fusion of monocular vision measurements of a single feature with inertial rate sensor measurements forms a feasible sensing strategy for determining the relative position between a moving observer and a stationary object* (see Figure 1.1). Robust relative position sensing is a core requirement as well as a significant challenge for many types of robots, especially those operating in real terrain. This research is motivated by the sensing requirements of underwater vehicles, which operate with the additional challenges presented by the underwater environment. Vision-based solutions are typically proposed to solve this problem. This dissertation explores a new sensing strategy with the potential for increased robustness by combining a very simple vision system and inertial rate sensors to solve the problem.

The main contribution of this work is the design of a new nonlinear, recursive estimator that has been adapted to the unique requirements of this sensing strategy. In particular, the types of nonlinearity and the dynamic observability associated with the sensing strategy lead to poor estimator performance for standard solutions. A new solution, based on a modified Kalman Filter approach, a specific representation of the system equations, and the application of the Unscented Transform, has been designed, implemented and evaluated.

1.1 Motivation and Background

Sensing the relative position between a robot and objects in its environment is a core requirement for many robot tasks. These include obstacle avoidance, robot localization, object mapping and object manipulation. Relative position sensing is an important capability for robots that operate in factories, in the home, on construction sites, in space, in hazardous environments and in the ocean. Many different relative position sensing systems have been developed to enable these applications.

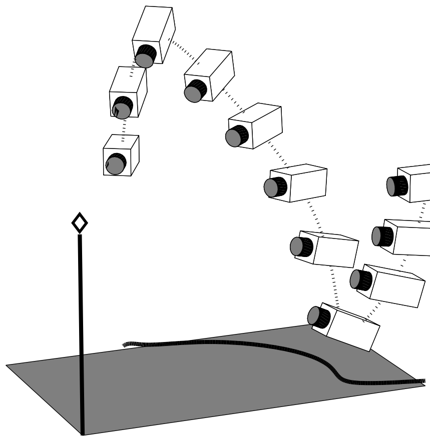

Figure 1.1: Moving Observer with a Camera Tracking a Stationary Object

The observer is composed of a single camera and an inertial rate sensor package (which is not shown). The two sensors are rigidly attached to each other. Each observer location denotes a different point in time. The observer moves subject to the constraint that the stationary object (denoted by ◇) remains in view of the camera. This motion is an integral part of the sensing strategy.

Underwater manipulation, the use of underwater vehicles to perform object manipulation tasks in the ocean, is the particular application that motivated this research. This powerful capability extends our ability to affect the ocean environment at depths that are unsafe for human divers. Underwater manipulation is used extensively for science-based exploration, like collecting samples from the seafloor, placing sensors, and maintaining equipment that has been deployed to build ocean observatories. In addition, there are many industrial and military applications for underwater manipulation (e.g., construction and maintenance of oil drilling equipment and telecommunication cables, and mine countermeasures).

Various types of underwater vehicles are in use for ocean exploration. Manned submersibles offer marine scientists a direct presence in the subsea environment, but their availability is limited by the cost of carrying humans safely into the high-pressure environment of the deep ocean. Remotely-operated vehicles (ROVs) provide an alternative for manned exploration. These vehicles are connected to a support vessel on the surface with a tether that provides high-bandwidth, two-way communication as well as power for extended operations. Very skilled pilots operate these vehicles using a teleoperation approach—commanding the vehicle thrusters and actuators from the surface based on live video signals from on-board cameras. While ROVs are in wide-spread use, their cost (about $10,000 per day, to cover the underwater vehicle as well as the support vessel and crew) is still a major limitation on scientific progress.

Untethered and unmanned vehicles, called Autonomous Underwater Vehicles (AUVs), have been developed more recently to reduce cost and to enable missions for which tethers and support vessels are not feasible. Without a tether, AUVs cannot be teleoperated by human pilots. Therefore, all aspects of AUV control, including task planning, navigation, sensing and low-level control of thrusters and actuators have to be autonomous. If human pilots are involved in AUV control, they do so at the supervisory level, via low-bandwidth acoustic modems, to monitor progress, to update mission goals, or to provide high-level interpretation of sensor data.

Developing robust autonomous sensing and control capabilities for AUVs has been extremely difficult, and as a result, the tasks that current AUVs perform are typically limited to navigation and obstacle avoidance. These capabilities define the survey-class of AUVs, which are used primarily to collect sensor profiles along predefined trajectories.

Currently, most underwater manipulation requires human involvement in the low-level control of the robot. Teleoperation of tethered vehicles is the state-of-the-art in underwater manipulation. However, even though ROV pilots can accomplish many impressive tasks, their work is difficult and very time-consuming. Much of this difficulty results from the large number of degrees-of-freedom (DOFs) that have to be controlled and the relatively limited sensor information that human pilots can absorb. The degrees-of-freedom include the 6-DOF motion (translation and rotation) of floating underwater vehicles as well as the manipulator joints (sometimes up to seven). Humans are very adept at interpreting video signals from on-board cameras as well as any available position measurements (like depth, altitude, and heading), but have greater difficulty with other useful sensor measurements, like joint angle measurements from manipulators and rate measurements.

Future missions could benefit from greater reliance on computer-based control of underwater manipulation tasks. While human pilots surpass computers in their ability to interpret tasks and to reason about the environment, computers have a clear advantage in controlling large numbers of coupled degrees of freedom and in assimilating disparate sources of sensor information. Computer control of underwater manipulation can lead to two useful capabilities: pilot-aids for tethered vehicles and autonomous underwater manipulation for untethered vehicles. Pilot-aides reduce the workload of human pilots—freeing them to focus on higher-level tasks—and potentially improve the efficiency and success rate of the current underwater manipulation capability. Autonomous underwater manipulation is a new capability for AUVs that could be applied when human pilots cannot participate directly in the low-level control of underwater vehicles.

Many useful applications of computer control for underwater manipulation could be developed, depending on mission requirements and vehicle capabilities. These applications include grasping, manipulation, and precise placement of objects. To execute all of these manipulation tasks, the underwater vehicle has to control the position of its manipulation tool relative to the object of interest. A robust and precise relative position sensor is required to enable this type of control. This dissertation presents a relative position sensing strategy suitable for floating underwater vehicles and describes an experiment that shows how this new sensing strategy could be used to implement a simple autonomous manipulation task.

1.2 Relative Position Sensing for Underwater Vehicles

There are two main sensor modalities that can be used to determine relative position in the underwater environment. Sonar is well suited for navigation and obstacle avoidance. It measures the range to natural objects as well as active beacons. The main advantage of sonar is its low attenuation in water, which permits long-range measurements.

Vision-based sensing is better suited for precise, short-range measurements, like those required for manipulation tasks. Consequently, this research has focused on a vision-based relative-position sensing system. Although the bearing resolution of sonar systems is improving rapidly, vision tends to have greater resolution than sonar, both in time and bearing angle. Although light attenuates more quickly than sonar in water, range measurements of several meters are possible with vision.

Vision-based sensing has been used very effectively for the control of robots in natural terrain (underwater, in space and on land). This is true for human-operated robots, for which camera images provide the principal source of operator information, and for autonomous robots, whose ability to extract useful information from images is gradually increasing. Vision is an effective sensor because it provides several types of information, like bearing, color and texture, and because it can be used for many tasks, like object modeling and identification, obstacle avoidance, user interfaces, and relative position sensing.

Vision is the primary source of sensor information for ROVs and manned submersibles, which are controlled by human pilots. These pilots visually track objects in the environment (e.g., sea floor, animals, equipment) either electronically through camera views or directly through port holes. The capabilities of these human-robot teams demonstrate that vision as a sensor provides sufficient information to perform very sophisticated tasks in the unstructured underwater environment.

However, replicating the impressive capabilities of the human vision system using computer-based vision-sensing techniques has been challenging. Nevertheless, new capabilities based on vision-based relative position sensing have been introduced on ROVs to assist human pilots. In 1995, Marks [35] introduced a computer-vision system to determine the position of an underwater vehicle along a planar surface (e.g., the sea floor or a canyon wall) using a monocular vision system. The system performs texture-based correlations of subsequent camera images to determine two-dimensional position offsets relative to a reference image. Range was determined with a sonar proximity sensor. This capability was used to build an autonomous station-keeping capability, which has been successfully demonstrated in numerous ocean trials [33]. Others have developed similar capabilities [17, 34, 38].

Rife and Rock [45] have successfully demonstrated an underwater stereo-vision system for determining the relative position between an underwater vehicle and a jelly fish. This sensor was used to construct an automatic control capability for autonomous tracking of a jelly fish with an underwater vehicle. These pilot-aides demonstrate the effectiveness of computer vision as a sensor for automatic control of underwater vehicles.

The main drawback of vision is that a single camera image does not provide a complete measurement of relative position to an unknown object in the environment. It provides only the bearing to features that have been identified in the field-of-view. Range information is lost. Relative position, which includes bearing and range to an object, must be determined by combining bearing information from multiple images using some form of triangulation. Alternatively, bearing information from vision can be combined with another source of range information to determine relative position.

Many techniques have been used in the laboratory and on operational robots to reconstruct relative position from camera measurements, including: stereo vision, in which two cameras with a fixed baseline observe the same feature; photogrammetry, in which a single camera observes multiple known features on the object; structured light, in which a light source and a camera each contribute a bearing to the feature; structure-from-motion, in which a camera subject to unknown motion observes multiple features from several locations; and motion-based tracking, in which a camera subject to known motion tracks a single feature in time.

While several important vision-based sensing capabilities have been demonstrated on operational underwater vehicles, the underwater environment still presents many significant challenges to successful implementation of computer vision. Among them are the unstructured scenery with a scarcity of strong visual features; non-uniform lighting; image noise; failures due to occlusions, visibility constraints, data association errors and detection errors; and the difficulty of maintaining accurate calibrations. All of these factors contribute to the overall robustness issues of vision-only systems.

Fusion of dissimilar sensor measurements provides an effective way to mitigate the robustness issues of vision-only systems. While vision measurements are required to provide relative position information, other sensors, like inertial rate sensors, Doppler velocimetry loggers (DVL, a velocity sensor), sonar-based sensors (altitude and other range measurements), depth and heading sensors (compass), can provide additional information about the position, orientation and motion of the vehicle. Combining vision measurements with measurements from other sensors typically results in improved performance and robustness of the sensing system.

1.3 Fusing Vision and Inertial Rate Sensors

This dissertation considers a sensing strategy based on fusing only vision and inertial rate sensor measurements. Inertial rate sensors are composed of accelerometers, which measure specific force (linear acceleration plus the apparent acceleration due to gravity) and rate gyros, which measure angular velocity. Inertial rate sensors are a good complement to vision because they are very robust, requiring no external measurements; they provide motion information, which is difficult to extract from vision; and they provide information for all six degrees-of-freedom using a single sensor package.

More complex systems that fuse even more sensors could be constructed. Additional information provided by more sensor measurements has the potential to improve the performance of the system. However, the use of additional sensor measurements also introduces additional failure modes and calibration parameters (intrinsic parameters of the transducer, like scale factors, as well as extrinsic parameters, like the position and orientation of the sensor relative to the other sensors). Consequently, system complexity generates significant costs to usability and robustness. Adding only inertial rate sensors to a vision system effectively addresses typical robustness issues of vision-only systems with a sensor that is inherently very robust.

A recursive state estimator is required to handle the fusion of the relative position information from vision with the rate information from the inertial sensors. In this sensor fusion algorithm, the measurements from the accelerometers have to be integrated twice to compute velocity and position and the measurements from the rate gyros have to be integrated to compute orientation. Unfortunately, inertial rate sensors are subject to time-varying biases and measurement noise. These errors result in growing drift errors in the integrated camera motion. Therefore, an estimator has to be designed to identify the drift errors, as well as the bias and noise errors which caused the drift. This is especially important for low-cost inertial rate sensors, which are subject to significant drift errors.

1.4 A New Relative Position Sensing Strategy

The relative position sensing strategy that is enabled by this research fuses measurements from inertial rate sensors with measurements from the simplest possible vision system: using a single camera to measure the bearing to a single visual feature on the object of

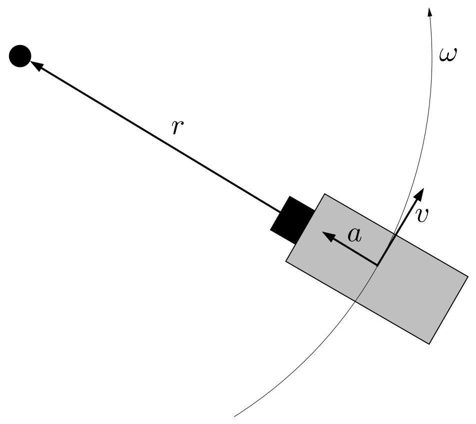

Figure 1.2: Simple Planar Example

interest. The choice of a very simple vision system was made to improve the potential robustness of the overall positioning system. By reducing the complexity of the vision system, the sensing system is susceptible to fewer vision outages.

This sensing strategy provides the relative position between a moving observer[1] and a stationary object, as shown in Figure 1.1. The motion of the observer between successive images generates a baseline for range computations by triangulation. The translation and rotation along this baseline is registered by the inertial rate sensors.

1.4.1 An Illustrative Example

To illustrate the concept of deriving relative position from a bearing measurement and inertial rate measurements, consider the simplified, planar example in Figure 1.2 in which the observer is moving in a circle centered at the object location. The observer could achieve this motion by pointing the optical axis of the camera directly towards the object while maintaining a constant velocity (v) perpendicular to and a constant acceleration (a) in the direction of the optical axis. At steady state, this would result in a constant rotation rate (ω) of the observer. In this case, $a = \omega^2 r$ and the range to the object (r) can be computed by dividing the centripetal acceleration by the square of the rotational velocity. Together, range and bearing describe a relative position. This example emphasizes the need for observer motion to generate observability of the object range.

In reality, the inertial rate sensors provide noisy and biased measurements, the observer motion cannot be controlled exactly (or at all), the observer motion is in three dimensions, and the accelerometer measurement contains a large, orientation-dependent contribution from gravity. Therefore, this simple approach does not provide a useful implementation

[1]In this dissertation, the term *observer* refers to the sensor platform that carries the camera and the inertial rate sensors, and not the estimation algorithm.

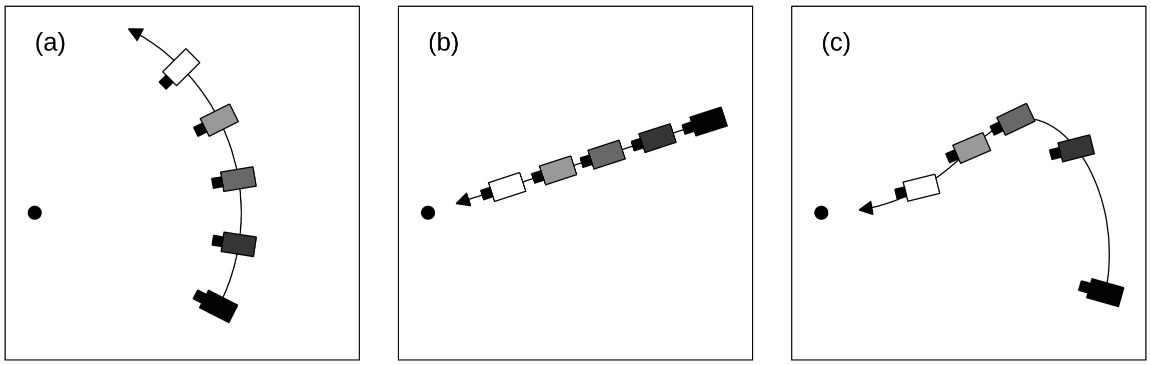

Figure 1.3: Example Observer Motions

of the sensing strategy. Consequently, this dissertation focuses on a more general sensor fusion approach that accommodates all of these challenges.

1.4.2 Observer Motion

The success of this sensing strategy depends on the motion of the observer. Motion generates the information required to determine relative position using an implicit triangulation approach. The bearing measurements obtained at multiple points of view combine with observer motion measurements to produce the relative position estimate. However, only the component of motion transverse to the line-of-sight produces new information about object range.

In addition to observability, the observer motion might also have to satisfy specific task requirements. For instance, if the sensing strategy is used to enable a manipulation task, the observer has a desired terminal position near the object and its motion has to satisfy that requirement.

Observability and task requirements can be competing objectives for the observer motion. Figure 1.3 displays three different observer motions. The observer is depicted with progressively lighter colors as it moves along the path. The object is shown as a black dot. The motion in (a) is optimized for observability because its circular path provides the maximum amount of motion transverse to the line-of-sight, which is critical for successful triangulation. However, the observer does not approach the object. The motion in (b) is the shortest path from the initial observer position to the goal position near the object. This type of motion might be optimal for moving the observer along the shortest path into position to manipulate the object, but does not generate any observability of the object range. The motion in (c) represents a compromise between the two objectives. At the beginning of the path, the observer moves primarily in a transverse motion to obtain a good estimate of object range. Towards the end of the path, the observer moves primarily towards the

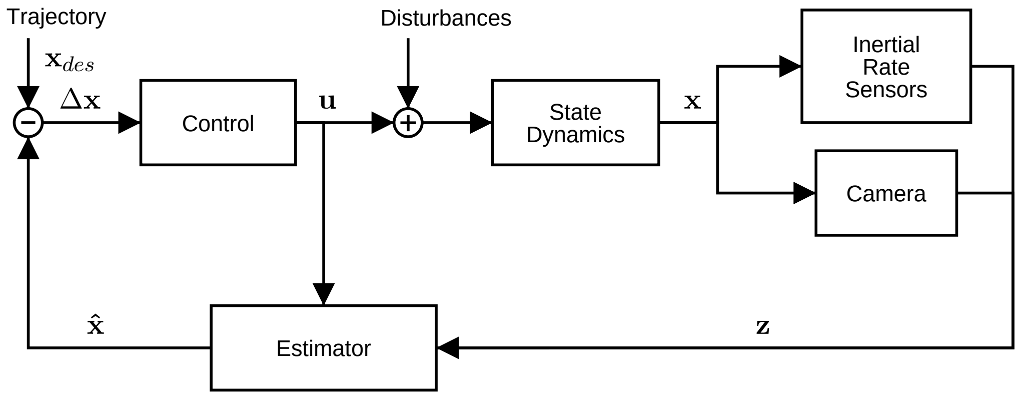

Figure 1.4: Overview of the Sensing Strategy with the Estimator, Trajectory, Controller, Observer Dynamics, and Sensors

object. Observability along the final portion of the path is reduced to allow the observer to reach its goal position.

A desired trajectory and a trajectory following controller are used in the sensing strategy to handle observer motion. While the sensing strategy could be used in applications for which the observer motion is uncontrolled, the trajectory approach, which is assumed throughout this dissertation, ensures sufficient observability and the ability to accomplish task-specific requirements. For the experiments presented in this dissertation, a simple trajectory has been designed. The design of optimal trajectories and the online generation of trajectories is an active area of research (see [14], which describes trajectory design for monocular vision-based target motion estimation).

1.4.3 Implementation of the Sensing Strategy

This approach to relative position sensing is called a sensing strategy because it requires a system-wide implementation. The observer motion, the observer actuator commands and dynamics, as well as the sensor measurements, are necessary for determining the relative position.

The implementation of this sensing strategy is shown in Figure 1.4. The block diagram includes the sensors, the estimator that is used to fuse the bearing and inertial rate sensor measurements, a trajectory that specifies desired relative position as a function of time, a relative position controller to execute that trajectory, and the state dynamics of the observer.

In Figure 1.4, the state of the system is represented by $\mathbf{x}$ and the state estimate is $\hat{\mathbf{x}}$. The trajectory specifies the desired state $\mathbf{x}_{des}$ as a function of time. The difference between the desired state and the state estimate is used by the controller to compute the actuator commands $\mathbf{u}$ for the observer. The vector $\mathbf{z}$ contains the sensor measurements.

1.4.4 Estimation Problem

The main contribution of this research is the design of a new recursive state estimator, which is required for fusing vision and inertial rate sensor measurements to determine relative position. This estimation problem includes two important nonlinearities: the projection nonlinearity in the camera sensor model and the geometric nonlinearity from the rotational motion of the observer. Therefore, a nonlinear estimator has to be designed. However, standard nonlinear estimation tools, like the Extended Kalman Filter (EKF), fail to provide an adequate solution.

The EKF uses linearization to approximate the nonlinear system equations. This approximation is based on the assumption that the estimate uncertainty is small. While the EKF provides good solutions for a broad range of nonlinear estimation problems, it fails for some problems because the estimate uncertainty cannot be assumed to be small. In this case, the linearization step introduces estimate and covariance errors that can cause the estimate to be strongly biased and cause the algorithm to diverge.

This sensor fusion problem causes problems for the EKF because convergence, which depends on observer motion, can be slow, which implies large estimate uncertainties. For example, the initial estimate does not converge until sufficient observer motion has occurred for the implicit triangulation to provide useful information about the object range. During this time, the estimate uncertainty remains at the value of the initial estimate uncertainty, which can be large. Consequently, the covariance of the EKF, which is sensitive to the size of the uncertainties, can be corrupted and the state estimate can converge to a false object range.

Therefore, a new estimator design that can produce accurate results for this sensor fusion problem is required. The new design is based on a Kalman Filtering approach with two key modifications. First, the nonlinear system equations are expressed with a specific representation that has a linear sensor model. This obviates linearization for the measurement update step of the Kalman Filter and leads to a more accurate solution. Second, the Unscented Transform, a relatively new technique for handling nonlinear stochastic problems without linearization, is used to implement the time update step of the estimator.

This estimator design solves the sensor fusion problem and enables the real-time implementation of the new sensing strategy.

1.5 Object Pick-Up Task

An object pick-up task has been defined to demonstrate the capabilities of the new sensing strategy. In this demonstration, the sensing strategy is used to enable a simple manipulation task. Although relative position sensing is a critical capability for many underwater

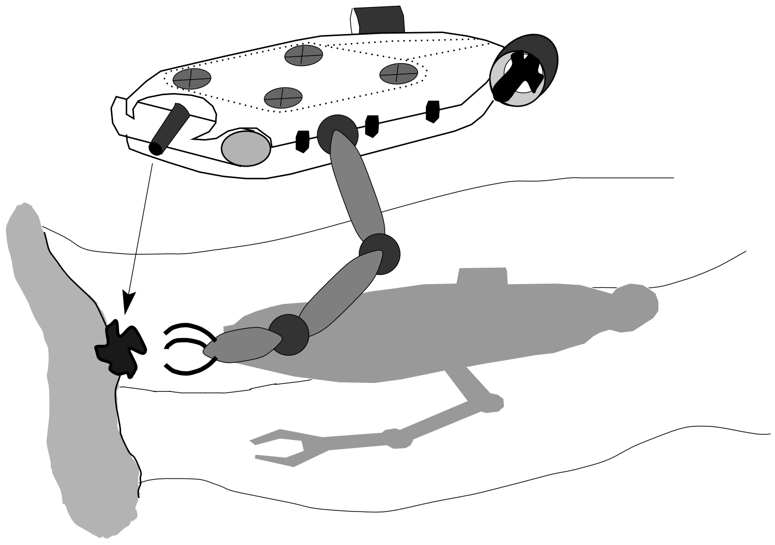

Figure 1.5: Object Pick-Up Task

vehicle control applications—including station-keeping, tracking and observation—object manipulation generates enhanced requirements on accuracy, robustness, and motion constraints. This demonstration task simulates the following application: a hover-capable, floating underwater vehicle is used to retrieve an object from the seafloor. This could be an instrument that has been deployed previously or a natural object of scientific value. The state-of-the-art in underwater manipulation requires ROVs to perform this type of task—this sensing strategy enables *autonomous* execution of the object pick-up task.

To perform a successful capture, the underwater vehicle has to determine its position relative to the stationary object and move to a desired relative position with sufficient accuracy to perform a successful grasp of the object. This scenario is illustrated in Figure 1.5.

The object pick-up task is implemented in the laboratory with a 7-DOF serial-link manipulator arm that is used to simulate the motion and manipulation capabilities of an underwater vehicle. Figure 1.6 shows the K-1607 manipulator by Robotics Research Corporation in position to pick up a cup, placed at a position unknown to the robot. The manipulator endpoint has a gripper for grasping the cup as well as a camera and inertial rate sensors. This cluster of sensors and the gripper on the endpoint simulate the underwater vehicle. The links and joints of the manipulator as well as the forces they transmit simulate the combined action of the vehicle thrusters and any disturbance from the underwater environment.

For the demonstration task to be realistic, the manipulator endpoint has to move according to the dynamics of an underwater vehicle. This can be done with a model following controller. This controller applies the assumed dynamics of an underwater vehicle to

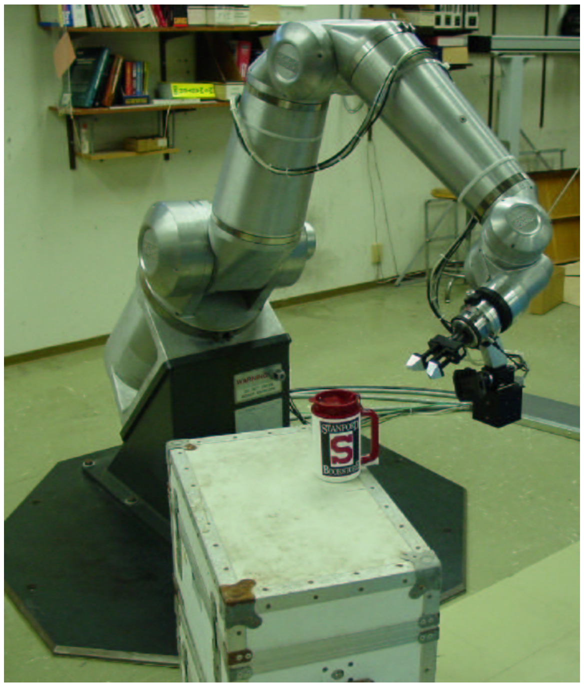

Figure 1.6: Manipulator Arm for Object Pick-Up Experiment

the vehicle disturbances and control commands to compute the desired vehicle motion. It then commands manipulator joint velocities that result in equivalent manipulator endpoint motion.

Figure 1.7 shows how the control system for the manipulator experiment maps into the State Dynamics block of Figure 1.4. $\mathbf{x}_{sim}$ represents the state of the system due to the input $\mathbf{u}_{obs}$ if the system evolves according to the dynamics model for the observer. The difference between $\mathbf{x}_{sim}$ and the actual manipulator state $\mathbf{x}$ drives the Endpoint Pose Controller, which generates the desired manipulator joint velocities $\mathbf{u}_{manip}$. These desired joint velocities are passed to the manipulator and its control system, which generates the new state $\mathbf{x}_{.}$ The motion capabilities and control authority of the manipulator readily exceed the requirements for simulating the motion of a hovering underwater vehicle.

The joint angle measurements from the manipulator are used to compute the actual endpoint motion. This can be used as a truth measurement to evaluate the sensing strategy and is an integral part of the model following controller.

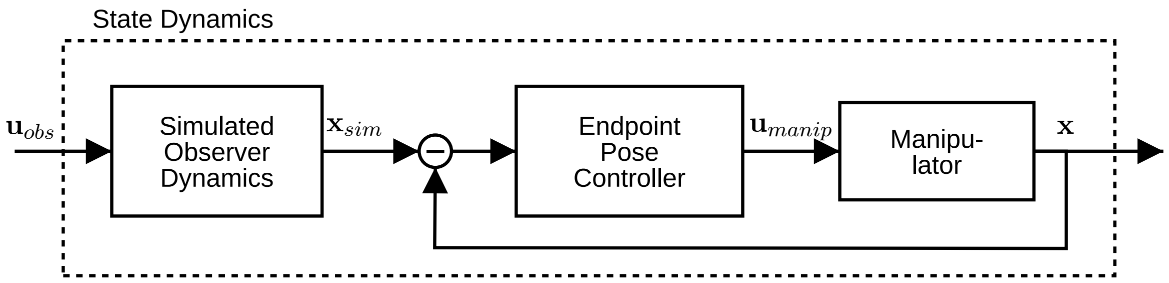

Figure 1.7: Simulated Observer Motion with the Manipulator Arm

1.6 Research Results

The design of a nonlinear estimator to solve the sensor fusion problem has enabled a new relative position sensing strategy based on fusing monocular vision measurements of a single feature and inertial rate sensor measurements. Simulations and hardware experiments demonstrate that the sensing strategy is feasible and that it can be used by an autonomous robot to implement a useful, closed-loop manipulation task.

The research addresses the difficulty in designing an estimator to solve the sensor fusion problem. Standard nonlinear estimation tools like the EKF failed to provide an adequate solution. By developing a modified Kalman Filter solution that does not use linearizations of the nonlinear system equations, a solution that works well for this problem has been attained. Simulations show that the estimator correctly computes the predicted standard deviation of the estimate error, which is a useful indicator for the performance of the estimator. The success of the hardware experiments confirms that the models and assumptions used in the estimator design are reasonable.

The main limitation of this sensing strategy is that its performance depends strongly on the size of disturbance forces, like ocean currents, and errors in the dynamics model of the observer. As the disturbances and model errors increase, the precision of the relative position estimate decreases. The significance of this conclusion depends on the application. When the observer models are accurate and the external disturbances are low, as is the case for the hardware experiment presented in this dissertation, then centimeter-level accuracy is easy to achieve. If the dynamics models are poorly understood and external disturbances are significant, as might be the case for a real underwater vehicle, then the performance might not be sufficient to perform precise manipulation tasks.

This limitation is caused primarily by the minimal sensor information that is fused to determine relative position. The goal of the research was to examine a sensing strategy that fuses inertial rate sensor measurements with only the simplest possible vision requirements. As a result, the sensing strategy has to rely heavily on the predictions generated by the dynamics model. The analysis of performance that was enabled by the new estimator design indicates that these sensors are sufficient to determine precise relative position, but only if disturbances are low and model errors are small. Although the potential exists for improving the estimator design to boost performance—both by applying different (and possibly more complex) algorithms and by increasing the accuracy of the models—the estimator is not the dominant cause for this limitation.

The recommended solution to this problem is to consider a modified sensing strategy which incorporates additional sensor measurements. A variety of options exist and are discussed in Chapter 7.

1.7 Related Research

This section introduces areas of research that are related to the research presented in this dissertation. Section 1.7.1 focuses on the general topic of relative position sensing for field robots. It reviews application areas and common sensors, discusses the relationship to absolute position sensing, and introduces the related topic of simultaneous localization and mapping. Other approaches for fusing vision and inertial rate sensors and approaches for determining relative position with a single bearing sensor are presented. The work of two groups that have also pursued a bearing plus inertial approach is described.

Section 1.7.2 covers related work in the area of autonomous manipulation. It describes the topics that comprise the broad area of autonomous manipulation and explains that a real-time relative position sensor is necessary to extend autonomous manipulation capabilities developed for land-based robots to underwater robots. It also reviews previous work in enabling autonomous manipulation capabilities for underwater vehicles.

An expanded discussion of related work in the design of recursive estimators for nonlinear problems is presented in Chapter 3.

1.7.1 Relative Position Sensing For Field Robots

The deployment of useful field robot capabilities is a very challenging problem, particularly in the area of relative position sensing in unstructured and unprepared environments. Consequently, the development of new sensing approaches for determining the relative position between a robot and objects in its environment is an active area of research. The purpose of relative position sensing includes obstacle avoidance, mapping (building a catalog of objects and their positions), localization (determining the robot position relative to mapped objects), and relative position control (controlling the position of the robot relative to an object to enable observation, modeling and manipulation). Relative position sensing has been identified as a challenge not only for underwater vehicles, but also for Mars rovers, unmanned aerial vehicles, underground mining vehicles [47] and service robots.

Many different types of sensors have been used to perform relative position sensing for field robots. These include vision (in all of its many forms, including multi-camera systems, motion-based systems, and structured light), laser scanners, radar, ultrasonic and tactile sensors. Often, sensor fusion is used to incorporate additional information, like inertial rate sensors and odometry. Redundant sensor information helps to develop robustness against the inherent failure modes of relative position sensors in real environments.

Absolute Position Sensing

The widespread availability of the global positioning system (GPS) has enabled many new field robot capabilities. GPS provides a low-cost, accurate measurement of robot position in global coordinates. Differential measurements can be used to describe robot motion. Although GPS is available to underwater vehicles while they are at the surface, the GPS signals do not penetrate to any useful depth in the ocean. In some cases, long-baseline (LBL) systems, which use sonar transponders, are available to provide absolute position measurements for underwater vehicles.

Although these positioning systems are very useful for the control of field robots, they do not replace the need for relative position sensing systems to perform obstacle avoidance, mapping, localization and relative position control. First, GPS and LBL systems provide absolute position and not the position relative to an object of interest. Second, they are not always available. GPS generally requires line-of-sight to several satellites and is susceptible to multipath interference. LBL systems operate only over a limited area and are very expensive to deploy.

Simultaneous Localization and Mapping

For some applications, the robot operates in an unknown environment and needs to map objects of interest while maintaining an estimate of its position relative to the objects that have already been mapped. This problem is straight forward if the robot can observe all of the objects at the same time or if it has an external position reference. But without an independent position measurement, the process of simultaneous localization and mapping (SLAM) generates a challenging estimation problem. Although the theoretical basis for SLAM is well understood (e.g. [11]), managing computational complexity and data association issues are areas of active research. Several applications of SLAM for field robots have been presented [13, 43, 58].

Bearing-only SLAM [9, 10] is a subset of SLAM using sensors that provide only bearing measurements and no range information. The bearing-only SLAM problem is closely related to the structure-from-motion problem in computer vision.

Fusing Multiple-Feature Vision and Inertial Rate Sensors

Although many vision-only techniques have been developed to determine relative position, several researchers have recognized the value in fusing vision measurements and inertial rate sensor measurements to improve the robustness, performance and computational cost of vision-only approaches. Usually, this work begins with a complete vision solution (i.e., the system has sufficient observability without the inertial rate sensors), which typically involves tracking more than one feature. Rate gyro measurements are then fused to simplify the problem of separating the contribution to image-plane feature motion due to camera translation and camera rotation. Sometimes, accelerometer measurements are fused to smooth the computation of camera velocity. Bhanu, Das, Roberts and Duncan [2] have used this approach to develop an obstacle avoidance system for low-flying rotorcraft. You, Neumann and Azuma [60] have developed an augmented reality system based on fusing vision and rate gyro measurements. Strelow and Singh [54] fuse measurements from multiple image features and inertial rate sensors to determine observer motion.

Although these systems perform better than vision-only approaches, they still incorporate the complexity of a complete vision solution. In contrast, this dissertation presents a system that requires only a very simple vision system (i.e., monocular vision measurements of a single feature) by fusing inertial rate sensors. The ability to reduce the vision requirements improves the potential robustness of the overall sensing system.

Target Motion Analysis

Target motion analysis (TMA) is a widely deployed capability that uses only a single bearing sensor together with observer motion to determine the relative position and velocity between an observer and a maneuvering target. TMA is generally used with passive sonar and passive radar measurements, which provide bearing to the target, but not its range. An extension to vision-based TMA is presented in [14]. The main difference between TMA and the sensing strategy described in this dissertation is that the observer motion is assumed to be known in TMA. This can be achieved with the use of an absolute positioning system like GPS or an LBL system. Much of the research in TMA is currently focused on the design of optimal trajectories that minimize the required observer motion while achieving sufficiently accurate target motion estimates.

Fusing Bearing and Inertial Rate Sensors

Two groups have recently discussed the problem of fusing measurements from inertial rate sensors and only one bearing sensor to determine relative position. They are Kaminer, Kang, Yakimenko and Pascoal [30] (airplane tracking a ship) and Gurfil and Kasdin [18] (missile intercepting a target). Both groups have made significant simplifying assumptions for this problem. They have assumed that the inertial rate sensor biases are not time-varying and that they have been perfectly calibrated. They assume that an independent, accurate measurement of observer orientation is available, which reduces the size of the problem and enables perfect gravity compensation in the accelerometer measurement. Both groups have presented only simulation work for which these assumptions can be satisfied easily.

However, even though both groups have worked with simplified systems, they have also recognized the difficulty of the estimation problem and have focused their research on developing new estimator designs that provide accurate solutions for this sensor fusion problem. The solution in Kaminer et. al. is based on exploiting the structure of the simplified problem and casting it as a linear parametrically varying system. To provide additional observability and performance, they have also developed an extension to a stereo system with two cameras as well as accelerometers. The solution by Gurfil and Kasdin uses the two-step estimator, a design which is also based on the choice of system representation with a linear sensor model.

However, the assumptions that were made to simplify the estimation problem in this related work do not hold for the application of interest in this dissertation. In particular, the observer orientation is not known exactly and has to be estimated from sensor measurements. Because the orientation estimate is uncertain, gravity compensation of the accelerometer measurements is also uncertain and has to be accounted for in the estimator design. This has a significant effect on the performance of the sensing strategy. Furthermore, the inertial rate sensor biases are time-varying, especially for low-cost inertial rate sensors, so these biases have to be estimated on-line as part of the solution. Consequently, the solution presented in this dissertation is capable of handling more complex and realistic sensor fusion problems and represents an extension of the previous work. This extension was required to enable a hardware demonstration of the sensing strategy.

1.7.2 Autonomous Manipulation

Autonomous manipulation is an active area of research with important applications for factory automation, service robots, Mars rovers, robots that operate in hazardous environments and underwater vehicles. Depending on the application, this capability requires fundamental advances in relative position sensing, object modeling, reasoning, task planning, motion planning, grasp planning and manipulator control. Many researchers are working on one or more of these components. Some have described full end-to-end demonstrations of autonomous manipulation of known objects (e.g., [50]) and unknown and *a priori* unmodeled objects [39, 49, 52].

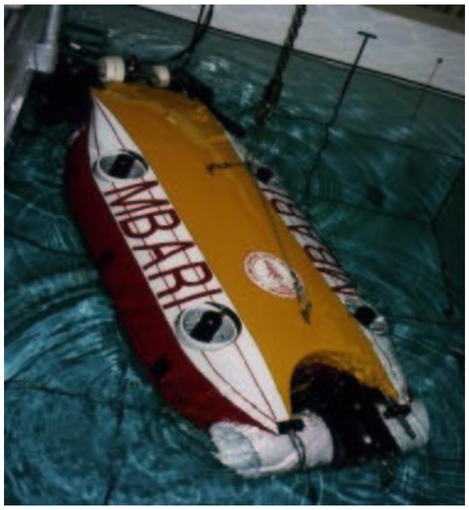

Figure 1.8: OTTER: A Small Underwater Vehicle Operated in MBARI's Test Tank

Computer vision is often used to determine the relative position between the robot and the object—both for feedback control and task specification. Visual servoing of robot manipulators [6, 19], both image-based and position-based, has been implemented successfully for many autonomous manipulation tasks.

In principle, most of the research on autonomous manipulation can be extended to autonomous underwater manipulation. However, because much of the autonomous manipulation research is focused on land-based robots, it tends to assume that the robot is fixed in inertial coordinates during manipulation tasks. In this case, two useful assumptions are valid: first, the motion of the manipulator and the relative position sensor can be measured easily and accurately, or they can be held stationary; second, a look-and-move approach, in which the robot can carefully interpret sensor information about the object before committing to motion, is feasible. This is not the case for autonomous manipulation by floating underwater vehicles. These vehicles are not stationary during the manipulation task and their motion is difficult to measure. Furthermore, the control of the manipulator has to incorporate real-time sensing to adjust for the motion of the floating vehicle.

The relative position sensing strategy presented in this dissertation provides a real-time estimate suitable for position control of a floating underwater vehicle relative to the object of interest. This estimate can be used to extend autonomous manipulation capabilities (such as online object modeling, motion planning, grasp planning, and manipulator control) that have been developed for land-based robots to make them suitable for floating underwater vehicles.

In 1996, Wang [57] introduced intervention-capable AUVs, a new class of AUVs that can manipulate objects in their environment. Wang demonstrated that OTTER (see Figure 1.8),

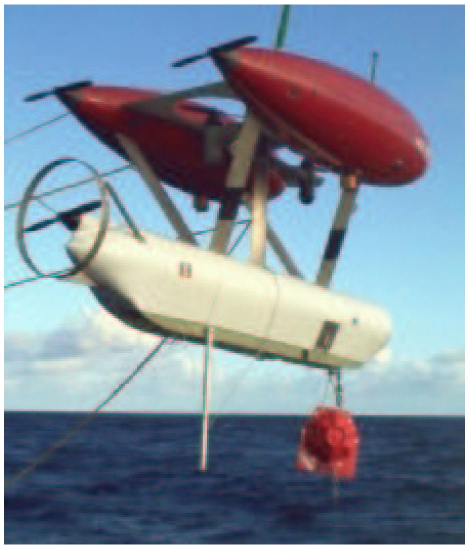

Figure 1.9: Autonomous Benthic Explorer of the Woods Hole Oceanographic Institution [59]

a small test-tank AUV, could autonomously search for, grasp, and retrieve an object. This retrieval task requires a direct relative position measurement between the AUV and the object, which Wang implemented with a vision-based stereo system. Vision-based manipulator control for an underwater vehicle has also been demonstrated by Smith, Yu, Sarafis and Lucas [51].

Yoerger et. al. [59] demonstrated the first sample retrieval task with an ocean-going AUV. The Autonomous Benthic Explorer (ABE, see Figure 1.9) sampled volcanic glass by pushing a boom into the sea floor. This task did not involve a specific object and was accomplished with a navigation sensor. Although this was a very simple task, it demonstrates the utility of intervention-capable AUVs.

The Semi-Autonomous Underwater Vehicle for Intervention Missions [61, 62] is an active program to develop an ocean-going AUV with manipulation capabilities.

1.8 Contributions

This dissertation presents several contributions that support the development of the new sensing strategy.

- **Sensing Strategy**
  A new sensing strategy that fuses monocular vision measurements of a single feature and inertial rate sensor measurements to determine the relative position between a

moving observer and a stationary object has been enabled. The sensing strategy addresses potential robustness issues of vision-only approaches by fusing a vision system with very simple requirements (one bearing measurement) with observer motion data from the inertial rate sensors.

- **Estimator Design**
  Enabling this sensing strategy required the design of a new nonlinear, recursive state estimator that has been adapted to address the unique requirements of the sensor fusion problem. The estimator design is based on a modified Kalman Filter approach that avoids the use of linearization by choosing a specific system representation that has a linear sensor model and by incorporating the Unscented Transform to implement the time update for the nonlinear process model.

  Simulation results have demonstrated that the new estimator design provides an adequate solution to this sensor fusion problem. In particular, the estimator accurately predicts the standard deviation of the estimation error. This is a key indicator that the estimator design can handle the combination of nonlinear system equations and elevated estimate uncertainties that is present in this estimation problem. Consequently, the mean estimate errors tend to be small.

- **Performance Analysis**
  An analysis has been developed to determine the performance limitations of this sensing strategy. The analysis showed that the estimate accuracy depends strongly on the size of disturbance forces and errors in the dynamics model. This is a consequence of the limited observability afforded by the sensor measurements available in this sensing strategy. Useful applications of this sensing strategy have been presented. However, when the accuracy of the relative position estimate is not sufficient for a given application, additional sensor measurements need to be incorporated into the sensing strategy to improve performance.

- **Hardware Experiment**
  The first hardware demonstration of a manipulation task using the new sensing strategy was performed. To enable this demonstration, a laboratory testbed that integrates a real-time implementation of the estimator with monocular vision, low-cost inertial rate sensors, a robotic manipulator, a trajectory generator, a controller and truth sensors was constructed. The experiment demonstrates the effectiveness of the sensing strategy in a system with real sensor measurements, model and calibration errors, and time delays. It also demonstrates how the sensing strategy can be integrated into a simple and useful closed-loop manipulation task.

1.9 Reader's Guide

The remainder of this dissertation is organized into six chapters:

Chapter 2—System Equations presents the system equations that define the sensing strategy and describes the estimation problem that has to be solved to implement the sensing strategy.

Chapter 3—Recursive Nonlinear Estimation introduces several estimator designs and compares the performance of a subset of these based on a very simple example which captures the important features of the full sensor fusion problem. This comparison explains the failure of the EKF and helps to suggest a new design strategy.

Chapter 4—Estimator Design presents the new estimator design for the full sensor fusion problem.

Chapter 5—Experimental System describes the experimental system and the demonstration task that were used to perform the hardware experiments. It also defines an analogous simulation system and presents the trajectory design for the demonstration task.

Chapter 6—Results presents the simulation and experimental results, along with their interpretation.

Chapter 7—Conclusions provides conclusions and suggestions for future work.

Chapter 2

System Equations

The introduction has described a new sensing strategy for determining the relative position between between a moving observer and a stationary object. This sensing strategy is based on fusing monocular vision measurements of a single feature (a bearing measurement) and inertial rate sensor measurements. It is called a sensing strategy because it requires a system-level implementation that depends not only on the sensor measurements, but also on the dynamics and the motion of the observer.

This chapter provides the system equations that define a specific implementation of this sensing strategy. This provides the context in which various aspects of the sensing strategy can be discussed. First, the geometry and reference frames to describe the demonstration task are presented. Next, suitable models are derived for the bearing and inertial rate sensors, for the dynamics of the observer, for its actuators, and for the disturbances acting on the observer. These components can be combined to generate a complete mathematical description of the demonstration task and the associated estimation problem.

The choice of models to represent the parts of the sensing strategy are an important factor in the design of the solution and its overall complexity. Models are required for the camera, the inertial rate sensors, the observer dynamics, and the external disturbances acting on the observer. These models were chosen to be as simple as possible while capturing the significant aspects of the types of sensors that are expected on typical underwater vehicles and the types of dynamics and disturbances that are expected in the underwater environment.

For some applications, more sophisticated models might be appropriate. These would require modifications to the estimation problem described in this chapter. However, the focus in the dissertation is on the generic features of the sensing strategy and not on the particular choice of models. Therefore, one of the goals for the estimator design presented in Chapter 4 is to generate a flexible solution that can easily accommodate changes to the particular system equations as they are defined in this chapter.

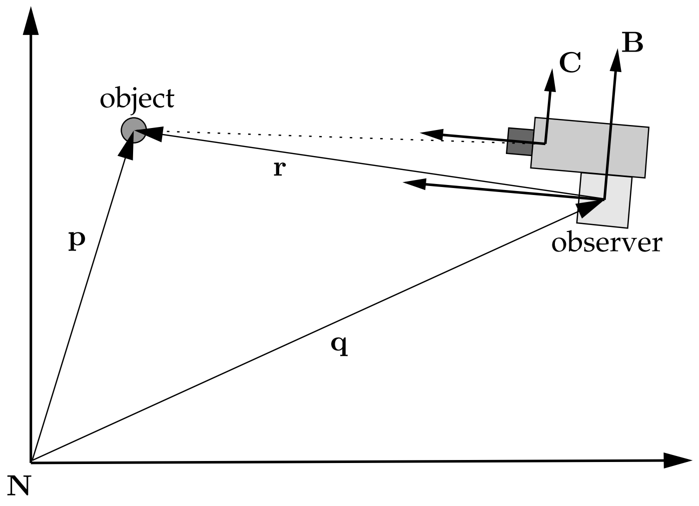

Figure 2.1: Geometry and Reference Frames for the Demonstration Task

Sections 2.1 to 2.3 define the models. Section 2.4 collects all of these models to write the complete system equations. Section 2.5 defines the estimation problem that has to be solved to implement the sensing strategy.

2.1 Geometry and Reference Frames

This section presents the geometry and reference frames to support the development of equations for the sensor fusion problem. Figure 2.1 shows the stationary object and the moving observer. Frame $\mathbf{N}$ is the inertial frame and $\mathbf{B}$ is the body frame of the moving observer. Frame $\mathbf{B}$ is fixed to the inertial rate sensors. The camera frame $\mathbf{C}$ is fixed with respect to Frame $\mathbf{B}$ and aligned with it. P_{cam} is the constant offset of $\mathbf{C}$ expressed in $\mathbf{B}$. The z-axes of the body and camera frames are aligned with the optical axis of the camera. A leading superscript (e.g., ${}^{B}\mathbf{x}$) indicates that the vector is expressed in a specific frame.

The vector $\mathbf{p}$ indicates the position of the tracked feature on the stationary object. The vector $\mathbf{q}$ is the position of the moving observer. The position of the feature relative to the observer is $\mathbf{r} = \mathbf{p} - \mathbf{q}$. The feature is stationary in the inertial frame, so $\dot{\mathbf{p}} = \ddot{\mathbf{p}} = 0$. Therefore, $\dot{\mathbf{r}} = -\dot{\mathbf{q}}$ and $\ddot{\mathbf{r}} = -\ddot{\mathbf{q}}$. Because of this assumption, a measurement of the observer acceleration $\ddot{\mathbf{q}}$ is directly useful for estimating the relative feature position $\mathbf{r}$.

The vector $\boldsymbol{\lambda}$ represents the orientation of the body frame relative to the inertial frame using the *roll-pitch-yaw* Euler angles (or X-Y-Z fixed angles in [7, Appendix B]). R is the rotation matrix that transforms a vector expressed in inertial coordinates to the body frame. The vector $\boldsymbol{\omega}$ is the associated rotational velocity expressed in the body frame. The following equations present all of the relationships required in this dissertation to represent orientation and angular velocity.

$$\begin{aligned}
\boldsymbol{\lambda} &= \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} roll \\ pitch \\ yaw \end{bmatrix} && (2.1)\\
R &= R_{BN}(\boldsymbol{\lambda}) && (2.2)\\
&= \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ -\sin\psi\cos\phi+\cos\psi\sin\theta\sin\phi & \cos\psi\cos\phi+\sin\psi\sin\theta\sin\phi & \cos\theta\sin\phi \\ \sin\psi\sin\phi+\cos\psi\sin\theta\cos\phi & -\cos\psi\sin\phi+\sin\psi\sin\theta\cos\phi & \cos\theta\cos\phi \end{bmatrix} && (2.3)\\
\frac{dR}{dt} &= -\boldsymbol{\omega}\times R && (2.4)\\
\frac{d\boldsymbol{\lambda}}{dt} &= E(\boldsymbol{\lambda})\, R^T\boldsymbol{\omega} && (2.5)\\
E(\boldsymbol{\lambda}) &= \begin{cases} \begin{bmatrix} 0 & 0 & sgn(\theta)/2 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1/2 \end{bmatrix} & \text{if } \cos\theta \text{ is small} \\ \begin{bmatrix} \cos\psi/\cos\theta & \sin\psi/\cos\theta & 0 \\ -\sin\psi & \cos\psi & 0 \\ \sin\theta\cos\psi/\cos\theta & \sin\theta\sin\psi/\cos\theta & 1 \end{bmatrix} & \text{otherwise} \end{cases} && (2.6)
\end{aligned}$$

Observer orientation is composed of attitude and heading. Attitude is the tilt away from the gravity vector and is captured by the *roll* and *pitch* angles. Heading is the rotation about the gravity vector and is captured by the *yaw* angle.

2.2 Sensor Models

2.2.1 Camera Model

The camera is represented by the standard perspective projection model shown in Figure 2.2. The x-, y-, and z-axes define the camera frame **C** and the coordinate system of the camera. The z-axis corresponds to the camera's optical axis. The image plane is shown at $z = f$. It is assumed, without loss of generality, that the camera measurements are scaled such that the effective focal length is 1. Therefore, an object located at $\mathbf{S} = \begin{bmatrix} S_x & S_y & S_z \end{bmatrix}^T$ in the camera frame appears as an image plane feature at $s_x = S_x/S_z$ and $s_y = S_y/S_z$. The resulting camera model in terms of ${}^N\mathbf{r}$ is

$$\mathbf{S} = \begin{bmatrix} S_x \\ S_y \\ S_z \end{bmatrix} = R\,{}^N\mathbf{r} - P_{cam} \qquad (2.7)$$

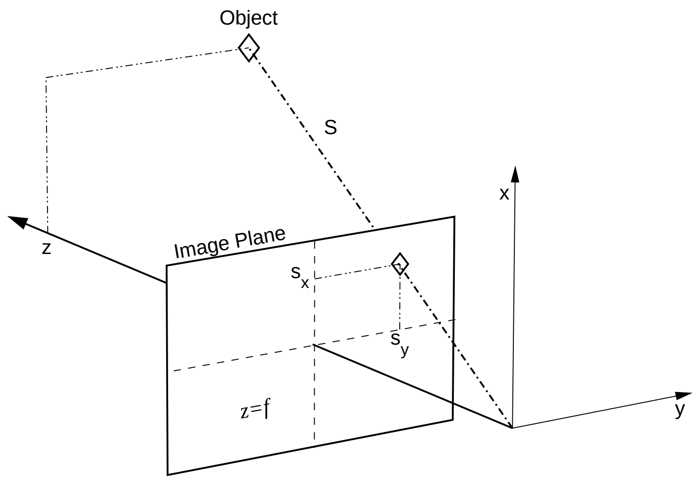

Figure 2.2: Perspective Projection Camera Model

$$\mathbf{z}_s = \begin{bmatrix} s_x \\ s_y \end{bmatrix} + \mathbf{n}_s = \frac{1}{S_z}\begin{bmatrix} S_x \\ S_y \end{bmatrix} + \mathbf{n}_s \tag{2.8}$$

where $\mathbf{n}_s$ is zero-mean white Gaussian measurement noise.

A bearing is usually defined to be an angle, but for this work, it is useful to generalize the concept to include the perspective geometry of the camera model. In this model, objects in the 3D world are projected onto a plane rather than a sphere or cylinder. A pixel measurement on the image plane could be converted easily to an angle with the tangent function. However, the equations are simpler without this conversion. Therefore, a pixel measurement is considered to be a type of bearing measurement. Range usually means the Euclidean distance between observer and object. However, a more useful quantity in the context of the perspective geometry camera model is the distance along the optical axis, which is the interpretation used throughout this work.

2.2.2 Other Bearing Sensors

This sensing strategy is presented in the context of monocular vision from a camera with perspective projection. But it applies equally well to other bearing sensors including omnidirectional cameras, passive radar and passive sonar.

2.2.3 Inertial Rate Sensor Models

The models for the inertial rate sensors are derived from those in [15]. The accelerometer measures specific force in body coordinates, which includes the acceleration $\ddot{\mathbf{q}}$ of the observer and an apparent acceleration due to gravity $\left(\mathbf{g} = \begin{bmatrix} 0 & 0 & -g \end{bmatrix}^T\right)$. The accelerometer measurement $\mathbf{z}_a$ also includes sensor biases $\mathbf{b}_a$ and zero-mean white Gaussian sensor noise $\mathbf{n}_a$.

$$\mathbf{z}_a = \boldsymbol{\alpha} R(\boldsymbol{\lambda}) \left(-\ddot{\mathbf{q}} + \mathbf{g}\right) + \mathbf{b}_a + \mathbf{n}_a \tag{2.9}$$

$\boldsymbol{\alpha} \approx I_{3\times3}$ represents scale factor errors induced by the sensor. The estimator design includes two simplifying assumptions on the accelerometer model. First, it is assumed that $\boldsymbol{\alpha} = \alpha I_{3\times3}$. Second, it is assumed that the scale factor variation can be ignored for the contribution due to real accelerations, which is much smaller than the contribution from gravity (i.e., $\alpha\ddot{\mathbf{q}} \approx \ddot{\mathbf{q}}$). This assumption has a small impact on overall accuracy and greatly simplifies the estimator design. The modified accelerometer model is

$$\mathbf{z}_a = R(\boldsymbol{\lambda}) \left(-\ddot{\mathbf{q}} + \alpha\mathbf{g}\right) + \mathbf{b}_a + \mathbf{n}_a \tag{2.10}$$

The rate gyro measurement includes the rotational velocity $\boldsymbol{\omega}$ of the observer, sensor biases $\mathbf{b}_\omega$, and zero-mean white Gaussian sensor noise $\mathbf{n}_\omega$.

$$\mathbf{z}_\omega = \boldsymbol{\omega} + \mathbf{b}_\omega + \mathbf{n}_\omega \tag{2.11}$$

Random walk models are used to capture the dynamics of the inertial rate sensor parameters. $\mathbf{n}_{ba}$, $\mathbf{n}_{b\omega}$ and n_α are zero-mean white Gaussian driving terms. These trivial models are sufficiently accurate for the short-duration experiments described in Section 5.2.

$$\frac{d}{dt}\mathbf{b}_a = \mathbf{n}_{ba} \tag{2.12}$$

$$\frac{d}{dt}\mathbf{b}_\omega = \mathbf{n}_{b\omega} \tag{2.13}$$

$$\frac{d}{dt}\alpha = n_\alpha \tag{2.14}$$

2.3 Actuator and Disturbance Models

The actuator and disturbance models are required by the estimator to predict the motion of the observer given the current state and control inputs. For an underwater vehicle, the disturbances are caused primarily by drag on the vehicle from ocean currents. The control inputs drive thrusters on the vehicle which generate force and torque components depending on how the thrusters are arranged.

The success of the sensing strategy depends strongly on the accuracy of these models and on the size of the disturbances. While the size of disturbances is determined by the operational environment, the accuracy of models could be optimized. However, accurately determining the dynamics of underwater vehicles is difficult and results in very complicated models. Furthermore, even small changes to the vehicle can cause significant deviations in the model.

This research has assumed very simple models that capture the first-order characteristics of the disturbances and the vehicle dynamics. These models have enabled the design of an estimator and the evaluation of the sensing strategy. Developing more sophisticated models and assessing the trade-off between complexity and performance has been identified as future work in Chapter 7.

The observer command input is $\mathbf{u} = \left[\begin{array}{cc} \mathbf{u}_1^T & \mathbf{u}_2^T \end{array} \right]^T$, where $\mathbf{u}_1$ is a vector of force inputs and $\mathbf{u}_2$ is a vector of torque inputs. Both are expressed in observer body coordinates and in units of m/s^2 and rad/s^2, respectively.

A linear drag model relates the control input $\mathbf{u}_1$ to observer velocity.

$$\frac{d}{dt}\dot{\mathbf{q}} = \ddot{\mathbf{q}} = R(\boldsymbol{\lambda})^T \mathbf{u}_1 + \gamma \left(\mathbf{w} - \dot{\mathbf{q}}\right) \tag{2.15}$$

For the underwater vehicle application, $\mathbf{w}$ represents the water velocity and can be interpreted as the source for a disturbance $\mathbf{d}_1 = \gamma \mathbf{w}$. The vector $\mathbf{d}_1$ represents unknown external forces on the observer as well as errors in the actuator model. It is modeled by a first-order Gauss-Markov process.

$$\frac{d}{dt}\mathbf{d}_1 \quad = \quad -\frac{1}{\tau_1}\mathbf{d}_1 + \mathbf{n}_{d1} \tag{2.16}$$

where $\mathbf{n}_{d1}$ is zero-mean white Gaussian noise.

A $1/s^2$ model relates the control input $\mathbf{u}_2$ to angular velocity. A simpler model has been assumed for the rotational motion of an underwater vehicle because angular velocity is easily measured with the rate gyros. Therefore, the accuracy of the model for angular velocity is not as important. In fact, Section 2.5 explains how to eliminate this model from the estimator design.

$$\frac{d}{dt}\boldsymbol{\omega} \quad = \quad \mathbf{u}_2 + \mathbf{d}_2 \tag{2.17}$$

The disturbances, again modeled by first-order dynamic equations, represent unknown external torques on the observer and errors in the actuator model.

$$\frac{d}{dt}\mathbf{d}_2 = -\frac{1}{\tau_2}\mathbf{d}_2 + \mathbf{n}_{d2} \tag{2.18}$$

The vector $\mathbf{n}_{d2}$ represents zero-mean white Gaussian noise.

2.4 Complete System Equations

The system equations, composed of a vector of states, the dynamics of those states, and the sensor models that relate the states to the measurements, describe the sensor fusion problem at the core of the sensing strategy. This section collects all of these parts and highlights the nonlinearities in the sensor and process model.

The state of the system includes the relative position ${}^{N}\mathbf{r}$, the velocity of the observer $\dot{\mathbf{q}}$, the observer orientation $\boldsymbol{\lambda}$ and angular velocity $\boldsymbol{\omega}$, the disturbance forces $\mathbf{d}_1$ and torques $\mathbf{d}_2$, the inertial rate sensor biases $\mathbf{b}_a$ and $\mathbf{b}_\omega$ and the scaling parameter α. The total state vector $\mathbf{X}$ can be written as a combination of the estimator states $\mathbf{x}$ and some additional states $\mathbf{x}_\omega$ which describe the rotational motion.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_\omega \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} {}^{N}\mathbf{r} \\ \dot{\mathbf{q}} \\ \mathbf{d}_1 \\ \mathbf{b}_a \\ \alpha \\ \boldsymbol{\lambda} \\ \mathbf{b}_\omega \end{bmatrix} \qquad \mathbf{x}_\omega = \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{d}_2 \end{bmatrix} \tag{2.19}$$

The dynamics of $\mathbf{X}$ have already been defined in Sections 2.2 and 2.3.

$$\frac{d}{dt}\mathbf{X} = \begin{bmatrix} \frac{d}{dt}\mathbf{x} \\ \frac{d}{dt}\mathbf{x}_\omega \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}, \mathbf{u}_1, \boldsymbol{\omega}, \mathbf{n}_{ba}, \mathbf{n}_{d1}, \mathbf{n}_{b\omega}, n_\alpha) \\ f_\omega(\mathbf{x}_\omega, \mathbf{u}_2, \mathbf{n}_{d2}) \end{bmatrix} \tag{2.20}$$

$$f = \begin{cases} \frac{d}{dt}{}^{N}\mathbf{r} &= -\dot{\mathbf{q}} \\ \frac{d}{dt}\dot{\mathbf{q}} &= R(\boldsymbol{\lambda})^T\,\mathbf{u}_1 + \mathbf{d}_1 - \gamma\dot{\mathbf{q}} \\ \frac{d}{dt}\mathbf{d}_1 &= -\frac{1}{\tau_1}\mathbf{d}_1 + \mathbf{n}_{d1} \\ \frac{d}{dt}\mathbf{b}_a &= \mathbf{n}_{ba} \\ \frac{d}{dt}\alpha &= n_\alpha \\ \frac{d}{dt}\boldsymbol{\lambda} &= E(\boldsymbol{\lambda})\,R(\boldsymbol{\lambda})^T\,\boldsymbol{\omega} \\ \frac{d}{dt}\mathbf{b}_\omega &= \mathbf{n}_{b\omega} \end{cases} \tag{2.21}$$

$$f_\omega = \begin{cases} \frac{d}{dt}\boldsymbol{\omega} &= \mathbf{u}_2 + \mathbf{d}_2 \\ \frac{d}{dt}\mathbf{d}_2 &= -\frac{1}{\tau_2}\mathbf{d}_2 + \mathbf{n}_{d2} \end{cases} \tag{2.22}$$

The vector $\mathbf{Z}$ contains the camera and inertial rate sensor measurements.

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}_s \\ \mathbf{z}_a \\ \mathbf{z}_\omega \end{bmatrix} = \begin{bmatrix} h_s(\mathbf{x}, \mathbf{n}_s) \\ h_a(\mathbf{x}, \mathbf{u}_1, \mathbf{n}_a) \\ h_\omega(\mathbf{x}, \mathbf{x}_\omega, \mathbf{n}_\omega) \end{bmatrix} \tag{2.23}$$

$$h_s(\mathbf{x}, \mathbf{n}_s) = \begin{cases} \mathbf{S} = \begin{bmatrix} S_x \\ S_y \\ S_z \end{bmatrix} = R(\boldsymbol{\lambda})\ {}^N\mathbf{r} - P_{cam} \\ \mathbf{z}_s = \begin{bmatrix} s_x \\ s_y \end{bmatrix} + \mathbf{n}_s = 1/S_z \begin{bmatrix} S_x \\ S_y \end{bmatrix} + \mathbf{n}_s \end{cases} \tag{2.24}$$

$$h_a(\mathbf{x}, \mathbf{u}_1, \mathbf{n}_a) = -\mathbf{u}_1 + R(\boldsymbol{\lambda})\left(-\mathbf{d}_1 + \gamma\dot{\mathbf{q}} + \alpha\mathbf{g}\right) + \mathbf{b}_a + \mathbf{n}_a \tag{2.25}$$

$$h_\omega(\mathbf{x}, \mathbf{x}_\omega, \mathbf{n}_\omega) = \boldsymbol{\omega} + \mathbf{b}_\omega + \mathbf{n}_\omega \tag{2.26}$$

2.5 Estimation Problem

Although the complete system equations presented in Section 2.4 already define an estimation problem, this problem can be partitioned to permit a simpler estimator design. It is possible to eliminate the $\mathbf{x}_\omega$ states without significant loss in estimator performance.

It is possible to eliminate the torque disturbance $\mathbf{d}_2$ from the estimation problem because it does not contribute significantly to the performance of the estimator. For operational underwater vehicles, the model parameters for the actuator and disturbance models defined in Section 2.3 are difficult to determine. Although the models can predict the motion of the observer, the accuracy of that prediction is at best moderate. The purpose of these models in the estimator is to improve the convergence of the estimate by mitigating the errors in the acceleration and angular velocity measurements. However, the presence of the gravity component in the accelerometer measurement has the effect of enhancing the accuracy of the angular velocity measurement and degrading the accuracy of the acceleration measurement. Consequently, while the models that predict linear velocity are important, those that predict angular velocity are not. Therefore, measuring the angular velocity is sufficient and the estimate of $\mathbf{d}_2$ can be eliminated.

Eliminating $\mathbf{d}_2$ has two advantages. First, the length of the state vector is reduced. Second, fewer model parameters have to be determined.

The estimation problem can be written more compactly by incorporating the inertial rate sensor measurements directly into the process model. This is often done for aided inertial navigation systems because it partitions the estimator design according to the rate of sensor measurements. The process model can then be implemented at the faster rate of the inertial rate sensor measurements and the sensor model can be implemented at the rate of the aiding sensor, which is typically slower. This can provide an advantage in the computational cost of the estimator.

The main advantage of incorporating the rate gyro measurement into the process model of this estimator is that the explicit representation of $\boldsymbol{\omega}$ can also be eliminated. As a result, $\mathbf{x}_\omega$ is not required in the state vector. However, incorporating the accelerometer measurement does not result in eliminating any states. Therefore, the accelerometer measurement remains in the sensor model.

Equation 2.19 defines the estimator state vector

$$\mathbf{x} = \begin{bmatrix} {}^{N}\mathbf{r} \\ \dot{\mathbf{q}} \\ \mathbf{d}_1 \\ \mathbf{b}_a \\ \alpha \\ \boldsymbol{\lambda} \\ \mathbf{b}_\omega \end{bmatrix}. \tag{2.27}$$

The process model can depend explicitly on the rate gyro measurements by rewriting the rate gyro sensor model in (2.11) as an expression for $\boldsymbol{\omega}$.

$$\boldsymbol{\omega} = \mathbf{z}_\omega - \mathbf{b}_\omega - \mathbf{n}_\omega \tag{2.28}$$

This generates the following estimation problem.

$$\frac{d}{dt}\mathbf{x} = f(\mathbf{x}, \mathbf{u}_1, \mathbf{z}_\omega, \mathbf{n}_p) = \begin{cases} \frac{d}{dt}{}^{N}\mathbf{r} &= -\dot{\mathbf{q}} \\ \frac{d}{dt}\dot{\mathbf{q}} &= R(\boldsymbol{\lambda})^T \mathbf{u}_1 + \mathbf{d}_1 - \gamma\dot{\mathbf{q}} \\ \frac{d}{dt}\mathbf{d}_1 &= -\frac{1}{\tau_1}\mathbf{d}_1 + \mathbf{n}_{d1} \\ \frac{d}{dt}\mathbf{b}_a &= \mathbf{n}_{ba} \\ \frac{d}{dt}\alpha &= n_\alpha \\ \frac{d}{dt}\boldsymbol{\lambda} &= E(\boldsymbol{\lambda})\, R(\boldsymbol{\lambda})^T (\mathbf{z}_\omega - \mathbf{b}_\omega - \mathbf{n}_\omega) \\ \frac{d}{dt}\mathbf{b}_\omega &= \mathbf{n}_{b\omega} \end{cases} \tag{2.29}$$

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_s \\ \mathbf{z}_a \end{bmatrix} = h(\mathbf{x}, \mathbf{u}_1, \mathbf{n}_z) \tag{2.30}$$

$$= \left\{ \begin{array}{lcl} \mathbf{S} & = & \begin{bmatrix} S_x \\ S_y \\ S_z \end{bmatrix} = R(\boldsymbol{\lambda})\ {}^{N}\mathbf{r} - P_{cam} \\ \mathbf{z}_s & = & \begin{bmatrix} s_x \\ s_y \end{bmatrix} + \mathbf{n}_s = 1/S_z \begin{bmatrix} S_x \\ S_y \end{bmatrix} + \mathbf{n}_s \\ \mathbf{z}_a & = & -\mathbf{u}_1 + R(\boldsymbol{\lambda})\left(-\mathbf{d}_1 + \gamma\dot{\mathbf{q}} + \alpha\mathbf{g}\right) + \mathbf{b}_a + \mathbf{n}_a \end{array} \right. \tag{2.31}$$

$$\mathbf{n}_p = \begin{bmatrix} \mathbf{n}_{d1} \\ \mathbf{n}_{ba} \\ \mathbf{n}_{\omega} \\ n_{\alpha} \\ \mathbf{n}_{b\omega} \end{bmatrix} \tag{2.32}$$

$$\mathbf{n}_z = \begin{bmatrix} \mathbf{n}_s \\ \mathbf{n}_a \end{bmatrix} \tag{2.33}$$

The vectors $\mathbf{n}_p$ and $\mathbf{n}_z$ collect all of the process and sensor noise terms.

2.6 Summary

This chapter provides an explicit definition of the sensing strategy by presenting the models that were used to represent the components of the sensing strategy. These generate the system equations from which the estimation problem can be derived. The next chapter evaluates various techniques that could be used to design an estimator for this sensor fusion problem and suggests an appropriate design strategy. Chapter 4 presents the design of a recursive, nonlinear estimator to solve this estimation problem.

Chapter 3

Recursive Nonlinear Estimation

Fusing measurements from a bearing sensor and inertial rate sensors to determine relative position involves a recursive nonlinear estimation problem that is difficult to solve with standard tools like the Extended Kalman Filter (EKF). Chapter 4 presents a new solution for this estimation problem. This chapter lays the groundwork for this solution by comparing several potential approaches for solving this nonlinear estimation problem. This comparison is based on a simple three-state example problem that captures the critical features of the full sensor fusion problem. Applying the EKF to the three-state example illustrates why this popular design method provides a poor solution to the sensor fusion problem. After examining the performance of several other approaches, the chapter concludes by identifying the design strategy used to create the full solution in Chapter 4.

Section 3.1 introduces the broad topic of optimal nonlinear estimation. Although extensive theory has been developed to handle these problems, practical solutions to all but a few specific cases have been elusive. Therefore, most solutions to nonlinear estimation problems involve approximations. Choosing appropriate approximations for the particular problem is a critical step in developing a successful solution for many of these problems.

The EKF has become a standard design tool for nonlinear estimators. It approximates the estimation problem by linearizing the nonlinear system equations to generate a similar, linear system to which the Kalman Filter equations can be applied. The Kalman Filter produces an optimal solution for linear Gaussian estimation problems. Section 3.2 describes the Kalman Filter and its extension to nonlinear problems.

Section 3.3 introduces the three-state example, a very simple estimation problem that captures those features of the full sensor fusion problem that cause problems for the EKF estimator design. This example is used to compare several different estimator designs using Monte Carlo simulations. The rest of the chapter introduces these estimator designs and discusses their performance on the three-state example.

Section 3.4 shows how the EKF is used to implement the estimator for the three-state example and discusses the EKF results. Dynamic observability is defined as a property of the system. This property is useful to explain how the EKF fails and to suggest an alternative solution approach.

Changing the representation of the nonlinear system is a powerful tool for improving the performance of the estimator. The particular representation of the state of the system determines the types of nonlinearity and in what part of the estimator the nonlinearities appear. Section 3.5 describes an estimator design based on the EKF, but with an alternative system representation. The key feature of this representation is a linear sensor model. This leads to a significant improvement in the estimator performance for the three-state example.

While approximations are necessary to implement estimators for most nonlinear problems, there are different approximations that can be applied. For example, the EKF uses linearization of the system equations. Two other techniques, Particle Filters and Sigma Point Methods, approximate the probability density function for the estimate uncertainty with a set of points sampled according to the density function. The advantage of these techniques is that their accounting of uncertainty is more complete so they produce more accurate solutions; that they are simple to derive; and that they can handle a broad range of nonlinearities. Particle Filters (Section 3.6), which use a large set of points sampled randomly from the density function, could be used to solve this problem, but are very computationally expensive, especially for large estimation problems. Sigma Point Methods, on the other hand, use a small, deterministic sampling of the density function. They generate good results at a computational cost that is similar to an equivalent EKF solution.

The Unscented Transform (UT) is a particular implementation of a Sigma Point method. Section 3.7 describes the UT, presents an estimator design based on the UT, and discusses its results for the three-state example.

The solution for the full sensor fusion problem, which is presented in Chapter 4, is based on the techniques described in this chapter. The estimator is a Kalman Filter using a representation of the system that results in a linear sensor model. The Kalman Filter framework is modified by the application of the Unscented Transform to handle the nonlinearities in the process model. This solution avoids all linearization of the nonlinear system equations. The Monte Carlo simulations in this chapter for the three-state example problem demonstrate the benefits of this combined approach.

3.1 Optimal Nonlinear Estimation

Although the optimal nonlinear estimation problem is well defined, its solution, which relies on infinite-dimensional descriptions of probability density functions, can generally not be solved exactly. To generate a solution, some part of the problem has to be approximated and several techniques for suboptimal solutions have been proposed. Linear Gaussian problems, which are finite dimensional, are an obvious exception. For these, the Kalman Filter provides an exact, optimal solution.

The problem of solving the optimal nonlinear estimation problem has been widely discussed in the literature. Stengel [53, Section 4.6] and Gelb [16, Ch. 6] provide an overview of the problem. Jazwinski [23, Ch. 9] and Maybeck [36, Ch. 12] analyze the problem in greater detail. All of these provide further references into the literature.

3.2 Kalman Filter Solutions

The Kalman Filter has become an ubiquitous method for designing optimal state estimators for dynamic systems. For linear problems with white Gaussian noise sources, the Kalman Filter generates an optimal solution. For nonlinear problems, many modifications to the Kalman Filter have been developed to generate good, sub-optimal solutions. In general, no guarantees for optimality or even convergence can be stated for the solutions to nonlinear problems. Nevertheless, countless nonlinear problems have been solved successfully with this approach.

The Kalman Filter is popular because it is a flexible approach that can generate very good solutions to complex problems. Once the system equations, the initial conditions, and the noise sources have been described, it provides a method to write the solution. Each design can then take advantage of a significant body of engineering solutions to tune the result.

This section describes the Kalman Filter for linear, Gaussian problems and then presents some of the extensions that have been developed to handle nonlinear problems. The most popular of these extensions is the EKF.

3.2.1 Kalman Filter Equations

The Kalman Filter and the Extended Kalman Filter have been described in countless textbooks, including [16, 29, 53]. This section introduces the particular interpretation used in this dissertation.

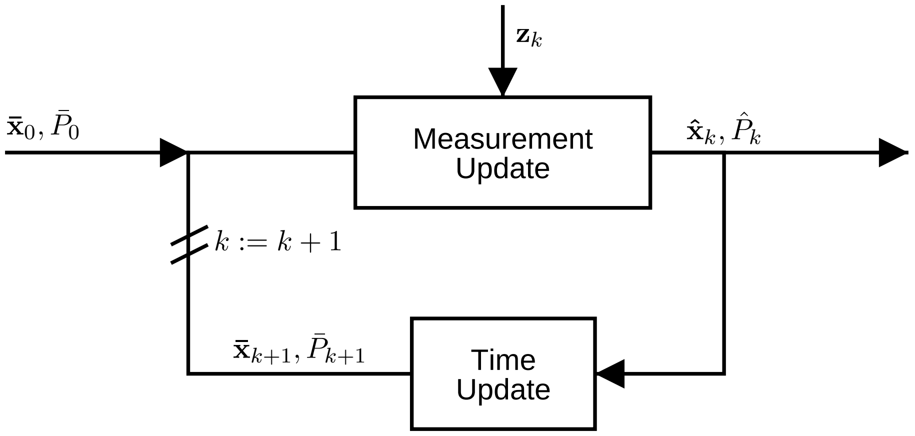
$\mathbf{z}_k$
$\bar{\mathbf{x}}_0, \bar{P}_0$
Measurement Update
$\hat{\mathbf{x}}_k, \hat{P}_k$
$k := k+1$
$\bar{\mathbf{x}}_{k+1}, \bar{P}_{k+1}$
Time Update

Figure 3.1: Prediction-Correction Framework of the Kalman Filter

Consider a linear, discrete-time system described by this state space representation:

$$\mathbf{x}_{k+1} = F_k \mathbf{x}_k + G_k \mathbf{u}_k + G_{p,k} \mathbf{n}_{p,k} \tag{3.1}$$

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{n}_{z,k}. \tag{3.2}$$

$\mathbf{x}_k$ is the system state of length n; $\mathbf{z}_k$ are the measurements; $\mathbf{n}_{p,k}$ is Gaussian white process noise; $\mathbf{n}_{z,k}$ is Gaussian white measurement noise; and F_k, G_k, $G_{p,k}$, and H_k are the system matrices at time $t = k\Delta T$. The deterministic driving terms have been omitted without loss of generality.

Assume that the initial estimate error $\tilde{\mathbf{x}}_0 = \mathbf{x}_0 - \hat{\mathbf{x}}_0$, the process noise $\mathbf{n}_{p,k}$ and the sensor noise $\mathbf{n}_{z,k}$ are all zero-mean Gaussian random variables that satisfy

$$E\left(\begin{bmatrix} \tilde{\mathbf{x}}_0 \\ \mathbf{n}_{p,i} \\ \mathbf{n}_{z,i} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_0^T & \mathbf{n}_{p,j}^T & \mathbf{n}_{z,j}^T \end{bmatrix} \right) = \begin{bmatrix} P_0 & 0 & 0 \\ 0 & Q_{i,j} & 0 \\ 0 & 0 & R_{i,j} \end{bmatrix} \tag{3.3}$$

$$Q_{i,j} = \begin{cases} Q & i = j \\ 0 & i \neq j \end{cases} \tag{3.4}$$

$$R_{i,j} = \begin{cases} R & i = j \\ 0 & i \neq j \end{cases}. \tag{3.5}$$

The Kalman Filter solution for $\hat{\mathbf{x}}_k$ and its covariance $\hat{P}_k$ can then be written in two steps:

time update:

$$\bar{\mathbf{x}}_{k+1} = F_k \hat{\mathbf{x}}_k + G_k \mathbf{u}_k \tag{3.6}$$

$$\bar{P}_{k+1} = F_k \hat{P}_k F_k^T + G_{p,k} Q G_{p,k}^T \tag{3.7}$$

measurement update:

$$\begin{aligned}\hat{\mathbf{x}}_k &= \bar{\mathbf{x}}_k + L_k\left(z - H_k\bar{\mathbf{x}}_k\right) &(3.8)\\ \hat{P}_k &= \left(I - L_kH_k\right)\bar{P}_k &(3.9)\\ L_k &= \bar{P}_kH_k^T\left(R + H_k\bar{P}_kH_k^T\right)^{-1} &(3.10)\end{aligned}$$

These are the two steps shown in Figure 3.1, which shows the prediction-correction framework of the Kalman Filter. The estimate is propagated forward in time with the prediction step, or time update, and then corrected with the current measurements in the measurement update. The time update is based on the process model of the system equations and the correction step is based on the sensor model.

Note that $\bar{\mathbf{x}}_k$ and $\bar{P}_k$ are the predicted estimates based on measurements up to time step $k-1$ and $\hat{\mathbf{x}}_k$ and $\hat{P}_k$ are the corrected estimates based on measurements up to time step k. That is

$$\begin{aligned}\bar{\mathbf{x}}_k &= \hat{\mathbf{x}}_{k|k-1} &(3.11)\\ \bar{P}_k &= \hat{P}_{k|k-1} &(3.12)\\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_{k|k} &(3.13)\\ \hat{P}_k &= \hat{P}_{k|k}. &(3.14)\end{aligned}$$

3.2.2 Extended Kalman Filter

Consider the following nonlinear, discrete-time system, with a nonlinear process model and a nonlinear sensor model. This is a generalization of the linear system in (3.1) and (3.2).

$$\begin{aligned}\mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{n}_{p,k}) &(3.15)\\ \mathbf{z}_k &= h(\mathbf{x}_k) + \mathbf{n}_{z,k} &(3.16)\end{aligned}$$

This problem can be solved using the Kalman Filter equations for linear systems by constructing a linear system that approximates the nonlinear system near the current best estimate. This is $\hat{\mathbf{x}}_k$ for the time update and $\bar{\mathbf{x}}_k$ for the measurement update. If $f(.)$ and $h(.)$ are differentiable and $\mathbf{n}_{p,k}$ is zero mean, then a linearization based on the first-order Taylor-series expansion can be computed:

$$\begin{aligned}f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{n}_{p,k}) &\cong f(\hat{\mathbf{x}}_k, \mathbf{u}_k, \mathbf{0}) + F_k(\mathbf{x}_k - \hat{\mathbf{x}}_k) + G_{p,k}\mathbf{n}_{p,k} &(3.17)\\ h(\mathbf{x}_k) &\cong h(\bar{\mathbf{x}}_k) + H_k(\mathbf{x}_k - \bar{\mathbf{x}}_k) &(3.18)\\ \text{where:}\quad F_k &= \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k, \mathbf{u}_k} &(3.19)\\ G_{p,k} &= \left.\frac{\partial f}{\partial \mathbf{n}_p}\right|_{\hat{\mathbf{x}}_k, \mathbf{u}_k} &(3.20)\end{aligned}$$

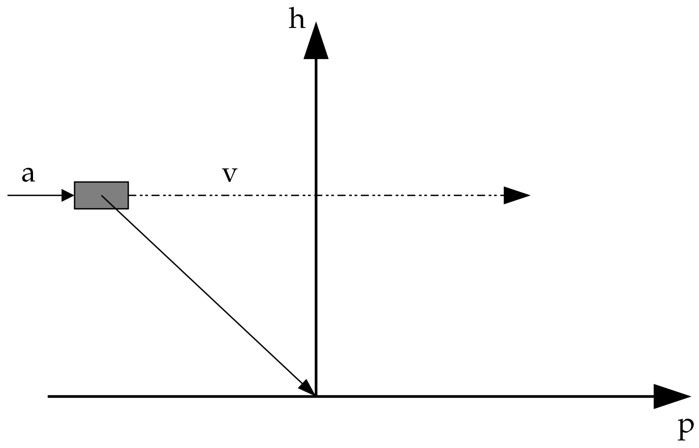

Figure 3.2: Simple Example of an Accelerating Observer with a Bearing Measurement of the Origin Traveling at Constant, Uncertain Altitude and Uncertain Velocity

$$H_k = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}}_k} \tag{3.21}$$

This linearization is used in (3.7), (3.9) and (3.10) together with the following equations to build the EKF:

$$\textbf{time update:} \qquad \bar{\mathbf{x}}_{k+1} = f(\hat{\mathbf{x}}_k, \mathbf{u}_k, \mathbf{0}) \tag{3.22}$$

$$\textbf{measurement update:} \qquad \hat{\mathbf{x}}_k = \bar{\mathbf{x}}_k + L_k\,(z - h(\bar{\mathbf{x}}_k)) \tag{3.23}$$

3.3 A Three-State Example

A simple, three-state example has been developed to compare the performance of various estimator designs and to illustrate the advantages of the new design approach. The example captures the generic features of the full sensor fusion problem that make it a challenging estimation problem. These include the projection nonlinearity in the vision measurement and the uncertain observer motion. This section describes the three-state example and presents simulation results for three different estimator designs.

A simple example is necessary for comparing the Extended Kalman Filter to other approaches. The EKF fails to solve the full sensor fusion problem described in Chapter 2. The algorithm frequently diverges and the estimates are strongly biased. This complicates the analysis of the EKF performance. Much more insight about the performance of the EKF can be gleamed from the example problem which has been tuned so that the problems of the EKF are apparent but do not cause the estimator to fail. This permits comparisons with alternative approaches based on the mean estimate error and standard deviation of the estimate error.

For this example, an accelerating observer is traveling at an uncertain velocity and uses a noisy bearing measurement to localize itself. This problem is presented in Figure 3.2. The observer is located at position p and altitude h, traveling with a velocity $v = \dot{p}$. It is subject to a known acceleration a. Its only measurement is $z = p/h$, a bearing measurement of the origin. Let $\mathbf{x}_k$ be the state of the system at time $t = kT$.

$$\begin{aligned} \mathbf{x}_k &= \begin{bmatrix} p_k \\ v_k \\ h_k \end{bmatrix} && (3.24) \\ \mathbf{x}_{k+1} &= \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} \frac{1}{2}T^2 \\ T \\ 0 \end{bmatrix} a && (3.25) \\ z_k &= p_k/h_k + n_{z,k} && (3.26) \end{aligned}$$

The system has no process noise. Process noise can mask the effects of errors and approximations in an estimator implementation. An example without process noise provides more useful comparisons between estimators.

This example was inspired by a similar problem—one with a range measurement instead of a bearing measurement—presented in [31] to illustrate the performance of the two-step estimator (see Section 3.5). Both variations generate a difficult estimation problem with dynamic observability. By incorporating a bearing measurement, the example presented here is more relevant to the full estimation problem discussed in this dissertation.

A Monte Carlo simulation compares the performance of three different estimators that were designed for the three-state example. These estimators, labeled X EKF, Y EKF and Y UT, are described in Sections 3.4, 3.5, and 3.7, respectively. This section provides all of the simulation results, which are then used in the subsequent sections to motivate the progression towards more sophisticated designs.

Each run of the simulation is based on different initial conditions and a different measurement noise vector. An estimate $\hat{\mathbf{x}}$ is computed for each of the three estimator designs. N runs are averaged to produce the Monte Carlo results. The mean estimate error $\mu(\tilde{\mathbf{x}})$, the actual standard deviation of the estimate error $\sigma(\tilde{\mathbf{x}})$, and the predicted standard deviation $\hat{\sigma}(\tilde{\mathbf{x}})$ are given by:

$$\begin{aligned} \mu(\tilde{\mathbf{x}}_k) &= \frac{1}{N}\sum_{i=1}^{N} \tilde{\mathbf{x}}_k^i && (3.27) \\ \sigma(\tilde{\mathbf{x}}_k) &= \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} \left(\tilde{\mathbf{x}}_k^i - \mu(\tilde{\mathbf{x}}_k)\right)^2} && (3.28) \end{aligned}$$

Table 3.1: Parameters for the Three-State Example

Parameter	Value
T	$0.1\,s$
a	$0.1\,m/s^2$
$\hat{\mathbf{x}}_0$	$\begin{bmatrix} -1.0 \\ 0.1 \\ 1.0 \end{bmatrix}$
$\bar{\mathbf{x}}_0$	$\begin{bmatrix} -1.0 \\ 0.08 \\ 0.8 \end{bmatrix}$
P_0	$\begin{bmatrix} 0.3^2 & 0 & 0 \\ 0 & 0.1^2 & 0 \\ 0 & 0 & 0.3^2 \end{bmatrix}$
R	0.001^2
N	500

$$\hat{\sigma}(\tilde{\mathbf{x}}_k) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} diag\left(\hat{P}_k^i\right)} \tag{3.29}$$

$$\tilde{\mathbf{x}}_k^i = \mathbf{x}_k^i - \hat{\mathbf{x}}_k^i \tag{3.30}$$

where the superscript i denotes different runs of the experiment. Note that the actual standard deviation is computed from the estimate error and the predicted standard deviation is computed from the estimator covariance.

The initial conditions $\mathbf{x}_0$ were sampled from a uniform distribution such that $\bar{\mathbf{x}}_0 = E(\mathbf{x}_0^i)$ and $P_0 = E\Big(\left(\mathbf{x}_0^i - \bar{\mathbf{x}}_0\right)\left(\mathbf{x}_0^i - \bar{\mathbf{x}}_0\right)^T\Big)$. A uniform distribution was chosen because sampling initial conditions from a Gaussian distribution, which extends to infinity, can make the problem non-sensical. For example, the actual altitude h could turn out to be negative. Sampling from a uniform distribution, which also satisfies P_0, avoids these problems. The measurement noise vector n_z was sampled from a zero-mean Gaussian distribution with covariance R. Table 3.1 shows the parameters for this simulation.

Figure 3.3 presents the results of the Monte Carlo simulations. The mean estimate error $\mu(\tilde{\mathbf{x}})$ is shown in the left column and the standard deviation $\sigma(\tilde{\mathbf{x}})$ is shown in the right column. Each row corresponds to one state of the system. The markers on the traces identify the particular estimation algorithm. For the plots of $\sigma(\tilde{\mathbf{x}})$, solid lines indicate the actual standard deviation and dashed lines indicate the predicted standard deviation.

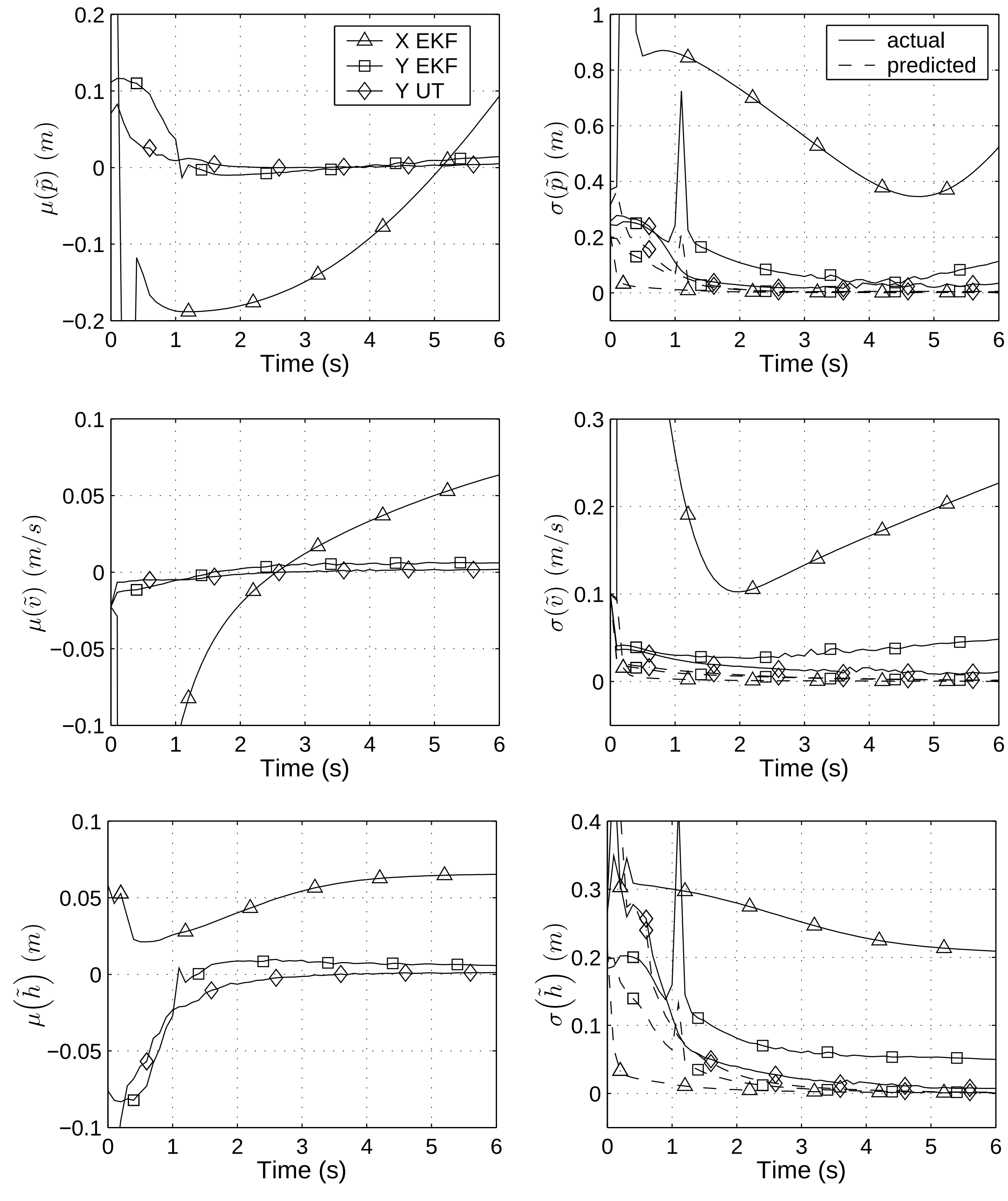

Figure 3.3: Mean (μ) and Standard Deviation (σ) of the Estimate Error for the Monte Carlo Simulations of the Three-State Example

3.4 X EKF: A Standard EKF Solution

The Extended Kalman Filter is the standard estimator design for nonlinear systems like the three-state example. This section derives the EKF equations for the three-state example and discusses the performance of the EKF in the Monte Carlo Simulations. These simulations show that the EKF is not always an appropriate estimator design. The solution can produce biased estimates, estimates with sub-optimal accuracy, or fail to converge at all. This example illustrates how the linearization of the EKF can introduce significant sources of error into the estimate and thereby motivates the search for a more suitable approach.

3.4.1 EKF Solution for the Three-State Example

The EKF applies the Kalman Filter equations to the linearized system equations of the nonlinear problem. The process model of the three-state example is linear and can be used directly. However, the sensor model is nonlinear and has to be linearized to implement the EKF. This results in

$$F_x = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.31}$$

$$G_x = \begin{bmatrix} \frac{1}{2}T^2 \\ T \\ 0 \end{bmatrix} \tag{3.32}$$

$$H_{x,k} = \begin{bmatrix} 1/\bar{h}_k & 0 & -\bar{p}_k/\bar{h}_k^2 \end{bmatrix} \tag{3.33}$$

$$\bar{\mathbf{x}}_{k+1} = F_x\hat{\mathbf{x}}_k + G_x a \tag{3.34}$$

$$\bar{P}_{x,k+1} = F_x\hat{P}_{x,k}F_x^T \tag{3.35}$$

$$\hat{\mathbf{x}}_k = \bar{\mathbf{x}}_k + L_k\left(z_k - \bar{p}_k/\bar{h}_k\right) \tag{3.36}$$

$$\hat{P}_{x,k} = \left(I - L_k H_{x,k}\right)\bar{P}_{x,k} \tag{3.37}$$

$$L_k = \bar{P}_{x,k}H_{x,k}^T\left(R + H_{x,k}\bar{P}_{x,k}H_{x,k}^T\right)^{-1}. \tag{3.38}$$

The traces marked with a triangle in Figure 3.3 correspond to the X EKF solution. It is clear from these traces that the straight-forward application of the EKF to this problem produces unsatisfactory results. Ideally, both the mean estimate error and the standard deviation will approach zero as time increases. The EKF fails to properly estimate the observer state as it travels past the origin (the region of greatest observability). As observability decreases with increasing time, the altitude estimate remains biased and as a result, the position and velocity bias continue to increase.

3.4.2 Problems with the EKF

A comparison of the actual and predicted standard deviation for the EKF can be used to explain the failure of this algorithm. The standard deviation computed from the estimator covariance significantly underpredicts the actual standard deviation and its evolution in time has a very different shape. The shape of the predicted standard deviation corresponds to an exponential decrease in uncertainty even though the actual uncertainty drops much more slowly and is almost constant in the initial part of the simulation.

The main problem of the EKF is that the covariance matrix fails to accurately describe the uncertainty in the estimate. This leads to the underprediction of the actual standard deviation of the estimate error seen in Figure 3.3. However, the main purpose for the covariance matrix is computing the Kalman gain L_k, which is used to correct the estimate with the latest sensor measurement. Errors in L_k lead to biases in the estimates.

The application of the Kalman Filter equations to the linearized system equations causes the inaccuracy in the covariance matrix. This process fails to account for all of the uncertainty in the estimation problem. Consider the linearization in (3.34) to compute H_k for the three-state example. Although it depends on $\bar{\mathbf{x}}_k$, which is uncertain, it is used in the Kalman Filter equations as if it were a known constant. A large uncertainty in $\bar{\mathbf{x}}_k$ (e.g., due to an unknown initial condition), can represent a large variation in H_k. The Kalman Filter does not provide an inherent mechanism to capture the uncertainty due to this variation, which is therefore ignored. This results in the observed underprediction of the covariance matrix.

The sensor fusion problem suffers from a confluence of two effects, which combine to generate poor EKF performance. First, H_k has a strong dependence on the uncertain state vector, which promotes the inaccuracy of the covariance matrix. Second, the accuracy of the covariance matrix is particularly important in this problem because it stores critical information about previous measurements. Triangulation, which is based on combining information from current and previous measurements, depends on the accuracy of the covariance matrix. However, these factors are not present for all nonlinear estimation problems. This explains why the EKF provides satisfactory solutions for many nonlinear problems, but fails spectacularly for some, including this sensor fusion problem.

3.4.3 Dynamic Observability

Several researchers have reported poor EKF performance for systems that exploit dynamic observability. This section describes dynamic observability and explains the failure of the EKF in terms of dynamic observability. The term *dynamic observability* was used by Carvalho, Del Moral, Monin and Salut [3] to describe a GPS positioning system that uses motion to satisfy observability requirements with fewer than three visible satellites. Kasdin [31] has discussed problems with "reliance on the dynamics to make the system observable".

A system exploits *dynamic observability* if it is observable, but does not satisfy *static observability*. In this sense, static observability describes a system whose state can be computed from sensor measurements at one time instance provided that all of the derivatives of these measurements are also available. Dynamic observability occurs when the system relies on variation in the system state or time-varying system equations to achieve observability.

The implicit triangulation that occurs in the three-state example illustrates the difference between static and dynamic observability. A single bearing measurement is never sufficient to determine the position and altitude of the observer, even if derivatives of the bearing were also available. To determine the observer state, two bearing measurements taken at different locations are necessary.

Dynamic observability is a significant factor in the estimator design when the necessary variation of the system state (e.g., the motion) is slow compared to the rate at which new measurements are acquired. This implies that long intervals of subsequent measurements do not generate observability of the complete state. The system provides measurements that make the complete state observable only after a significant period of time. During this time, some states are relatively uncertain, which can violate typical assumptions that the estimate uncertainty is small.

Dynamic observability exacerbates the deficiencies of the EKF in two ways. First, the uncertainty of the state estimates can be large for significant durations of the task. This increases the cost of ignoring the estimate uncertainty due to linearization and exacerbates the corruption of the covariance matrix. Second, the accuracy of the covariance matrix is more critical because it captures important information from past measurements which has to be combined with later measurements to generate observability (e.g., the implicit triangulation for the three-state example).

3.4.4 Other Kalman Filter Extensions for Nonlinear Problems

Many other extensions for the Kalman Filter have been proposed to addresses some of the problems with the EKF (see [16, 23, 36]). The iterated EKF is designed to improve the EKF by linearizing $h(.)$ about the corrected estimate $\hat{\mathbf{x}}$ instead of the predicted estimate $\bar{\mathbf{x}}$. Various second-order extensions have been developed (e.g., the Gaussian second-order filter and the truncated second-order filter). A fourth-order filter has also been presented. However, while these extensions tend to provide better results than the EKF, their implementations can add a lot of complexity to the algorithm. More importantly, these approaches do not directly address the inherent problem with the EKF, which is its failure to account for the uncertainty of the linearization step.

3.5 Y EKF: An EKF with an Alternate System Representation

The choice of representation of the system equations is a powerful tool to minimize the effect of errors from approximations in the optimal nonlinear estimation problem. For example, Aidala and Hammel [1] were able to improve the performance of a bearings-only tracking system by partitioning the representation into observable and unobservable states. Choosing a representation that yields a linear sensor model is the cornerstone of Kasdin's two-step estimator [31].

The choice of representation can be used to shift nonlinearities from one part of the problem to another and to affect the types of nonlinearities that are present. For example, the three-state example can be rewritten to shift the nonlinearity from the sensor model to the process model and to replace the ratio of states with a product of states. Even without changing estimation algorithm, both of these modifications have a positive effect on the quality of the solution.

Consider the vector $\mathbf{y}$ to represent the state of the three-state example:

$$\mathbf{y}_k = \begin{bmatrix} \alpha_{,k} \\ \beta_k \\ \gamma_k \end{bmatrix} = \begin{bmatrix} p_k/h_k \\ v_k \\ 1/h_k \end{bmatrix} \tag{3.39}$$

This leads to a linear sensor model and a nonlinear process model

$$\mathbf{y}_{k+1} = f_y(\mathbf{y}_k) = \begin{bmatrix} \alpha_k + \left(\beta_k + \frac{1}{2}aT\right)\gamma_k T \\ \beta_k + aT \\ \gamma_k \end{bmatrix} \tag{3.40}$$

$$z_k = \alpha_k + n_{z,k} \tag{3.41}$$

which can be linearized to construct a different EKF for the same problem:

$$F_{y,k} = \begin{bmatrix} 1 & \hat{\gamma}_k T & \left(\hat{\beta}_k + \frac{1}{2}aT\right)T \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.42}$$

$$H_y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{3.43}$$

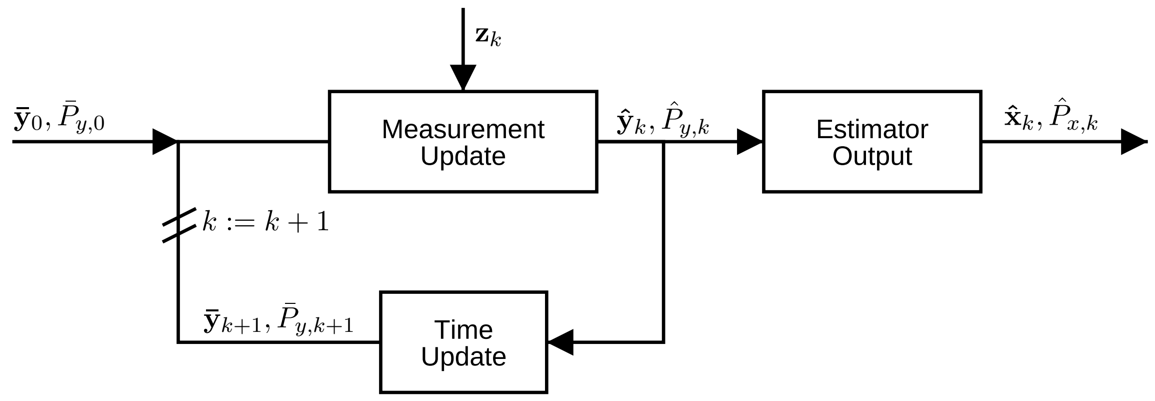
$\mathbf{z}_k$
$\bar{\mathbf{y}}_0, \bar{P}_{y,0}$
Measurement Update
$\hat{\mathbf{y}}_k, \hat{P}_{y,k}$
Estimator Output
$\hat{\mathbf{x}}_k, \hat{P}_{x,k}$
$k := k+1$
$\bar{\mathbf{y}}_{k+1}, \bar{P}_{y,k+1}$
Time Update

Figure 3.4: Estimator Implementation for Alternative System Representation $\mathbf{y}$

$$\bar{\mathbf{y}}_{k+1} = \begin{bmatrix} \hat{\alpha}_k + \left(\hat{\beta}_k + \frac{1}{2}aT\right)\hat{\gamma}_k T \\ \hat{\beta}_k + aT \\ \hat{\gamma}_k \end{bmatrix} \tag{3.44}$$

$$\bar{P}_{y,k+1} = F_{y,k}\hat{P}_{y,k}F_{y,k}^T \tag{3.45}$$

$$\hat{\mathbf{y}}_k = \bar{\mathbf{y}}_k + L_k\left(z_k - \bar{\alpha}_k\right) \tag{3.46}$$

$$\hat{P}_{y,k} = \left(I - L_k H_y\right)\bar{P}_{y,k} \tag{3.47}$$

$$L_k = \bar{P}_{y,k}H_y^T\left(R + H_y\bar{P}_{y,k}H_y^T\right)^{-1}. \tag{3.48}$$

Although the estimator operates on the $\mathbf{y}$ state vector, the output should, in general, be in terms of the $\mathbf{x}$ vector. Therefore, an additional step is required to transform $\hat{\mathbf{y}}_k$ and $\hat{P}_{y,k}$ to $\hat{\mathbf{x}}_k$ and $\hat{P}_{x,k}$. This corresponds to the estimator output step in Figure 3.4. Let

$$\mathbf{x} = g(\mathbf{y}) = \begin{bmatrix} \alpha/\gamma \\ \beta \\ 1/\gamma \end{bmatrix}. \tag{3.49}$$

Using a linearization approach to transfer one random variable to the other results in

$$\hat{\mathbf{x}}_k = g(\hat{\mathbf{y}}_k) \tag{3.50}$$

$$\hat{P}_{x,k} = \frac{\partial g}{\partial y}\hat{P}_{y,k}\frac{\partial g}{\partial y}^T. \tag{3.51}$$

The results of applying this estimator to the three-state example are labeled Y EKF in Figure 3.3. These plots show that the performance of this estimator is significantly better than that of the EKF based on $\mathbf{x}$. Both the mean and standard deviation of the estimate error are significantly smaller and decay much more quickly to values near zero. Furthermore, the predicted standard deviations are closer to the actual values, which explains the improved performance. Similar results were presented in [21].

The improvement in performance that can be achieved simply by choosing a different representation of the system equations is surprising. Shifting all of the nonlinearities to the process model and eliminating the need for linearization in the measurement update tends to reduce the adverse effects of the approximations used to construct the EKF. While it is not clear why a linear sensor model improves performance, this has been observed for a variety of problems and forms the core idea of the two-step estimator developed by Kasdin [31].

However, the difference between the predicted and actual standard deviations is still significant and can be attributed to the linearization that produces $F_{y,k}$. Again, Equation 3.45 assumes that $F_{y,k}$ is constant even though it depends on the uncertain $\hat{\mathbf{y}}_k$.

Kasdin and Weaver [32] have shown that the two-step estimator can overcome this problem in special cases if it is possible to define a state vector that also yields linear dynamics. Achieving a linear sensor and process model for a nonlinear system was possible by defining a much larger state vector with nonlinear state constraints.

The standard deviations for the Y EKF algorithm in Figure 3.3 show a spike just after $t = 1\,s$. This was generated because of the singularities in the function $g(.)$ used in the estimator output. For some of the runs, the estimate of γ_k was close to zero, which is a possible estimator state even if it is a non-sensical condition for the problem. It was therefore ignored.

3.6 Particle Filters

Particle Filters, or Sequential Monte Carlo Methods, are emerging nonlinear estimation tools that utilize the rapidly increasing power of the computer to improve the performance of nonlinear estimators [3, 12]. They turn optimal nonlinear estimation into a suboptimal, finite-dimensional problem by approximating the representation of uncertainty instead of approximating the nonlinearity of the system representation (as is done for the EKF). The advantage of this approach is that the uncertainty, or the probability density function associated with the estimate, can be approximated more effectively without developing ever more complex solutions.

Particle Filters represent the stochastic nature of the system with a sampling of the probability distribution, which generates many particles, each representing one point in the probability space. The weight of each particle, which is updated and corrected whenever new sensor measurements become available, represents the likelihood that this particle corresponds to the true system state. The deterministic dynamics are applied to each particle to predict the distribution at a future time.

Particle Filters can deal effectively with probability distributions that are not Gaussian nor unimodal. Furthermore, they can handle any nonlinear function and do not require differentiability or continuity.

The advantage of Particle Filters is that in the limit of increasing numbers of particles, they can solve arbitrary nonlinear estimation problems with arbitrary accuracy (at least in theory—some practical problems are still being addressed in current research). Their key disadvantage is that they still require significant computational power, especially for large systems, which tends to preclude real-time implementations. However, they are a useful benchmark against which other estimators can be evaluated (see Section 7.1.3).

3.7 Y UT: Incorporating the Unscented Transform

The Unscented Transform (UT) is a relatively new tool for managing nonlinear stochastic problems [25, 26]. Like Particle Filters, the UT represents random variables with a set of points, called sigma points, and propagates these through the deterministic nonlinear transformations. The UT differs from Particle Filters in two key points. First, the UT uses a much smaller number of sigma points (or particles) than Particle Filters, which makes the UT a viable tool even for large real-time problems. These sigma points are chosen deterministically using the prior mean and covariance. Second, after the UT creates a set of sigma points and applies the nonlinear transformation to these points, it immediately reconstructs a new mean and covariance and discards the points. Thus, the representation of the random variable iteratively changes from mean and covariance to sigma points and back. Although the UT requires several approximations, the method tends to be more accurate than many of the other techniques that have been used to propagate random variables through nonlinear transformations.

3.7.1 Illustrative Example

The difference between linearization and the Unscented Transform is illustrated by a simple example of a stochastic nonlinear transformation[1]. Both methods can be compared to

[1]This example is motivated by a similar example in [56].

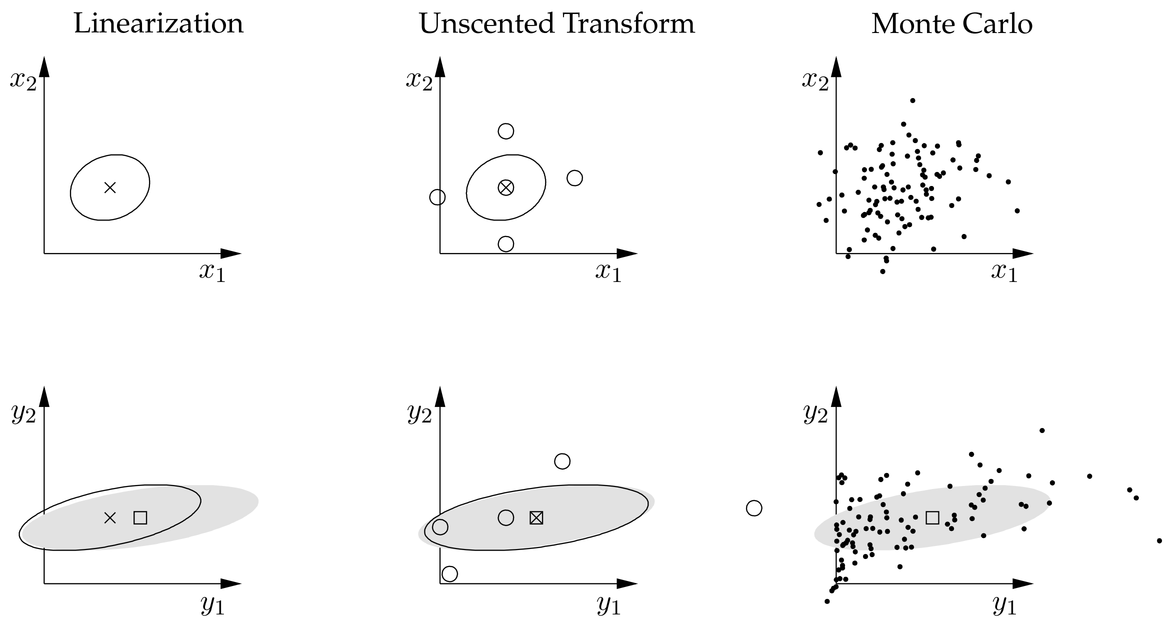

Figure 3.5: Example of a Stochastic Nonlinear Transformation

a Monte Carlo approach to determine their accuracy. Consider the nonlinear function

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = f(\mathbf{x}) = \begin{bmatrix} x_1^2 x_2 \\ x_2 \end{bmatrix} \tag{3.52}$$

where $\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$. Suppose that $\mathbf{x}$ is a Gaussian random variable with a known distribution.

$$\bar{\mathbf{x}} = E(\mathbf{x}) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{3.53}$$

$$P_x = E\left(\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T\right) = \begin{bmatrix} 0.36 & 0.05 \\ 0.05 & 0.25 \end{bmatrix} \tag{3.54}$$

$$\tilde{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}} \tag{3.55}$$

Each method can be used to compute $\bar{\mathbf{y}}$ and P_y, the first two moments of the distribution for $\mathbf{y}$. Figure 3.5 shows graphically how the different methods work. The top row of plots shows the domain (input) for each method and the bottom row shows the range (output). Each of the three columns of plots represents one of the methods.

The mean and covariance of $\mathbf{x}$ and $\mathbf{y}$ are depicted with an X and an ellipse on the plots in the first two columns. For linearization, the statistics of $\mathbf{y}$ are computed as follows:

$$\bar{\mathbf{y}} = f(\bar{\mathbf{x}}) \tag{3.56}$$

$$P_y = \frac{\partial f}{\partial x} P_x \frac{\partial f}{\partial x}^T \tag{3.57}$$

The linearization is encoded in $\frac{\partial f}{\partial x}$.

The Unscented Transform first computes five sigma points for this two-dimensional example. These are shown as circles on the plots. One sigma point is at the mean and the other four capture the size of the covariance ellipse. Therefore, this is a deterministic sampling of the distribution of $\mathbf{x}$. These sigma points are then propagated to the output, where they are used to compute the mean and covariance of the distribution of $\mathbf{y}$. The actual equations are shown in the next section.

The Monte Carlo approach involves a random sampling of the distribution of $\mathbf{x}$ to create a cloud of points, or particles. Again, these points are propagated through the nonlinear function to create a new cloud from which the statistics of $\mathbf{y}$ can be computed. This is the core principle of Particle Filters. The right column in Figure 3.5 shows the Monte Carlo approach. For this example, $100,000$ points were actually sampled, although only 100 points are plotted. Because the accuracy of the Monte Carlo approach improves with the number of points, a very large cloud has been used in order to compute the *true* distribution of $\mathbf{y}$. The mean and covariance of this distribution are depicted with a square and the gray region in the bottom row of the plots.

This example shows the advantage of the Unscented Transform over the linearization approach. The linearization develops a significant bias in the mean and underpredicts the covariance relative to the Monte Carlo approach. For the Unscented Transform, any bias in the mean is too small to see and although the covariance does not match the Monte Carlo covariance exactly, it is very close. While the Monte Carlo approach is computationally very expensive, the Unscented Transform has a similar computational cost as the linearization approach.

3.7.2 Applying the UT to the Three-State Example

The UT leads to further performance improvements for the three-state example. The new representation for the three-state example in Section 3.5 yields a linear sensor model and the nonlinear process model shown in (3.40). The UT provides an alternative method to compute the time update in (3.44-3.45). The specific implementation was suggested in [56] and is explained fully in Section 4.3.1.

$$\hat{\mathcal{Y}}_k = \hat{\mathbf{y}}_k + \eta \left[\begin{array}{ccc} \mathbf{0}_{l\times 1} & \sqrt{\hat{P}_{y,k}} & -\sqrt{\hat{P}_{y,k}} \end{array} \right] \tag{3.58}$$

$$\bar{\mathcal{Y}}_{k+1} = f_y\left(\hat{\mathcal{Y}}_k\right) \tag{3.59}$$

$$\bar{\mathbf{y}}_{k+1} = \sum_{i=0}^{2l} W_i \bar{\mathcal{Y}}_{k+1,i} \tag{3.60}$$

$$\begin{aligned}\bar{P}_{y,k+1} &= \sum_{i=0}^{2l} W_i \left(\bar{\mathcal{Y}}_{k+1,i} - \bar{\mathbf{y}}_{k+1}\right)\left(\bar{\mathcal{Y}}_{k+1,i} - \bar{\mathbf{y}}_{k+1}\right)^T + \\ &\quad \left(1 - \eta^2/l + \beta\right)\left(\bar{\mathcal{Y}}_{k+1,0} - \bar{\mathbf{y}}_{k+1}\right)\left(\bar{\mathcal{Y}}_{k+1,0} - \bar{\mathbf{y}}_{k+1}\right)^T\end{aligned} \tag{3.61}$$

$$l = 3 \tag{3.62}$$

$$N = 2l + 1 = 7 \tag{3.63}$$

$$\eta = \sqrt{3} \tag{3.64}$$

$$\beta = 2 \tag{3.65}$$

$$W_i = \begin{cases} \frac{\eta^2 - l}{\eta^2}, & i = 0 \\ \frac{1}{2\eta^2}, & i > 0 \end{cases} \tag{3.66}$$

The number of states is l and the number of sigma points is N. The sigma points are captured as columns in $\hat{\mathcal{Y}}_k, \bar{\mathcal{Y}}_{k+1} \in \mathbb{R}^{l\times N}$. The function f_y operates once on each column $\hat{\mathcal{Y}}_{k,i}$ of $\hat{\mathcal{Y}}_k$ to create the columns $\bar{\mathcal{Y}}_{k+1,i}$ of $\bar{\mathcal{Y}}_{k+1}$. η and β are design parameters. While the UT seems like a complicated alternative to (3.44-3.45), it is easy to implement and similar to the EKF in computational cost.

The results labeled Y UT in Figure 3.3 are created with an estimator that combines the UT time update with the standard Kalman Filter measurement update for linear sensor models in (3.46-3.48). The estimator output step, which computes the function $g(.)$ as shown in (3.49), can also be implemented with a UT to improve the estimator accuracy.

These results indicate a clear improvement of the Y UT over both of the EKF cases. The mean and standard deviation of the estimate error are smaller. Furthermore, the disparity between the predicted and actual standard deviation is reduced. Although the remaining disparity is not zero—which is not surprising given that there are approximations in the algorithm—the disparity is very small and the shape of the predicted and actual standard deviation traces is very similar.

The disparity between the actual and predicted standard deviation can be used to measure the severity of the approximations of each of the methods. This implies that the change to a representation with a linear sensor model is already very advantageous. The UT further improves the accuracy of the implementation of the resulting nonlinearities in the time update.

3.7.3 Unscented Kalman Filter

The UT can be used to build a complete optimal recursive estimator, called the Unscented Kalman Filter (UKF) [56]. The UKF uses the UT to implement the transformations for both nonlinear process and sensor models. Like the Kalman Filter, it propagates the mean and covariance of the random variables using a recursive prediction/correction scheme. The UKF is an interesting alternative when generating a representation with a linear sensor model is not possible or not convenient.

3.8 Other Nonlinear Estimation Techniques

Many strategies exist for developing useful estimators for nonlinear systems. The nonlinear estimation techniques described so far in this chapter have aimed at optimal estimators for nonlinear systems. However, all of these techniques are suboptimal, require approximations in the implementation, and therefore do not provide reliable guarantees on performance, accuracy, or even convergence.

Estimators for nonlinear systems can also be designed in an *ad hoc* fashion, in which arbitrary filter structures, with parameters, are proposed and then tuned according to various criteria. The most important test is convergence, which can sometimes be proven using a Lyapunov function. The motivation for choosing a particular filter structure is often the ability to find a Lyapunov function that proves convergence.

A key advantage of these *ad hoc* methods is that they are potentially very simple and can handle arbitrary nonlinearities in a robust manner—if a stability proof can be established. The main drawbacks is that their solutions are often hard to generalize to other problems, even closely related ones, that performance (optimality) is usually not considered, and that any unaccounted modeling errors can still cause the filter to fail to converge.

3.9 Summary

This chapter provides the necessary background in nonlinear estimation to support the design of the full estimator in the next chapter. The field of nonlinear estimation has generated a diverse set of solutions. The EKF is the most popular approach because of its flexibility. However, it performs poorly for some nonlinear problems, including this one, because it fails to account for a significant source of uncertainty in the linearizations. In that case, an approach that manages these approximations more carefully is required.

This chapter presents several alternatives to the EKF. It focuses on reformulating the problem to obtain a representation with a linear sensor model and on using the Unscented Transform to implement the nonlinear time update. Combining these two techniques with the Kalman Filter framework results in an estimator that inherits many of the benefits of the Kalman Filter while avoiding the problems of the EKF. This is the basis for the estimator design in the next chapter.

The conclusions in this chapter are supported by a simple three-state example that captures the key features of the full estimation problem. This example demonstrates the performance improvements over the EKF due to reformulating the problem and implementing the time update with the UT.

Chapter 4

Estimator Design

This chapter provides a solution to the estimation problem that was described in Chapter 2. The design is motivated by a combination of the techniques already discussed in Chapter 3. The solution is based on the Kalman Filter framework applied to a new representation of the system equations. This representation leads to a linear sensor model and forces all of the nonlinearities into the process model. The Unscented Transform (UT) is used to propagate the statistics of the estimate through these nonlinearities.

This design eliminates errors due to the linearization of system equations, which is a key problem for the application of the Extended Kalman Filter (EKF) to this problem. The linear sensor model obviates the need for linearization to implement the measurement update of the Kalman Filter. The UT handles nonlinear problems without linearization.

Although linearization has been removed as a source of error, the design cannot eliminate all error sources. The nature of optimal nonlinear estimation implies that any solution is based on some approximations and will be subject to errors from those approximations (see Section 3.1). Although the solution assumes that all uncertain quantities can be modeled as zero-mean Gaussian random variables, this assumption is violated by the nonlinear process model. The UT is itself based on an approximation, even if it is not as significant as linearization.

The goal in the design is therefore to avoid the most serious sources of error while producing a flexible, intuitive, real-time estimator. This is the motivation for the design choices in this chapter.

Section 4.1 defines a new state vector, $\mathbf{y}$, and the associated sensor and process models that define the new system representation. Section 4.2 introduces the square-root implementation of the estimator. Section 4.3 shows how the UT is applied to the process model to implement the time update (prediction step) of the Kalman Filter. Section 4.4 describes the implementation of the measurement update (correction step).

A further transformation is required to generate the output of the estimator. The relative position is not explicitly part of the state vector ($\mathbf{y}$) of the new system representation. The nonlinear transformation between the state vector and the relative position, also implemented using the UT, is presented in Section 4.5.

4.1 System Representation

The first step of the solution is to rewrite the system equations in terms of a new set of states. This new formulation has several objectives. It results in a linear sensor model, which removes the need to linearize the system to implement the measurement update. Furthermore, it transforms the nonlinearities in the system in an effort to reduce the errors due to approximations.

The choice of representation for the system can have a significant impact on the estimator performance. This is true for linear systems, where the choice of representation can affect the numerical properties of the solution. For nonlinear problems, this choice can have an even greater effect, because it determines the types of nonlinearities in the system equations and where they appear. Therefore, it controls the errors from approximations of the nonlinear equations and their effect on the overall solution.

Recall the nonlinear sensor model defined in Section 2.4.

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_s \\ \mathbf{z}_a \end{bmatrix} = h(\mathbf{x}, \mathbf{u}_1, \mathbf{n}_z) \tag{4.1}$$

$$= \begin{cases} \mathbf{S} = \begin{bmatrix} S_x \\ S_y \\ S_z \end{bmatrix} = R(\boldsymbol{\lambda})\ {}^{N}\mathbf{r} - P_{cam} \\ \mathbf{z}_s = \begin{bmatrix} s_x \\ s_y \end{bmatrix} + \mathbf{n}_s = 1/S_z \begin{bmatrix} S_x \\ S_y \end{bmatrix} + \mathbf{n}_s \\ \mathbf{z}_a = -\mathbf{u}_1 + R(\boldsymbol{\lambda})\left(-\mathbf{d}_1 + \gamma\dot{\mathbf{q}} + \alpha\mathbf{g}\right) + \mathbf{b}_a + \mathbf{n}_a \end{cases} \tag{4.2}$$

The design requires that the sensor model be expressed as a linear relationship by defining a new state vector $\mathbf{y}$:

$$\mathbf{z} = H\mathbf{y} + \mathbf{n}_z \tag{4.3}$$

$$\mathbf{x} = g(\mathbf{y}) \tag{4.4}$$

such that H is a constant matrix and $g(.)$ is a nonlinear mapping.

This requirement does not fully constrain the new state vector $\mathbf{y}$, so a number of additional design choices have to be made. These will be explained after the new state vector is introduced.

$$\mathbf{y} = \begin{bmatrix} s_x \\ s_y \\ \zeta \\ \mathbf{v} \\ \mathbf{a} \\ \mathbf{b}_a \\ \mathbf{Z} \\ \psi \\ \mathbf{b}_\omega \end{bmatrix} \tag{4.5}$$

The first three coordinates of $\mathbf{y}$ represent the relative position between the observer and the object. s_x and s_y are the position of the feature in the image plane. These two, together with the inverse feature range ζ, describe the relative position in the camera frame ($\mathbf{S}$):

$$\mathbf{S} = \frac{1}{\zeta} \begin{bmatrix} s_x \\ s_y \\ 1 \end{bmatrix} \qquad \begin{bmatrix} s_x \\ s_y \\ \zeta \end{bmatrix} = \frac{1}{S_z} \begin{bmatrix} S_x \\ S_y \\ 1 \end{bmatrix} \tag{4.6}$$

The representation of feature range S_z presents a design choice. Because feature range does not appear in the sensor model, it is not constrained by the linearity requirement. A useful representation for feature range is $\zeta = 1/S_z$. This choice leads to low-order polynomials as the dominant nonlinearity in the process model. Polynomials tend to result in more accurate estimator time-updates than, for example, ratios, which are induced by representing range with S_z.

The new state vector contains four vectors used to generate the linear accelerometer sensor model. $\mathbf{v}$ and $\mathbf{a}$ are the velocities and disturbance forces expressed in observer body coordinates, $\mathbf{b}_a$ is the accelerometer bias term, and $\mathbf{Z}$ is a new vector parallel to the direction of gravity. The sensor model is linear in those components:

$$\mathbf{z}_a = -\mathbf{u}_1 + \gamma\mathbf{v} - \mathbf{a} + g\mathbf{Z} + \mathbf{b}_a + \mathbf{n}_a \tag{4.7}$$

where

$$\begin{aligned}
\mathbf{v} &= \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = R\dot{\mathbf{q}} && (4.8)\\
\mathbf{a} &= R\mathbf{d}_1 && (4.9)\\
\mathbf{Z} &= \begin{bmatrix} Z_x \\ Z_y \\ Z_z \end{bmatrix} = R\begin{bmatrix} 0 \\ 0 \\ \alpha \end{bmatrix} && (4.10)\\
\alpha &= \|\mathbf{Z}\| && (4.11)
\end{aligned}$$

The apparent acceleration due to gravity is captured by $g\mathbf{Z}$. $\mathbf{Z}$ is the unit vector describing the inertial z-direction in the body frame modified by the accelerometer scale factor α. $\mathbf{Z}$ captures the observer attitude, but not its heading. Therefore, $\mathbf{y}$ also contains ψ to represent heading, or rotation about $\mathbf{Z}$. Together, $\mathbf{Z}$ and ψ define the observer orientation $(\boldsymbol{\lambda} = f(\mathbf{Z}, \psi))$. A similar representation for orientation is described in [42]. Finally, the rate gyro bias $\mathbf{b}_\omega$ is the last component of the new state vector $\mathbf{y}$.

The new state vector induces new sensor and process models. Because of the way the state vector was designed, the sensor model is now linear, with a constant H:

$$\begin{aligned}
\mathbf{z} &= H\mathbf{y} + \begin{bmatrix} 0 \\ 0 \\ -\mathbf{u}_1 \end{bmatrix} + \mathbf{n}_z && (4.12)\\
H &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \gamma I_{3\times3} & -I_{3\times3} & I_{3\times3} & gI_{3\times3} & 0 & 0 \end{bmatrix} && (4.13)
\end{aligned}$$

All of the nonlinearities now appear in the process model:

$$\frac{d}{dt}\mathbf{y} = f(\mathbf{y}, \mathbf{u}_1, \mathbf{z}_\omega, \mathbf{n}_p) \qquad (4.14)$$

$$f = \begin{cases} \frac{d}{dt}s_x &= -v_x\zeta + s_x v_z \zeta + s_x \rho_y \omega_x - (\rho_z + s_x \rho_x)\omega_y + \rho_y \omega_z \\ \frac{d}{dt}s_y &= -v_y\zeta + s_y v_z \zeta + (\rho_z + s_y \rho_y)\omega_x - \rho_x s_y \omega_y - \rho_x \omega_z \\ \frac{d}{dt}\zeta &= v_z \zeta^2 + \zeta \rho_y \omega_x - \zeta \rho_x \omega_y \\ \frac{d}{dt}\mathbf{v} &= \mathbf{u}_1 + \mathbf{a} - \gamma \mathbf{v} - \boldsymbol{\omega} \times \mathbf{v} \\ \frac{d}{dt}\mathbf{a} &= -\frac{1}{\tau}\mathbf{a} - \boldsymbol{\omega} \times \mathbf{a} + \mathbf{n}_{d1} \\ \frac{d}{dt}\mathbf{b}_a &= \mathbf{n}_{ba} \\ \frac{d}{dt}\mathbf{Z} &= -\boldsymbol{\omega} \times \mathbf{Z} + \frac{1}{\alpha}\mathbf{Z} n_\alpha \\ \frac{d}{dt}\psi &= \frac{1}{\alpha\left(Z_y^2 + Z_z^2\right)} \begin{bmatrix} 0 & Z_y & Z_z \end{bmatrix} \boldsymbol{\omega} \\ \frac{d}{dt}\mathbf{b}_\omega &= \mathbf{n}_{b\omega} \end{cases} \tag{4.15}$$

where

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \mathbf{z}_\omega - \mathbf{b}_\omega - \mathbf{n}_\omega \tag{4.16}$$

$$\begin{bmatrix} \rho_x \\ \rho_y \\ \rho_z \end{bmatrix} = P_{cam}\zeta + \begin{bmatrix} s_x \\ s_y \\ 1 \end{bmatrix} \tag{4.17}$$

4.2 Square-Root Implementation

The Kalman Filter framework is implemented with a square-root algorithm in which a square-root factor of the covariance is represented instead of the actual covariance. This approach has two distinct advantages. First, the UT already operates on a square-root factor of the covariance, so the square-root factor has to be computed anyways. However, the full covariance is not explicitly required by the algorithm. Therefore, storing and operating on only the square-root factor is more efficient. Second, for the covariance to be meaningful, it has to be nonnegative-definite. This is easy to satisfy for linear problems that are implemented with sufficient numerical precision. But modifications to the Kalman Filter which accommodate nonlinear system equations do not generally guarantee nonnegative-definiteness. However, squaring a square-root of the covariance always generates a nonnegative-definite covariance. Therefore, the square-root approach generates more reliable results more efficiently.

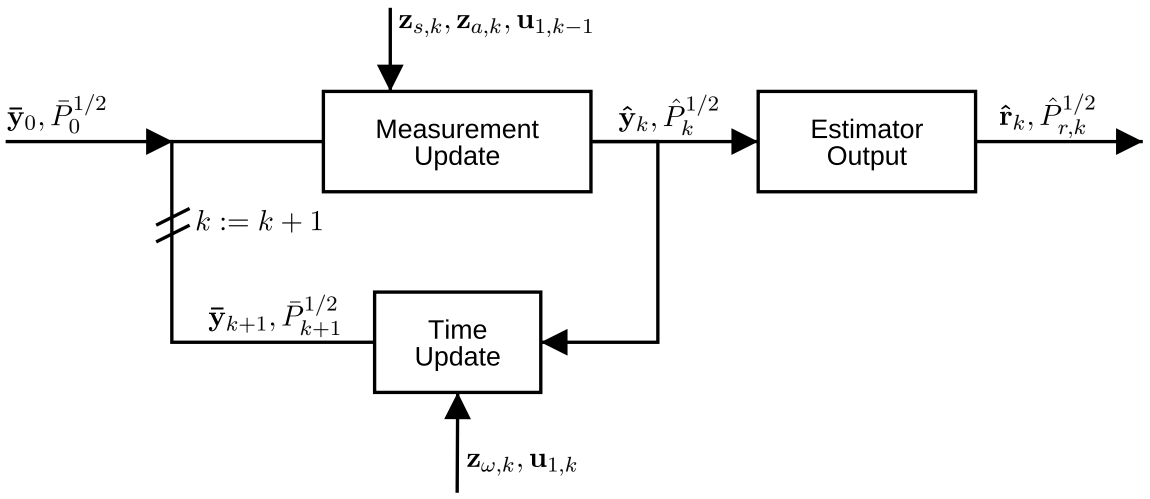
$\mathbf{z}_{s,k}, \mathbf{z}_{a,k}, \mathbf{u}_{1,k-1}$
$\bar{\mathbf{y}}_0, \bar{P}_0^{1/2}$
Measurement Update
$\hat{\mathbf{y}}_k, \hat{P}_k^{1/2}$
Estimator Output
$\hat{\mathbf{r}}_k, \hat{P}_{r,k}^{1/2}$
$k := k+1$
$\bar{\mathbf{y}}_{k+1}, \bar{P}_{k+1}^{1/2}$
Time Update
$\mathbf{z}_{\omega,k}, \mathbf{u}_{1,k}$

Figure 4.1: Square-Root Implementation of the Estimator

Figure 4.1 illustrates the three components of the estimator and defines their inputs and outputs. All of the signals represent both a mean value ($\mathbf{y}$ or $\mathbf{r}$) and the associated covariance factor ($P^{1/2}$ or $P_r^{1/2}$). The overbar indicates predicted estimates (e.g., $\bar{\mathbf{y}}_k = \hat{\mathbf{y}}_{k|k-1}$) and the hat indicates corrected estimates (e.g., $\hat{\mathbf{y}}_k = \hat{\mathbf{y}}_{k|k}$). The diagram shows that the past command $\mathbf{u}_{1,k-1}$ and the current measurement from the camera ($\mathbf{z}_{s,k}$) and from the accelerometers ($\mathbf{z}_{a,k}$) are inputs to the measurement update. The current command ($\mathbf{u}_{1,k}$) and the rate gyro measurement ($\mathbf{z}_{\omega,k}$) are inputs to the time update. The estimator output block extracts the relative position estimate from the full estimator state.

The estimator stores and operates on the square-root factor of the covariance. A square-root factor of a real-valued[1], nonnegative-definite matrix P is any matrix A that satisfies [29, Chapter 12]

$$P = AA^T.$$

Square-root factors are not unique, which is a property that is exploited in both the time and measurement updates. Consider a unitary matrix Θ (i.e., $\Theta\Theta^T = I$). Then, $P = AA^T = A\Theta\Theta^T A^T$ implies that $A\Theta$ is also a square-root factor of P. Sometimes, $P^{1/2}$ is reserved for symmetric square-root factors of P $\left(\text{such that } P = \left(P^{1/2}\right)^2\right)$, but in this dissertation, $P^{1/2}$ denotes any square-root factor of P.

[1]The scope of this presentation is limited to real-valued matrices, although complex matrices can also be handled by applying appropriate conjugates.

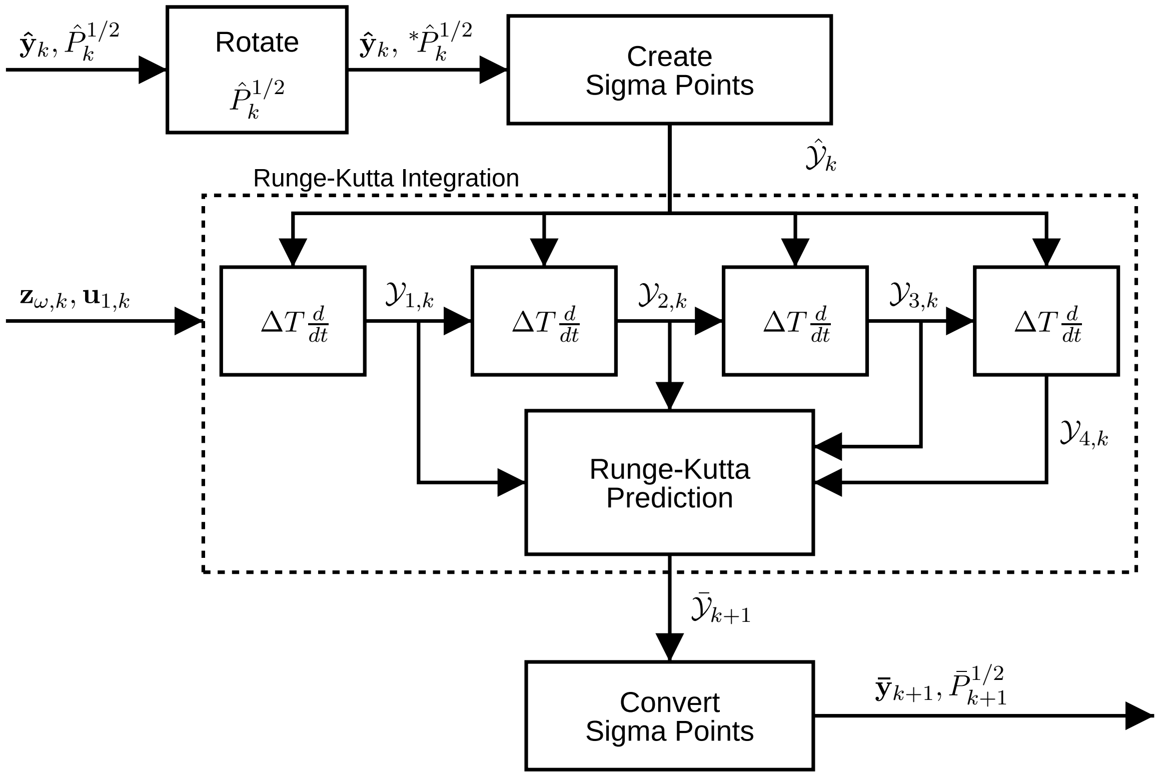

Figure 4.2: Implementation of the Time Update

4.3 Time Update

The time update of the Kalman Filter applies the process model to the current estimate to predict the state estimate at a future time. The nonlinearities of the process model are handled with the Unscented Transform (UT). The continuous-time dynamics are implemented with a Runge-Kutta integration.

The UT results in a more accurate solution because the nonlinear system equations can be handled without linearization. Furthermore, the UT yields a simpler implementation because it does not require the computation of any derivatives. Although the UT still requires approximations whose effect cannot be bounded, it tends to produce better results for this problem.

Figure 4.2 shows how the UT and the Runge-Kutta integration interact. The corrected state estimates $\left(\hat{\mathbf{y}}_k \text{ and } \hat{P}_k^{1/2}\right)$ are converted to sigma points $\hat{\mathcal{Y}}_k$, which are then propagated forward in time by the Runge-Kutta integration. The output of the integration is a new set of sigma points $\bar{\mathcal{Y}}_{k+1}$ which can be converted back to a mean $\bar{\mathbf{y}}_{k+1}$ and a square-root factor $\bar{P}_{k+1}^{1/2}$.

4.3.1 Square-Root Unscented Transform

The implementation of the UT requires two parts: creating sigma points from a mean and covariance and converting sigma points back to a mean and covariance. The square-root version of the UT, which produces a square-root factor instead of a proper covariance in the second step, has been described in van der Merwe and Wan [56].

Before creating the sigma points, the state vector has to be augmented to handle nonlinear process noise $\mathbf{n}_p$. This involves adding states that are set to zero but have a covariance equal to Q. Let the augmented state be $\mathbf{Y} \in \mathbb{R}^{l\times 1}$, ($l = n+p$) and the associated square-root factor of the covariance be $S \in \mathbb{R}^{l\times l}$.

$$\hat{\mathbf{Y}}_k = \begin{bmatrix} \hat{\mathbf{y}}_k \\ 0_{l\times 1} \end{bmatrix} \tag{4.18}$$

$$\hat{S}_k = \begin{bmatrix} {}^*\hat{P}_k^{1/2} & 0 \\ 0 & Q^{1/2} \end{bmatrix} \tag{4.19}$$

The UT uses $2l+1$ sigma points for vectors of length l. The first point is the actual mean. The remaining points are computed by adding and subtracting columns of the square-root factor of the covariance to the mean. The sigma points are stacked horizontally as columns of the matrix $\hat{\mathcal{Y}}_k \in \mathbb{R}^{l\times(2l+1)}$.

$$\hat{\mathcal{Y}}_k = \hat{\mathbf{Y}}_k + \eta \begin{bmatrix} 0_{l\times 1} & \hat{S}_k & -\hat{S}_k \end{bmatrix} \tag{4.20}$$

η is a design parameter that controls the spread of the sigma points. The optimal value for η is $\sqrt{3}$.

The power of the sigma point representation is that the mean and covariance of the original distribution matches that of the sigma points:

$$\hat{\mathbf{Y}}_k = \frac{1}{2l+1}\sum_{i=0}^{2l} \hat{\mathcal{Y}}_{k,i} \tag{4.21}$$

$$\hat{S}_k\hat{S}_k^T = \frac{1}{2\eta^2}\sum_{i=0}^{2l} \left(\hat{\mathcal{Y}}_{k,i} - \hat{\mathbf{Y}}_k\right)\left(\hat{\mathcal{Y}}_{k,i} - \hat{\mathbf{Y}}_k\right)^T \tag{4.22}$$

where $\hat{\mathcal{Y}}_{k,i}$ represents the ith column of $\hat{\mathcal{Y}}_k$.

Equations 4.21 and 4.22 can be generalized to provide additional design parameters. Equations 4.23 and 4.24 correspond to the implementation suggested in [56].

$$\hat{\mathbf{Y}}_k = \sum_{i=0}^{2l} W_i \hat{\mathcal{Y}}_{k,i} \tag{4.23}$$

$$\begin{aligned} \hat{S}_k\hat{S}_k^T &= \sum_{i=0}^{2l} W_i \left(\hat{\mathcal{Y}}_{k,i} - \hat{\mathbf{Y}}_k\right)\left(\hat{\mathcal{Y}}_{k,i} - \hat{\mathbf{Y}}_k\right)^T \\ &\quad + \left(1 - \eta^2/l + \beta\right)\left(\hat{\mathcal{Y}}_{k,0} - \hat{\mathbf{Y}}_k\right)\left(\hat{\mathcal{Y}}_{k,0} - \hat{\mathbf{Y}}_k\right)^T \end{aligned} \tag{4.24}$$

$$W_i = \begin{cases} \frac{\eta^2 - l}{\eta^2}, & i = 0 \\ \frac{1}{2\eta^2}, & i > 0 \end{cases} \tag{4.25}$$

The two parameters η and β are used to control properties of the sigma points that are beyond matching their first and second moments. η controls the spreading of the sigma points about the mean. A useful setting is $\eta = \sqrt{3}$. β can be used to match higher moments. If the distribution is in fact Gaussian, $\beta = 2$ is optimal.

Propagating these sigma points through a nonlinear transformation generates a new set of sigma points that approximates the true distribution of the output. This mapping corresponds to the nonlinear continuous-time dynamics.

$$\bar{\mathcal{Y}}_{k+1} = \mathcal{F}\left(\hat{\mathcal{Y}}_k, \mathbf{u}_{1,k}, \mathbf{z}_{\omega,k}, \Delta T\right). \tag{4.26}$$

Section 4.3.4 explains how (4.26) is implemented. Applying (4.23) and (4.24) to $\bar{\mathcal{Y}}_{k+1}$ generates an approximation of the true mean and covariance of the predicted state.

However, the desired output includes the square-root factor $\bar{S}_{k+1}$ of the covariance. This can be obtained by rewriting these equations as follows:

$$\bar{\mathbf{Y}}_{k+1} = \sum_{i=0}^{2l} W_i \bar{\mathcal{Y}}_{k+1,i} \tag{4.27}$$

$$\begin{aligned} \bar{S}_{k+1}\bar{S}_{k+1}^T &= \sum_{i=0}^{2l} W_i \left(\bar{\mathcal{Y}}_{k+1,i} - \bar{\mathbf{Y}}_{k+1}\right)\left(\bar{\mathcal{Y}}_{k+1,i} - \bar{\mathbf{Y}}_{k+1}\right)^T \\ &\quad + \left(1 - \eta^2/l + \beta\right)\left(\bar{\mathcal{Y}}_{k+1,0} - \bar{\mathbf{Y}}_{k+1}\right)\left(\bar{\mathcal{Y}}_{k+1,0} - \bar{\mathbf{Y}}_{k+1}\right)^T && (4.28) \\ &= \mathcal{A}\mathcal{A}^T && (4.29) \\ &= \begin{bmatrix} A & 0_{l\times(l+1)} \end{bmatrix} \Theta\Theta^T \begin{bmatrix} A^T \\ 0_{(l+1)\times l} \end{bmatrix} && (4.30) \\ &= AA^T && (4.31) \end{aligned}$$

where

$$\mathcal{A} = \left[\begin{array}{cc} \sqrt{W_0^*}\left(\bar{\mathcal{Y}}_{k+1,0} - \bar{\mathbf{Y}}_{k+1}\right) & \sqrt{W_1}\left(\bar{\mathcal{Y}}_{k+1,1:2l} - \bar{\mathbf{Y}}_{k+1}\right) \end{array} \right] \tag{4.32}$$

$$W_0^* = W_0 + \left(1 - \eta^2/l + \beta\right) \tag{4.33}$$

$$I = \Theta\Theta^T \tag{4.34}$$

If a Θ that satisfies $\mathcal{A} = \left[\begin{array}{cc} A & 0_{l\times(l+1)} \end{array} \right]\Theta$ can be found, this approach produces a square-root factor without first computing the covariance by associating $\bar{S}_{k+1} = A$. Of course, a Θ can always be found with a QR decomposition, which is a standard matrix algorithm [29, Appendix A].

Equation 4.32 assumes that $W_0^* \geq 0$, which is not necessarily the case, especially for small η (note that $W_1 \geq 0$ is always true). The correct way to handle $W_0^* < 0$ is with a Cholesky downdate algorithm. However, a much simpler alternative is to set

$$W_0^* = max\left\{0, W_0 + \left(1 - \eta^2/l + \beta\right)\right\} \tag{4.35}$$

This approximation results in no meaningful difference in performance and has therefore been integrated into the design.

4.3.2 Related Work on Sigma Point Methods

Sigma Point Methods approximate a probability distribution by a deterministic set of sigma points. This enables the propagation of the distribution through a nonlinear mapping without approximating the mapping. The Symmetric Unscented Transform, which has been discussed in this section, is one of those methods. Much of the research on Sigma Point Methods is recent and on-going, and consequently, new variations, with specific advantages for particular problems, are still presented and compared in the literature.

Julier, Uhlmann and Durrant-Whyte [24, 25] first introduced this technique. Ito and Xiong [22] have presented Sigma Point Methods as Gaussian filters and derived several variations, one of which is the Unscented Transform. They also present the Central Difference Filter (CDF) and claim that it is more accurate than the UT with similar cost. Van der Merwe and Wan have introduced the square-root form of the UT and the CDF in [55]. More recently, Julier and Uhlmann [28] have presented a new point selection method that reduces the required number of sigma points to just $l + 1$. Julier [27] has also presented an improved method to scale sigma points.

All of these methods avoid the linearization of the system equations and significantly outperform the EKF. The effect of their differences has not been evaluated for this problem. Additional performance improvements could be gained by considering other Sigma Point

Methods. The UT was chosen because it is simple to implement and affords an efficient square-root implementation.

4.3.3 Continuous-Time Dynamics

The nonlinear transformation of (4.26) is based on the integration of the continuous-time dynamics presented in (4.15). The function $\mathcal{F}()$ (defined in Section 4.3.4) implements a Runge-Kutta integration that uses the function $F()$ below. Let $\mathcal{Y} \in \mathbb{R}^{l\times(2l+1)}$. It contains $2l+1$ sigma points, each composed of a state $\mathbf{y}$ and a process noise $\mathbf{n}_p$.

$$\mathcal{Y} = \begin{bmatrix} \mathbf{y}_0 & \mathbf{y}_1 & \cdots & \mathbf{y}_{2l} \\ \mathbf{n}_{p,0} & \mathbf{n}_{p,1} & \cdots & \mathbf{n}_{p,2l} \end{bmatrix} \tag{4.36}$$

Now define

$$F(\mathcal{Y}, \mathbf{u}_1, \mathbf{z}_\omega, \Delta T) \tag{4.37}$$
$$= \Delta T \frac{d}{dt}\mathcal{Y} \tag{4.38}$$
$$= \Delta T \begin{bmatrix} f(\mathbf{y}_0, \mathbf{u}_1, \mathbf{z}_\omega, \mathbf{n}_{p,0}) & f(\mathbf{y}_1, \mathbf{u}_1, \mathbf{z}_\omega, \mathbf{n}_{p,1}) & \cdots & f(\mathbf{y}_{2l}, \mathbf{u}_1, \mathbf{z}_\omega, \mathbf{n}_{p,2l}) \\ 0_{p\times1} & 0_{p\times1} & \cdots & 0_{p\times1} \end{bmatrix} \tag{4.39}$$

Note that the function $f()$ in (4.15) is singular for $\frac{d}{dt}\psi$ if $Z_y = Z_z = 0$. However, the actual implementation can avoid this singularity by using an alternative formulation. From the inverse of (2.5), the angular velocity can be related to the rate of change of the Euler angles.

$$R^T\boldsymbol{\omega} = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi & 0 \\ \sin\psi\cos\theta & \cos\psi & 0 \\ -\sin\theta & 0 & 1 \end{bmatrix} \frac{d\boldsymbol{\lambda}}{dt} \tag{4.40}$$

$\mathbf{R}_z$ is the last column of (2.3) and is given by

$$\mathbf{R}_z = \begin{bmatrix} R_{zx} \\ R_{zy} \\ R_{zz} \end{bmatrix} = R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{bmatrix}. \tag{4.41}$$

Although $\mathbf{R}_z$ does not appear explicitly in the state vector $\mathbf{y}$, it can be computed from the $\mathbf{Z}$ component of $\mathbf{y}$.

$$\mathbf{R}_z = \frac{\mathbf{Z}}{\|\mathbf{Z}\|} \tag{4.42}$$

A new expression can be obtained by extracting the last row of (4.40), substituting $\mathbf{R}_z$ and $\frac{d\boldsymbol{\lambda}}{dt} = \begin{bmatrix} \frac{d\phi}{dt} & \frac{d\theta}{dt} & \frac{d\psi}{dt} \end{bmatrix}^T$, and rewriting.

$$\begin{aligned} \mathbf{R}_z^T\boldsymbol{\omega} &= \begin{bmatrix} -\sin\theta & 0 & 1 \end{bmatrix} \frac{d\boldsymbol{\lambda}}{dt} && (4.43) \\ &= \begin{bmatrix} R_{zx} & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{d\phi}{dt} \\ \frac{d\theta}{dt} \\ \frac{d\psi}{dt} \end{bmatrix} && (4.44) \\ &= R_{zx}\frac{d\phi}{dt} + \frac{d\psi}{dt} && (4.45) \end{aligned}$$

Integrating over time ΔT and rearranging yields a new expression for $\Delta\psi$ which does not have any singularities.

$$\Delta\psi = \mathbf{R}_z^T\boldsymbol{\omega}\Delta T - R_{zx}\Delta\phi \tag{4.46}$$

This expression depends on $\Delta\phi$, which can be computed from $\mathbf{R}_z$ and $\Delta\mathbf{R}_z$.

$$\begin{aligned} \phi &= \tan^{-1}\frac{R_{zy}}{R_{zz}} && (4.47) \\ \Delta\phi &= tan^{-1}\frac{R_{zy}+\Delta R_{zy}}{R_{zz}+\Delta R_{zz}} - tan^{-1}\frac{R_{zy}}{R_{zz}}. && (4.48) \end{aligned}$$

The ability to take advantage of this singularity-free approach is an advantage of the Unscented Transform, which does not require Jacobians of the system equations. The manner in which the nonlinear transformation is computed is not important. This is in contrast to methods like the EKF, which requires explicit Jacobians.

4.3.4 Runge-Kutta Integration

The time update uses a standard fourth-order Runge-Kutta algorithm [40, Section 15.1] to integrate the process model. This generates the function $\mathcal{F}()$ used by the UT to propagate the sigma points.

$$\bar{\mathcal{Y}}_{k+1} = \mathcal{F}\Big(\hat{\mathcal{Y}}_k, \mathbf{u}_{1,k}, \mathbf{z}_{\omega,k}, \Delta T\Big) \tag{4.49}$$

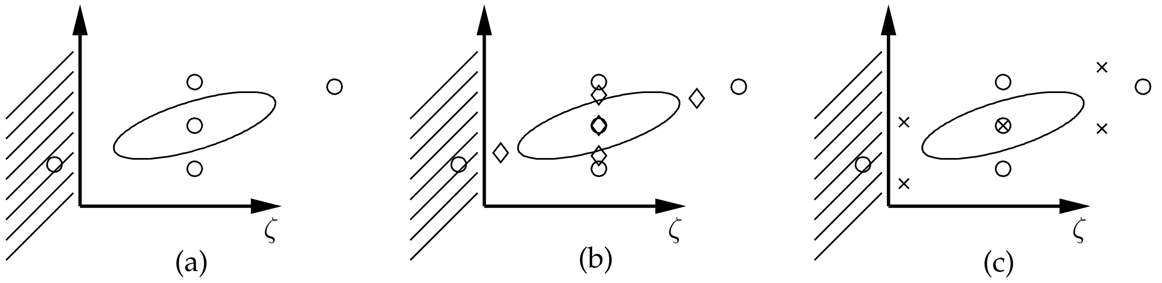

Figure 4.3: Rotation of $\hat{\mathbf{P}}_{\mathbf{k}}^{\mathbf{1/2}}$

For large uncertainties, it is possible for the sigma points to be chosen such that $\zeta < 0$ for some of the points, which is non-sensical. Plot (a) shows a two-dimensional example for which the sigma points ($\circ$) have been chosen from a distribution indicated by the ellipse. ζ corresponds to the horizontal axis. Plot (b) shows a second set of sigma points ($\diamond$) which have been chosen with a reduced η such that all sigma points satisfy $\zeta > 0$. Plot (c) shows an improved choice of sigma points ($\times$) found by rotating the square root factor of the co-variance. Note that these sigma points are stacked vertically, minimizing the spreading of ζ for a given η.

$$\mathcal{F} = \begin{cases} \mathcal{Y}_{RK}^1 &= F\Big(\hat{\mathcal{Y}}_k, \mathbf{u}_{1,k}, \mathbf{z}_{\omega,k}, \Delta T\Big) \\ \mathcal{Y}_{RK}^2 &= F\Big(\hat{\mathcal{Y}}_k + a_{21}\mathcal{Y}_{RK}^1, \mathbf{u}_{1,k}, \mathbf{z}_{\omega,k}, \Delta T\Big) \\ \mathcal{Y}_{RK}^3 &= F\Big(\hat{\mathcal{Y}}_k + a_{32}\mathcal{Y}_{RK}^2, \mathbf{u}_{1,k}, \mathbf{z}_{\omega,k}, \Delta T\Big) \\ \mathcal{Y}_{RK}^4 &= F\Big(\hat{\mathcal{Y}}_k + a_{43}\mathcal{Y}_{RK}^3, \mathbf{u}_{1,k}, \mathbf{z}_{\omega,k}, \Delta T\Big) \\ \bar{\mathcal{Y}}_{k+1} &= \hat{\mathcal{Y}}_k + \sum_{i=1}^4 \alpha_i \mathcal{Y}_{RK}^i \\ a_{21} = a_{32} &= 1/2 \\ a_{43} &= 1 \\ \alpha_1 = \alpha_4 &= 1/6 \\ \alpha_2 = \alpha_3 &= 1/3 \end{cases} \tag{4.50}$$

4.3.5 Rotation of $\hat{\mathbf{P}}_{\mathbf{k}}^{\mathbf{1/2}}$

The inverse range estimate ζ in this sensor fusion problem is constrained to be a positive quantity ($\zeta > 0$), which expresses the requirement that the object is positioned in front of the camera. However, if the uncertainty of ζ is large, it is possible to use (4.20) to construct sigma points with negative values of ζ. Figure 4.3 illustrates this problem and its solution using a simple two-dimensional example. Let S_ζ denote that row of $\hat{P}_k^{1/2}$ which corresponds to ζ. If

$$\eta\,(max\,|S_\zeta|) \quad > \quad \hat{\zeta} \tag{4.51}$$

then at least one sigma point will have a negative ζ-value, which is not permissible.

To avoid negative values of ζ, there are two degrees-of-freedom that can be exploited in the selection of sigma points, shown in (4.20). First, η can be reduced below its optimal value of $\sqrt{3}$ to reduce the spread of the sigma points. The following rule uses this approach to compute a permissible value of η.

$$\eta \;=\; min\left\{\sqrt{3}, \frac{\zeta - \zeta_{min}}{max\,|S_\zeta|}\right\} \tag{4.52}$$

where $0 < \zeta_{min} < \zeta$ represents a lower bound for the ζ-values of the sigma points.

However, reducing η below its optimal value has adverse consequences for the accuracy of the unscented transform. By contracting the sigma points towards the mean value, less of the uncertainty of the estimate is exposed to the nonlinearity. Therefore, adjusting η should be used as a last resort.

The second available degree-of-freedom is the square-root factor of the covariance, which is not unique. This section explores a particular rotation of $\hat{P}_k^{1/2}$ that leads to a set of sigma points with the least amount of spreading in the ζ-value for a given η value. When this optimization is combined with (4.52), the largest possible values of η are generated. This rotation is the first step of the time update depicted in Figure 4.2.

It is possible to rotate $\hat{P}_k^{1/2}$ by a unitary matrix Θ.

$$\hat{P}_k \;=\; \hat{P}_k^{1/2}\hat{P}_k^{T/2} = \hat{P}_k^{1/2}\Theta\Theta^T\hat{P}_k^{T/2} = {}^*\hat{P}_k^{1/2}\,{}^*\hat{P}_k^{T/2} \tag{4.53}$$

$${}^*\hat{P}_k^{1/2} \;=\; \hat{P}_k^{1/2}\Theta \tag{4.54}$$

$$I \;=\; \Theta\Theta^T \tag{4.55}$$

Let ${}^*S_\zeta$ be the corresponding row of ${}^*\hat{P}_k^{1/2}$. Then a Θ can be chosen that yields any desired values of ${}^*S_\zeta$ subject to a constraint on the variance of ζ.

$$\sigma^2(\zeta) \;=\; S_\zeta S_\zeta^T = {}^*S_\zeta\,{}^*S_\zeta^T \tag{4.56}$$

If ${}^*S_\zeta$ has length n, a useful objective is to satisfy

$$max\,|\,{}^*S_\zeta| \;<\; \kappa\sigma(\zeta)\,/\sqrt{n} \tag{4.57}$$

$$\kappa \;\geq\; 1 \tag{4.58}$$

If $\kappa = 1$, all entries of ${}^*S_\zeta$ have absolute values of $\sigma(\zeta)\,/\sqrt{n}$. This is the rotation that generates the least amount of spreading of the sigma points in the ζ-component. Values of $\kappa > 1$ lead to more variation in the entries of ${}^*S_\zeta$. The objective can be satisfied by iteratively applying j Givens-like rotations [29, Appendix B.2] between the two entries with the largest and the smallest absolute values in the remaining ${}^*S_\zeta$ $(j \leq n)$.

4.4 Measurement Update

As a result of the design choices described above, the measurement update of the Kalman Filter framework is identical to that of a linear system. It has been implemented as an array algorithm to improve its numerical properties and to interoperate more easily with the square-root UT. The measurement update produces the corrected estimates $\left(\hat{\mathbf{y}}_k, \hat{P}_k^{1/2}\right)$ from the predicted estimates $\left(\bar{\mathbf{y}}_k, \bar{P}_k^{1/2}\right)$ by incorporating the current measurement $\mathbf{z}_k$.

This section presents a summary of the procedure presented in [29, Section 12.3]. It involves assembling A, the pre-array, rotating it into a lower-triangular post-array B, associating the required components from B, computing the Kalman gain L, and correcting the estimate.

The pre-array is assembled from the square-root factor of the predicted covariance estimate $\bar{P}_k^{1/2}$, the square-root of the measurement noise covariance $R^{1/2}$, and H, the matrix describing the linear sensor model in (4.13).

$$A = \begin{bmatrix} R^{1/2} & H\bar{P}_k^{1/2} \\ 0_{n\times m} & \bar{P}_k^{1/2} \end{bmatrix} \tag{4.59}$$

The lower-triangular post-array is computed from the pre-array with a rotation by the unitary matrix Θ.

$$B = A\Theta \tag{4.60}$$

$$= \begin{bmatrix} X & 0_{m\times n} \\ Y & Z \end{bmatrix} \tag{4.61}$$

$$I = \Theta\Theta^T \tag{4.62}$$

$$X \in \mathbb{R}^{m\times m} \tag{4.63}$$

$$Y \in \mathbb{R}^{n\times m} \tag{4.64}$$

$$Z \in \mathbb{R}^{n\times n} \tag{4.65}$$

By multiplying (4.60) by its transpose, it is possible to associate X, Y and Z with useful quantities.

$$BB^T = A\Theta\Theta^T A^T = AA^T \tag{4.66}$$

$$B = \begin{bmatrix} \hat{R}_{e,k}^{1/2} & 0_{m\times n} \\ \bar{P}_k H^T \hat{R}_{e,k}^{-T/2} & \hat{P}_k^{1/2} \end{bmatrix} \tag{4.67}$$

Because $\hat{R}_{e,k}^{1/2}$ is lower triangular, it is possible to compute L, the Kalman gain, with an efficient backward substitution.

$$L_k = \left[\bar{P}_k H^T \hat{R}_{e,k}^{-T/2}\right] / \left[\hat{R}_{e,k}^{1/2}\right] \tag{4.68}$$

Finally, L is used to correct the predicted estimate with the current measurement.

$$\hat{\mathbf{y}}_k = \bar{\mathbf{y}}_k + L_k \left(\mathbf{z}_k - H\bar{\mathbf{y}}_k + \begin{bmatrix} 0 \\ 0 \\ \mathbf{u}_{1,k} \end{bmatrix} \right) \tag{4.69}$$

4.5 Estimator Output

The estimator output, a relative position between the observer and the object, does not appear explicitly in the state vector $\mathbf{y}$. Instead, the output $\mathbf{r}$ (in body coordinates) is given by

$$\mathbf{r} = g(\mathbf{y}) \tag{4.70}$$

$$= \frac{1}{\zeta} \begin{bmatrix} s_x \\ s_y \\ 1 \end{bmatrix} \tag{4.71}$$

Again, the nonlinearity, coupled with potentially large uncertainty of ζ, can lead to a biased result. To reduce errors caused by linearization, this nonlinear transformation can also be implemented with a straight-forward application of the UT using 7 sigma points.

4.6 Summary

This chapter presents a solution for the nonlinear estimation problem described in Chapter 2. The design is based on a Kalman Filter framework and incorporates several modifications that address the unique requirements of this estimation problem. The solution includes a new representation of the system representation that yields a linear sensor model. The square-root version of the unscented transform is used to propagate the estimate through the nonlinear state dynamics. The next chapter describes the experimental system that was used to validate this design. Results from these experiments are presented in Chapter 6.

Chapter 5

Experimental System

An important part of this research is the experimental demonstration of an autonomous object pick-up task that is enabled by the sensing strategy. This chapter describes the experimental hardware, the demonstration task, the simulation system, and the trajectory that was used to generate these results.

This relative position sensing strategy, based on fusing monocular vision measurements of a single feature and inertial rate sensor measurements, has been pursued in this research because it has the potential to be very robust in real underwater environments and because it can be implemented with sensors commonly available on underwater vehicles. A robust measurement of relative position is required to enable autonomous manipulation for underwater vehicles. However, to evaluate the sensing strategy and to demonstrate how it could be applied for autonomous manipulation with an underwater vehicle, an experiment with a fixed-base manipulator arm has been constructed. In this experiment, the endpoint of the manipulator arm simulates the motion of an underwater vehicle. This experiment, which is described in detail in this chapter, is more useful than an underwater vehicle for this research because it provides a flexible development environment and a truth measurement of relative position.

Section 5.1 describes the robot, sensors, computer systems and other hardware required to perform the demonstration task. Section 5.2 describes the demonstration task that was used to validate the sensing strategy on the experimental system. Section 5.3 focuses on the control system for the robot, including the implementation of the desired trajectory and the simulation of typical dynamics and disturbances of an underwater vehicle. Section 5.4 presents some calibrations of the experimental hardware. Section 5.5 explains the difference between the relative position that is produced by the sensing system and the relative position that is used for control of the manipulator arm. It also describes a method for determining a truth measurement for the relative position. Section 5.6 summarizes all of the important parameter values for the experimental system. Section 5.7 presents the

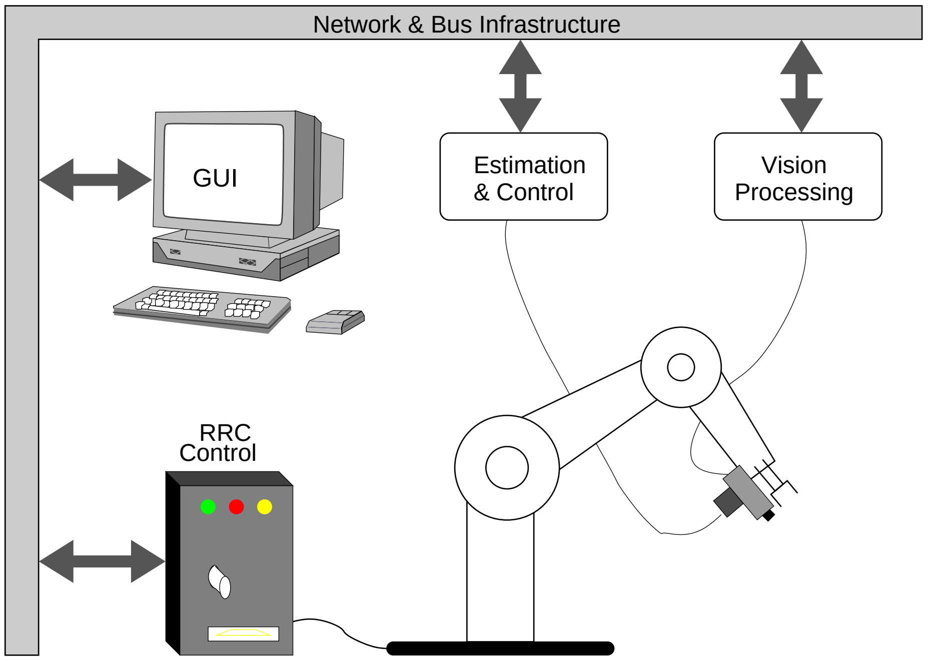

Figure 5.1: Functional Diagram of the Experimental System

software-based simulation of the hardware experiment. Finally, Section 5.8 presents the trajectory design for the object pick-up task.

5.1 Hardware

The experimental system is constructed around the K-1607 manipulator arm by Robotics Research Corporation (RRC). This manipulator simulates the motion and manipulation capabilities of an intervention-capable underwater vehicle. The endpoint of the manipulator carries a pneumatic gripper, a camera, and an inertial rate sensor package.

An important choice in the design of the experimental system was to utilize only sensors that can also be used with underwater vehicles. This approach ensures that the estimator design is based on adequate sensor models and that the results from experiments are meaningful in the context of an underwater vehicle application.

In addition to the manipulator and the sensors, the experimental system includes several computers, the network and bus infrastructure to connect these computers, and some special purpose hardware. All of these components are shown in Figure 5.1. The main computer handles the estimation and most of the control algorithms. This is a PowerPC running the VxWorks operating system. It sends manipulator motion commands to and receives joint angle updates from the RRC Control cabinet via a special purpose bus interface. This cabinet contains a computer, some analog circuits for control and the power amplifiers for the motors in the manipulator. The video signal from the camera is connected to the vision processing system, which reports bearing measurements to the network. The inertial rate sensors are connected directly to the estimation and control computer via a serial link. The graphical user interface (GUI) runs on a UNIX workstation connected to the network.

5.1.1 RRC Manipulator

The K-1607 manipulator arm (see Figure 1.6) by Robotics Research Corporation [46] is the core of the experimental system. It is a large, 7-DOF, kinematically redundant, serial-link electric manipulator located in the Aerospace Robotics Lab at Stanford University. The purpose of the manipulator is to simulate the motion of an underwater vehicle, to transport the vision and inertial rate sensors and to perform a simple, closed-loop manipulation task.

The commands to the manipulator can include desired joint angles, desired joint torques, or desired endpoint positions. The first approach—specifying desired joint angles at $200\,Hz$ to the manipulator—is the most useful for this experiment because it is simple and robust to implement, can easily support the control bandwidth of a typical underwater vehicle, and satisfies the simple requirements for the object manipulation task. Specifying only desired endpoint positions would be useful if the dynamics of the endpoint are not important. Specifying desired joint torques would be necessary to achieve greater control bandwidth and to perform manipulation tasks that require force control.

The joint angle commands are transfered to the RRC computer via a Bit3 bus interface. The RRC computer interfaces to the analog electronics and handles other functions, like homing the robot joints and applying the brakes when the robot is disabled. The difference between desired and actual joint angles is used by the controller to compute a pulse-width modulated (PWM) joint torque command.

Each joint of the manipulator is driven by an electric motor and has encoders to measure the actual joint angle. This joint angle measurement is used by the joint controller and is also published to the Bit3 bus interface.

5.1.2 Vision System

The experiment has a very simple vision system that tracks points in the camera view. This system uses a Pulnix TM-7EX [41] camera connected to the PointGrabber II hardware [5]. The PointGrabber segments points from the background of the video signal using a simple, constant-threshold algorithm. In the lab, this is feasible because it was possible to attach an infrared LED to the object and an infrared filter to the camera. This results in a dramatic contrast in the video signal between the bright spot of the LED and the dark background. It has the additional benefit of simplifying the interpretation of the scene and demonstrating that the estimation algorithm works with only one bearing measurement.

The VisionServer [48] software reads the point locations detected by the PointGrabber II hardware, applies a third-order polynomial to correct for lens distortion, tracks points in time, and reports point locations to the network. The estimator runs at $10\,Hz$ and therefore incorporates vision measurements from every third video frame. There is a delay of about three iterations (i.e., $0.3\,s$) in the system between the motion command and the vision measurements. The measurement noise of this vision system is approximately $\sigma(s_{x,y}) = 0.01$.

5.1.3 Inertial Rate Sensors

All six inertial rate sensors are collocated in the DMU-6X inertial measurement unit by Crossbow [8]. This is a low-cost solid-state device. The accelerometers measure differential capacitance between micro-machined cantilevers and the rate gyros are based on vibrating ceramic plates.

The random walk of the accelerometers due to measurement noise is rated at $0.15\,m/s/\sqrt{Hr}$. At a sample rate of $10\,Hz$, this is equivalent to measurement noise with a $0.008\,m/s^2$ standard deviation. The turn-on accelerometer bias has a standard deviation of $0.1\,m/s^2$. The random walk of the rate gyros due to measurement noise is rated at $0.35°/\sqrt{Hr}$. At a sample rate of $10\,Hz$, this is equivalent to measurement noise with a $0.0003\,rad/s$ standard deviation. The turn-on rate gyro bias has a standard deviation of $0.018\,rad/s$.

The inertial measurement unit generates measurements at about $160\,Hz$ and sends them to the main computer over a serial data link. They are down-converted and synchronized to the vision measurements using a symmetric finite impulse response (FIR) filter with 37 taps and a $3\,dB$ bandwidth of $3\,Hz$. This introduces a delay of $0.11\,s$, which was chosen to synchronize the inertial rate sensor measurements with the vision measurements.

Actual outputs from the inertial rate sensors led to the parameter values actually used for the inertial rate sensor models. The measurement noise is larger at $0.1\,m/s^2$ and $0.001\,rad/s$. The turn-on bias has been identified as $\begin{bmatrix} 0 & -0.23 & -0.12 \end{bmatrix} m/s^2$ with a standard deviation of $0.04\,m/s^2$ for the accelerometers and $\begin{bmatrix} 0.004 & -0.003 & 0.010 \end{bmatrix} rad/s$ with a standard deviation of $0.01\,rad/s$ for the rate gyros.

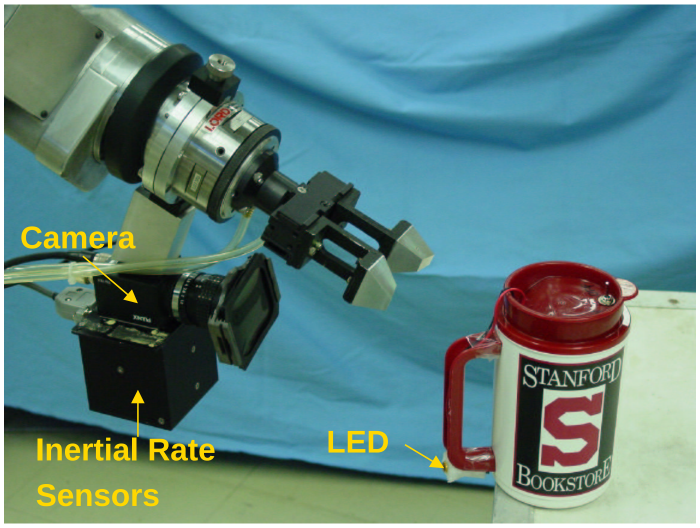

Figure 5.2: Manipulator Endpoint with the Camera, Inertial Rate Sensors, and Gripper; and the Cup with an Infrared LED

5.2 Object Pick-Up Task

A simple robotic task—picking up a cup with the manipulator arm—has been developed to demonstrate the capabilities of the sensing strategy. Figure 1.6 shows the K-1607 manipulator by Robotics Research Corporation in position to pick up a cup, placed at a position unknown to the robot. To complete the task, the robot has to determine its relative position and move to a desired relative position with sufficient accuracy to close its gripper successfully on the handle of the cup.

Figure 5.2 is a close-up of the endpoint of the robot arm and the object. It shows the pneumatic gripper, the camera, and the inertial rate sensors mounted on the robot endpoint. An infrared LED on the bottom of the cup handle provides an easy point feature for the camera to track.

The demonstration task is performed autonomously by the robot using a real-time implementation of the sensing strategy. Complete results are presented in the next chapter. Figure 5.3 shows a sequence of snapshots of the demonstration task. The task is initialized in Frame 1 such that the LED of the cup is visible in the camera's field of view. The relative position of the cup is not known by the estimator or the controller. The robot begins by rotating down (Frame 2) and back up (Frame 3 and 4) while controlling its relative position based on the current estimator output and the desired trajectory. Finally, it moves into the contact position (Frame 5), grasps the cup, and demonstrates success by raising the cup (Frame 6).

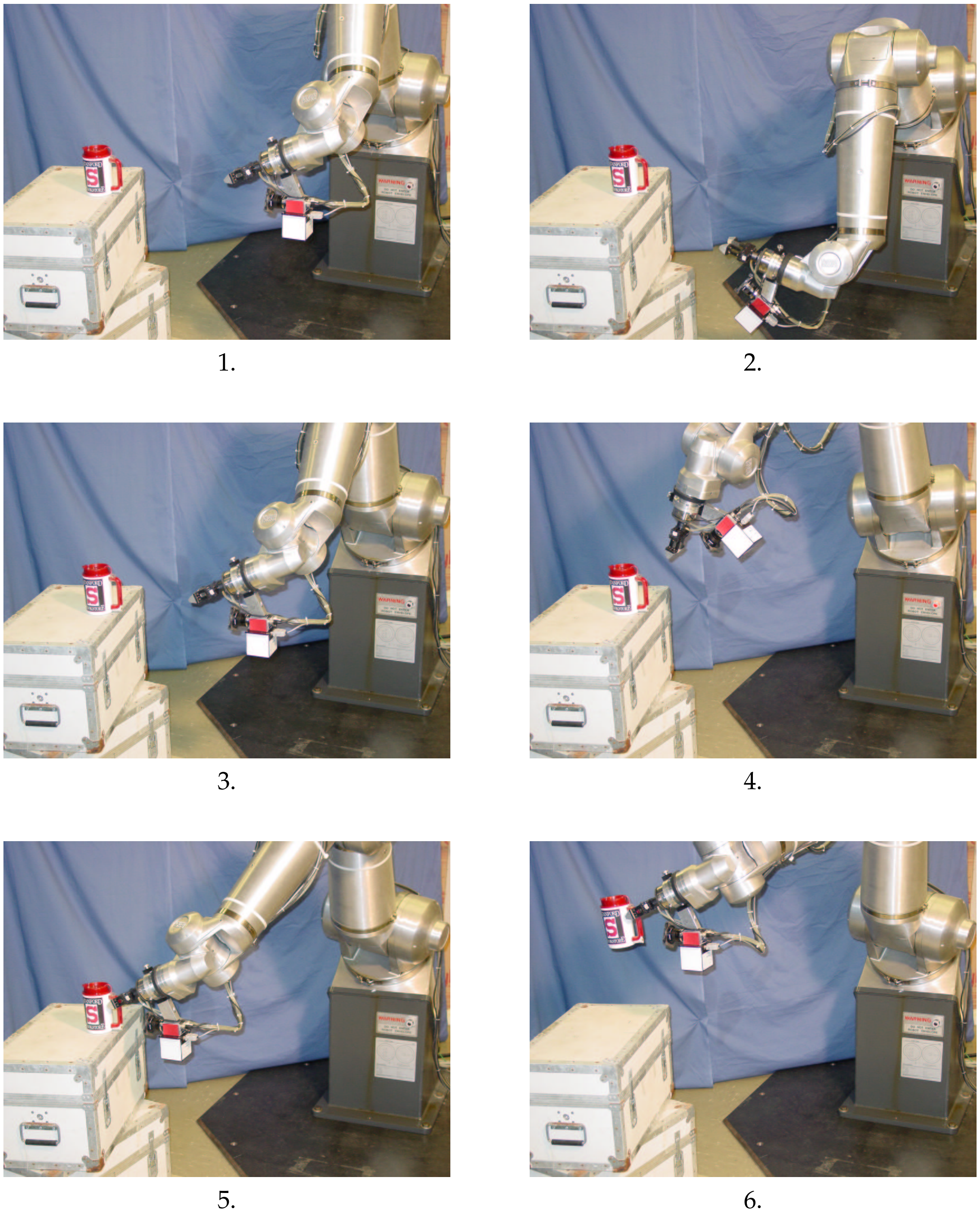

Figure 5.3: Sequence of Snapshots from a Successful Demonstration of the Object Pick-Up Task

This task can be interpreted in more than one context. For example, it shows how the sensing strategy can be employed on a fixed-base manipulator that requires an independent estimate of endpoint motion. However, the robot can also simulate an intervention-capable AUV. In this case, the manipulator endpoint, including the sensors and the gripper, represents the AUV. The links of the manipulator and the forces they transmit simulate the combined action of the AUV thrusters and any disturbances from the environment. It is this interpretation that is most useful for this dissertation.

The purpose of this demonstration task is to validate the relative position sensing strategy on a hardware experiment. Consequently, several assumptions have been applied to the task to avoid many of the other research issues associated with autonomous manipulation tasks. To eliminate the need for online object modeling and grasp planning, a known object is used. Is is assumed that the LED on the handle of the cup is the only feature visible in the camera image. This obviates the need for any reasoning about the scene and interpretation of the task.

This demonstration also takes a simplified approach to vision processing. With the help of an infrared filter on the camera and an infrared LED on the object, vision processing is reduced to applying a threshold to the image and extracting the centroid of the only suitable blob, which are routine capabilities. Effective processing of images in the field, especially in the underwater environment, is much more difficult, very application dependent, and beyond the scope of this dissertation. An important contribution of this work is to reduce the number of robust, trackable features required to determine relative position with a monocular vision system. Reducing this number has the potential to simplify the requirements on vision systems for real environments in a significant manner.

The demonstration assumes that the object's geometry and orientation are known. Without these assumptions, realistic tasks other than station-keeping cannot be defined by specifying only a single point feature. While only a single feature is required to determine relative position, more than one feature are necessary to define a general manipulation task. By assuming that the cup is always upright, that is always faces the same direction, and that the position offset between the object's LED and the grasp point is known, estimating the relative position between the robot and the LED is sufficient to perform the pick-up task.

Applying this sensing strategy in the context of real autonomous manipulation tasks requires more knowledge about the object than can be conveyed with a single feature. However, if the object is stationary, only the current bearing to a single feature is necessary in real-time to execute the task. All other information could be obtained before the manipulation task begins from non-real-time reasoning or from low-bandwidth interactions with a human supervisor.

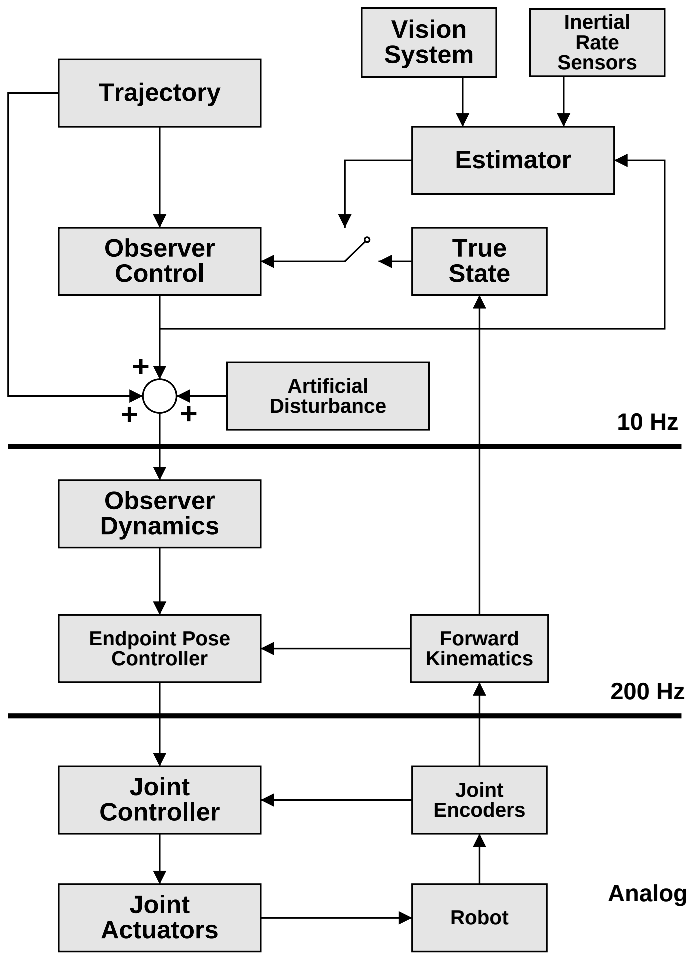

Figure 5.4: Block Diagram for the Experimental System

5.3 Control System

The control of the manipulator arm to simulate the motion of an underwater vehicle and to perform a manipulation task is distributed into three layers. The top layer, operating at $10\,Hz$ implements the task, which includes the relative position sensing system, the desired trajectory, and the observer control. The output of this level are observer force and torque commands. The middle level, operating at $200\,Hz$, computes desired robot joint angles to generate the endpoint motion that is implied by the observer force and torque commands. The bottom level, operating on analog hardware and some high-sample-rate digital hardware, implements the joint control for the robot. This system is depicted in Figure 5.4. This section describes the important parts of this control system except for the estimator, which has been discussed at length in Chapter 4.

5.3.1 Trajectory

The trajectory describes the observer motion as a function of time for the duration of the task. This description includes the desired relative position and velocity, the desired orientation and angular rates, and the nominal force and torque commands required to achieve this motion. The trajectory-following control includes a feedback and a feedforward component. The desired relative position and orientation, as well as their desired rates of change, are compared to the sensed values by the observer control, which computes the feedback forces and torques. These are added to the nominal commands (i.e., the feedforward control), which have been pre-computed in the trajectory design. The design of the trajectory is described in Section 5.8.

5.3.2 Observer Control

The observer control generates the feedback control commands to align the observer motion with the nominal trajectory. It is implemented in body coordinates as a PD controller with saturation.

$$\mathbf{u}_{1,fb} \quad = \quad sat(-K_{1,p}\left(\mathbf{r}_{des} - \hat{\mathbf{r}}\right) + K_{1,d}\left(\mathbf{v}_{des} - \hat{\mathbf{v}}\right), u_{1,max}) \tag{5.1}$$

The control input for the rotational degrees-of-freedom is computed from the difference between the desired and estimated values of three parts: $\mathbf{R}_z$ (the direction of gravity in body coordinates), ψ (heading) and $\boldsymbol{\omega}$. The contribution from $\mathbf{R}_z$ should be perpendicular to $\mathbf{R}_z$ and the contribution from ψ should be parallel to $\mathbf{R}_z$. Consider the last column of (2.4):

$$\frac{d\mathbf{R}_z}{dt} \quad = \quad -\boldsymbol{\omega} \times \mathbf{R}_z \tag{5.2}$$

Let $\boldsymbol{\omega} = \boldsymbol{\omega}_1 + \boldsymbol{\omega}_2$ such that $\boldsymbol{\omega}_1$ is perpendicular to $\mathbf{R}_z$ (i.e., $\boldsymbol{\omega}_1^T\mathbf{R}_z = 0$) and $\boldsymbol{\omega}_2$ is parallel to $\mathbf{R}_z$ (i.e., $\boldsymbol{\omega}_2 \times \mathbf{R}_z = 0$). Taking the cross-product between $-\mathbf{R}_z$ and (5.2) results in a useful expression for motivating the proportional term in the control law.

$$\begin{aligned} -\mathbf{R}_z \times \frac{d\mathbf{R}_z}{dt} &= \mathbf{R}_z \times (\boldsymbol{\omega} \times \mathbf{R}_z) && (5.3) \\ &= \mathbf{R}_z \times (\boldsymbol{\omega}_1 \times \mathbf{R}_z) && (5.4) \\ &= \left(\mathbf{R}_z^T\mathbf{R}_z\right)\boldsymbol{\omega}_1 - \left(\mathbf{R}_z^T\boldsymbol{\omega}_1\right)\mathbf{R}_z && (5.5) \\ &= \boldsymbol{\omega}_1 && (5.6) \end{aligned}$$

Param.	Value	Units
$K_{1,p}$	2	$1/s^2$
$K_{1,d}$	2	$1/s$
$u_{1,max}$	0.2	m/s^2
$K_{2,p}$	2	rad/s^2
$K_{2,\psi}$	1	$1/s^2$
$K_{2,d}$	2	$1/s$
$u_{2,max}$	0.2	rad/s^2

Table 5.1: Parameters for the Observer Control

Now, the error in $\mathbf{R}_z$ can be associated with $d\mathbf{R}_z/dt$ and the related torque command with $\boldsymbol{\omega}_1$.

$$\begin{aligned}\mathbf{u}_{2,fb} &= sat\Big(K_{2,p}\left(-\hat{\mathbf{R}}_z \times \left(\mathbf{R}_{z,des} - \hat{\mathbf{R}}_z\right)\right) \\ &\quad + K_{2,\psi}\hat{\mathbf{R}}_z\left(\psi_{des} - \hat{\psi}\right) + K_{2,d}\left(\boldsymbol{\omega}_{des} - \hat{\boldsymbol{\omega}}\right), u_{2,max}\Big) \qquad (5.7) \\ &= sat\Big(K_{2,p}\left(\mathbf{R}_{z,des} \times \hat{\mathbf{R}}_z\right) \\ &\quad + K_{2,\psi}\hat{\mathbf{R}}_z\left(\psi_{des} - \hat{\psi}\right) + K_{2,d}\left(\boldsymbol{\omega}_{des} - \hat{\boldsymbol{\omega}}\right), u_{2,max}\Big) \qquad (5.8)\end{aligned}$$

Useful parameters for a $10\,Hz$ implementation of this controller are given in Table 5.1.

5.3.3 Observer Control Input

The feedback controller can operate in two different configurations based on its input. In the "control-from-estimate" configuration, it computes the feedback control from the estimator output of relative position and velocity, orientation, and angular rate. In the "control-from-truth" configuration, it computes the feedback control from the true state.

Both of these configurations will be useful for the experiments described in Chapter 6. The "control-from-estimate" configuration enables the object manipulation task by guiding the robot towards the object, which can be randomly placed in the robot workspace. The "control-from-truth" configuration is useful for controlling the robot independently from the estimator and is available if the object location has already been calibrated. Section 5.5.1 explains how the true state can be computed from the known object location and the endpoint pose, which is derived from the joint angle measurements. The switch on the observer control input in Figure 5.4 indicates this choice of configuration.

5.3.4 Simulated Disturbances

Free-floating underwater vehicles are subject to significant disturbances from the environment. Typical underwater disturbances are a dominant factor in the performance of the sensing strategy. Therefore, the experimental system includes the capability to generate artificial disturbance forces and to add these to the control commands for the manipulator. This approach provides insight into the performance of a potential underwater vehicle application by investigating the effect of disturbances of various magnitudes on the estimator results.

The experimental system is also subject to some real disturbance forces. These are caused by unmodeled effects like calibration errors, time delays and friction. However, the effect on performance from these disturbances is small. The contribution from the artificial disturbances is much more significant.

The Gauss-Markov models that have been defined to capture typical disturbances in the underwater environment have been described in Section 2.3 and are reprinted below. The vector $\mathbf{d}_1$ represents forces and $\mathbf{d}_2$ represents torques.

$$\frac{d}{dt}\mathbf{d}_1 = -\frac{1}{\tau_1}\mathbf{d}_1 + \mathbf{n}_{d1} \tag{5.9}$$

$$\frac{d}{dt}\mathbf{d}_2 = -\frac{1}{\tau_2}\mathbf{d}_2 + \mathbf{n}_{d2} \tag{5.10}$$

This section develops realistic parameters for τ_1, τ_2, $\sigma(\mathbf{n}_{d1})$ and $\sigma(\mathbf{n}_{d2})$.

The parameter values are based loosely on an experiment with the Ventana ROV [44]. In this experiment, a slowly varying bias force, with a time constant of approximately $\tau_1 = 60\,s$, was detected in the fore-aft direction of the vehicle. A typical size for the bias force in units of acceleration was $0.012\,m/s^2$. If this can be assumed to be the standard deviation of the bias force, then $\sigma(\mathbf{n}_{d1}) = \sqrt{2/\tau_1}\sigma(\mathbf{d}_1) = 0.0022\,m/s^3$. These values have been used for all three of the translational degrees-of-freedom.

Parameter values for the rotational motion were more difficult to derive from the experiment. Nevertheless, useful quantities have been identified: $\tau_2 = 60\,s$ and $\sigma(\mathbf{d}_2) = 0.01\,rad/s^2$. This leads to $\sigma(\mathbf{n}_{d2}) = \sqrt{2/\tau_2}\sigma(\mathbf{d}_2) = 0.0018\,rad/s^3$. In the experiment, discrete-time versions of these equations were implemented at $10\,Hz$.

5.3.5 Observer Dynamics

The manipulator has sufficiently powerful motors to simulate the dynamics of an underwater vehicle. This is achieved by propagating the commanded forces and torques through the dynamics defined in Section 2.3. This generates motion commands (desired velocities and rotational rates) specified in Cartesian coordinates. These motion commands are the input to the endpoint pose controller.

The input to the dynamics includes the feedback control computed by the observer control, the feedforward commands from the trajectory, and the simulated disturbances.

Discrete time versions of (2.15) and (2.17) were implemented at $200\,Hz$. The drag parameter in (2.15) was estimated to be $\gamma = 0.2 \ \ 1/s$ from the same experiment that was described in the previous section.

5.3.6 Endpoint Pose Controller

The purpose of the endpoint pose controller is to compute a set of robot joint velocities such that the robot endpoint moves according to the desired velocities and rotational rates. Pose represents the 6-DOF combination of position and orientation of the endpoint expressed in the robot base frame. The manipulator forward kinematics compute the pose of the endpoint from joint angle measurements.

The robot is kinematically redundant because it has seven degrees-of-freedom (joints), which is more than the necessary number to perform a 6-DOF task. Therefore, the joint velocities are not unique until a null space objective has been defined.

The implementation of the pose controller follows the standard designs for resolved rate controllers described in [7, 37] with the following customizations: The pseudo-inverse of the Jacobian matrix is computed using the singularity-robust method described in [37]. A specific joint-velocity weighting is computed with an algorithm described in [4] that guarantees joint-limit avoidance. The nullspace objective is to draw the joint angles towards the midpoint of their range of motion.

When the robot operates in the "control-from-estimate" configuration, the pose controller is the highest level in the control architecture where robot joint angle measurements are used.

5.4 Calibration

5.4.1 Camera Calibration

The camera measurements are corrected for lens distortion by two third-order polynomials, one for each axis of the image plane. The camera calibration involves computing a set of twenty constants that define these polynomials. The vision system described in Section 5.1.2 is typically installed for 2D applications with a fixed overhead camera above a table with constant object heights. In this case, the camera calibration is most easily performed with a calibration board that has many LEDs, each at known locations (see [5]). However, in this application, the camera is moving and the object range is variable, from about $20\,cm$ in front of the camera to up to $3\,m$. Therefore, a different calibration approach was developed.

If a and b are the uncalibrated vision plane measurements, then the calibrated measurements are given by

$$\begin{aligned} s_x &= C_{x,0}a^3 + C_{x,1}a^2 + C_{x,2}a + C_{x,3}b^3 + C_{x,4}b^2 + C_{x,5}b \\ &\quad + C_{x,6}a^2b + C_{x,7}ab^2 + C_{x,8}ab + C_{x,9} \qquad (5.11) \\ s_y &= C_{y,0}a^3 + C_{y,1}a^2 + C_{y,2}a + C_{y,3}b^3 + C_{y,4}b^2 + C_{y,5}b \\ &\quad + C_{y,6}a^2b + C_{y,7}ab^2 + C_{y,8}ab + C_{y,9} \qquad (5.12) \end{aligned}$$

By associating the actual pixel measurements with known values of s_x and s_y, these polynomials can calibrate not only the lens distortion, but also the conversion from pixel measurements to the dimensionless s-measurements and any offsets between the center of the lens and the center of the CCD.

The camera was calibrated with a single LED at a fixed position. The camera was then moved to various positions such that it covered all the expected object ranges and that it recorded feature positions across the entire field of view. At each position, the location of the camera in robot coordinates and the raw image plane position of the LED feature was recorded. Throughout this process, the orientation of the camera remained constant. A nonlinear optimization was then applied to determine both the calibration coefficients ($C_{x,n}$ and $C_{y,n}$) and the position of the LED in robot coordinates.

5.4.2 Calibration of Position Offsets

The parts of the system used to accomplish the object pick-up task are not collocated. The position and orientation offsets between the manipulator tool plate, inertial rate sensors, camera, gripper, LED, and cup handle are important for estimating the relative position between the inertial rate sensors and the LED and for using this estimate to control the position between the gripper and the cup handle. This section presents the calibration of these offsets.

Figure 5.5 shows the parts of the manipulator endpoint. The tool plate is the interface between the RRC manipulator and the custom parts that have been attached to the manipulator. The point marked *A1* is the center of the tool plate. The forward kinematics typically compute the position and orientation of the tool plate as a function of the joint angles. However, the observer body frame is centered on the inertial rate sensors (i.e., at the point marked *A2*), so the position offset P_{12} and the rotation Φ_{12} between *A1* and *A2* is

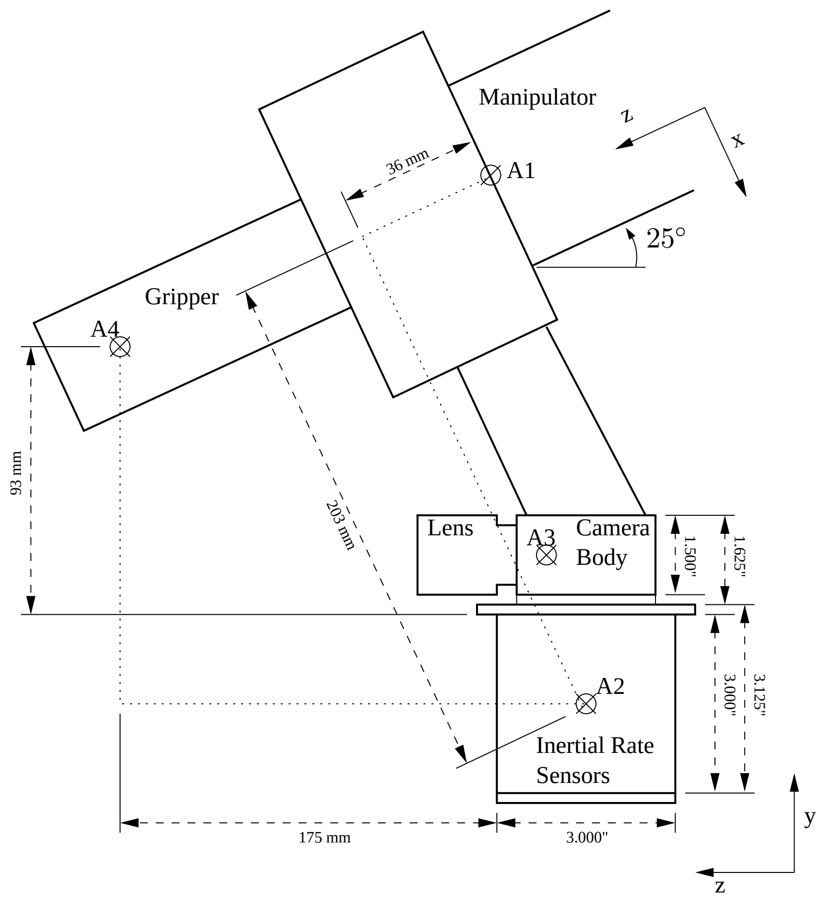

Figure 5.5: Details of the Endpoint Geometry

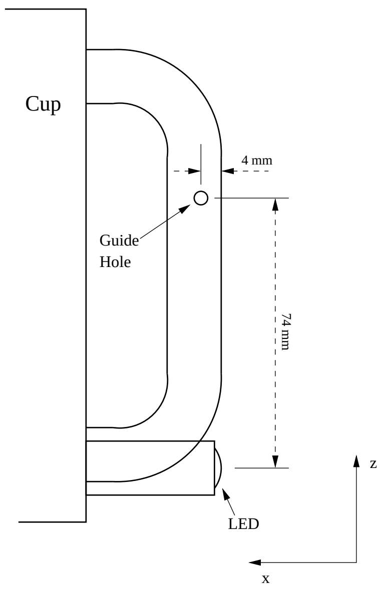

Figure 5.6: Details of the Cup Geometry

required to compute the correct forward kinematics.

$$P_{12} = \begin{bmatrix} 0.203 \\ 0 \\ 0.036 \end{bmatrix} m \qquad \Phi_{12} = \begin{bmatrix} -25^\circ \\ 0 \\ 90^\circ \end{bmatrix} \tag{5.13}$$

P_{12} is expressed in the tool plate coordinates. Φ_{12} is a *roll-pitch-yaw* vector also starting from the tool plate.

The remaining positions are all resolved in the observer body frame. P_{cam} is the camera position between *A3* and *A2* and P_{tool} is the tool position between *A4* and *A2*.

$$P_{cam} = \begin{bmatrix} 0 \\ 0.064 \\ 0.040 \end{bmatrix} m \qquad P_{tool} = \begin{bmatrix} 0 \\ 0.131 \\ 0.213 \end{bmatrix} m \tag{5.14}$$

However, the values

$$P_{tool} = \begin{bmatrix} -0.003 \\ 0.130 \\ 0.203 \end{bmatrix} m \tag{5.15}$$

produce more accurate results in the experiment, so these have been used instead of the measured values.

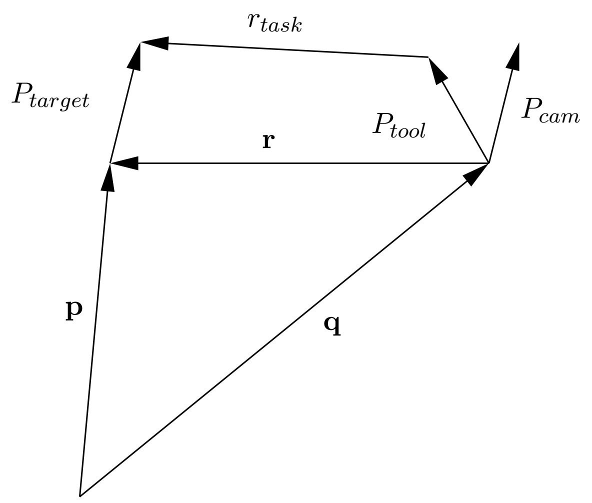

Figure 5.7: Geometry of the Object Pick-Up Task

Figure 5.6 shows the geometry of the cup. The LED is attached to the bottom of the cup handle. The grasp location of the cup is on the upper part of the handle. With this geometry, the LED will continue to be visible in the camera image during the grasping phase of the task. P_{target} is the position of the grasp position (the location of the guide hole) relative to the LED. This is expressed in inertial coordinates.

$$P_{target} = \begin{bmatrix} 0.004 \\ 0 \\ 0.074 \end{bmatrix} m \tag{5.16}$$

5.5 Task Geometry

The sensing system determines the relative position $\mathbf{r}$ between the LED and the observer body frame whereas the object pick-up task is based on the relative position $\mathbf{r}_{task}$ between the cup handle and the gripper. Figure 5.7 shows the difference between $\mathbf{r}$ and $\mathbf{r}_{task}$ based on the position offsets defined in the previous section.

$$\begin{aligned} \mathbf{r} &= \mathbf{p} - \mathbf{q} && (5.17) \\ \mathbf{r}_{task} &= \mathbf{p} + P_{target} - \mathbf{q} - P_{tool} && (5.18) \\ &= \mathbf{r} + P_{target} - P_{tool} && (5.19) \end{aligned}$$

The observer control (presented in Section 5.3.2) should be defined in terms of $\mathbf{r}_{task}$ so that (5.1) operates on the difference

$$\begin{aligned}
\tilde{\mathbf{r}}_{task} &= \mathbf{r}_{task,des} - \hat{\mathbf{r}}_{task} && (5.20)\\
&= \mathbf{r}_{task,des} - \hat{\mathbf{r}} - P_{target} + P_{tool} && (5.21)\\
&= (\mathbf{r}_{task,des} + P_{tool}) - (\hat{\mathbf{r}} + P_{target}) && (5.22)
\end{aligned}$$

In the experimental system, this is handled by defining the trajectory in terms of $\mathbf{r}_{traj,des} = \mathbf{r}_{task,des} + P_{tool}$, which is convenient because both terms are expressed in observer body coordinates. In the control, P_{target} is first rotated from inertial to body coordinates based on the current orientation and then added to the relative position estimate before passing it to the controller.

5.5.1 Relative Position Truth Measurement

The experimental system does not provide a direct truth measurement for the relative position between the observer and the object, but the observer position $\mathbf{q}$, computed from the encoders on the manipulator joints, can be used to calibrate and derive the relative position. This section explains the truth measurement and discusses its accuracy.

When the gripper is in the correct location to pick up the cup, $\mathbf{r}_{task} = 0$. If this occurs at $\mathbf{q} = \mathbf{q}^0$, then

$$\begin{aligned}
\mathbf{r}_{task} = 0 &= \mathbf{p} + P^0_{target} - \mathbf{q}^0 - P^0_{tool} && (5.23)\\
\mathbf{p} &= \mathbf{q}^0 + P^0_{tool} - P^0_{target} && (5.24)\\
\mathbf{r} &= \mathbf{p} - \mathbf{q} && (5.25)\\
&= \mathbf{q}^0 - \mathbf{q} + P^0_{tool} - P^0_{target} && (5.26)
\end{aligned}$$

Therefore, by calibrating $\mathbf{q}^0$ at a grasp position of the observer, the relative position at other positions $\mathbf{q}$ can be computed. Although P^0_{tool} and P^0_{target} are constants, the superscript has been added to indicate that frame transformations have to be computed using the orientation of the observer when $\mathbf{q}^0$ was measured.

This is a derived *truth* measurement and is itself subject to systematic errors. These are introduced by errors in the calibration of the offsets and in the forward kinematics, which is the mapping from robot joint angles to endpoint position and orientation. These errors are small when $\mathbf{q}^0 - \mathbf{q}$ is small, or more precisely, when the actual joint angle differences between the current and grasp positions are small. As this difference increases, the systematic errors also grow.

Table 5.2: Estimator Initial Conditions

$\hat{\mathbf{y}}_0$

Entry	Value	Units
s_x	-0.006	
s_y	-0.013	
ζ	1.79	$1/m$
$\mathbf{v}$	0	m/s
$\mathbf{a}$	0	m/s^2
$\mathbf{b}_a$	$\begin{bmatrix} 0 \\ -0.23 \\ -0.12 \end{bmatrix}$	m/s^2
$\mathbf{Z}$	$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$	
ψ	1.57	rad
$\mathbf{b}_\omega$	$\begin{bmatrix} 0.004 \\ -0.003 \\ 0.010 \end{bmatrix}$	rad/s

Σ_0

Entry	Value	Units
s_x	0.2	
s_y	0.2	
ζ	0.53	$1/m$
$\mathbf{v}$	0.1	m/s
$\mathbf{a}$	0.012	m/s^2
$\mathbf{b}_a$	$\begin{bmatrix} 0.04 \\ 0.04 \\ 0.04 \end{bmatrix}$	m/s^2
$\mathbf{Z}$	$\begin{bmatrix} 0.15 \\ 0.02 \\ 0.15 \end{bmatrix}$	
ψ	0.01	rad
$\mathbf{b}_\omega$	$\begin{bmatrix} 0.003 \\ 0.003 \\ 0.003 \end{bmatrix}$	rad/s

5.6 Parameter Values

This section provides the parameter values required for implementing the estimator. Some of these parameters have already been defined in previous sections. The parameters can be grouped into initial conditions, the process noise covariance and the sensor noise covariance.

The initial estimate covariance P_0, the process noise covariance Q and the sensor noise covariance R are diagonal matrices.

$$P_0 = E\left(\mathbf{y}_0\mathbf{y}_0^T\right) = (diag(\Sigma_0))^2 \tag{5.27}$$

$$Q = E\left(\mathbf{n}_p\mathbf{n}_p^T\right) = (diag(\Sigma_p))^2 \tag{5.28}$$

$$R = E\left(\mathbf{n}_z\mathbf{n}_z^T\right) = (diag(\Sigma_z))^2 \tag{5.29}$$

where the Σ-vectors specify the standard deviations along the diagonals.

The initial conditions $\hat{\mathbf{y}}_0$ and Σ_0 are defined in Table 5.2. The entries corresponding to s_x, s_y, ζ, $\mathbf{v}$, and $\mathbf{Z}$ are representative of the typical initial conditions at the beginning of the manipulation task. The entries for $\mathbf{a}$ are derived from Section 5.3.4. The entries from $\mathbf{b}_a$ and $\mathbf{b}_\omega$ are from Section 5.1.3.

The system does not have an absolute measurement of heading. Therefore, any initial error in the heading measurement cannot be eliminated. Consequently, the system is

Table 5.3: Process and Measurement Noise Parameters

Σ_p

Entry	Value	Units
$\mathbf{n}_{d1}$	0.0022	m/s^3
$\mathbf{n}_{ba}$	0.00001	m/s^3
$\mathbf{n}_{\omega}$	0.001	rad/s
n_{α}	0.000001	$1/s$
$\mathbf{n}_{b\omega}$	0.00001	rad/s^2

Σ_z

Entry	Value	Units
$\mathbf{z}_s$	0.01	
$\mathbf{z}_a$	0.1	m/s^2

initialized with the known heading of $\psi = 90°$ and a low uncertainty. An analysis of the results then shows how well the estimator can maintain the estimate of heading based only on integrating the rate gyro measurements.

The Σ-vectors corresponding to the process noise $\mathbf{n}_p$ and the measurement noise $\mathbf{n}_z$ are defined in Table 5.3. The standard deviations for $\mathbf{n}_{d1}$, $\mathbf{n}_{\omega}$, $\mathbf{z}_s$ and $\mathbf{z}_a$ are defined earlier in this chapter.

The standard deviations for $\mathbf{n}_{ba}$, $\mathbf{n}_{b\omega}$ and n_{α} are set to be very low because the duration of the experiments is short—$20\,s$ is the length of the manipulation task—so any variability in the bias and scale factors is negligible. To use this estimator for longer-duration experiments for which changes in the bias and scale factors are significant, appropriate parameters for these process noise components should be chosen.

The values of Σ_p are used by the simulation system. For use by the estimator, $\sigma(\mathbf{n}_{\omega})$ is increased by $0.0015\,rad/s$ and $\sigma(n_{\alpha})$ is increased by $0.001\ \ 1/s$. Fictitious process noise is added to cover the effects of approximations in the estimator algorithm. The fictitious process noise has minimal effect during normal conditions. However, it is important when entries in Σ_p are scaled down for some of the experiments in Chapter 6 that simulate low-noise conditions.

5.7 Simulation System

A simulation system has been developed to improve the efficiency of the design process and to provide a means to compute Monte Carlo results based on a large number of runs. Whenever possible, the simulation system is based on the same parameter values as the hardware system. However, the complexity of the simulation system is significantly reduced. The transformations between observer motions and robot joint motions are not necessary. The state dynamics and simulated measurements can all be implemented at $10\,Hz$ using the same equations that were developed in Section 4.1. Finally, only the

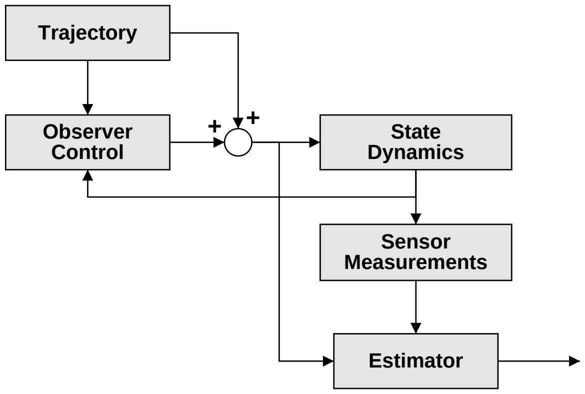

Figure 5.8: Block Diagram for the Simulation System

"control-from-truth" configuration needs to be implemented. The simulation system was developed in Matlab using the structure shown in Figure 5.8.

5.8 Trajectory

The trajectory specifies the desired relative positions, velocities, orientation and angular rates for performing the demonstration task, as well as the nominal force and torque commands required to achieve this desired motion. The trajectory design has to satisfy two competing objectives. First, it has to generate sufficient observer motion for the relative position estimates to converge. This tends to require motion transverse to the line-of-sight. Second, it has to result in observer motion towards the object—from the initial observer position to a final position that permits a grasp of the object.

The design of this trajectory is based on an *ad hoc* optimization of the estimate uncertainty subject to reaching the desired final grasp position. Candidate trajectories are represented by a superposition of motion primitives, each defined by a number of parameters like amplitude, start-time and duration. Using properly designed primitives and parameter ranges ensures that each candidate trajectory starts at the nominal initial relative position, stops at the relative position and orientation required for successful grasping, and provides sufficient degrees-of-freedom to ensure estimator convergence. Parameters can then be optimized to balance the estimator performance with the amount of motion and the duration of the trajectory. The *ad hoc* design method is based on proposing various trajectories based on intuition and choosing the best trajectory based on the performance of the estimator.

A trajectory is proposed by defining the desired relative position and the desired orientation as a function of time. From these, the desired velocity and the desired angular rates can be computed. This intermediate trajectory is then propagated through the closed-loop

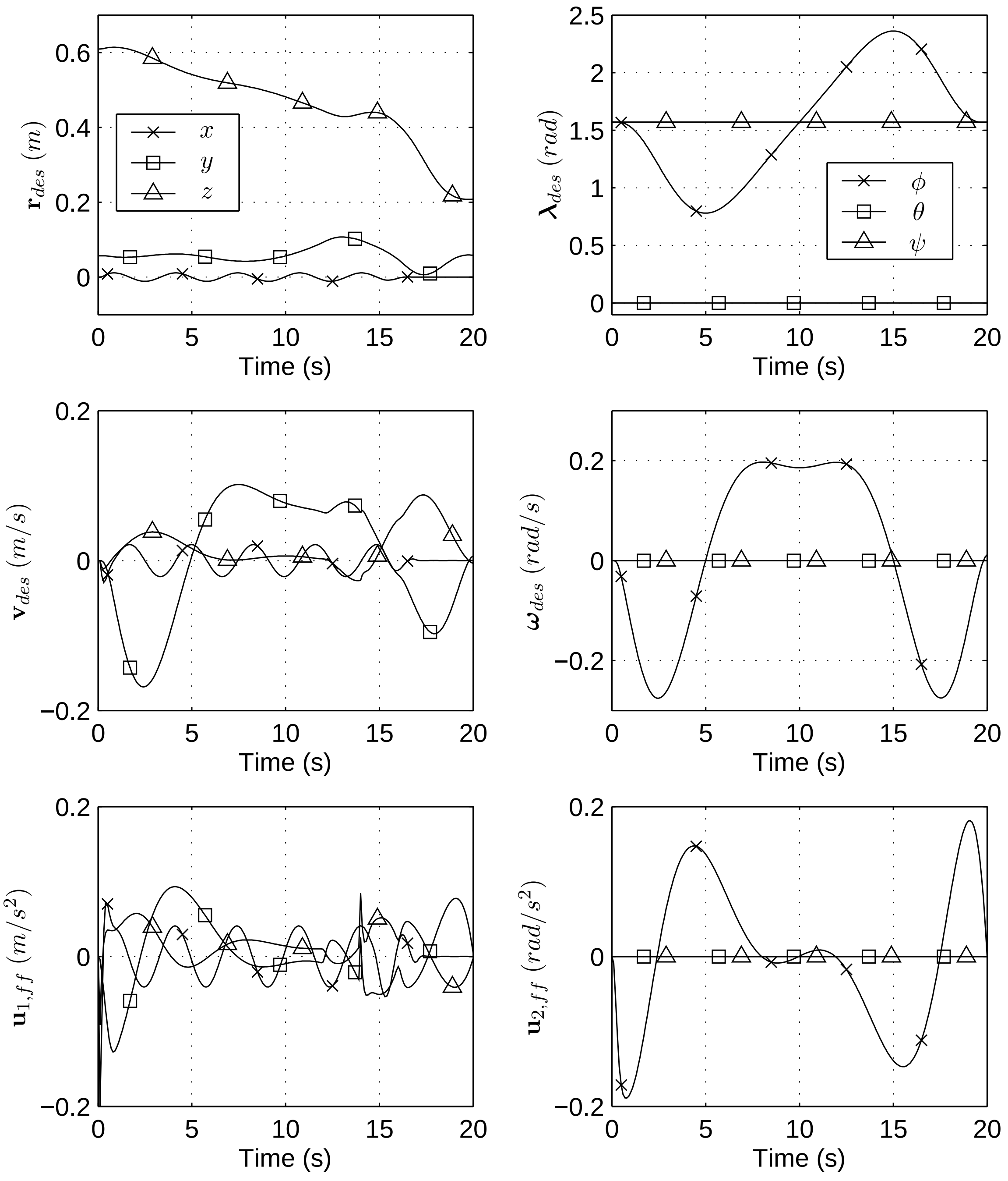

Figure 5.9: Trajectory

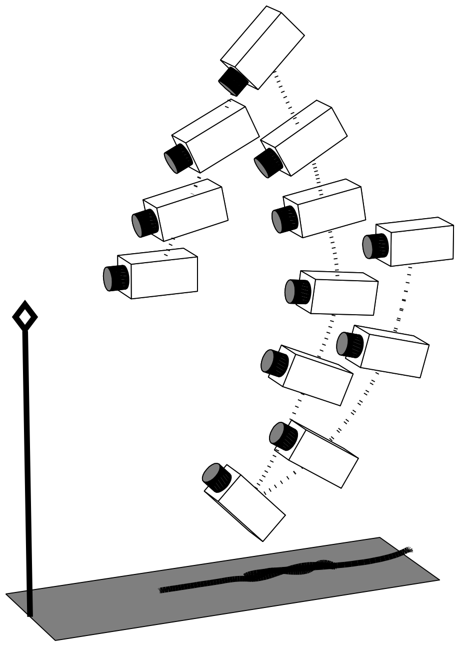

Figure 5.10: Desired Observer Motion Corresponding to the Trajectory in Figure 5.9.

control of the simulation system, which is subject to the modeled bandwidth limitations and actuator saturation of the experimental system. This generates an achievable trajectory of motion commands including the forces and torques that produce that motion.

Figure 5.9 presents the trajectory for the demonstration task. This trajectory is used for all of the simulation and experimental results in Chapter 6. The first row shows the desired relative position $\mathbf{r}_{des}$ and the desired orientation $\boldsymbol{\lambda}_{des}$ in the *roll-pitch-yaw* representation. The second row shows the associated desired velocity $\mathbf{v}_{des}$ and the desired angular rates $\boldsymbol{\omega}_{des}$. The third row shows the feedforward commands $\mathbf{u}_{1,des}$ and $\mathbf{u}_{2,des}$. Figure 5.10 presents this motion on a 3D plot.

The two dominant motions in frames 1 to 5 of Figure 5.3 are the motion towards the object, represented by $r_{z,des}$ (the z-axis in the body coordinates is parallel to the camera axis), and the rotational motion in the vertical plane represented by ϕ_{des}. As discussed in Section 5.5, $\mathbf{r}_{des}$ includes the P_{tool} offset and therefore does not converge to zero at the end of the trajectory.

5.9 Summary

This chapter has presented the experimental system and the demonstration task used to validate the sensing strategy, its feasibility, and it applicability to a closed-loop manipulation task. The description focused on the actual hardware, the object pick-up task, the control system, calibration and parameters, the development of a simulation system, and the trajectory design. Simulation and experimental results using this system are presented in the next chapter.

Chapter 6

Results

This chapter presents results from computer simulations and hardware experiments that demonstrate that the sensing strategy—fusing monocular vision measurements of a single feature with inertial rate sensor measurements to determine relative position—is feasible and that it is a useful capability for the closed-loop control of an autonomous robot. Computer simulations and hardware experiments provide complementary advantages. Computer simulations provide greater insight into the performance of the estimator because modeling errors can be avoided, parameters can be varied more easily, more experiments can be performed and all truth measurements are immediately available. Hardware experiments provide confidence that all critical issues have been addressed, that the problem can be solved in real time, and that the capability is suitable for an autonomous manipulation task.

All of the experiments are performed in the context of the object pick-up task introduced in Section 5.2. Chapter 2 describes the estimation problem that has to be implemented to perform the pick-up task. The design of an estimator to solve this problem appears in Chapter 4. Chapter 5 describes the experimental hardware and the associated simulation environment used to generate the results presented in this chapter.

This chapter is divided into three main sections. The first section presents simulation results for the baseline problem. These results are generated with the "control-from-truth" configuration in which the observer control is independent of the estimator results. Estimator results demonstrate centimeter-level accuracy for the relative position sensing capability. Analyzing these results confirms that the estimator design can exploit the nonlinearities inherent in this sensing strategy.

To develop additional insight into the performance of a potential underwater vehicle demonstration, the second section presents simulations for environments with larger and smaller disturbances and higher-quality inertial rate sensors. This analysis shows that the effectiveness of the sensing strategy depends strongly on the disturbance environment and only weakly on the inertial rate sensor quality. Therefore, employing more expensive sensors to improve performance does not help, which is a fundamental limitation of this sensing strategy. This limitation can be addressed by fusing additional sensor measurements, like tracking additional features or incorporating additional sensors. These extensions are discussed in Chapter 7.

Results from the hardware experiment are presented in the third section. This includes results for the baseline experiment, which validate the simulation results and show that the implementation of the sensing strategy can tolerate the effects of unmodeled errors that are present in the hardware experiment. A second experiment demonstrates the effectiveness of the sensing strategy for an autonomous manipulation task in which the estimator results are used for the closed-loop control of the observer. This hardware experiment represents the first real-time implementation of this sensing strategy to perform a useful manipulation task with an autonomous robot.

6.1 Simulation Results

This section presents simulation results for the baseline experiment. The baseline experiment focuses on the estimator performance by using the "control-from-truth" configuration (described in Section 5.3.3). In this configuration, the robot control depends on the known relative position and velocity. This configuration avoids any estimator performance issues that result from closing the control loop around the estimator. The parameters that define this estimation problem are summarized in Section 5.6.

These simulation results show that the estimator works well in the presence of modeled uncertainties and provide insight on the observability issues of the sensing strategy. Hardware experiments (presented later in Section 6.3.1) validate these simulation results and show that the estimator also performs well with real sensor measurements.

This section is split into two parts. The first part shows the results for one simulation run. This provides a context for later results by plotting the true states, the associated estimates and their predicted uncertainty. The second part presents a Monte Carlo analysis, which provides averages of the estimate error—the difference between the true state and the estimate—over many runs.

6.1.1 One Simulation Run

Figures 6.1 to 6.6 present the results from one simulation run. The simulation environment is described in Section 5.7. For each run, the actual system state is initialized at $\mathbf{y}_0 = \mathbf{y}_{init} + \Delta\mathbf{y}_0$, where $\Delta\mathbf{y}_0$ is sampled according to P_0. The estimator is initialized at $\bar{\mathbf{y}}_0 = \mathbf{y}_{init}$. The relative position encoded in $\mathbf{y}_{init}$ corresponds to the desired relative position at the start of the trajectory. The control $\mathbf{u}_k$ is computed from the known state $\mathbf{y}_k$ and causes the state to follow the relative position and velocity described in the trajectory.

Figure 6.1 shows the evolution in time for s_x, s_y, ζ and the relative velocity; Figure 6.2 shows the disturbance and $\mathbf{Z}$-vector components; Figure 6.3 shows the inertial rate sensor biases; and Figure 6.4 shows the observer heading. Finally, Figure 6.5 shows the relative position, which is computed from s_x, s_y and ζ. All of these plots show the true state with a solid line, the estimate with a dashed line, and the uncertainty envelope in gray. The uncertainty envelopes $\check{\mathbf{y}}$ and $\check{\mathbf{r}}$ are defined to be one estimated standard deviation above and below the true state.

$$\check{\mathbf{y}} = \begin{cases} \mathbf{y} + \sqrt{diag\left(\hat{P}\right)} \\ \mathbf{y} - \sqrt{diag\left(\hat{P}\right)} \end{cases} \tag{6.1}$$

$$\check{\mathbf{r}} = \begin{cases} \mathbf{r} + \sqrt{diag\left(\hat{P}_r\right)} \\ \mathbf{r} - \sqrt{diag\left(\hat{P}_r\right)} \end{cases} \tag{6.2}$$

The uncertainty of s_x and s_y is low because these components are directly observable in the camera measurements. In contrast, the uncertainty of ζ is larger because it cannot be measured directly. The inverse object range, ζ, is determined by an implicit triangulation which depends critically on the uncertainty of several other states.

The $\mathbf{Z}$-vector, scaled by the acceleration due to gravity, is the dominant component of the accelerometer measurement, which leads to a relatively good estimate. The disturbance force $\mathbf{a}$ and the accelerometer bias $\mathbf{b}_a$, which are the smaller contributions to the accelerometer measurement, are more difficult to estimate accurately.

Note that the uncertainty in $\hat{b}_{a,x}$ shows almost no improvement during this experiment. This is a consequence of the trajectory design and does not affect the accuracy of the range estimate. Because Z_x and $b_{a,x}$ are mainly constant throughout the trajectory, the estimator cannot assign contributions in the corresponding accelerometer measurement to one or the other state. Introducing more variation in Z_x by modifying the trajectory leads to more pronounced convergence in $\hat{b}_{a,x}$, but this is not necessary for good overall estimator performance.

The heading ψ is not actually observable in the sensor measurements—the estimator can only integrate angular velocity around the $\mathbf{Z}$-vector to compute changes in heading. Therefore, the estimator is initialized with the true heading. This implies that the initial uncertainty of the heading is zero. As time progresses, that uncertainty grows to accommodate the noise and bias errors of the rate gyro measurements. Although absolute heading is not observable, it helps to include it in the estimator to enable heading control. Adding

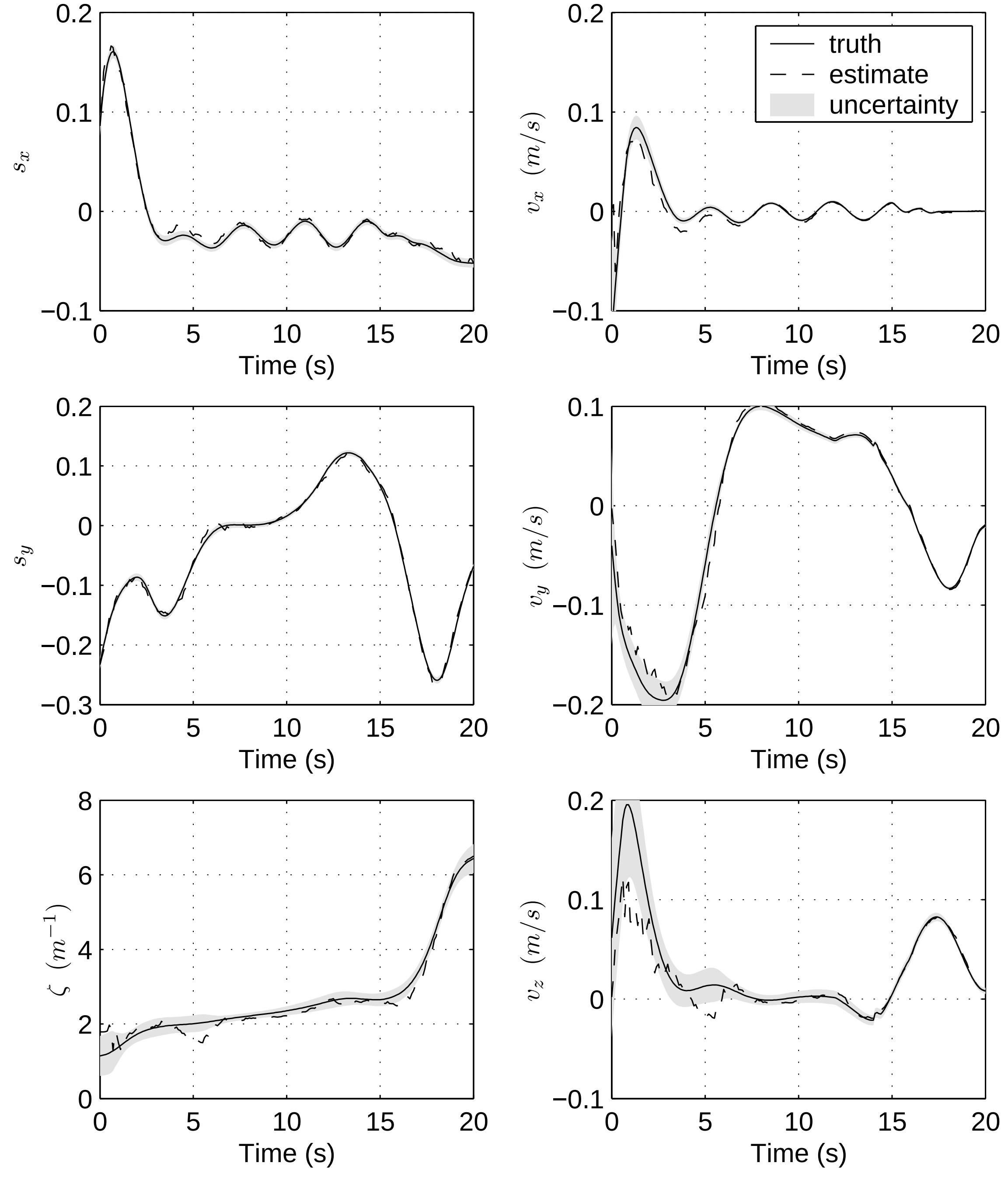

Figure 6.1: One Simulation Run: s_x, s_y, ζ and Velocity

s_x and s_y are the image plane feature positions. ζ is the inverse range. The range is the object position along the optical axis of the camera. The velocity $\mathbf{v}$ *is the relative velocity between object and observer expressed in the observer body frame.*

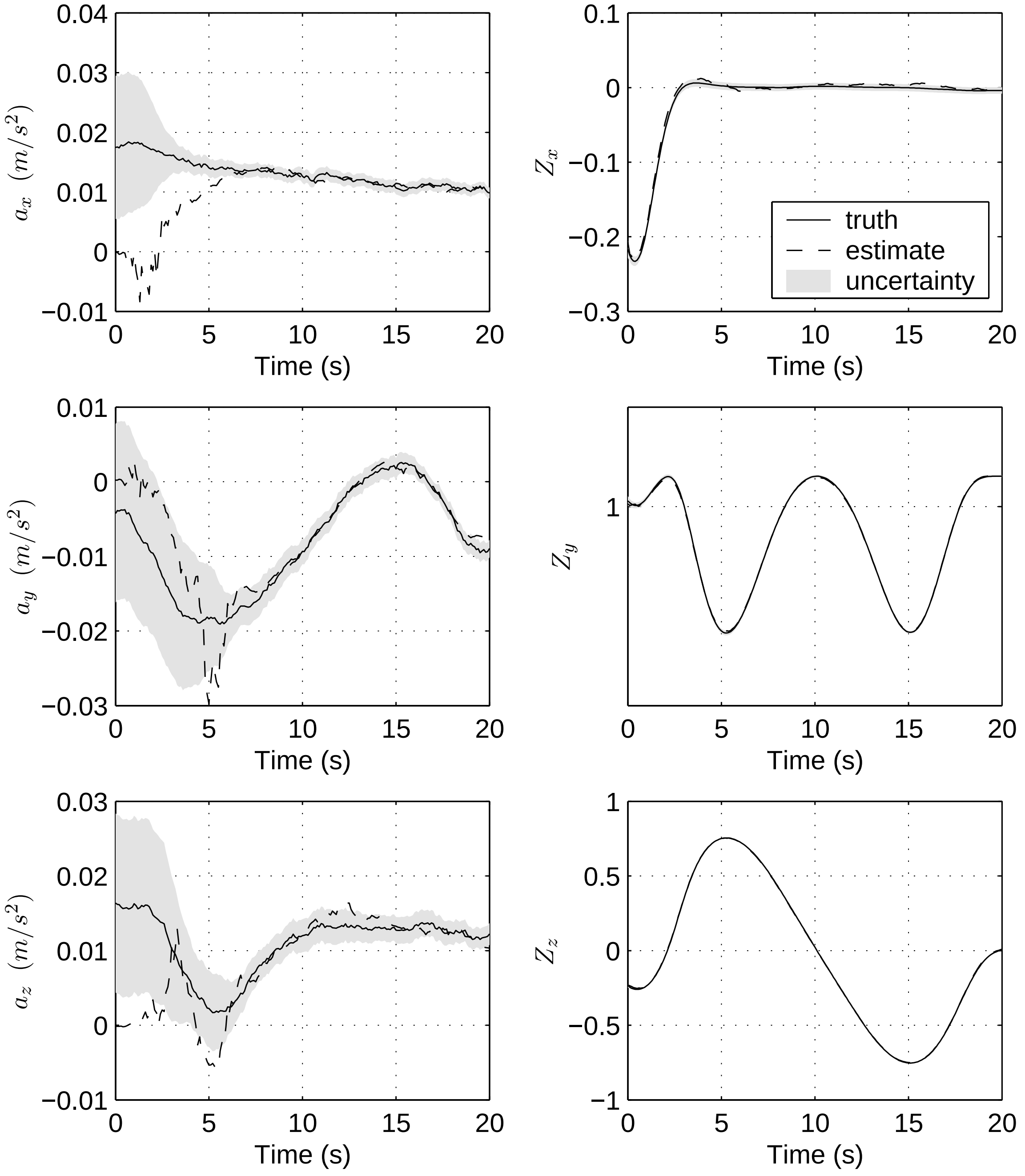

Figure 6.2: One Simulation Run: Disturbances and $\mathbf{Z}$-vector

$\mathbf{a}$ *is the force disturbance, in units of acceleration and expressed in the observer body coordinates.* $\mathbf{Z}$ *is the scaled gravity direction in the observer body coordinates.*

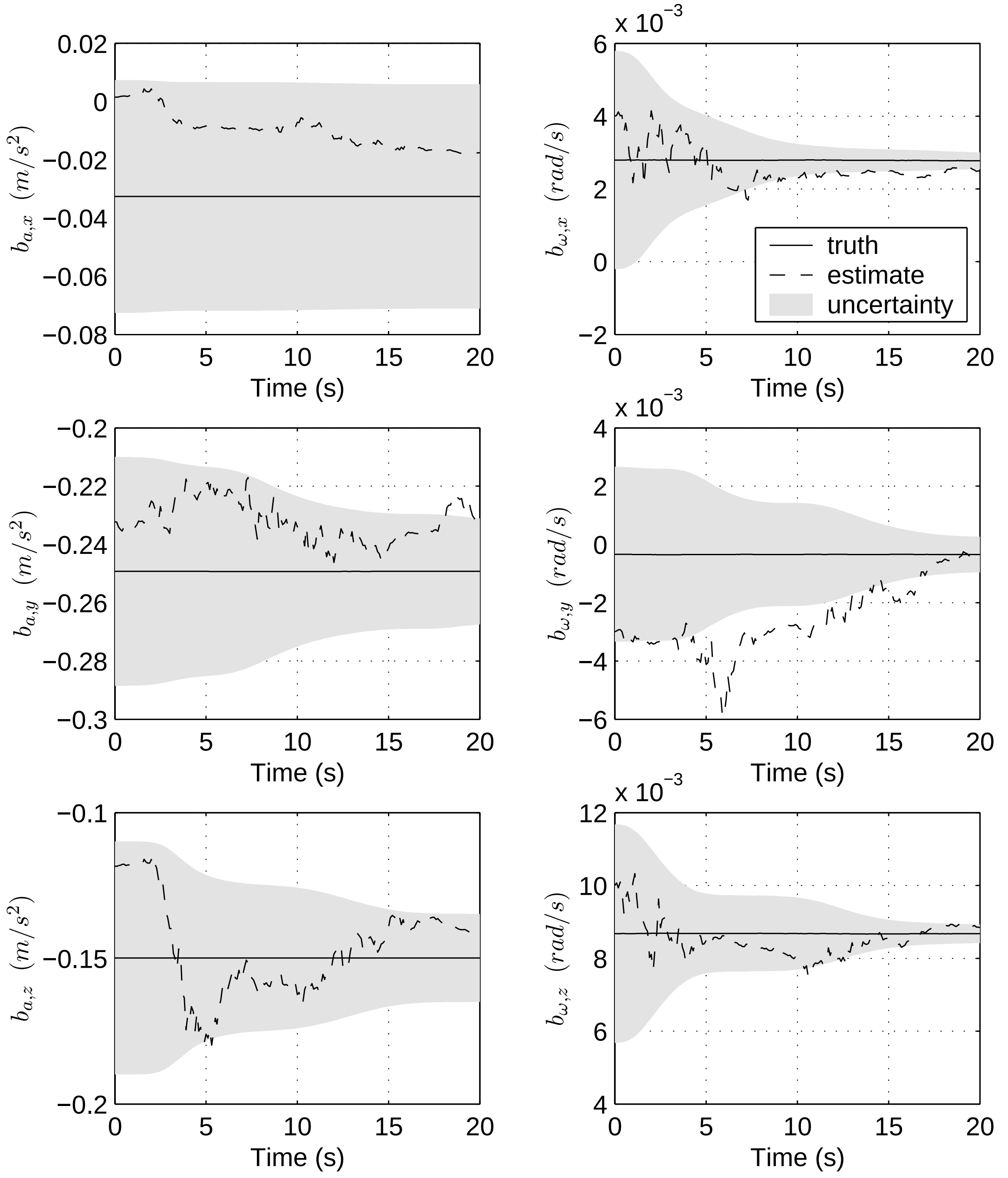

Figure 6.3: One Simulation Run: Inertial Rate Sensor Biases

$\mathbf{b}_a$ *are the biases on the accelerometers.* $\mathbf{b}_\omega$ *are the biases on the rate gyros.*

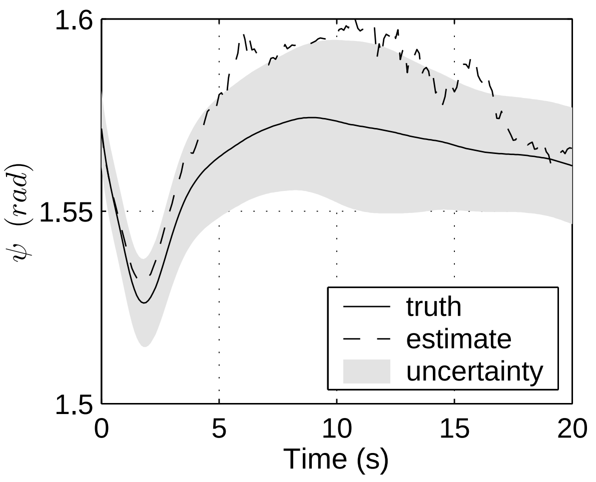

Figure 6.4: One Simulation Run: Observer Heading

The observer heading ψ is the rotation about the gravity vector or the $\mathbf{Z}$-vector.

a compass to the sensor strategy would result in a significant improvement of the heading estimate.

The relative position in Figure 6.5 is computed from s_x, s_y and ζ using the nonlinear transformation $\mathbf{r} = g(\mathbf{y})$ (see Section 4.5). In the estimator, this is implemented with another UT algorithm. The object range r_z depends directly on ζ and is the least certain estimate. As shown in these plots, the other two components of $\mathbf{r}$ are easier to estimate.

Figure 6.6 shows the innovations $\tilde{\mathbf{z}} = \mathbf{z} - \bar{\mathbf{z}}$ for the vision and accelerometer measurements. These plots are a powerful analysis tool because the innovations can be computed without an independent truth measurement and most estimation problems are detectable in the innovations. The gray region indicates the uncertainty envelope defined as

$$\check{\mathbf{z}} = \begin{cases} \sqrt{diag\left(\hat{R}_e\right)} \\ -\sqrt{diag\left(\hat{R}_e\right)} \end{cases} \tag{6.3}$$

where $\hat{R}_e$ is the covariance of the innovations (see Section 4.4).

The results for this single simulation run provide a context for the other results in this chapter. Clearly, the estimator converged and the innovations are reasonable. However, this is only a single run which is not representative of a stochastic nonlinear problem. The next section presents a Monte Carlo analysis, which provides confidence that these results are valid over a large number of simulations. Section 6.3.1 presents results from a hardware experiment, which is subject to a variety of unmodeled errors.

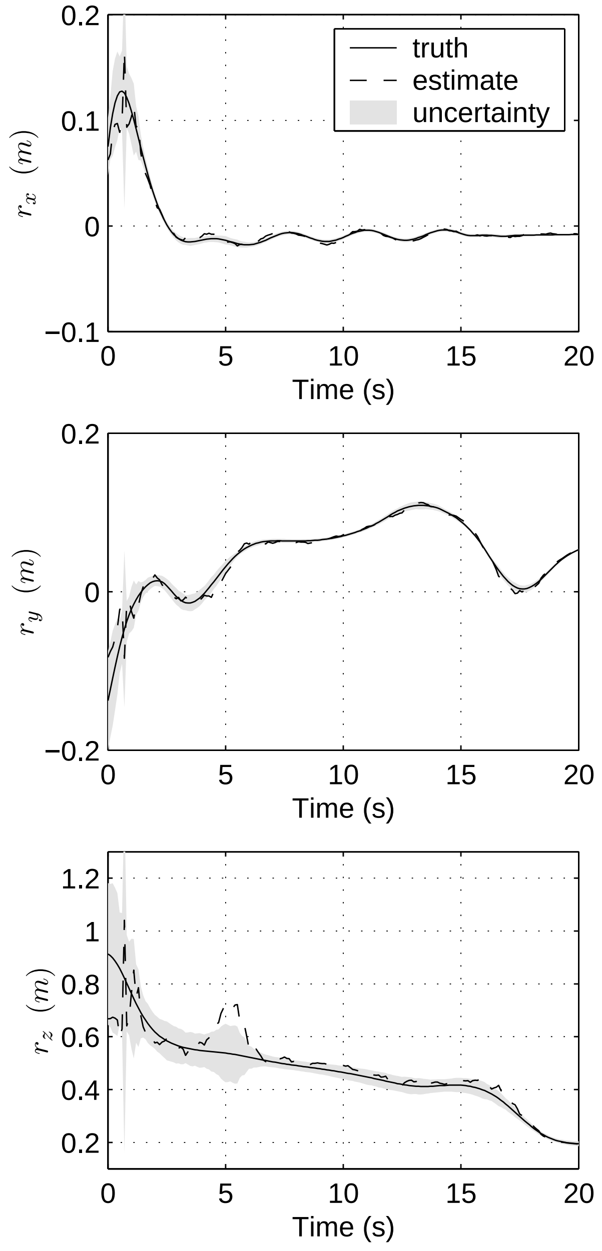

Figure 6.5: One Simulation Run: Relative Position

The relative position between the object and the observer is expressed in observer body coordinates. It is computed from s_x, s_y and ζ.

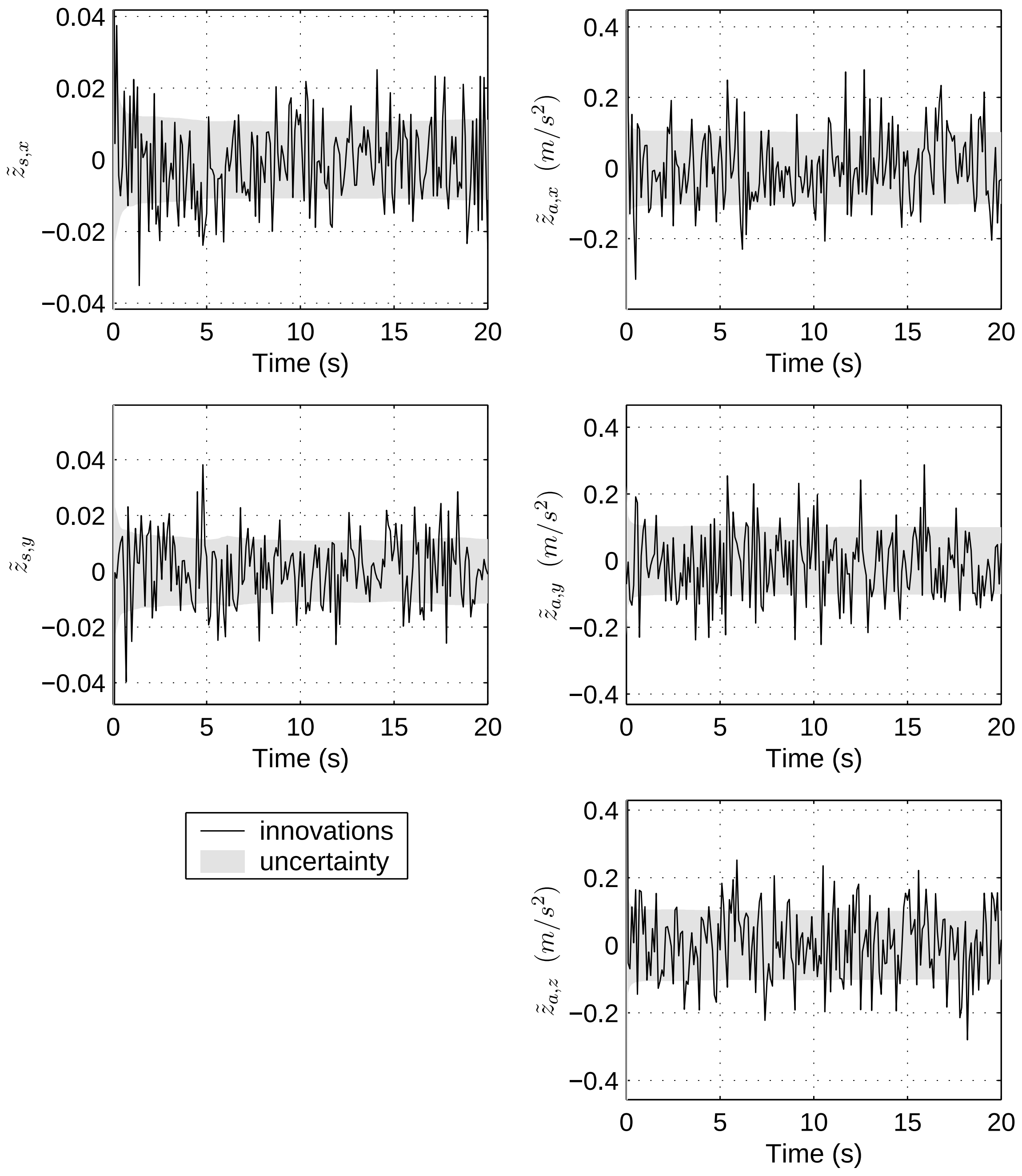

Figure 6.6: One Simulation Run: Innovations

$\mathbf{z}_s$ is the image plane measurement of the feature. $\mathbf{z}_a$ is the accelerometer measurement.

6.1.2 Monte Carlo Simulations

Monte Carlo simulations capture the averaged behavior of the estimator and summarize the performance of the estimator over many different runs. This type of analysis is especially useful for nonlinear systems for which the estimator performance cannot be predicted accurately from analysis. The Monte Carlo results in this chapter are all generated from $N = 1000$ runs like the one described in the previous section. If N runs are averaged to produce the Monte Carlo results, the mean estimate error $\mu(\tilde{\mathbf{y}})$, the actual standard deviation of the estimate error $\sigma(\tilde{\mathbf{y}})$, and the predicted standard deviation $\hat{\sigma}(\tilde{\mathbf{y}})$ are given by

$$\mu(\tilde{\mathbf{y}}_k) = \frac{1}{N}\sum_{i=1}^{N}\tilde{\mathbf{y}}_k^i \tag{6.4}$$

$$\sigma(\tilde{\mathbf{y}}_k) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(\tilde{\mathbf{y}}_k^i - \mu(\tilde{\mathbf{y}}_k)\right)^2} \tag{6.5}$$

$$\hat{\sigma}(\tilde{\mathbf{y}}_k) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} diag\left(\hat{P}_k^i\right)} \tag{6.6}$$

$$\tilde{\mathbf{y}}_k^i = \mathbf{y}_k^i - \hat{\mathbf{y}}_k^i \tag{6.7}$$

where the superscript i denotes a given run. Equivalent relationships can be written for the relative position $\mathbf{r}$ and the innovations $\tilde{\mathbf{z}}$.

The Monte Carlo analysis contains several integrity checks which are applied to each run. These checks are evaluated only from information available to the estimator (i.e., not from truth data) and are used to remove runs that have generated non-sensical results. A variety of conditions can cause these anomalies in a nonlinear stochastic estimator. For example, ζ should be positive, but neither of the estimate updates explicitly enforces that. The integrity checks can trap for this situation. Ideally, all of these failures are trapped by the integrity checks and η, the percentage of successful runs, is large. In these results, η is reported as a metric for the estimator design.

Two different integrity checks are used. The first triggers if ζ falls outside of the range $[0.2, 100]$. This corresponds to object ranges above $5\,m$ and below $1\,cm$. The second triggers if the innovations for a measurement fall outside of the uncertainty bound for more than 10 consecutive measurements.

Figures 6.7 to 6.11 present Monte Carlo results for the baseline experiment. For these results, $\eta = 99.7\%$, that is, 3 out of 1000 runs were culled.

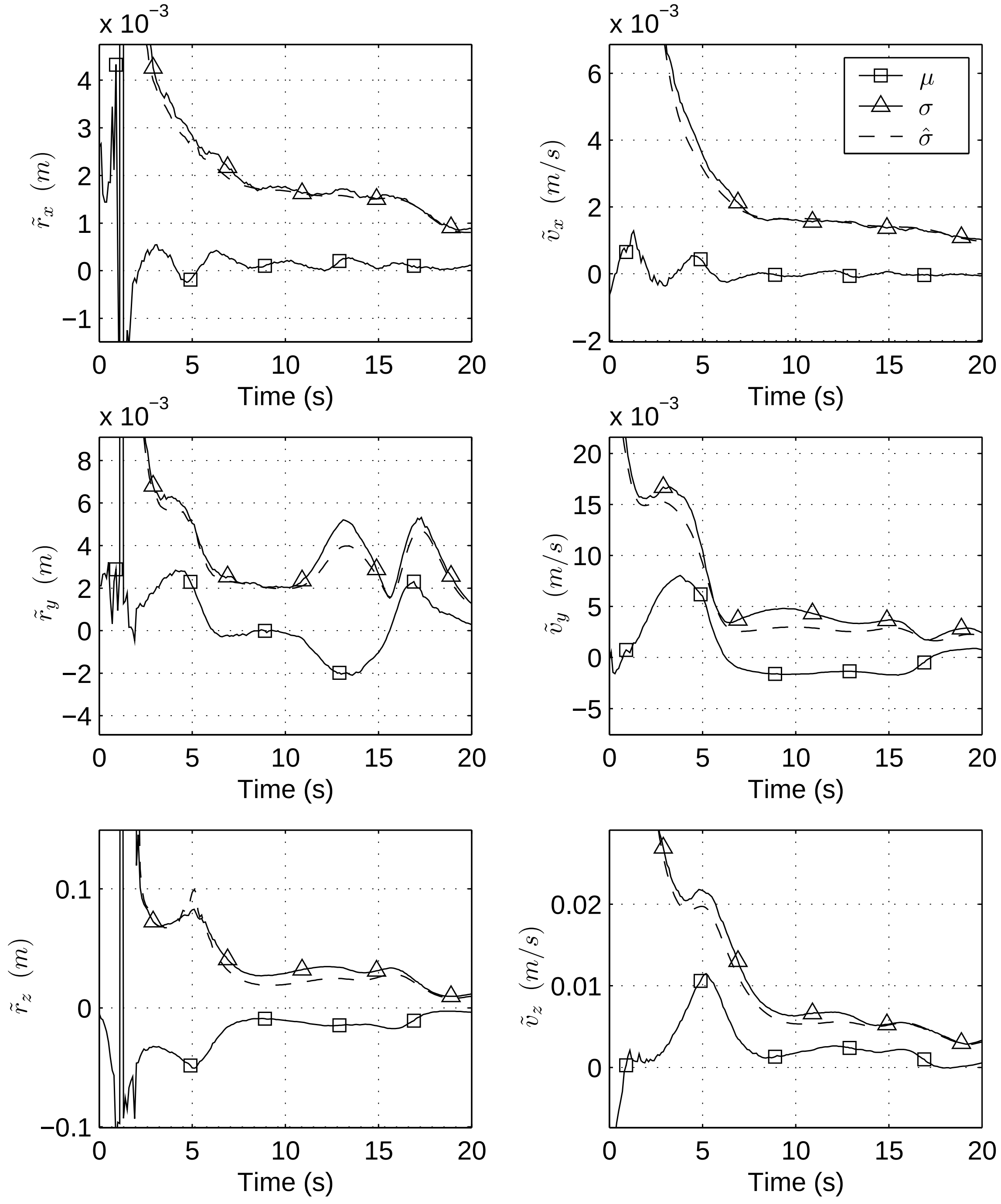

Figure 6.7: Baseline Monte Carlo Experiment: Relative Position and Velocity

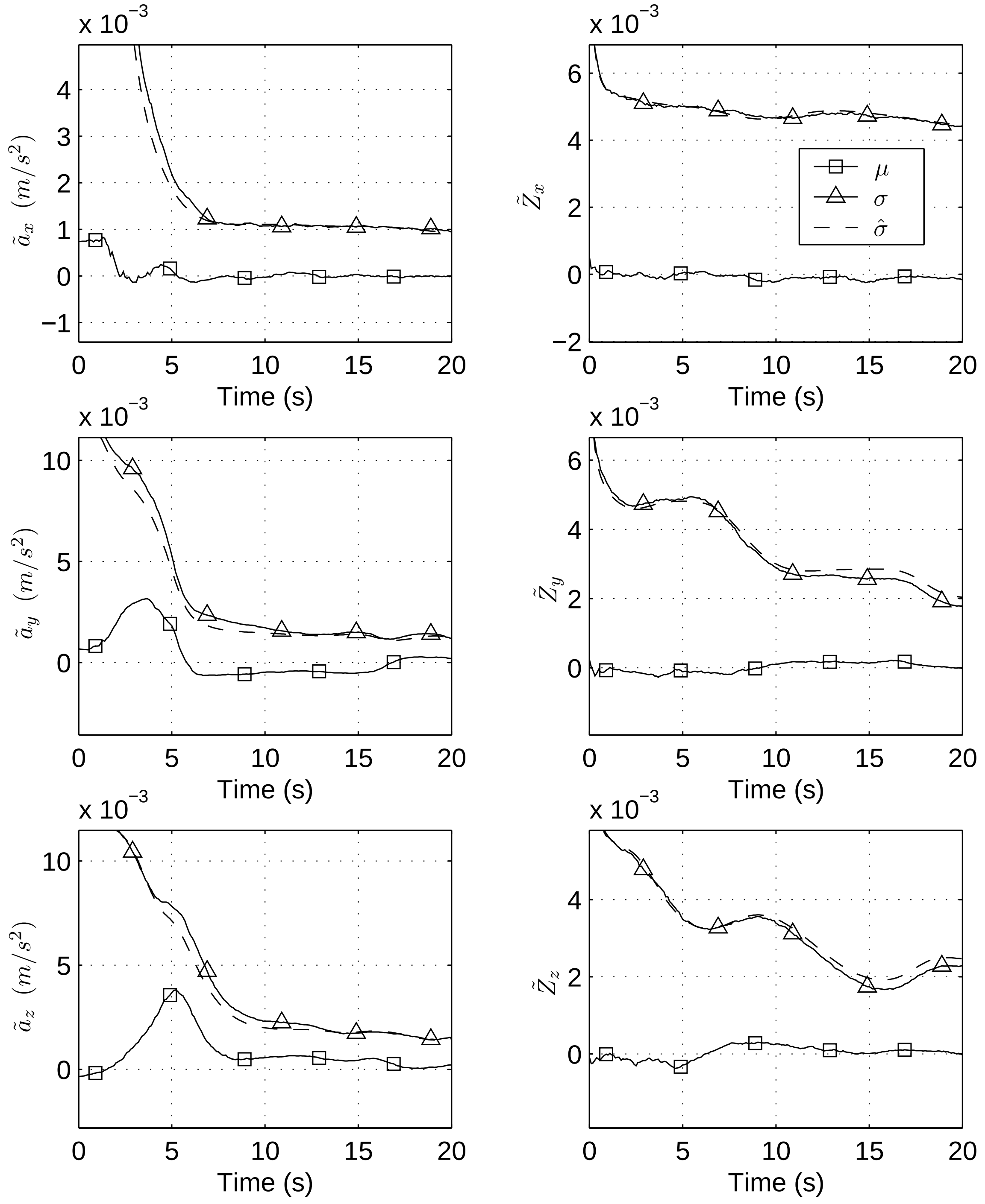

Figure 6.8: Baseline Monte Carlo Experiment: Disturbances and $\mathbf{Z}$-vector

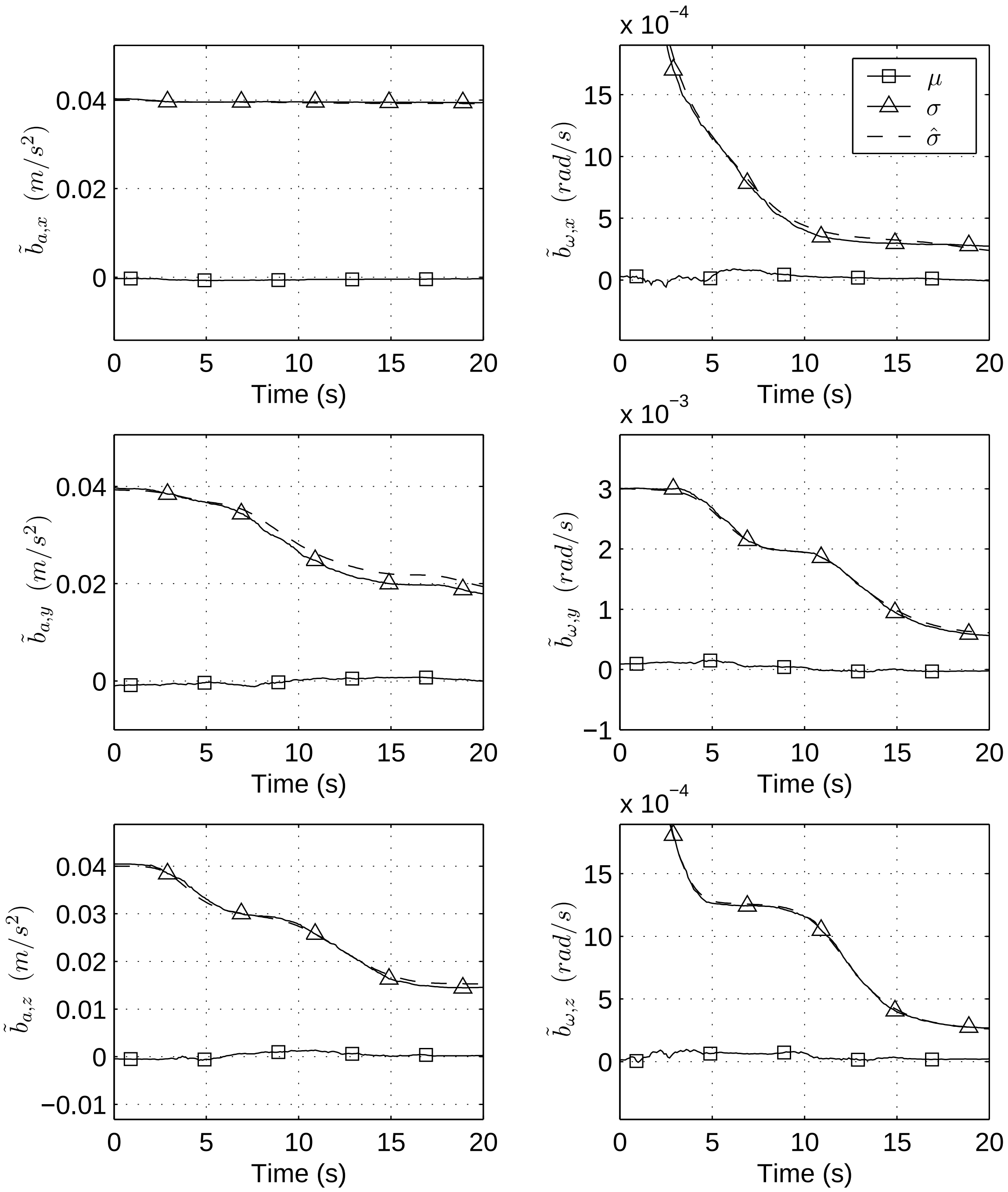

Figure 6.9: Baseline Monte Carlo Experiment: Inertial Rate Sensor Biases

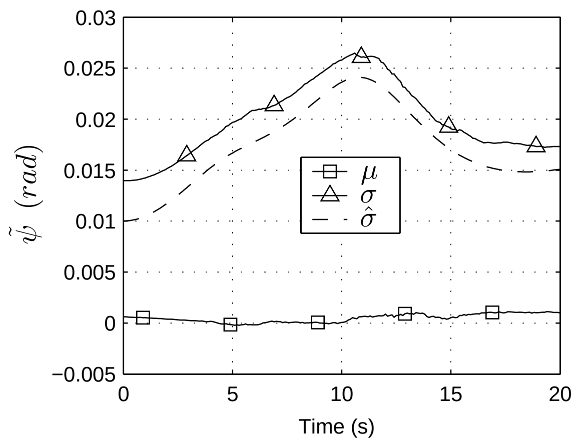

Figure 6.10: Baseline Monte Carlo Experiment: Observer Heading

These plots demonstrate some important characteristics of this estimator design. First, the mean estimate error (square symbols) is small compared to its standard deviation (triangle symbols), so it does not represent a dominant source of error. Second, the predicted standard deviation ($\hat{\sigma}$, dashed line) correctly predicts the actual standard deviation (σ). Third, states that are unobservable for this particular trajectory (e.g., $b_{a,x}$ in Figure 6.9) do not diverge. Fourth, the innovations have minimal structure. These observations indicate that problems which typically arise when applying optimal estimators to nonlinear systems with dynamic observability have been avoided with this estimator design.

Estimates are unbiased if the mean estimate error approaches zero. Although the simulations show that the estimate biases are small, the estimator is not completely unbiased. But this is not surprising given that the problem is nonlinear and the estimator design contains necessary approximations. Furthermore, all of the runs that have been averaged to produce these results have used the same desired relative position trajectory. Therefore, any systematic errors that arise from the approximations are reinforced by every run.

These observations about the merits of the estimator design are similar to those of Section 3.7, where the design was applied successfully to the much simpler three-state example. The results here demonstrate that the estimator design can successfully scale to much more complicated problems. Note that unlike the three-state example, this estimation problem includes process noise (real and fictitious) which can sometimes mask small nonlinear effects.

The purpose of the baseline experiment is to explore the performance of the estimator in the context of the application and system defined in Chapter 5. The Monte Carlo simulations show that the uncertainty in the object range, the most important output of the estimator, has a standard deviation of about $1\ cm$. These results assume a known model for

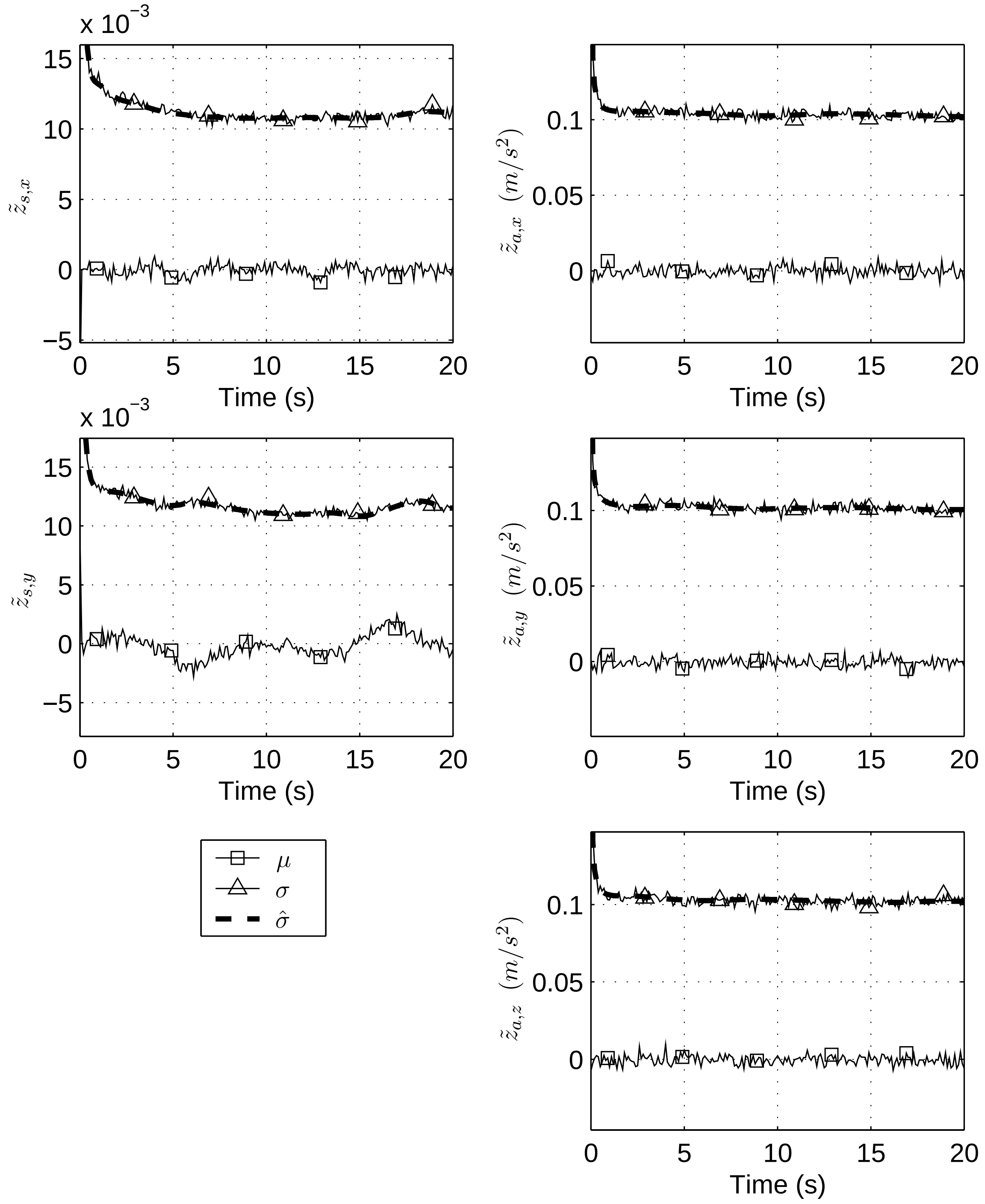

Figure 6.11: Baseline Monte Carlo Experiment: Innovations

disturbance forces and torques in the underwater environment and assume no modeling errors.

6.2 Parameter Study

The size of the external disturbances from the environment is a key factor that determines the estimator performance. Larger disturbance levels lead to more uncertain results because the observer motion is more difficult to estimate. The results presented in this section demonstrate that smaller disturbances lead to corresponding improvements in the estimator performance. They also show that when disturbances grow by about an order of magnitude without modifying the observer trajectory, the estimator performance quickly degrades.

An obvious question that arises is whether better inertial rate sensors, with lower measurement noise and more stable sensor biases, lead to a corresponding increase in estimator performance? The simulations in this section predict no improvement in the relative position estimate with increased inertial rate sensor accuracy. This effect can be explained as a consequence of the sensing strategy and leads to specific recommendations for future work.

Unfortunately, the factors that have the greatest influence over the performance of the sensing strategy are the most difficult to affect. For instance, the size of external disturbances is given by the environment. Model errors, which have a similar effect as disturbances, are difficult to eliminate. Conversely, the quality of inertial rate senors, which can be improved by orders of magnitude at a dollar cost, has little effect. Therefore, any improvements in the performance of the sensing strategy will likely come from increased observability due to additional sensor measurements.

6.2.1 Parameter Variations

This analysis considers various parameter variations that are controlled by two scale factors. The disturbance factor, or β_D, controls the size of the external disturbances. The noise factor, or β_N, controls the quality of the inertial rate sensors. These two scale factors define a two-dimensional parameter study. While there are many ways to construct an analysis to compare parameter variations, this provides a simple approach to highlight two core trends. The parameters listed below are modified by these scale factors. The underbar indicates that the parameters are a function of β_D and β_N. These parameters replace the corresponding baseline parameters defined in Section 5.6.

The disturbance factor modifies the size of the noise source that drives the models for the external disturbances and the initial uncertainty of the disturbance force.

$$\underline{\sigma}(n_{d1}) = \beta_D \sigma(n_{d1}) \tag{6.8}$$

$$\underline{\sigma}(n_{d2}) = \beta_D \sigma(n_{d2}) \tag{6.9}$$

$$\underline{\sigma}(a_0) = \beta_D \sigma(a_0) \tag{6.10}$$

The noise factor affects all of the parameters that define the inertial rate sensors, including the measurement noise

$$\underline{\sigma}(n_a) = \beta_N \sigma(n_a) \tag{6.11}$$

$$\underline{\sigma}(n_\omega) = \beta_N \sigma(n_\omega)\,, \tag{6.12}$$

the process noise on the sensor bias and scale factor states

$$\underline{\sigma}(n_{ba}) = \beta_N \sigma(n_{ba}) \tag{6.13}$$

$$\underline{\sigma}(n_{b\omega}) = \beta_N \sigma(n_{b\omega}) \tag{6.14}$$

$$\underline{\sigma}(n_\alpha) = \beta_N \sigma(n_\alpha)\,, \tag{6.15}$$

and the initial uncertainty in the bias states

$$\underline{\sigma}(b_{a,0}) = \beta_N \sigma(b_{a,0}) \tag{6.16}$$

$$\underline{\sigma}(b_{\omega,0}) = \beta_N \sigma(b_{\omega,0})\,. \tag{6.17}$$

The baseline experiment at the beginning of the chapter corresponds to $\beta_D = 1$ and $\beta_N = 1$. This section introduces results for combinations of $\beta_D = \begin{bmatrix} 0.1 & 0.32 & 1 & 3.2 & 10 & 32 & 100 \end{bmatrix}$ and $\beta_N = \begin{bmatrix} 0.001 & 0.01 & 0.1 & 1 \end{bmatrix}$.

6.2.2 Results

Figures 6.12 to 6.16 present results from the Monte Carlo simulations at each combination of (β_D, β_N). Each simulation represents 1000 runs and is generated with the same algorithm except for the parameter modifications listed above. The parameter modifications affect both the data generated for the simulation and the estimator gains used to process the data.

Figure 6.12 shows a comparison between the actual and predicted standard deviation of the estimate error for the object range (r_z). This plot is based on data with $\beta_N = 1$ and various values of β_D. The actual standard deviations, $\sigma(\tilde{r}_z)$, are computed at the end of the trajectory (when the observer makes contact with the object) from the difference between

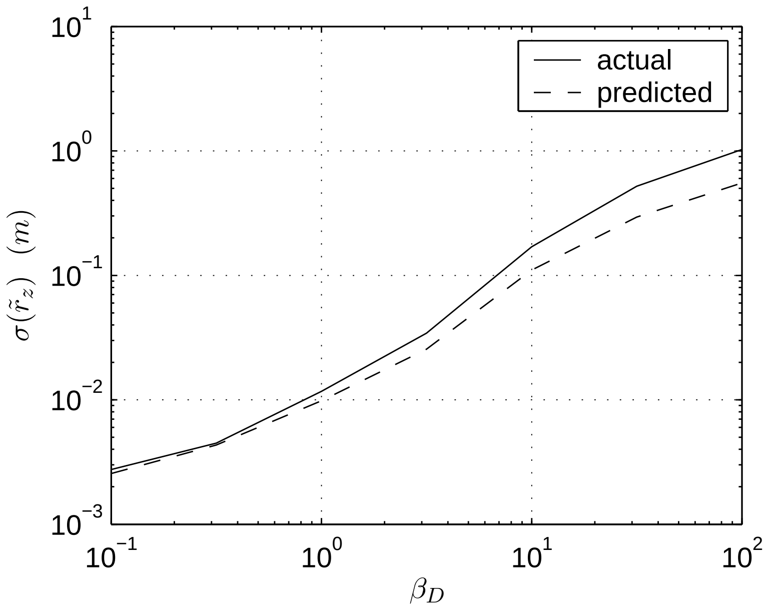

Figure 6.12: Parameter Study: Actual and Predicted Standard Deviation of the Range Estimate Error for $\beta_N = 1$

the estimated and true states. The predicted standard deviations, $\hat{\sigma}(\tilde{r}_z)$, are computed from the estimate covariance.

This comparison shows that the estimate covariance is a useful predictor for the actual standard deviation of the estimate error, which broadens a conclusion from Section 6.1.2. However, it also shows that as the disturbance factor increases, the disparity between the actual and predicted standard deviation also increases.

This effect can be explained by the presence of greater uncertainty in the estimate, particularly in the ζ state, as β_D increases. Although the UT algorithm (not unlike most other optimal estimators) assumes that the probability distribution is Gaussian, the actual distribution is not necessarily Gaussian in a nonlinear system. In fact, the probability distribution for ζ is expected to be quite skewed. When the uncertainty is small, this difference in shape is not very important and the assumption that all distributions are Gaussian is a good one. However, when the uncertainty is large, this assumption induces errors that are not accounted for. This effect is illustrated in Stengel [53, pp. 382-3]. When errors caused by this approximation begin to dominate, an estimator design that can handle non-Gaussian distributions, like a Particle Filter, should be used (see Section 7.1.3).

The plots show that as the disturbance grows, the accuracy of the range estimate becomes much larger than typical objects in an underwater environment. This occurs because the trajectory used for all of these runs was designed for $\beta_D = 1$ and is constrained by assumed observer actuator limits. When the disturbance factor is greater than 10, typical disturbance forces actually exceed the assumed observer actuator limits, which would also lead to a breakdown in controller performance. A more realistic scenario would consider an observer with actuators that are more powerful than the expected disturbances and are capable of executing a faster trajectory. Such a trajectory would provide greater observability and presumably better performance than is indicated in these plots. However, in this analysis, the trajectory remains the same, regardless of the size of the disturbances.

Next, a comparison between multiple values of β_D and β_N is presented. Figure 6.13 shows the standard deviation of the estimate error for the object range (r_z) as well as the z-components of the disturbance force, the $\mathbf{Z}$-vector and accelerometer bias vector (a_z, Z_z, and $b_{a,z}$, respectively). Only the standard deviation computed from the estimate error (σ) is plotted. These are computed at the end of the trajectory. Each line represents a different value of β_N.

The plot for object range shows an obvious trend towards better estimator performance as the disturbance factor decreases and worse performance as it increases. It also shows that changes in β_N have very little effect on the accuracy of the range estimate. The same is true for the disturbance estimate. Not surprisingly, changes in β_N have a direct effect on the accuracy of the accelerometer bias estimate. However, for the bias estimates, β_D has very little effect. The dependence on β_D and β_N of the accuracy of the $\mathbf{Z}$-vector is discussed later.

Figure 6.14 shows η, the success rate of the estimator, which represents the number of runs that were not culled based on the integrity checks described in Section 6.1.2. The degradation in performance above $\beta_D = 10$ coincides with the separation between the actual and predicted standard deviations discussed above.

The next plot compares the mean and standard deviation of the range estimate error for different values of β_D and β_N. Again, these quantities are computed at the end of the trajectory. Figure 6.15 shows that the standard deviation of the estimate error ($\sigma(\tilde{r}_z)$, solid lines) is greater than the mean estimate error ($\mu(\tilde{r}_z)$, dashed lines) as long as $\beta_D < 10$. Although the estimates are slightly biased—which is not uncommon for nonlinear estimators—the standard deviation represents the dominant source of error. For larger values of β_D, this relationship reverses. This is another manifestation of the assumption that all probability distributions are Gaussian. As the uncertainty increases, errors due to this assumption also increase and cause the estimates to become biased.

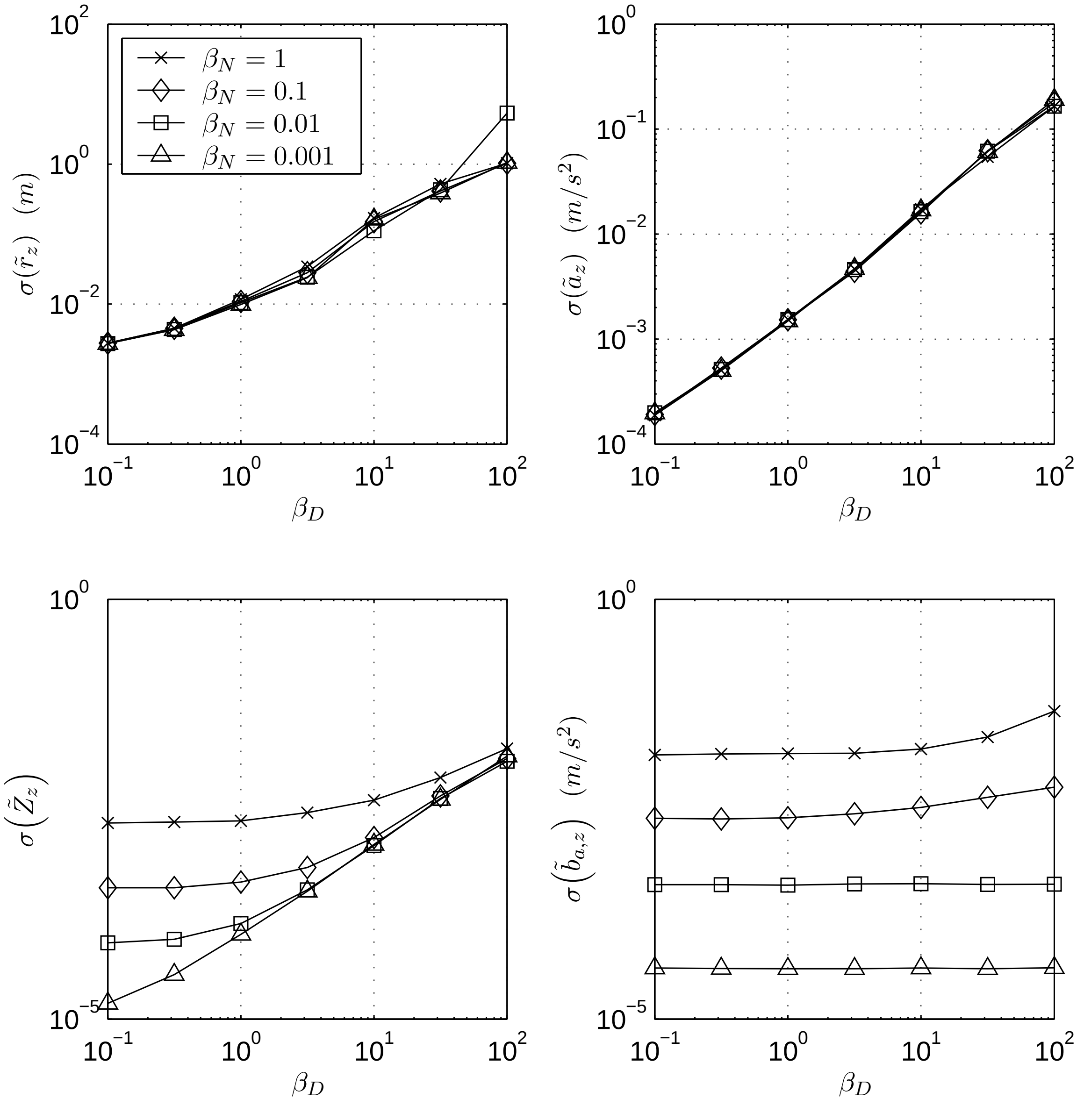

Figure 6.13: Parameter Study: Standard Deviations of the Estimate Error for Four Representative States

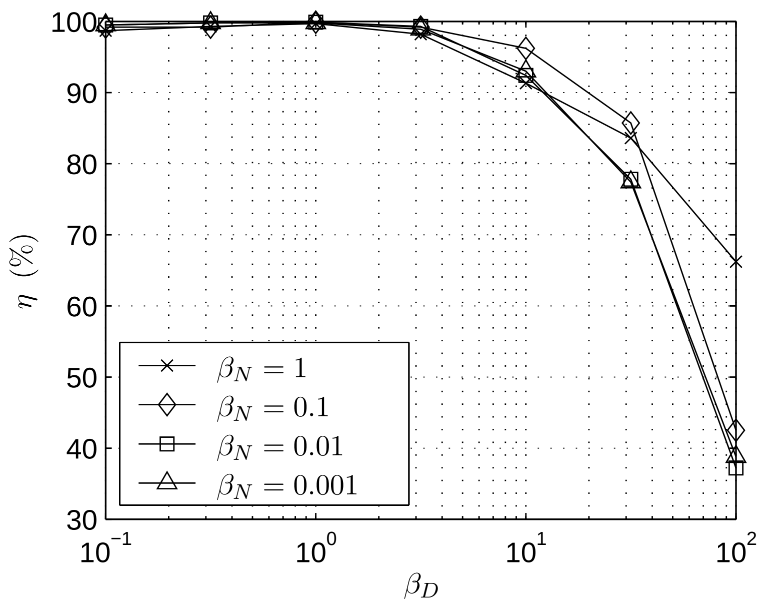

Figure 6.14: Parameter Study: Success Rate

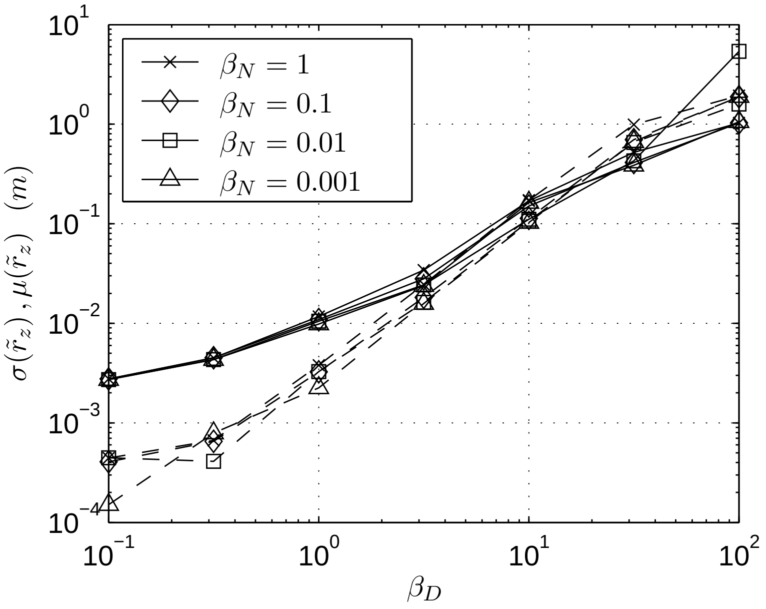

Figure 6.15: Parameter Study: Mean and Standard Deviation of Estimate Error for Range Estimate

6.2.3 Discussion

These simulations, which explore the effect on estimator performance of different disturbance levels and higher quality inertial rate sensors, provide insight into the sensing strategy and the operation of the estimator. As shown in Figure 6.13, the positioning accuracy does not improve if the quality of the inertial rate sensors improves. This is a surprising observation which indicates that another uncertainty controls the achievable relative position accuracy. The simulation suggests that a key factor for determining the achievable accuracy is the size of the external disturbances.

This effect can be explained by focusing on the accelerometer sensor model:

$$\mathbf{z}_a = -\mathbf{u}_1 + \gamma\mathbf{v} - \mathbf{a} + \mathbf{b}_a + g\mathbf{Z} + \mathbf{n}_a \tag{6.18}$$

The force input $\mathbf{u}_1$ is deterministic, so the innovations from this measurement depend on five terms: the linear drag term ($\gamma\mathbf{v}$), the disturbance ($\mathbf{a}$), the accelerometer bias estimate ($\mathbf{b}_a$), the $\mathbf{Z}$-vector scaled by the apparent acceleration due to gravity ($g\mathbf{Z}$), and the measurement noise ($\mathbf{n}_a$). Figure 6.16 compares the standard deviation for these five terms (only the z-components) on the same plot. Each plot corresponds to a different value of β_N.

These plots suggest that the $\mathbf{Z}$-vector dominates the uncertainty of the accelerometer measurement. The drag (γv_z) and disturbance (a_z) estimates behave independently from the size of the measurement noise. This suggests that these estimates are corrected by a different mechanism within the estimator. One possibility is an implicit differentiation of the relative position, which explains their susceptibility to the size of disturbances. The accelerometer measurement update corrects primarily the bias ($b_{a,z}$) and gravity vector (gZ_z) terms. The accelerometer bias is constant for the short time-scales of each experiment. Therefore, it has substantially different dynamics than the other states and can be separated easily from them. However, the gZ_z term has similar dynamics to the drag and disturbance terms, which makes these terms more difficult to separate. Therefore, the accelerometer measurement update cannot improve the gravity vector estimate below the accuracy of the drag and disturbance terms. This is evident in the plots, which show that the gZ_z term is approximately bounded below by all the other terms. This leads to the knee in the curve for the $\mathbf{Z}$-vector trace.

The presence of the gravity vector in the accelerometer measurement has the effect of improving the attitude estimate and degrading the disturbance estimate. Even though the $\mathbf{Z}$-vector, converted to units of acceleration by g, is the least certain term, it provides a very good estimate of attitude. This helps to reduce the effect of measurement noise in the rate gyro measurement and leads to good convergence of the rate gyro biases. Because of this ability to correct errors in the rate gyro measurements, the quality of the rate gyros is not

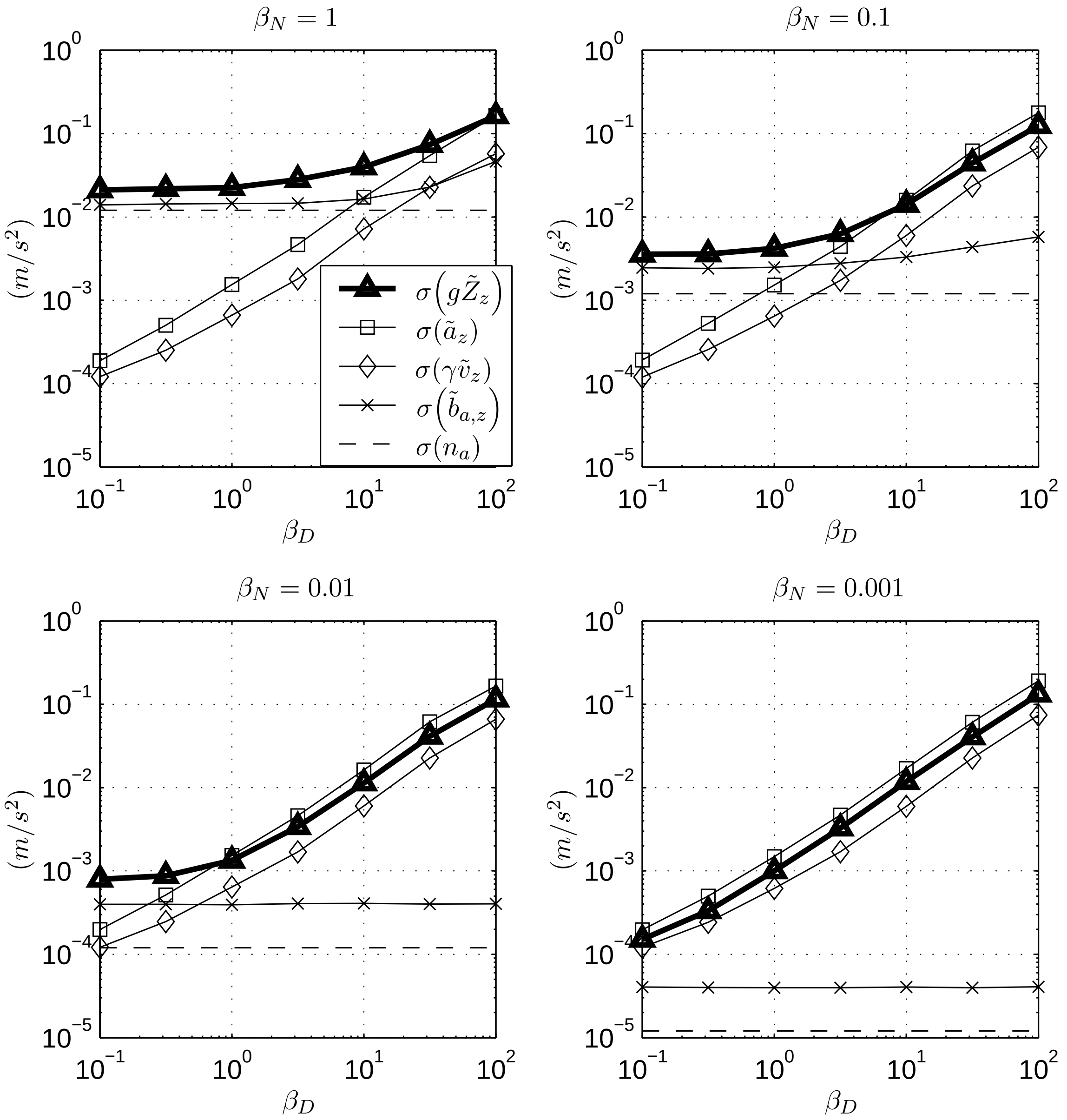

Figure 6.16: Parameter Study: Terms of the Accelerometer Measurement in the z-direction

very important. However, the uncertainty of the gravity vector is large compared to typical disturbances, so that estimating disturbances from the accelerometer measurements is difficult, regardless of the quality of the accelerometers.

If observer attitude were known exactly, then the effect of gravity could be removed from the accelerometer measurement and the measurement would be able to correct the estimate of disturbance to an accuracy close to the accelerometer measurement noise.

This explains why the relative position estimates are insensitive to the inertial rate sensor quality. Any measurement or bias errors in the rate gyro measurement are corrected by the accelerometer measurement, which tends to be sufficiently accurate for this purpose regardless of the inertial rate sensor quality. However, the gravity component in the accelerometer measurement limits the accuracy with which real accelerations can be measured. In practice, this limit is high enough so that the accelerometer measurements contribute little to the estimate of acceleration and relative position.

This analysis reveals a key limitation of the sensing strategy. The achievable accuracy of the relative position estimate is determined by the disturbance environment. Disturbances are determined by the given application and not by design parameters, like inertial rate sensor quality, which can be affected by the sensor, estimator, or trajectory design. Therefore, if more than the achievable accuracy is required, a modified sensor strategy has to be developed. This will be suggested as future work in the final chapter.

6.3 Hardware Experiments

The purpose of the hardware experiments is to validate the simulation results presented in the previous section, to identify and bound the effect of those features of a potential application that were not part of the simulation, to demonstrate the usefulness of this sensing strategy for an autonomous object manipulation task, and to show that the estimator can run in real time. Section 6.3.1 presents *open-loop* hardware results that complement the Monte Carlo simulations presented in Section 6.1.2. Section 6.3.2 shows a *closed-loop* experiment for in which the estimator output is used for the motion control of the robot.

These experiments are performed with the experimental system described in Chapter 5. The results demonstrate the performance of the estimator in the context of real sensor measurements and other challenging aspects of a hardware experiment, like model errors, calibration errors and time delays.

6.3.1 Baseline Experiment

This section presents results from the baseline hardware experiment. This experiment consists of twenty runs in the "control-from-truth" configuration. In this configuration, the observer motion does not depend on the estimator output. The observer control is based on the truth measurement of relative position.

Variation between the runs of this experiment are caused by a variety of factors. Although the object position remained the same for the entire experiment, each of the twenty runs started with a different initial observer position. Therefore, the observer motion—determined by the initial observer position, the fixed object location, the desired relative position specified in the trajectory, the simulated disturbances on the observer, and the performance of the controller—is different for each run. This variation in the observer motion together with variation in the process noise and measurement noise leads to variations in the estimator performance between runs.

The results in Figures 6.17 to 6.20 are averages of the estimate error and the innovations computed over all twenty runs. Estimate errors can only be computed for states with a truth measurement. These include relative position, disturbance forces and observer orientation. Each of the estimate error plots shows the mean estimate error (μ), the actual standard deviation of the error (σ), and the predicted standard deviation of the error ($\hat{\sigma}$), computed from the covariance matrix of the estimator. Equations 6.4 to 6.7 show how these statistics are computed.

The relative position estimate is the main output from the estimator. The three plots in Figure 6.17 show the consistency between the actual and predicted standard deviations for the relative position estimate error. The actual standard deviation (σ) is sometimes larger than the predicted standard deviation ($\hat{\sigma}$), but the excess is small and the shape of the two curves remains similar. This consistency is a significant achievement of the estimator design for this sensor fusion problem. This was already observed for the simulation results. This plot extends this conclusion to the experimental results. The estimate error shown in these plots is due in part to errors in the truth measurement (see Section 5.5.1). At the end of the trajectory, the standard deviation of the range estimate is $2.8\,cm$ compared to the predicted standard deviation of $1.0\,cm$. The mean range estimate error is $-1.7\,cm$.

Figure 6.18 shows results for the estimate of disturbance forces in the left column and observer attitude in the right column. For the disturbance forces, the standard deviation of the estimate error is larger than predicted by the covariance matrix. This indicates sources of process noise in the hardware experiment that are not fully modeled. For the comparison of observer attitude, the estimate is normalized from the $\mathbf{Z}$-vector representation used by the estimator to the $\mathbf{R}_z$-vector representation available on the robot. This is done according to (4.42).

Figure 6.19 shows the results for the heading estimate. The heading estimate is the result of an implicit open-loop integration of rate gyro measurements. That is, no absolute measurement of heading has been integrated into the sensor fusion algorithm. Because

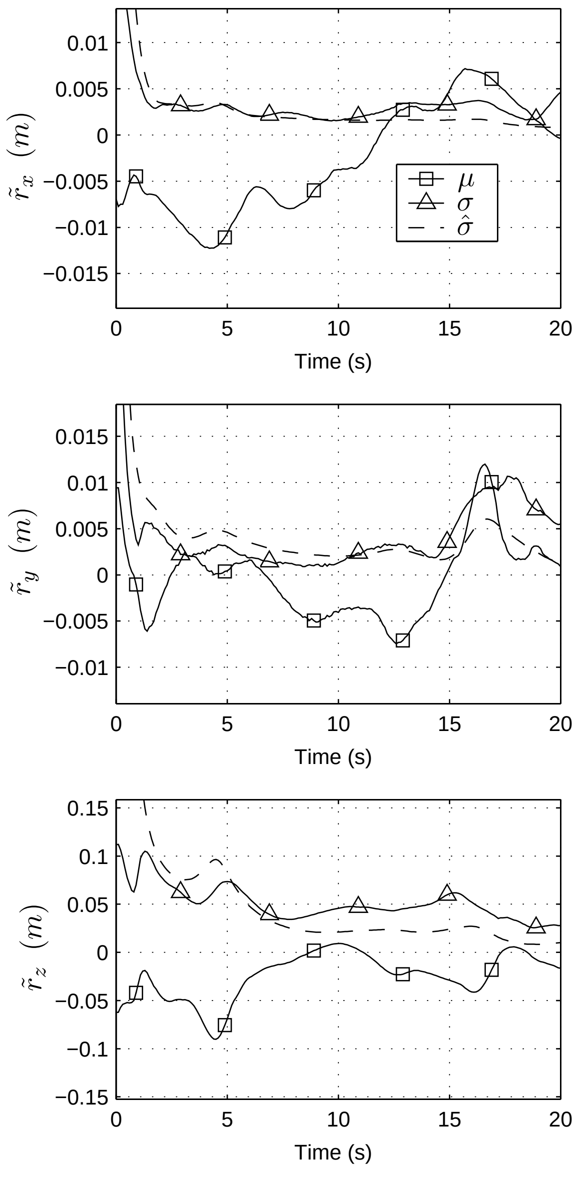

Figure 6.17: Baseline Hardware Experiment: Relative Position

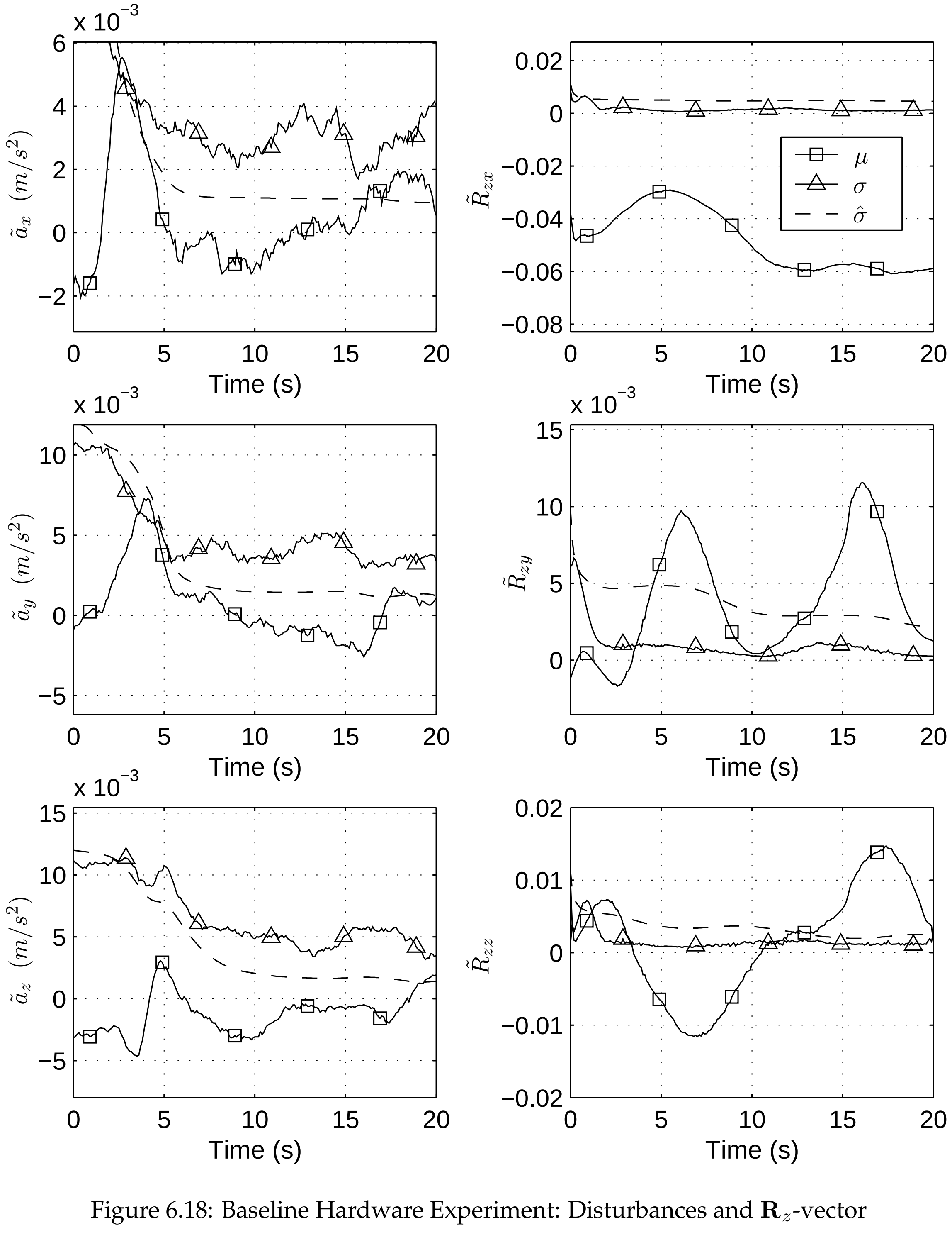

Figure 6.18: Baseline Hardware Experiment: Disturbances and $\mathbf{R}_z$-vector

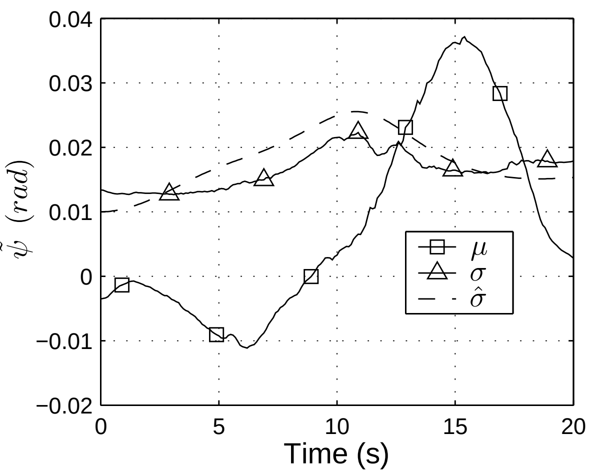

Figure 6.19: Baseline Hardware Experiment: Observer Heading

the estimator can only determine changes in heading, it was initialized with the actual heading at the beginning of every run.

Figure 6.20 shows the innovations, which are the difference between the predicted and actual measurements and can be computed without truth measurements. These plots were used to set the measurement noise parameters that have been used throughout this dissertation.

For these hardware results, which include a variety of unmodeled errors, the standard deviation in the error of the range estimate is about $3\,cm$. This is significantly larger than the corresponding simulation result of $1\,cm$. This shows the sensitivity of the performance to small errors, which is a consequence of the limited observability of the sensing strategy. This effect has already been described in Section 6.2.

Nevertheless, the hardware experiments confirm the validity of the Monte Carlo simulations. Although the simulation results indicate better accuracy than the hardware experiment, the quality of the performance (e.g., the shape of the standard deviation plots, relative accuracies, convergence properties) are similar. Therefore, the simulations are a useful tool for evaluating the sensing strategy and for predicting the performance of hardware experiments.

6.3.2 Object Pick-Up Task

The sensing strategy has been designed to be integrated into the control of an observer to perform autonomous manipulation tasks. This section presents an experiment in which a real-time implementation of the sensing strategy is used to guide the manipulator towards an object whose position is unknown. This is the first time that a relative position estimate

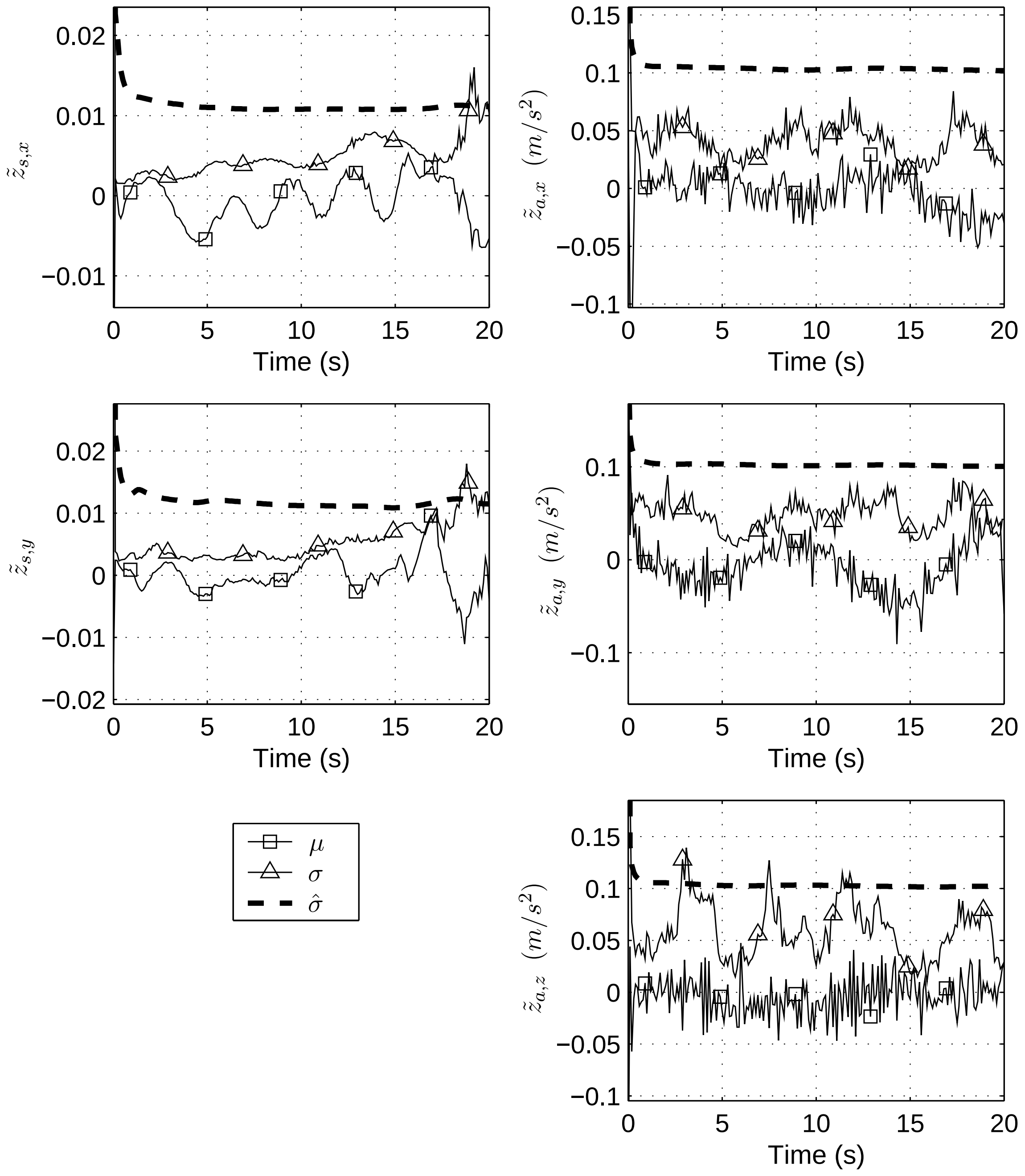

Figure 6.20: Baseline Hardware Experiment: Innovations

derived from a bearing measurement and inertial rate sensor measurements has been used to perform such a manipulation task.

This experiment includes three runs with a different unknown cup position for each run. Therefore, the relative position between the moving observer and the stationary object is unknown at the beginning of each run. For each run, the robot is commanded to execute the trajectory, which defines the desired relative position and velocity between the observer and the robot as a function of time. For this experiment, the relative position and velocity used by the controller is determined by a real-time implementation of the estimator, which corresponds to the "control-from-estimate" configuration. Each run terminates with the manipulator endpoint in position to grasp the cup.

Figure 6.21 shows the endpoint position of the robot for all three runs, each of which begins at the same starting position and terminates at one of the three cup positions. The picture on the right shows the robot at the starting position. The three pictures on the left show the robot at one of the three final positions. The arrows connect the pictures to the corresponding robot location on the plot.

This experiment shows that a real-time $(10\,Hz)$ implementation of the estimator is possible and that the estimator can be incorporated into the closed-loop control of an autonomous robot.

These results have been obtained by eliminating the external disturbance input to improve the accuracy of the control. The preceding experiments in this Chapter have highlighted the estimator uncertainty that can be achieved at disturbance levels that are considered typical for the underwater environment. However, these uncertainties are large enough to cause significant failure rates for this particular object pick-up task. The geometry of the object and the gripper can only accommodate deviations of about $\pm 1\,cm$. The Monte Carlo simulations in Section 6.2 indicate that the disturbance level is a key parameter that affects positioning accuracy. It is also easily controlled in the hardware experiment because the dominant disturbances are simulated. Setting the disturbance level to zero improves the estimator accuracy sufficiently to provide for successful execution of the object pick-up task.

In a real operational setting, in which the external disturbance cannot be manipulated, improving accuracy by reducing the disturbance is not an option. In this case, only tasks that can handle greater misalignment than the object pick-up task described in this dissertation can be performed. For example, a gripper could be designed that can grasp an object in the presence of larger variation in the relative position between the two. Also, a variety of observation tasks that do not involve contact could be performed.

However, this limitation is due only in part to the estimator performance. Even with a perfect measurement of relative position, designing a controller to position an underwater

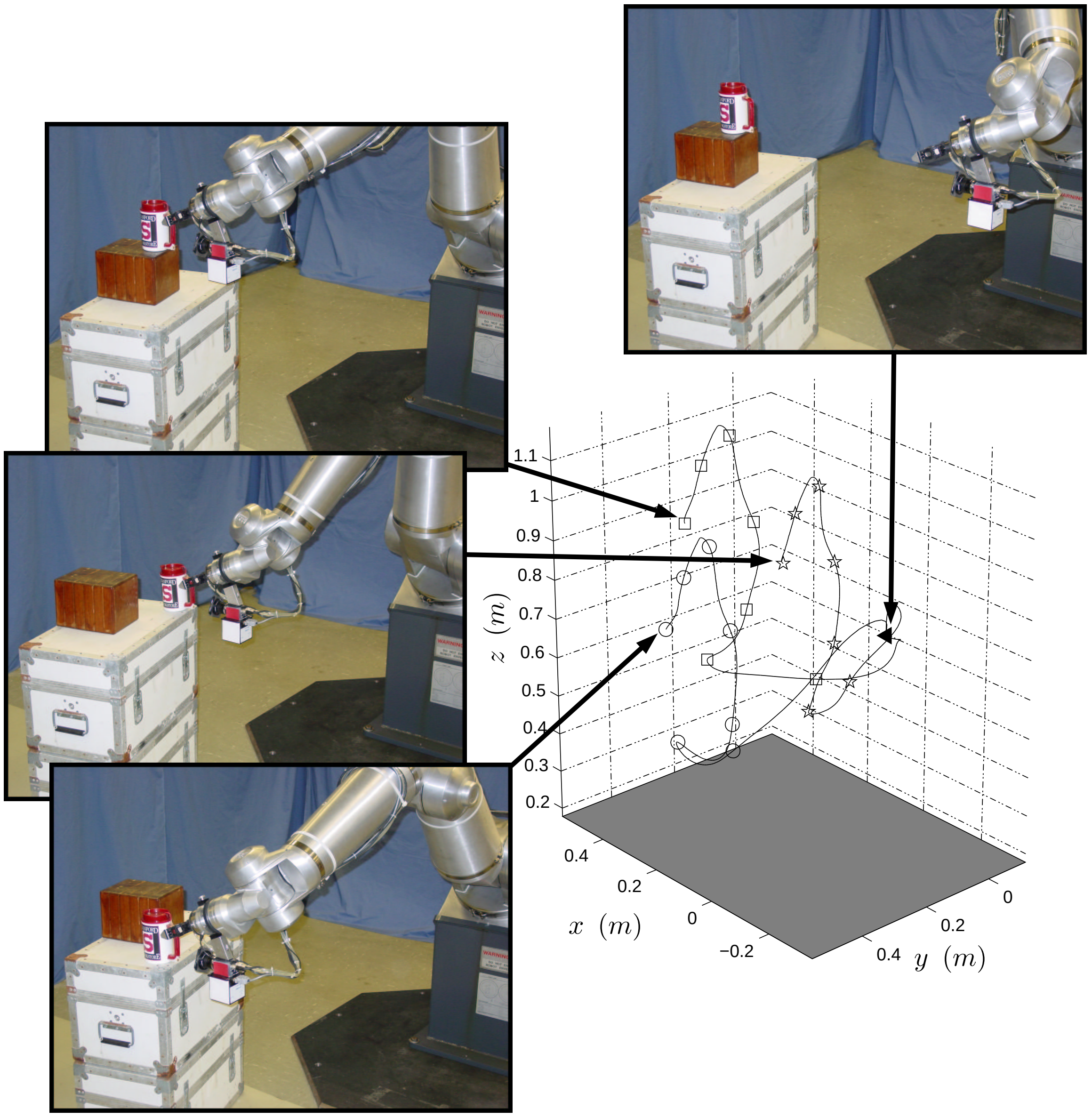

Figure 6.21: Multiple Runs of the Object Pick-Up Task with Three Different, Unknown Object Locations

vehicle in the presence of significant disturbance forces with centimeter-level accuracy is challenging.

6.4 Summary

This chapter has presented results from Monte Carlo simulations and from hardware experiments. The first section focused on simulation results for the baseline experiment, which demonstrate the estimator performance with default parameters. The simulation results show that the new estimator design generates results with only small estimation biases and that the estimate covariance accurately predicts the uncertainty of the estimate. The second section presented a parameter study where both the level of disturbances and the inertial rate sensor quality have been varied. This analysis shows that the estimator performance depends strongly on the disturbance levels, but is largely insensitive to the inertial rate sensor quality. The third section described two hardware experiments. The baseline experiment performed on the hardware validates the simulation results and highlights the effects of model errors associated with real hardware. The object pick-up task is an experiment in which a real-time implementation of the sensing strategy has been used to perform a simple manipulation task with an autonomous robot. This is the first implementation of this sensing strategy that has been used to perform an autonomous object manipulation task.

Chapter 7

Conclusions

Fusing monocular vision measurements of a single feature with inertial rate sensor measurements generates a feasible sensing strategy for determining relative position between a moving observer and stationary object. This capability enables relative position control for floating underwater vehicles, and can be adopted to other applications.

The main contribution of this research is the design of a real-time, nonlinear estimator to perform the fusion of these measurements. This estimator design has enabled the new sensing strategy. The dissertation explains why standard nonlinear estimation tools, like the Extended Kalman Filter, perform poorly for this sensor fusion problem; it shows that the new estimator leads to adequate performance, producing results with only small estimation errors and accurately predicting the standard deviation of the estimate errors; it presents the first hardware demonstration of this sensing strategy; and it shows how to integrate the sensing strategy into a useful closed-loop manipulation task.

The research provides important contributions to a range of applications. The estimator design strategy—avoiding linearizations by choosing a representation with a linear sensor model and implementing the time update with an Unscented Transform—should provide improved results for any nonlinear estimation problem that suffers from error introduced by linearizations. The relative position estimate can be used to perform a variety of underwater vehicle tasks including station-keeping, tracking, and observation. The sensing strategy is a useful capability for other types of *floating* robots—especially those that already use inertial rate sensors to determine their motion, like aerial and space robots.

7.1 Future Work

This dissertation represents a first step in the exploration of a new sensing strategy. At the outset of this work, it was not obvious that determining relative position with this approach would in fact work, or what the most difficult issues would be. This dissertation has presented a detailed description of the sensing strategy, a design for the nonlinear, real-time estimator to fuse the sensor measurements, and a demonstration of a useful task based on this sensor strategy.

In the course of this research, several opportunities for future work have been identified. These include issues that have not been fully explored in this dissertation, ideas for extensions of the sensing strategy, and additional advances required to enable a robust demonstration on an operational vehicle.

7.1.1 Uncertain Dynamics

The main limitation of this sensing strategy is that its performance degrades in the presence of large disturbances and significant errors in the dynamics model of the observer. Although it is possible, determining accurate dynamics models for underwater vehicles is very challenging. Moreover, the effect of disturbances, which tend to be large for underwater vehicles, can result in diminishing returns for more sophisticated dynamics modeling. Some applications, especially those with small disturbances, are not affected by this limitation. However, when disturbances and model errors cause errors in the relative position estimate that exceed the requirements, the sensing strategy should be extended to avoid this limitation. Section 7.1.2 recommends possible extensions based on additional sensor measurements.

If disturbances are not the primary source of estimate error, an analysis of the estimator sensitivity to errors in the observer dynamics model would identify the accuracy with which these models should be known for a given estimate accuracy. This would be a useful step towards determining the level of complexity that is required for these models.

7.1.2 Additional Sensor Measurements

This research was focused on a minimal set of sensor measurements to determine relative position between a moving observer and a stationary object. The goal was to expose fundamental issues in the implementation of the sensing strategy by reducing it to its core requirements. However, the sensing strategy has a significant limitation—in the presence of large disturbances, the performance of the estimator degrades.

In many cases, more than the minimal set of sensor measurements are available and the performance of the system can be improved by incorporating additional measurements. A key strength of this sensing strategy is that it is flexible enough to accept many additional measurements given only minor modifications to the estimator that solves the sensor fusion problem.

- **Additional Features**

  The ability to determine relative position by tracking just one feature is a very useful property, especially underwater, where robust features can be scarce. However, taking advantage of additional features, when available, should provide significant benefits, like improved accuracy and increased robustness. This approach provides additional sensor measurements without adding additional hardware. It also adds flexibility to the sensing strategy, whereby a feature that is traveling out of the field of view of the camera can be augmented by a more appropriate feature before the first feature is lost.

  Tracking additional features with the monocular vision system requires a simple extension of the estimator. For each additional feature, three new states ($s_{x,j}$, $s_{y,j}$, and ζ_j, $j = 2, 3, \ldots$) need to be added to the state vector, with obvious extensions to the sensor and process models. Each additional feature should correspond to objects that are also stationary.

  Of particular interest with a multiple feature design is the ability to detect when not all tracked features correspond to the same stationary object. This situation will lead to poor estimator results unless the problem features are removed from the estimator.

- **Additional Cameras**

  Integrating additional cameras into the sensing strategy combines the benefits of multi-camera solutions with the robustness of this sensing strategy. When accurate feature correspondences can be established in a multi-camera system, they provide range to an object without the need for motion. However, during vision drop-outs of one or more cameras or when feature correspondences are unreliable, the sensing strategy can operate without the multi-camera constraint and the inertial rate sensors can be used to propagate a useful estimate.

  Solutions based on multiple cameras and inertial rate sensors can be constructed using either tight or loose integration. In a loose integration, the camera measurements are combined to form a relative position measurement, which is subsequently fused with the motion estimate computed from the inertial rate sensors. In a tight integration, which is enabled by this research, all of the vision measurements and inertial rate sensor measurements are fused in the same estimator.

  The advantage of tight integration appears in the application of the multi-camera constraint. For loose integration, the constraint has to be applied in the vision algorithm to corresponding features in order to get any output from the vision system. However, for tight integration, the constraint can be applied selectively, that is, only when a correspondence exists and the system has confidence in it. When the constraint is applied correctly, the estimate accuracy improves the most. But even without the constraint, the separate vision measurements provide a useful contribution to the estimator output.

The estimator requires a triplet of states $\begin{bmatrix} s_x & s_y & \zeta \end{bmatrix}^T$ for each feature measurement in each camera, with obvious extensions to the sensor and process models. The constraint between corresponding features, when it exists, can be applied in the time-update. Being able to incorporate a nonlinear constraint between redundant states highlights a key advantage of the choice to use the Unscented Transform to implement the time-update.

- **DVL**

  Many underwater vehicles now carry a Doppler velocimetry logger, or DVL. The DVL measures the body velocity either over the seafloor or relative to particles suspended in the water, using a sonar Doppler approach. In the first mode, the velocity over the seafloor corresponds one-to-one to the velocity states in the estimator and could therefore be incorporated with a linear sensor model. In the second mode, the velocity relative to particles provides a measurement proportional to the disturbance state $\mathbf{a}$, which is linear in the relative velocity between the water and the vehicle (see Section 2.3).

  A DVL measurement could provide significant performance improvements, but would also incur additional cost and complexity. A measurement of observer velocity would improve performance in two ways: First, observer velocity has to be integrated only once to obtain position. This reduces the accumulation of drift errors due to measurement and bias errors. Second, differentiating the velocity measurement in the estimator provides a means to separate actual observer accelerations from the other components in the accelerometer measurement. Together, these performance improvements should reduce the dependence on accurate dynamics models. However, incorporating additional hardware into the sensing strategy also introduces new failure modes (like losing bottom-lock in the DVL measurement) and creates more calibration parameters (both intrinsic, like transducer scale factors, and extrinsic, like sensor position and orientation with respect to the other sensors).

- **Compass**

  A compass would provide the sensing strategy with an absolute measurement of observer heading. Without it, the estimator simply integrates the rate gyro measurements, with corresponding growth of uncertainty. The estimator already has a state for observer heading, so the compass can be added with a linear sensor model. However, compass measurements are not trivial to integrate into an estimation algorithm because they can be biased by spurious fields generated by the vehicle and by the environment. Without careful modeling, these biases can easily distort other estimates.

7.1.3 Analysis of Achievable Performance

The results in Chapter 6 indicate that the estimator presented in this dissertation generates results that have only small biases, that it accurately predicts the RMS estimate error, and that the innovations are appropriate. Although these are useful metrics for the performance of a nonlinear estimator, they do not compare the estimator results to the best achievable performance. The results cannot be used to claim that no other estimator, given the same constraints and data, can provide a significant improvement in the estimate accuracy.

An analysis of achievable performance would be useful in deciding whether additional work in developing a better estimator design is warranted. A bound that is significantly lower than the current estimator performance would motivate further research on the estimator design. But if this estimator approaches the achievable performance, then research effort should be directed toward some of the other issues described in this section.

An observability analysis could help to identify achievable performance. However, simple nonlinear extensions of observability analyzes developed for linear systems are not suitable for this sensing strategy. They suffer form the same problems with linearization that were discussed in the context of the Extended Kalman Filter. Full nonlinear observability analysis is possible in theory, but rarely practical.

This sensing strategy requires a real-time solution, so that the estimate can be used for the closed-loop control of the observer. However, by lifting the requirement for a real-time, recursive estimate, building a potentially more accurate estimator should be possible, at least for the purpose of analysis.

An interesting approach would involve a Particle Filter implementation. At least in theory, Particle Filters provide a continuous trade-off between accuracy and computational cost by varying the number of particles. As the number of particles is allowed to grow to infinity, the Particle Filter should reach an arbitrarily accurate solution. Solving the sensor fusion problem with progressively larger numbers of particles should identify a lower bound on estimator accuracy. At the same time, it would provide a measure of the computational cost involved in achieving this bound.

As research on Particle Filters progresses and as computers become faster, a Particle Filter solution might also become a candidate for a real-time solution.

7.1.4 Trajectory Design

The observer trajectory determines the quality of the information that is provided to the estimator by the sensor measurements. Even a very good estimator will generate poor results if the observer follows a poorly designed trajectory. The trajectory presented in Section 5.8 for the demonstration task was designed in an *ad hoc* fashion (using heuristics and trial-and-error) to satisfy the task constraints (moving to a goal position that enables a grasp of the object) and to provide sufficient observer motion for convergence of the estimator.

Future extensions could include trajectories that are planned online (quickly and without human tweaking) and trajectories that are optimal with respect to a useful cost function. Online trajectory planning is the first step toward online task planning based on *in situ* object modeling and reasoning. Optimal trajectories provide the best trade-off between the amount of observer motion and estimator performance.

Much of the literature on optimal trajectory design for bearing sensors has come from the passive sonar tracking community. More recently, Frew [14] has extended this work for a system with a bearing measurement from vision and has developed an online solution. The solution computes the predicted estimator performance for a subset of possible observer paths and chooses the best one. Although optimality is not guaranteed, the search runs quickly enough so that a lot of potential observer paths can be considered and good solutions are typically found.

Although this approach could be extended for the sensing strategy discussed in this dissertation, it would have to be modified to account for some important differences in the problem. First, due to significant differences in the complexity of the estimation problem, the cost of computing the predicted estimator performance increases dramatically for this problem and Frew's method becomes intractable. Second, additional task-specific constraints have to be incorporated. For instance, the trajectory has to be constrained to end at a specific relative position for grasping.

An initial attempt to develop an online trajectory capability for this problem based on an extension of Frew's work has already been presented in [20], but this did not generate optimal solutions, was still too slow, and has not yet been integrated with the experimental system.

7.1.5 Moving Objects

All the work described in this dissertation assumes that the object to be tracked is stationary. A more exact requirement is that the object be non-accelerating: this ensures that the accelerometer measurements actually describe the relative position between the object and the observer. However, objects that travel at constant velocity are not very common in the underwater environment. For applications with moving, non-accelerating objects, the sensing strategy, as is, should provide useful results.

An interesting extension to this work would allow moving objects that can accelerate. To enable this capability, the estimator would need to estimate both the observer and the object velocity. Extra help in the form of additional measurements, motion information from the object, or dynamics models for the object might be required for adequate performance.

7.1.6 Demonstration in an Operational Setting

This section describes several topics that should be addressed in preparation for a demonstration of this sensing strategy in an operational setting. The research so far has been focused on the development of the sensing strategy and the laboratory demonstration. The goal was to highlight fundamental issues, to determine whether the strategy is robust enough to succeed on a hardware system, and to demonstrate that it can be implemented in real time. Section 7.1.1 has already discussed the need to deal with uncertain dynamics and large disturbance forces in real environments. This section considers the need for a real vision processing capability, the opportunity to refine the sensor models, and the construction of a multi-rate estimator.

- **Vision Processing for Real Environments**

  Robust and effective vision processing algorithms for real underwater environments are still a topic of current research. Before the sensing strategy can be applied in these environments, it has to be matched with a vision processing algorithm that can track features that are relevant to the target application. Although the research described in this dissertation does not represent any progress toward better vision processing algorithms for real environments, it enables useful vision-based tasks that can operate in environments, like underwater, that provide few robust trackable features.

- **Refined Sensor Models and Improved Calibration**

  For this research, very simple sensor models were chosen and the model parameters were determined with quick calibration methods. Now that the sensing strategy has been developed and the effects of different design choices can be evaluated, an analysis of sensitivity for many of these choices would focus attention on those that are most critical to the design.

  Many researchers have developed adaptive estimators which compute the best covariance matrices for the process noise (i.e., Q) and the measurement noise (i.e., R)

based on the innovations process. These methods could be applied to this estimator design in order to improve performance, to reduce the dependence on the calibration parameters, and to adapt to slowly-changing conditions.

- **Multi-Rate Estimator**

  In the current estimator, all measurements are incorporated at $10\,Hz$. A preprocessing step synchronizes and combines all measurements to conform to this rate. However, the measurements are inherently generated at multiple rates. Inertial rate sensors with digital interfaces typically produce measurements at rates greater than $100\,Hz$. Those with analog interfaces could be sampled even faster. Cameras used in vision systems typically produce new images at rates of up to $60\,Hz$, but these are subject to interpretation. Some features can be extracted quickly, but others require computationally more expensive algorithms, which might depress the rate of available vision measurements. Some vision systems produce outputs at variable rates.

  The rate at which measurements are incorporated can affect the overall accuracy of the estimator. This factor should be examined more closely to quantify the advantages of a multi-rate estimator.

Bibliography

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

[2] Bir Bhanu, Subhodev Das, Barry Roberts, and Dave Duncan. A system for obstacle detection during rotorcraft low altitude flight. *IEEE Transactions on Aerospace and Electronic Systems*, 32(3):875–897, July 1996.

[3] H. Carvalho, P. Del Moral, A. Monin, and G. Salut. Optimal nonlinear filtering in GPS/INS integration. *IEEE Transactions on Aerospace and Electronic Systems*, 33(3):835–850, July 1997.

[4] Tan Fung Chan and Rajiv V. Dubey. A weighted least-norm solution based scheme for avoiding joint limits for redundant manipulators. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 395–402, May 1993.

[5] Vincent W. Chen. *Experiments in Nonlinear Adaptive Control of Multi-Manipulator, Free-Flying Space Robots*. PhD thesis, Stanford University, Stanford, CA 94305, December 1992. Also published as SUDAAR 631.

[6] Peter I. Corke. *Visual Control of Robots: high-performance visual servoing*. John Wiley & Sons, Inc., New York, 1996.

[7] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley, 1989.

[8] Crossbow Technology, Inc., http://www.xbow.com/.

[9] Matthew Deans and Martial Hebert. Experimental comparison of techniques for localization and mapping using a bearings only sensor. In Daniela Rus and Sanjiv Singh, editors, *Experimental Robotics VII*, pages 395–404. Springer Verlag, 2001. Presented at the *International Symposium on Experimental Robotics, ISER 2000*. Also available at *http://www.ri.cmu.edu/events/iser00/papers/deans-final.pdf*.

[10] Matthew C. Deans. Maximally informative statistics for localization and mapping. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, volume 2, pages 1824–1829, Washington, DC, May 2002.

[11] 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, June 2001.

[12] A. Doucet, J. F. G. de Freitas, and N. J. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, New York, 2001.

[13] Hans Jacob S. Feder, John J. Leonard, and Christopher M. Smith. Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research*, 18(7):650–668, July 1999.

[14] Eric W. Frew. *Trajectory Design For Target Motion Estimation Using Monocular Vision*. PhD thesis, Stanford University, Stanford, CA 94305. in preparation.

[15] Demoz Gebre-Egziabher. *Design and Performance Analysis of a Low-Cost Aided Dead Reckoning Navigator*. PhD thesis, Stanford University, 2001. Chapter 3.

[16] Arthur Gelb, editor. *Applied Optimal Estimation*. The M.I.T. Press, Cambridge, Massachusetts, 1974.

[17] Nuno Gracias, Sjoerd van der Zwaan, Alexandre Bernardino, and Jose Santos-Victor. Results on underwater mosaic-based navigation. In *Proceedings of the Oceans 2002 Conference*, volume 3, pages 1588–1594, Biloxi, MS, October 2002. MTS/IEEE.

[18] Pini Gurfil and N. Jeremy Kasdin. Two-step optimal estimator for three dimensional target tracking. In *Proceedings of the 2002 American Control Conference*, volume 1, pages 209–214, Anchorage, AK, May 2002.

[19] Koichi Hashimoto, editor. *Visual Servoing: Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback*, volume 7 of *Robotics and Automated Systems*. World Scientific Publishing Company, Singapore, 1993.

[20] 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*, volume 3, pages 1857–1864, Biloxi, MS, October 2002. MTS/IEEE.

[21] Andreas Huster and Stephen M. Rock. Relative position estimation for intervention-capable AUVs by fusing vision and inertial measurements. In *Proceedings of the 12th*

International Symposium on Unmanned Untethered Submersible Technology, Durham, NH, August 2001. Autonomous Undersea Systems Institute.

[22] Kazufumi Ito and Kaiqi Xiong. Gaussian filters for nonlinear filtering problems. *Transactions on Automatic Control*, 45(5):910 –927, May 2000.

[23] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*, volume 64 of *Mathematics in Science and Engineering*. Academic Press, New York, 1970.

[24] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proceedings of the American Control Conference*, volume 3, pages 1628–1632, Seattle, WA, June 1995.

[25] Simon Julier, Jeffrey Uhlmann, and Hugh F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482, March 2000.

[26] Simon Julier and Jeffrey K. Uhlmann. Data fusion in nonlinear systems. In David L. Hall and James Llinas, editors, *Handbook of Multisensor Data Fusion*, chapter 13. CRC Press, 2001.

[27] Simon J. Julier. The scaled unscented transformation. In *Proceedings of the 2002 American Control Conference*, volume 6, pages 4555–4559, Anchorage, AK, USA, May 2002.

[28] Simon J. Julier and Jeffrey K. Uhlmann. Reduced sigma point filters for the propagation of means and covariances through nonlinear transformations. In *Proceedings of the 2002 American Control Conference*, volume 2, pages 887–892, Anchorage, AK, USA, May 2002.

[29] Thomas Kailath, Ali H. Sayed, and Babak Hassibi. *Linear Estimation*. Prentice Hall, 2000.

[30] Isaac Kaminer, Wei Kang, Oleg Yakimenko, and Antonio Pascoal. Application of nonlinear filtering to navigation system design using passive sensors. *IEEE Transactions on Aerospace and Electronic Systems*, 37(1):158–172, January 2001.

[31] N. Jeremy Kasdin. The two-step optimal estimator and example applications. In Robert D. Culp and Eileen M. Dukes, editors, *Guidance and Control 2000*, volume 104 of *Advances in the Astronautical Sciences*, pages 15–34, San Diego, CA, 2000. American Astronautical Society, Univelt Incorporated.

[32] N. Jeremy Kasdin and Thomas J. M. Weaver. Satellite quaternion estimation from vector measurements with the two-step optimal estimator. In Robert D. Culp and

Steven D. Jolly, editors, *Guidance and Control 2002*, volume 111 of *Advances in the Astronautical Sciences*, pages 15–34. Univelt Inc. Publishers, 2002.

[33] K. N. Leabourne, S. M. Rock, S. D. Fleischer, and R. L. Burton. Station keeping of an ROV using vision technology. In *Proceedings of the Oceans 97 Conference*, volume 1, pages 634 –640, Halifax, October 1997. MTS/IEEE.

[34] J.-F. Lots, D. M. Lane, and E. Trucco. Application of 2 1/2 D visual servoing to underwater vehicle station-keeping. In *Oceans 2000 MTS/IEEE*, volume 2, pages 1257–1264, Providence, RI, September 2000.

[35] Richard L. Marks. *Experiments in Visual Sensing for Automatic Control of an Underwater Robot*. PhD thesis, Stanford University, Department of Aeronautics and Astronautics, Stanford, CA 94305, June 1995. Also published as SUDAAR 681.

[36] Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 2. Academic Press, New York, 1982.

[37] Yoshihiko Nakamura. *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley, 1991.

[38] S. Negahdaripour, X. Xu, and L. Jin. Direct estimation of motion from sea floor images for automatic station-keeping or submersible platforms. *IEEE Journal of Oceanic Engineering*, 24(3):370–382, July 1999.

[39] Issa A. D. Nesnas, Mark W. Maimone, and Hari Das. Rover maneuvering for autonomous vision-based dexterous manipulation. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2296–2301, San Francisco, CA, April 2000. IEEE.

[40] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, 1990.

[41] Pulnix America, Inc., http://www.pulnix.com/.

[42] Henrik Rehbinder and Xiaoming Hu. Drift-free attitude estimation for accelerated rigid bodies. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 4244–4249, Seoul, South Korea, May 2001.

[43] W. D. Rencken. Concurrent localisation and map building for mobile robots using ultrasonic sensors. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2192–2197, Yokohama, Japan, July 1993.

[44] Jason Rife. e-mail correspondence, December 2002.

[45] Jason Rife and Stephen M. Rock. Field experiments in the control of a jellyfish tracking ROV. In *Proceedings of the Oceans 2002 Conference*, volume 4, pages 2031–2038, Biloxi, MS, October 2002. MTS/IEEE.

[46] Robotics Research Corporation, http://www.robotics-research.com/.

[47] S. Scheding, E. M. Nebot, M. Stevens, H. Durrant-Whyte, J. Roberts, P. Corke, J. Cunningham, and B. Cook. Experiments in autonomous underground guidance. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, volume 3, pages 1898–1903, Albuquerque, NM, April 1997.

[48] Stanley 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.

[49] Matthias Seitz. Towards autonomous robotic servicing: Using an integrated hand-arm-eye system for manipulating unknown objects. *Robotics and Autonomous Systems*, 26(1):23–42, January 1999.

[50] Christopher E. Smith and Nikolaos P. Papanikolopoulos. Grapsing of static and moving objects using a vision-based control approach. *Journal of Intelligent and Robotic Systems*, 19(3):237–270, July 1997.

[51] J. S. Smith, R. Yu, I. Sarafis, and J. Lucas. Computer vision control of an underwater manipulator. In *Proceedings of Oceans 94*, volume 1, pages 187–192. IEEE, September 1994.

[52] S. A. Stansfield. Robotic grasping of unknown objects: A knowledge-based approach. *The International Journal of Robotics Research*, 10(4):314–326, August 1991.

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

[54] Dennis Strelow and Sanjiv Singh. Online motion estimation from image and inertial measurements. In *Workshop on Integration of Vision and Inertial Sensors (INERVIS 2003), Proceedings of the 11th International Conference on Advanced Robotics (ICAR 2003)*, pages 1930–1937, Coimbra, Portugal, July 2003. IEEE.

[55] Rudolph van der Merwe and Eric A. Wan. Efficient derivative-free Kalman filters for online learning. In *European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium, April 2001.

[56] Rudolph van der Merwe and Eric A. Wan. The square-root unscented Kalman filter for state and parameter-estimation. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 3461–3464, Salt Lake City, Utah, May 2001. IEEE.

[57] Howard H. Wang. *Experiments in Semi-Autonomous Underwater Intervention Robotics*. PhD thesis, Stanford University, Stanford, CA 94305, November 1996. Also published as SUDAAR 693.

[58] Stefan B. Williams, Paul Newman, Gamini Dissanayake, and Hugh Durrant-Whyte. Autonomous underwater simultaneous localisation and map building. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1793–1798, San Francisco, CA, April 2000. IEEE.

[59] Dana R. Yoerger, Albert M. Bradley, Marie-Helene Cormier, William B. F. Ryan, and Barrie B. Walden. High resolution mapping of a fast spreading mid ocean ridge with the Autonomous Benthic Explorer. In *Proceedings of the 11th International Symposium on Unmanned Untethered Submersible Technology*, Durham, NH, August 1999. Autonomous Undersea Systems Institute.

[60] Suya You, Ulrich Neumann, and Ronald Azuma. Hybrid inertial and vision tracking for augmented reality registration. In *Proceedings of IEEE Virtual Reality*, pages 260–267, Houston, March 1999.

[61] J. Yuh, S. K. Choi, C. Ikehara, G. H. Kim, G. McMurty, M. Ghasemi-Nejhad, N. Sarkar, and K. Sugihara. Design of a semi-autonomous underwater vehicle for intervention missions (SAUVIM). In *Proceedings of the 1998 International Symposium on Underwater Technology*, pages 63–68. IEEE, April 1998.

[62] SAUVIM project page, http://www.eng.hawaii.edu/$\sim$asl/sauvim.html.