Stability and Robustness Analysis Tools for Marine Robot Localization and Mapping Applications

by

Brendan J. Englot

S.B. Mechanical Engineering
Massachusetts Institute of Technology, 2007

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2009

Signature of Author:_____
Department of Mechanical Engineering
May 8, 2009

Certified by: _____
Franz S. Hover
Assistant Professor of Mechanical and Ocean Engineering
Thesis Supervisor

Accepted by:_____
David E. Hardt
Graduate Officer, Department of Mechanical Engineering

Stability and Robustness Analysis Tools for Marine Robot Localization and Mapping Applications

by

Brendan J. Englot

Submitted to the Department of Mechanical Engineering
on May 8, 2009 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Mechanical Engineering

ABSTRACT

The aim of this analysis is to explore the fundamental stability issues of a robotic vehicle carrying out localization, mapping, and feedback control in a perturbation-filled environment. Motivated by the application of an ocean vehicle performing an autonomous ship hull inspection, a planar vehicle model performs localization using point features from a given map. Cases in which the agent must update the map are also considered. The stability of the marine robot controller and estimator duo is investigated using a pair of theorems requiring boundedness and convergence of the transition matrix Euclidean norm. These theorems yield a stability test for the feedback controller. Perturbations are then considered using a theorem on the convergence on the perturbed system transition matrix, yielding a robustness test for the estimator. Together, these tests form a set of tools which can be used in planning and evaluating the robustness of marine vehicle survey trajectories, which is demonstrated through experiment. An augmented A* kinodynamic path-planning algorithm is then implemented to search the control input space for the globally robustness-optimal survey trajectory.

Thesis Supervisor: Franz Hover
Title: Assistant Professor of Mechanical and Ocean Engineering

Acknowledgements

First and foremost I must thank Franz Hover for being a great mentor and advisor. I am most appreciative of the many hours you spent participating in technical discussions, offering creative ideas, reviewing my manuscripts, helping me debug my experiments, and of course for supporting my research and presenting me with this opportunity. I also thank John Leonard for being a wise and supportive co-PI, I benefitted greatly from your contributions to our HAUV brainstorming sessions. As the biggest contributors to my engineering education, I must thank Harry Asada, Neville Hogan, and David Trumper. I'm very lucky to have had you as instructors for my essential first courses on systems, control, estimation, and robotics.

Thanks to Jerome Vaganay of Bluefin Robotics for many hours of collaborative field testing in Boston Harbor. Thanks also to Andrew Patrikalakis for writing excellent DIDSON playback and data extraction software on a moment's notice. To Matt Greytak, thanks for allowing me to pester you on the topic of motion planning, you saved me many hours in the library with your insight. Josh Taylor, your quals anecdotes kept me fearful enough of the exam to keep me studying hard. Charlie Ambler, your enthusiasm and sense of humor has made 5-422 a great place to work. Hordur Johannsson and Michael Kaess, you have been great teammates and I look forward to continued collaboration with you. To the new folks, Lynn, Kyle, and Kyle, you're off to a great start and I know you are destined for a great future in research.

To my family, your unconditional love and support has made all of this possible, there isn't enough room on this page to fully express my love and appreciation. And finally, to Sherry, without the promise of seeing your beautiful face at the end of each day I don't think I ever could have passed the quals; may this be the first of many times that I acknowledge you.

Research was sponsored by the Office of Naval Research Grant
N00014-06-10043, monitored by Dr. T.F. Swean.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

For a robot to possess true autonomy, it must be endowed with several key functionalities. The robot must be capable of perceiving the world around it and using that perception to determine its own location. It must be able to interpret, organize, and store the information it obtains from its perception of the world. It must be able to make decisions about where to travel and how to travel there. And, as it will be argued in this analysis, the robot must be capable of tolerating disturbances that displace it from its intended path, it must understand when these disturbances pose the danger of instability, and it must remain robust to disturbances by choosing its actions carefully. The application of ship hull inspection has motivated a close look at the stability of an autonomous marine vehicle equipped with the first three capabilities mentioned above, with the goal of enabling it with the fourth.

1.1 Motivation: Autonomous Ship Hull Inspection

It is a goal of the US Navy to automate the procedure of ship hull inspection, a task that is typically carried out by a team of divers. Hull inspections are performed with the goal of searching for structural damages, including penetrations, cracks, and corrosion, the goal of observing and managing the buildup of sea life and biofouling (e.g., removing barnacles and other marine growth), the goal of performing routine maintenance, which includes the replacement of zinc anodes, and the most important goal of searching the hull for mines and security threats. In addition to the nominal challenge of achieving one hundred

percent coverage of a large marine structure and the dangers associated with finding and removing mines, many of the harbors in which these inspections occur contain murky water with low visibility, further complicating the hull inspection tasks.

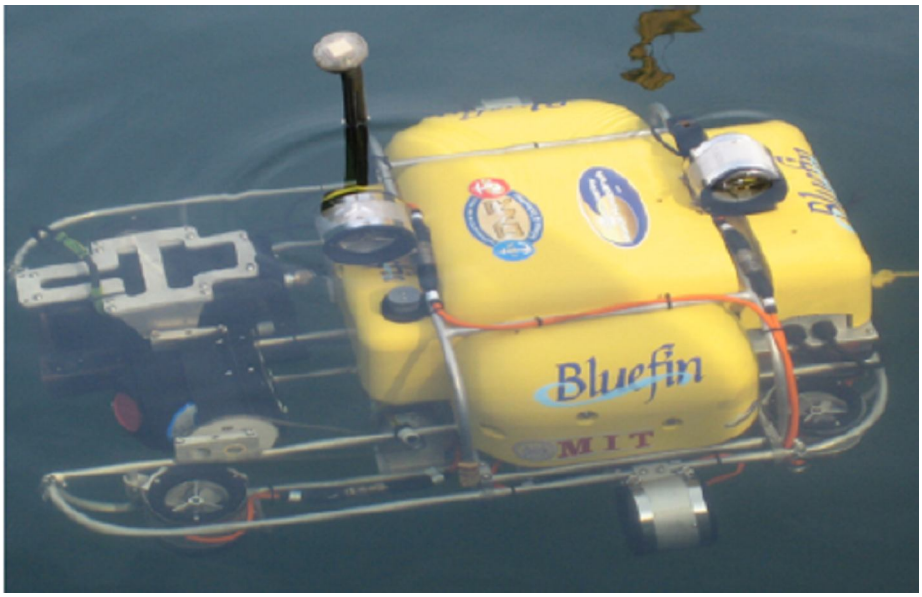

Figure 1-1: A photograph of the MIT-Bluefin Hovering Autonomous Underwater Vehicle (HAUV). Image provided courtesy of Bluefin Robotics.

Using an autonomous underwater vehicle (AUV) to perform hull inspections has been adopted as the preferred approach by the US Navy. To enable an AUV to perform this set of tasks, a number of sensing challenges must be overcome. Operation of an AUV near a large steel structure prevents a compass from being used to obtain heading. In addition, GPS, a reliable positioning sensor for surface, ground, and air vehicles, is not available underwater. Long baseline acoustic tracking, which uses a set of acoustic beacons mounted on the seafloor or a marine structure to accurately measure an AUV's position, won't work if the AUV is in close proximity to a ship hull, since the hull becomes an obstacle between the AUV and the baseline network. A very large baseline network would be needed to measure the vehicle's position on any side of the hull. A Doppler Velocimetry Log (DVL) is another commonly-used sensor for AUV navigation, which projects four acoustic beams onto a surface and measures range and velocity relative to the surface. In deep water, an AUV cannot project these beams onto the seafloor, and if it projects these beams onto a ship hull, it will periodically lose contact with the hull surface if it must navigate the sharp

corners and complex areas necessary to achieve one hundred percent coverage. The DVL is also prone to drift. Finally, an inertial measurement unit (IMU) is prone to drift, and operation over a long period of time will yield poor AUV position estimation. Given the various disadvantages of these sensors commonly used in underwater navigation, vision-based navigation methods offer a promising alternative (in addition to providing mine detection capability), and a vehicle has been developed which is capable of blending the aforementioned sensors with data obtained from vision.

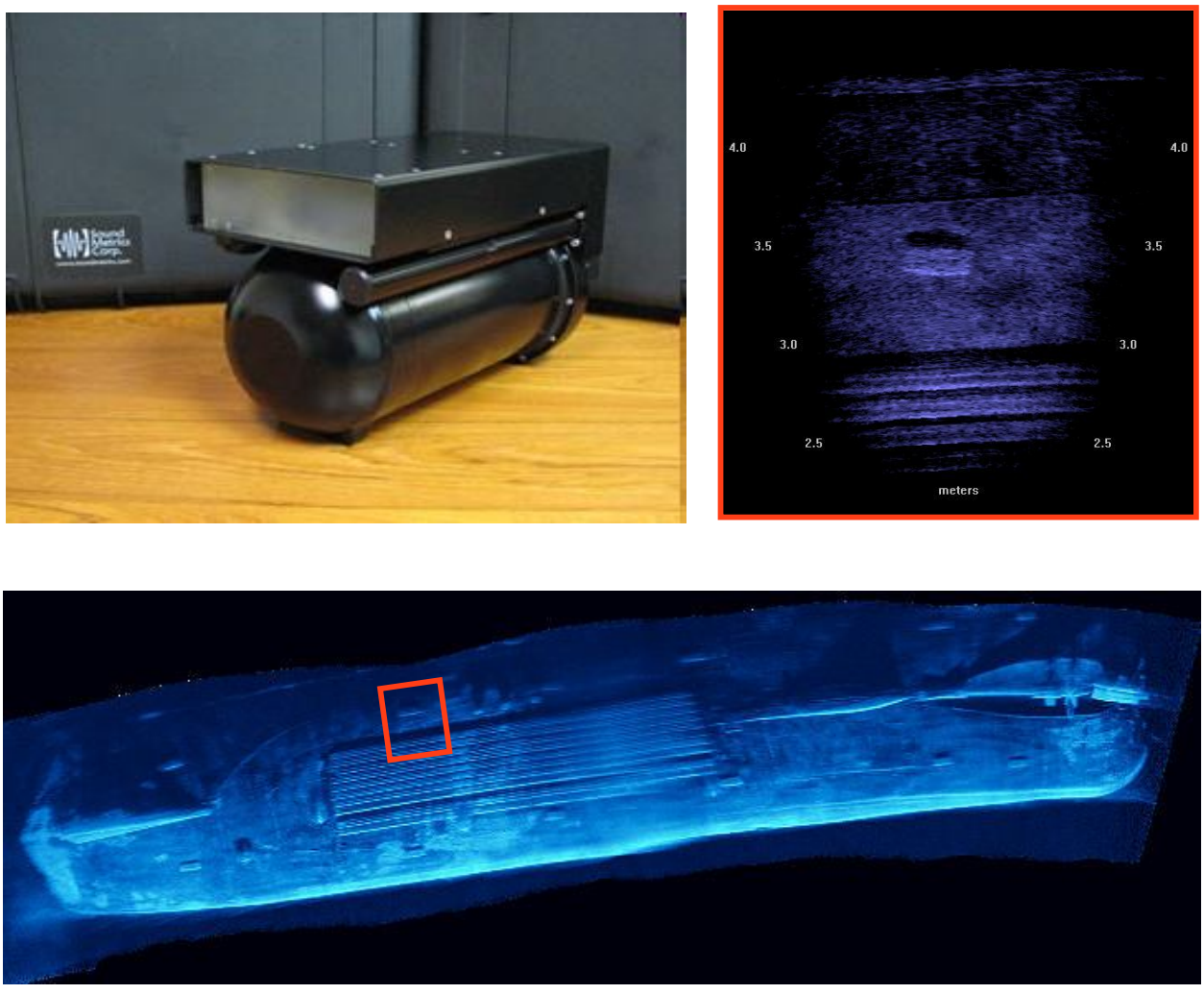

Figure 1-2: Clockwise from upper left: a photograph of a DIDSON, a single DIDSON image from a ship hull survey featuring cooling pipes and a zinc anode, and a mosaic of DISON images gathered from a complete hull survey, with the red box denoting the location of the single image). DIDSON photograph courtesy of Sound Metrics Corporation (http://www.soundmetrics.com/PRODUCTS/PR-3000m/didson_deepunits.html) and sonar image mosaic courtesy of AcousticView (http://www.acousticview.com).

1.1.1 Hovering Autonomous Underwater Vehicle (HAUV)

The vehicle prototype being developed for autonomous ship hull inspection is the Hovering Autonomous Underwater Vehicle (HAUV), a holonomic, hovering vehicle capable of actuation in any of its six degrees of freedom. The vehicle, pictured in Figure 1-1, was created by MIT and Bluefin Robotics and has successfully demonstrated many of the capabilities required for autonomous ship hull inspection [1], [2], [3]. To navigate during a ship hull survey it uses a combination of depth sensing, IMU, and DVL. Its vision, which at the current time is used only so the vehicle operator can look for mine-shaped features on the hull, is provided by a Sound Metrics Dual Frequency Identification Sonar (DIDSON). The DIDSON uses sonar to produce images similar in appearance to a visual image from a camera, as is depicted in Figure 1-2 [4], [5]. A key difference between the DIDSON and a camera is that a DIDSON can produce these images in any harbor environment, irrespective of water clarity. Although the DIDSON's current function is to allow a remote vehicle operator to detect mines, it can also be used to enable visual navigation by tracking an assortment of natural features found on a ship hull.

Aside from mines, a wide variety of features on a ship hull can be repeatedly recognized in DIDSON images (zinc anodes, inlets, the rudder, propeller, and shaft, and sometimes rivets and welds). DIDSON provides a range and bearing measurement associated with each pixel of each image, allowing the relative range and bearing between a feature and the HAUV to be measured repeatedly. Range-and-bearing measurement of a network of hull features, given a prior map of the hull containing these features, permits vehicle localization to be performed using vision as the only means of sensing. If a prior map is not available, then the vehicle can build a map of features while simultaneously using them to localize. In the analysis to follow, an estimation strategy of this type will be considered. Due to the random disturbances present in an ocean environment, a navigation process of this nature must be coupled with high-fidelity feedback control. By considering stability and robustness, this analysis will pursue improved understanding of the integrated localization, mapping, and dynamic control process.

1.2 Relevant Prior Work in Control, Estimation, and Motion Planning

Before considering the complexities of a process by which an autonomous vehicle plans its trajectory, employs feedback control to counteract disturbances, and simultaneously perceives the surrounding environment and its location within the environment, it is essential to consider fundamental work done in each of these separate areas. For a more comprehensive treatment of these topics, Thrun [6] provides a review of localization as well as simultaneous localization and mapping, and LaValle [7] and Choset [8] provide a review of motion planning algorithms.

1.2.1 Robot Localization

Localization is the estimation of a robot's pose (position and orientation). In mobile robotics research this problem is often interpreted more specifically as the task of estimating the pose of a robot relative to a given map of the surrounding environment. Sensors such as GPS and acoustic long baseline trivialize the task of localization, but if the robot can only use observations of the features in its surrounding environment to localize (e.g., trying to use sightings of nearby trees to navigate through a forest), then localization becomes a far richer problem. The localization problem is often divided into two categories, passive and active localization. Passive localization is purely an estimation process which exerts no control over the robot's motion, while active localization describes a process in which a robot is driven to minimize estimation error. Hereafter, the term localization will be used to refer only to passive localization for the purpose of discussing the control and estimation processes separately.

Kalman Filter Localization

The most common form of localization employs a Kalman filter or some variation thereof to estimate the robot's position and orientation states and sometimes also its velocity states. The Kalman filter is a Bayesian minimum variance estimator which is optimal in the sense

that it yields an estimate which minimizes mean square error. It relies on the assumption that both the evolution of the system state and the measurement of the system state are random processes, each consisting of a deterministic process based on a linear model added to an uncorrelated Gaussian random process. The Kalman filter is a recursive estimator, and thus only the most recent measurement and state estimate are needed to estimate the current state. The linearized Kalman filter (LKF) and extended Kalman filter (EKF) may be used when either the state evolution process or the measurement process includes a nonlinear deterministic model. Each filter employs a linearization of the nonlinear dynamics to estimate the system state. The estimate is no longer optimal once the dynamics are linearized, but both filters yield good results in many engineering applications. The extended Kalman filter and linearized Kalman filter algorithms will be presented in detail in Chapters 2 and 3, respectively, and a comprehensive derivation, analysis, and discussion of the Kalman filter may be found in Gelb [9]. The concept of using a Kalman filter to estimate robot pose was introduced by Smith and Cheeseman in 1986 [10], who recognized the potential for concurrently improving knowledge of the robot pose and the locations of the features being measured by the robot. These ideas were implemented by Leonard and Durrant-Whyte, who used the EKF for mobile robot localization using measurements of the geometric features in the surrounding environment [11].

Scan-Matching and Map-Matching

There are several other varieties of localization which have become popular in the mobile robotics community. The concept of localization by scan-matching, or comparing a range scan with previously obtained scans to estimate a robot's changing pose, was first introduced by Lu and Milios in 1994 [12]. The Iterative Closest Point algorithm (ICP), which was developed by Besl and McKay for matching 3D point clouds [13], was eventually applied to scan-matching localization tasks [14] and remains a popular algorithm for use in mobile robot localization and mapping [15], [16]. Another method of localization is map-matching, which uses occupancy grid maps rather than point clouds for comparison. An occupancy grid map is a discrete grid of the robot's environment which stores the

probability of an obstacle being present in each unit of the space. By matching a grid map of the global environment with a local grid that is instantaneously perceived by the robot, the robot's pose relative to the global map can be estimated [17].

Markov Localization

A noteworthy and very broad localization category is Markov localization, classified by Simmons and Koenig in 1995 [18]. Based on measurements of the surrounding environment and a given map, a robot propagates its belief, which is a probability density function representing the probability of being at each location on the map at the present instant in time. The updated belief resulting from each subsequent measurement is computed using Bayes' rule. The name of this localization strategy comes from the definition of a Markov process; a random process in which each state depends only on the state which came immediately before. Markov localization is independent of the state space chosen to represent the robot, and it can tolerate ambiguities in associating each measured feature with a feature on the map (e.g., if we observe a door in a hallway, we don't need to know whether it's door number one, two, or three). Kalman filter localization, although developed earlier, is a specialized version of Markov localization which always assumes a unimodal Gaussian belief distribution and represents the belief using the mean and covariance only. These restrictions prevent Kalman filter localization from tolerating feature association ambiguities. For this reason, it is also important that the robot pose is known when Kalman filter localization is initialized.

Grid and Monte Carlo Localization

Other popular specialized versions of Markov localization are grid localization and Monte Carlo localization. Grid localization discretizes the robot pose space and lumps belief probability density into discrete units using a histogram representation of the belief distribution [19]. Monte Carlo localization selects a random set of poses from the pose space to analyze (termed particles), and each particle is assigned a weight based on the probability of the most recent measurement given the pose of the particle [20]. Before the

next measurement is collected a new set of particles of equal weight is generated by randomly choosing each one (with replacement) from the set of previous particles, with probability proportional to the weights of the previous particles. This set of resampled particles comprises the belief distribution from which the pose estimate is obtained. In the analysis to follow, the Kalman filter is adopted as the localization vehicle of choice, as it can easily be included in the state space framework of a linear dynamic system implementing feedback control.

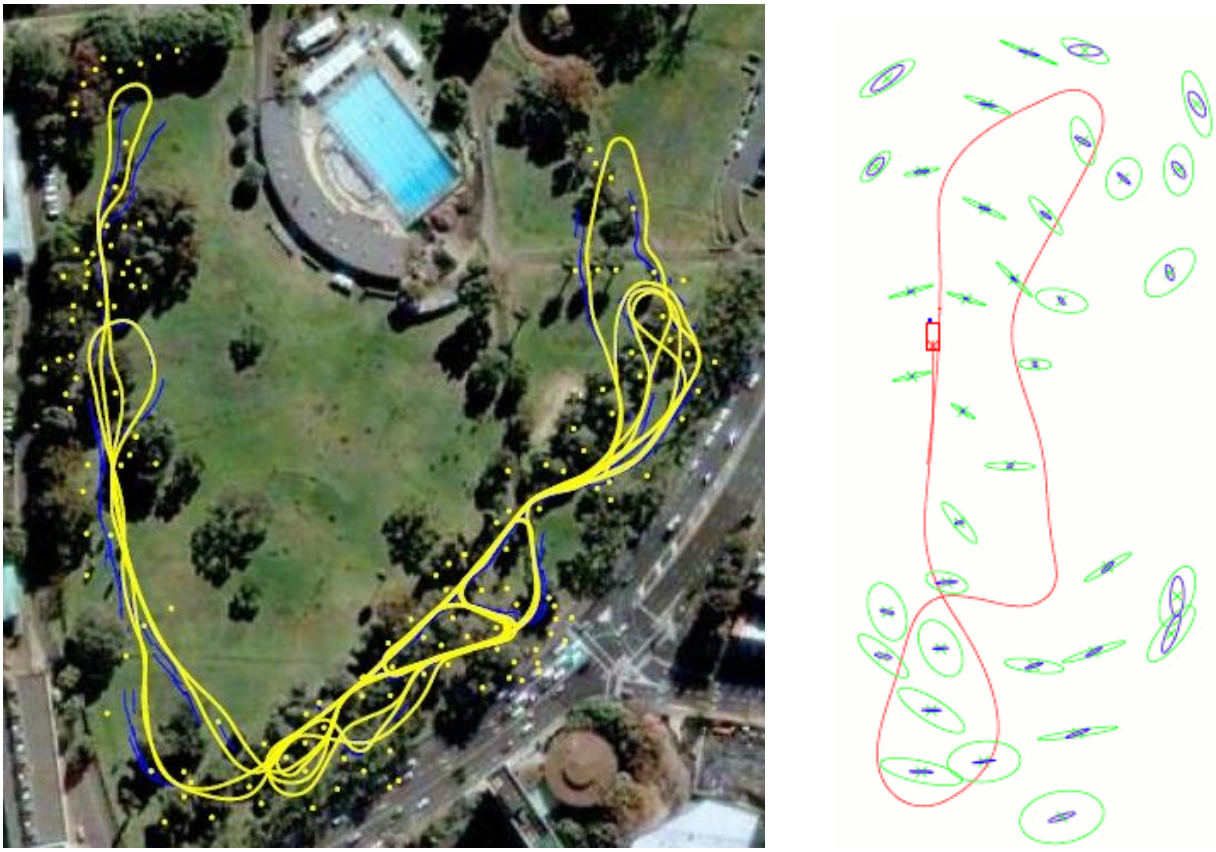

Figure 1-3: Examples of vehicle trajectories and maps constructed using SLAM. At left, feature and pose estimates generated using a data set from Victoria Park in Sydney, Australia. At right, a map which displays the error ellipses associated with each feature, indicating the relative confidence in each feature's estimated location. Images provided courtesy of Michael Kaess (http://www.cc.gatech.edu/~kaess/iSAM.html).

1.2.2 Simultaneous Localization and Mapping (SLAM)

Starting with the fundamental analysis of Smith and Cheeseman in 1986, the problems of localization and map-building have often been considered in tandem. For a robot to

possess true autonomy, it is often argued that it should be able to deploy in an unknown environment, build a map of its surroundings, and accurately understand its location on that map at all times. A foundational articulation of the SLAM problem using a Kalman filter was achieved by Smith, Self, and Cheeseman in 1990 when they showed that the estimates of the features observed by a robot grow increasingly correlated as the robot explores its environment [21]. This requires a SLAM estimation algorithm to possess a state vector containing every single one of the features on the map, with computation scaling as the square of the number of states. Fears of unbounded error growth were put to rest by Csorba, whose key results demonstrated convergence of the Kalman filter SLAM algorithm [22], [23]. It was demonstrated that errors in the estimates of features on the map ultimately converge, in the limit of many observations, to a lower bound determined by the error in the initial pose of the vehicle. Despite these convergence results, the quadratic computational complexity imposed by the necessary inversion of the Kalman filter error covariance matrix has motivated other approaches. By splitting a large map into submaps, Leonard and Feder improved the limitations of an EKF-SLAM framework [24]. Estimation algorithms other than the Kalman filter have also been used to improve the efficiency and accuracy of SLAM.

Graphical SLAM

A drawback of the EKF-SLAM algorithm is that errors in feature association cannot be undone, and cases where feature associations are ambiguous can be problematic (e.g., if the robot is looking at a door but cannot determine which door it's looking at, the Kalman filter will fail to converge). One solution to this has been to perform SLAM offline, once all of the data has been gathered. A graph is formed connecting each pose in the entire pose history of the robot (the precise pose is still unknown, but knowledge of the measurement gathered at each pose and the control command implemented between poses is sufficient to form the graph), and each pose is subsequently connected to each feature sighted at the time the pose was achieved (any errors in feature association are corrected later). A map and vehicle trajectory are then estimated over the entire pose history and measurement history executed by the vehicle. This technique was introduced by Lu and Milios in 1997,

who generated a graph of range scans associated with the robot pose history and found an optimal map and trajectory solution using maximum likelihood estimation [25].

Information Representation

More recent offline graphical SLAM implementations have used an information representation. This approach constructs the graph of pose and measurement history within the information matrix, which stores the connections between each pose in the entire pose history and each feature on the map [26]. This matrix is equivalent to the inverse of the estimation error covariance matrix, and so it must be inverted back in order to obtain estimates of the robot trajectory. A unique feature of the information matrix is that it can be sparsified by condensing feature information into the information connecting poses, and so the computation required to invert this sparse matrix and recover the pose history is only linear in the number of unknown poses. In addition, recovering estimates of each feature observed by the vehicle is linear in the number of poses from which the feature was sighted. Online filters using the information representation have been developed to achieve a middle ground between the EKF, which resolves measurements into a probability distribution at every measurement step, and the graphical approach, which is resolved after all data is acquired. The sparse extended information filter is one example of such a filter [27]. Only the current pose is maintained in the information matrix, past robot poses are removed and links to the features observed from those poses are transferred to neighboring features. The entire information matrix is not inverted at each measurement step; only a part of the matrix is used to produce a vehicle pose estimate and updated estimates of a few active features which are in close proximity to the vehicle. This results in efficient computation which is independent of the size of the state space.

Particle Filters

Finally, another noteworthy approach to SLAM is the use of particle filters. The most ubiquitous particle filter SLAM algorithm is FastSLAM, described by Montemerlo in 2002 [28]. FastSLAM can accommodate robots with nonlinear governing equations and pose

belief distributions can be multimodal, accounting for different belief outcomes due to differing feature association outcomes. The algorithm maintains a set of particles, where each particle consists of a pose belief distribution and an individual Gaussian distribution for each feature on the map. Given a new control command and measurement, a new pose is sampled for each particle from the updated conditional belief distribution and each feature estimate is updated using an EKF. Each particle is assigned an importance weight based on the probability of the most recent measurement given this sampled pose and the previous history of measurements. Then, the step of resampling occurs, in which a new set of particles is drawn (with replacement) from the current set. The probability of choosing each particle is set proportional to the importance weight. This means that particles with high importance weights are likely to be picked more than once when resampling occurs. This resampled belief distribution is used to generate the next pose estimate. Particle filter SLAM is the most adept at handling feature association ambiguities, it can run online, and computation scales logarithmically with the number of states. It relies on the fundamental principle that the belief distribution for the entire particle (i.e., the pose and the feature locations) can be factored into distributions representing the pose and each individual feature.

Despite the computational advantages of the information theoretic and particle filtering methods, the appealing simplicity of applying linear system stability theory to a SLAM problem has led to the Kalman filter formulation as our choice for the analysis to follow. A Kalman filter can easily be included in the state space framework of a linear dynamic system implementing feedback control, as it can be represented using a single observer gain which acts on the entire system state vector.

1.2.3 Integrated Localization, Mapping, and Control

Merging a SLAM estimator with a control law is a natural extension to the body of research discussed thus far, and it is required to close the loop on an autonomous vehicle. In the robotics community, the integration of SLAM and control has mainly been addressed with the question, “To which location should the vehicle head next?” Some research has addressed this question by controlling a robot to achieve optimal coverage of its

environment. A variety of algorithms have been developed which achieve complete coverage of an area while minimizing the path length or travel time required to do so [29]. Others have devised control strategies which reduce estimation uncertainty, which is commonly referred to as maximizing the information gain. Information is gained when a robot state's estimation error variance is reduced (i.e., a belief distribution becomes less uniform and closer to being concentrated at a single point). A greedy method of integrated SLAM and control to optimize information gain, which looks only one step ahead into the future, was implemented experimentally in 1999 by Leonard and Feder [30], and a multi-step look ahead method was explored by Huang in 2005 [31]. Other research has sought to achieve a compromise between coverage and information gain, and the search for the optimal balance between the two is a widely pursued subject in robotics research today [32], [33], [34].

Although answering the question of "where to head next" is essential to achieving true autonomy in a robotic vehicle, there is another aspect of the control problem which demands attention, and it is control in the context of using feedback to stabilize, manipulate, and reject disturbances in a dynamic system. Although many of the seminal experiments in localization and mapping have been performed using wheeled robot platforms, wheeled robots are not as susceptible as aerial robots or marine robots are to being knocked off course by random disturbances. In particular, because marine robots are subject to ocean waves, currents, and wakes from nearby marine vessels, a closer look at the stability margins of an integrated localization, mapping, and control process is warranted.

Stability Analyses

Despite the widespread use of SLAM, a limited body of work exists on the stability of the integrated localization, mapping, and dynamic control process. Stability of the linear Kalman filter in the specialized case of a one degree-of-freedom monobot was assessed by Vidal-Calleja, Andrade-Cetto, and Sanfeliu [35], and its observability by Andrade-Cetto & Sanfeliu [36], leading to the conclusion that the partial observability of the filter yields marginally stable estimation error dynamics. Hover analyzed the stability margins of a

localization estimator with closed-loop control for one degree-of-freedom and planar three degree-of-freedom vehicles with double integrator plants [37]. This analysis considered the regulation problem, using a constant-gain controller and estimator only. In the present work, the focus will be extended to a planar vehicle with time-varying controller and estimator gains, allowing travel anywhere in the 2-D plane.

Visual Servoing

Although the localization and mapping community hasn't thoroughly explored the dynamic stability of an autonomous vehicle that navigates using relative measurements of features in its environment, stability has been more rigorously investigated in the related community of visual servoing. Visual servoing is the task of using vision (often a single monocular camera) and feedback control to servo a robot manipulator to a desired configuration with high precision. It is frequently the case that the camera is in-hand, mounted on or close to the robot's end effector. Solutions to the visual servoing problem are typically divided into two categories, position based and image based visual servoing [38]. Both processes use image data to extract relative bearing measurements between the manipulator and features in the image. Position based visual servoing computes an estimate of the robot's pose in 3D task space, and the error signal acted on by the controller is in 3D space as well. A localization strategy similar to those discussed in Section 1.2.1 can be implemented to solve this problem. Image based visual servoing computes the error signal in the 2-D image, and this error in image space is mapped to robot actuator commands. Although there is less emphasis on probabilistic estimation in the field of visual servoing, stability and robustness of the closed-loop manipulator is a commonly-visited topic [39], [40]. In particular, an analysis of visual servoing stability and robustness using Lyapunov stability theory performed by Deng in 2002 serves as a good example of an examination from which a mobile robot performing localization, mapping, and feedback control would benefit [41].

1.2.4 Motion Planning via Graph Search

Localization and mapping aside, the pure task of motion planning, or planning the trajectory a robot will execute as it navigates through its environment, is a rich topic which has been studied extensively. Assuming a robot is perfectly capable of localizing and perceiving its environment, the question of how to maneuver from point A to point B can be answered in many ways. Most motion planning algorithms attempt to discretize and reduce in quantity the actions and states which can be used to describe the motion of a robot. If we were to allow the space of states and actions which could be occupied and executed by the robot to be continuous, then searching through all of the possible combinations of actions that will propagate the robot through its state space often proves a very difficult problem to solve for the optimal path. A common simplification is to represent the robot's state space using a graph, an assembly of straight-lined edges connected by nodes.

Dijkstra's Algorithm

Graphs are used to describe a wide variety of spatial networks, from roadways to electric circuits. The problem of using a graph representation to determine the shortest path from one location to another has long been of interest, and one of the most famous and widely-used graph search algorithms was described by E. W. Dijkstra in 1959 [42]. Dijkstra's algorithm requires each edge of the graph to have an associated cost describing the difficulty of travel between its two connecting nodes. On a system of roads, this cost might be proportional to the length of the road connecting two cities, or the time required to travel that road. Every path of edges one could travel along the graph has an associated cost-to-come, the sum of costs incurred by traveling the path. The goal of the algorithm is to find the path from a given start node to a given goal node which incurs the minimum cost-to-come. From the start node, all adjacent nodes (nodes separated from the start node by a single edge only) are placed in a queue, and they are ranked by their cost-to-come (the cost of traveling to these nodes from the start node). The node with the lowest cost-to-come is chosen first, and all of its adjacent nodes are consequently evaluated. The paths

leading to these new nodes from the start node and the respective total costs required to reach them are added to the queue. The node from which we performed this evaluation is now "popped", and we may no longer evaluate adjacent nodes from the location of this "dead" node for the rest of the algorithm. Next, the path on the queue with the lowest cost-to-come is selected, and the algorithm repeats. We visit the node associated with the lowest cost and begin adding its adjacent nodes to the queue. This occurs until a path is found that reaches the goal. When the first goal-reaching path is achieved, all non-goal-reaching paths with higher costs-to-come are subsequently eliminated from the queue. This step occurs again for each goal-reaching path found afterward, until it is finally the case that all remaining paths on the queue have a higher cost-to-come than the most recent goal-reaching path. Then, all goal-reaching paths are evaluated and the one with the lowest cost-to-come is selected as the optimal path.

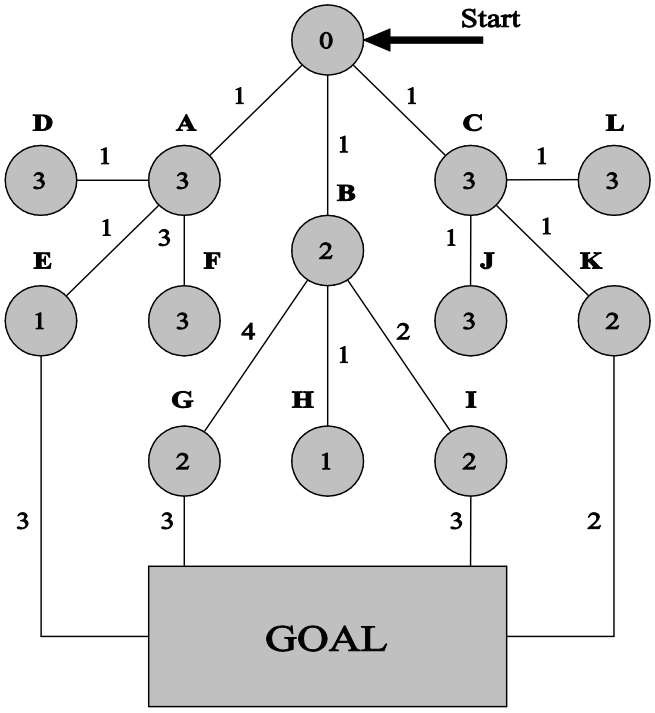

Figure 1-4: A graph which demonstrates the A* algorithm. Each edge is labeled with its associated cost of travel, and each and node is labeled with its heuristically formulated cost of reaching the goal from that node. From the perspective of any single node, the cost-to-come would consist of the sum of numbers along the edges used to travel to the node from the start position, and the cost-to-go would consist of the number written inside the node. Image provided courtesy of Howie Choset (http://www.cs.cmu.edu/~biorobotics/book).

A* Algorithm

By considering another type of cost that penalizes being at a node far from the goal (rather than just assigning penalty for being far from the start node), Dijkstra's algorithm has been augmented to converge to the optimal solution in a more efficient manner. The A* algorithm, introduced by Hart, Nillsson, and Raphael in 1968, evaluates paths along a graph by adding Dijkstra's cost-to-come to an additional cost which estimates the remaining effort required to travel to the goal from the node under consideration [43]. The use of this additional cost-to-go has been proven to find the optimal path if it is formulated using an admissible heuristic, or a procedure which always underestimates the true cost required to reach the goal from the node under consideration [44].

Kinodynamic Planning

Although many planning algorithms apply A* to graphs which represent grids in discretized 2D space, this approach neglects the dynamics of the agent that will carry out the path. This is suitable for robots which move slowly and are easily actuated, but many robots cannot be moved from one location in the state space to another without careful consideration of the relationship between commanded control inputs and robot motion. The problem of motion planning while obeying dynamic constraints on velocity and acceleration is known as kinodynamic planning. One of the earliest implementations of kinodynamic planning, achieved by Canny et al in 1988, considered a point-mass robot capable of a few discrete actions at any point in time; it could choose to apply a positive or negative acceleration of fixed magnitude for a fixed period of time along any of the three coordinate axes. A dynamic model of the point-mass was used to compute the state space configurations which would result from the discrete control actions, and a graph search algorithm was used to find the optimal combination of control actions for reaching the goal [45]. Choosing discrete control actions for more complex robotic systems can be challenging, and the concept of maneuver-based motion planning has addressed this [46]. For a robot with more degrees of freedom than a point-mass, carefully choosing a subset of maneuvers to use for planning can simplify the challenge of searching through a robot's

state space for the optimal trajectory.

1.3 Problem Statement: Identifying and Managing Vehicle Robustness

As briefly mentioned before, the objective of this analysis is to take a close look at the stability of an autonomous marine vehicle carrying out a fully dynamic localization, mapping, and control process. This will ultimately reveal how to plan trajectories that ensure robustness against random disturbances, and will also inform estimator and controller design. The motivation behind this analysis is autonomous ship hull inspection, and so all vehicle models considered are intended to represent the HAUV, although these models can be applied to any holonomic marine vehicle. Although the models will be simplified to some extent, they are intended to capture the essential aspects of the feature-based navigation and control problem. The measurement process produces range-and-bearing measurements of hull features and is a nonlinear function of the system states. Random noise influences both the measurement process and the dynamic control process.

Different aspects of the localization and mapping problem will be considered in the sections to follow. In Chapter 2, the basic map-building capability of the HAUV is explored, and a map is constructed with no prior knowledge of the number or configuration of features. In Chapters 3 and 4, feedback control is tied directly into the feature-based estimation loop, and a slightly more restrictive definition of mapping is used to enable a stability analysis. It is important to note that "mapping" is considered in the context of use and *refinement* of a map that is given *a priori*, and not the building of an entirely new map. Refinement of the map entails updating or correcting the features already present on the map, but does not include the addition of new features. Also considered is the case of map *exploitation*, in which the *a priori* map of known features is used for the sole purpose of localization, and is not updated or corrected. Chapter 3 will demonstrate that integrated map refinement and control can achieve uniform stability in the sense of Lyapunov using a theorem on the transition matrix Euclidean norm. A complementary theorem will be used to show that integrated map refinement and control can achieve uniform asymptotic stability. Chapter 4 considers the effect of perturbations on stability and introduces a

robustness performance metric. This metric can be used to evaluate the conditioning of the estimator, and in particular to evaluate the variation in robustness that results from a variation in the geometric pattern of features on the map. Experimental results which validate these predictions of geometry-dependent robustness will be presented. Chapter 5 discusses how this metric can be used in guiding the choice of vehicle survey trajectories, the first robustness-optimal form of robot motion planning. A modification is made to the A* algorithm to achieve this, and the robustness predictions are supported by experiment.

Chapter 2

Autonomous Underwater Map-Building in Real-Time

Before integrating the observation of features into a marine vehicle's feedback control process, the capability to detect features on a ship hull and use them to construct a map in real-time is demonstrated. At AUVfest 2008, an event organized for collective experiments and demonstrations of Navy-sponsored AUV technology, the goal of real-time autonomous map building was set for the MIT-Bluefin HAUV. The task was performed on the hull of the

Figure 2-1: A photograph of the USS Saratoga, which is currently retired and stationed at the Naval Undersea Warfare Center in Newport, Rhode Island. The rectangular box indicates the portion of the hull that was surveyed during the AUVfest 2008 real-time underwater mapping exercise. Photograph provided courtesy of DefenseImagery.mil (http://www.dodmedia.osd.mil/Assets/Still/1992/Navy/DN-ST-92-09908.JPEG).

USS Saratoga, a retired naval aircraft carrier. The HAUV performed hull relative surveys along a portion of the hull very similar in curvature and orientation to a flat vertical wall. The approximate region of the hull surveyed is indicated in Figure 2-1. The HAUV used its IMU, DVL, and depth sensor to localize along with layered proportional-integral-derivative (PID) control to correct perceived errors as it carried out the hull survey. While these standard hull survey processes were performed using the vehicle's internal computer, the feature extraction and map-building processes were carried out on a topside computer. The computer intercepted real-time image data from the DIDSON, odometry data from the DVL, and used these to produce real-time localization and map estimates. Two algorithms were run in series to produce these estimates, a feature extraction algorithm which identified the vehicle-relative range and bearing of mine-shaped training targets observed in DIDSON imagery, and an Extended Kalman Filter algorithm which recursively estimated the vehicle pose and velocity, and the location of each target on the hull.

2.1 Extended Kalman Filter

2.1.1 Dynamic Model

An EKF is employed to estimate the pose and velocity of the HAUV and to construct the map of hull features. Although the HAUV is a six degree-of-freedom vehicle subjected to hydrodynamic drag, a number of approximations are made to keep the vehicle model as simple as possible. First, a planar three degree-of-freedom vehicle model is used since the HAUV is surveying an approximately vertical wall at fixed range, fixed heading, and zero pitch. As a result, only the roll angle and *x*-*y* position are necessary to locate the vehicle and determine its orientation relative to the hull. In addition, the hull survey run on the Saratoga consists of horizontal and vertical straight-line trajectories only, with no planned variation in roll angle. For this reason, the vehicle's orientation in roll will be decoupled from its dynamics in *x* and *y*. In other words, it will always be assumed that propulsion in the sway direction corresponds to *x*-directed motion along the hull, and it will always be assumed that propulsion in the heave direction corresponds to *y*-directed motion along the hull. In reality the vehicle will be perturbed in range, heading, pitch, and roll, but because

the HAUV layered control acts to correct any errors in these degrees of freedom, they are assumed to be of constant magnitude as described above (although we will still estimate the roll angle for the purposes of monitoring angular perturbations). One final simplification is the approximation that the vehicle dynamics consists of double integrators in each of its three planar, uncoupled degrees of freedom (i.e., hydrodynamic drag is neglected). Since this model will not be used to compute a control action for the vehicle and is being used for estimation only, this assumption will not hinder performance of the algorithm. Given these assumptions, the HAUV dynamic equations, which use an Euler discretization, appear as follows:

$$\begin{bmatrix} u_{k+1} \\ v_{k+1} \\ \dot{\varphi}_{k+1} \\ x_{k+1} \\ y_{k+1} \\ \varphi_{k+1} \end{bmatrix} = \begin{bmatrix} I_{3x3} & 0_{3x3} \\ I_{3x3}\Delta T & I_{3x3} \end{bmatrix} \begin{bmatrix} u_k \\ v_k \\ \dot{\varphi}_k \\ x_k \\ y_k \\ \varphi_k \end{bmatrix} + \begin{bmatrix} I_{3x3} \\ 0_{3x3} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

$$\underline{x}_{v|k+1} = F\,\underline{x}_{v|k} + \Gamma\,\underline{w}_k \tag{2.1}$$

The body-referenced sway velocity, heave velocity, and roll rate are described by u, v, and $\dot{\varphi}$, respectively, and x, y, and φ represent the horizontal, vertical, and angular position of the vehicle relative to the ship hull. Process noise w_i, which is zero mean Gaussian white noise with diagonal covariance matrix Q, is applied in each degree of freedom. The notation $\underline{x}_v$ refers to a column vector which contains all six vehicle states. Although (2.1) does not explicity include the control commands sent to the vehicle by the HAUV layered control, this will not hinder the filter's ability to track the vehicle's trajectory in response to these control commands.

2.1.2 Aggregate State Vector

Features observed on the hull are approximated as point features defined by an x-y position. Because the EKF is used to estimate both the vehicle states and the feature locations, the feature locations must be included in the state vector. The features are

assumed to be permanently fixed to the ship hull and have no dynamics. The aggregate state vector containing both vehicle and feature states has the following structure:

$$\underline{x}_k = [\ \underline{x}_{v|k} \quad x_1 \quad y_1 \quad x_2 \quad y_2 \quad \ldots \quad x_n \quad y_n\ ]^T \tag{2.2}$$

The size of the system transition matrix *F* must be increased to accommodate these new states. Because the features are static, two rows and columns of zeros must be added to *F* for each new feature, with a single entry of magnitude one along the diagonal to propagate the constant value of each static feature state. Although the above state vector is depicted with *n* features, the number of features in the state vector changes over the course of the algorithm. The algorithm is initialized with zero features in the state vector, and new features are added as they are observed by the vehicle. Features are never removed from the state vector, regardless of the time passed since they were last sighted.

2.1.3 Measurement Model

The vehicle's measurement process, consisting of the measurement of a feature's range and bearing relative to the vehicle, is a nonlinear function of the system states, described by (2.3).

$$\begin{aligned} r_i &= \sqrt{(x_i - x)^2 + (y_i - y)^2} \\ b_i &= a\tan\left(\frac{\mathrm{y_i} - y}{x_i - x}\right) - \varphi \\ \underline{z}_k &= [\ u \quad v \quad \dot{\varphi} \quad r_1 \quad b_1 \quad \ldots. \quad r_m \quad b_m\ ]^T \\ &\quad + [\ v_u \quad v_v \quad v_{\dot{\varphi}} \quad v_{r_1} \quad v_{b_1} \quad \ldots. \quad v_{r_m} \quad v_{b_m}\ ]^T \end{aligned} \tag{2.3}$$

The relative range from the vehicle to feature *i* is given by r_i, the relative bearing to feature *i* is given by b_i, and the complete measurement vector for any sampling instant *k* is given by $\underline{z}_k$. Note that $\underline{z}_k$ also includes the sway velocity, heave velocity, and roll rate, which are obtained from the DVL. Let *m* represent the number of features observed by the vehicle at any given sampling instant (this can vary from zero to infinity). Added to each measurement is *v*, zero mean Gaussian white sensor noise with diagonal covariance matrix *R*. Because the measurement process is a nonlinear function of the system states, it must

be linearized in order to propagate the EKF estimation error covariance matrix. The measurement Jacobian *H* containing partial derivatives, and the values of these partial derivatives, is given by (2.4). The partial derivative of each entry in the measurement vector of (2.3) is taken with respect to each of the state variables in (2.2) to construct this matrix. It changes size from sampling instant to sampling instant depending on how many features are observed. At a minimum, when no features are observed *H* is a three-by-three identity matrix, receiving only the odometry measurements from the DVL.

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & . & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & . & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & . & 0 & 0 \\ 0 & 0 & 0 & \partial r_1/\partial x & \partial r_1/\partial y & 0 & \partial r_1/\partial x_1 & \partial r_1/\partial y_1 & 0 & 0 & . & 0 & 0 \\ 0 & 0 & 0 & \partial b_1/\partial x & \partial b_1/\partial y & -1 & \partial b_1/\partial x_1 & \partial b_1/\partial y_1 & 0 & 0 & . & 0 & 0 \\ 0 & 0 & 0 & \partial r_2/\partial x & \partial r_2/\partial y & 0 & 0 & 0 & \partial r_2/\partial x_2 & \partial r_2/\partial y_2 & . & 0 & 0 \\ 0 & 0 & 0 & \partial b_2/\partial x & \partial b_2/\partial y & -1 & 0 & 0 & \partial b_2/\partial x_2 & \partial b_2/\partial y_2 & . & 0 & 0 \\ . & . & , & . & . & . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & \partial r_m/\partial x & \partial r_m/\partial y & 0 & 0 & 0 & 0 & 0 & . & \partial r_m/\partial x_m & \partial r_m/\partial y_m \\ 0 & 0 & 0 & \partial b_m/\partial x & \partial b_m/\partial y & -1 & 0 & 0 & 0 & 0 & . & \partial b_m/\partial x_m & \partial b_m/\partial y_m \end{bmatrix}$$

$$\frac{\partial r_i}{\partial x} = \frac{(x - x_i)}{\sqrt{(x_i - x)^2 + (y_i - y)^2}} \qquad \frac{\partial r_i}{\partial x_i} = -\frac{\partial r_i}{\partial x}$$

$$\frac{\partial r_i}{\partial y} = \frac{(y - y_i)}{\sqrt{(x_i - x)^2 + (y_i - y)^2}} \qquad \frac{\partial r_i}{\partial y_i} = -\frac{\partial r_i}{\partial y}$$

$$\frac{\partial b_i}{\partial x} = \frac{(y_i - y)}{(x_i - x)^2 + (y_i - y)^2} \qquad \frac{\partial b_i}{\partial x_i} = -\frac{\partial b_i}{\partial x} \qquad (2.4)$$

$$\frac{\partial b_i}{\partial x} = \frac{(x - x_i)}{(x_i - x)^2 + (y_i - y)^2} \qquad \frac{\partial b_i}{\partial y_i} = -\frac{\partial b_i}{\partial y}$$

2.1.4 The EKF Algorithm

The various components of this model can now be used to produce an estimate of (2.2) based on the measurements of (2.3). The first step of the EKF algorithm is initialization. An initial guess of the vehicle state vector is required, and an initial error covariance matrix is also required. Because the correlations among the various vehicle states are not yet known, the initial covariance matrix is chosen to be diagonal, containing a variance for each

state derived from the presumed accuracy of the initial choice for that state. The algorithm then proceeds as follows:

1) Propagate the error covariance matrix P through the system dynamics (obtaining the *a priori* estimation error covariance):

$$P_{k+1|k} = FP_kF^T + \Gamma Q \Gamma^T \tag{2.5}$$

2) Propagate the state estimate through the system dynamics (obtaining the *a priori* state estimate):

$$\underline{\hat{x}}_{k+1|k} = F\,\underline{\hat{x}}_k \tag{2.6}$$

3) Compute the Kalman Filter gain:

$$K_{k+1} = P_{k+1|k}H_{k+1}^T(\underline{\hat{x}}_{k+1|k})[H_{k+1}(\underline{\hat{x}}_{k+1|k})P_{k+1|k}H_{k+1}^T(\underline{\hat{x}}_{k+1|k}) + R]^{-1} \tag{2.7}$$

4) Propagate the error covariance matrix P through the measurement dynamics:

$$P_{k+1} = [I - K_{k+1}H_{k+1}(\underline{\hat{x}}_{k+1|k})]P_{k+1|k} \tag{2.8}$$

5) Obtain a new measurement, and propagate the state estimate through the measurement dynamics (obtaining the *a posteriori* state estimate):

$$\underline{\hat{x}}_{k+1} = \underline{\hat{x}}_{k+1|k} + K_{k+1}[\underline{z}_{k+1} - h(\underline{\hat{x}}_{k+1|k})] \tag{2.9}$$

As the notation used in the algorithm indicates, the measurement Jacobian H_k is obtained by linearizing the measurement process about the *a priori* state estimate. H_k and $\underline{z}_k$ will vary in size throughout the algorithm depending on how many features are observed at each measurement step. The nonlinear measurement process from which H_k is derived is denoted by $h(\underline{\hat{x}})$. Although the above equations comprise the nominal EKF algorithm, a few special additions are necessary to accommodate the need to associate a feature observation with a specific feature on the map, and the need to introduce a new state into the state vector and covariance matrix at any point when a feature is observed for the first time.

2.1.5 Data Association Algorithm

When a feature is observed by the vehicle, it is important to understand whether this feature is new or already present on the map. If it's already present on the map, then the

specific feature in the state vector that the measurement corresponds to must be identified. This evaluation is performed using a three-step process. First, a heuristic estimate of the observed feature's *x*-*y* position is computed using the measurement and the most recent *a posteriori* state estimate. This procedure is described by (2.10):

$$\begin{aligned} x_i &= \hat{x} + r_i \cos(b_i + \hat{\varphi}) \\ y_i &= \hat{y} + r_i \sin(b_i + \hat{\varphi}) \end{aligned} \tag{2.10}$$

After this estimate is computed, a weighted Euclidean distance is computed between the measured feature and each feature currently on the map. This weighted distance is given by (2.11):

$$r_{assocation} = \sqrt{[W_x(x_i - \hat{x}_j)]^2 + [W_y(y_i - \hat{y}_j)]^2} \tag{2.11}$$

The final step is to identify the map feature which yields the smallest weighted Euclidean distance, and this is the map feature most closely associated with the measured feature. If the minimum weighted Euclidean distance is less than a chosen threshold, it is concluded that the new measurement is an observation of the feature with the winning association. If the minimum distance is larger than the threshold, then the measurement is declared to be the measurement of a new feature, and this feature is initialized into the state vector and error covariance matrix.

2.1.6 Feature Initialization Algorithm

A special procedure must be used to introduce a newly sighted feature into the error covariance matrix. First, the *a priori* error covariance matrix $P_{k+1|k}$ is extended in size to include two new rows and two new columns for the new feature that was just observed. This preparatory matrix will be denoted $P^*_{k+1|k}$, which is given by (2.12). The notation used

$$P^*_{k+1|k} = \begin{bmatrix} P^{vv}_{k+1|k} & P^{vf}_{k+1|k} & 0 \\ P^{vf}_{k+1|k}{}^T & P^{ff}_{k+1|k} & 0 \\ 0 & 0 & R \end{bmatrix} \tag{2.12}$$

in (2.12) indicates that prior to the addition of new rows and columns, the upper left corner of $P_{k+1|k}$ contains the covariance matrix for the vehicle states, the lower right corner

contains the covariance matrix for the feature position states, and the upper right and lower left corners contain the correlations between the feature states and the vehicle states. This notation is consistent with the aggregate state vector defined in (2.2). The bottom right corner where the newly added rows and columns intersect contains the sensor noise covariance matrix *R*, which stores the noise distribution information for the range and bearing measurement processes. Another two rows and columns and another *R* matrix would need to be inserted into $P^*_{k+1|k}$ for each additional feature being initialized. This preparatory covariance matrix is then propagated through the heuristic feature position estimation equations defined in (2.10). Since (2.10) is a nonlinear function of vehicle state estimates and feature measurements, a Jacobian must be assembled, which appears as follows:

$$\underline{g}(\underline{\hat{x}}, \underline{z}) = [\ x_1 \quad y_1 \quad \ldots \quad x_m \quad y_m\ ]^T \tag{2.13}$$

$$\nabla g = \left[\begin{array}{cccccc|ccccc|ccccc}
\frac{\partial x_1}{\partial \hat{u}} & \frac{\partial x_1}{\partial \hat{v}} & \frac{\partial x_1}{\partial \hat{\varphi}} & \frac{\partial x_1}{\partial \hat{x}} & \frac{\partial x_1}{\partial \hat{y}} & \frac{\partial x_1}{\partial \hat{\varphi}} & \frac{\partial x_1}{\partial \hat{x}_1} & \frac{\partial x_1}{\partial \hat{y}_1} & \cdot & \frac{\partial x_1}{\partial \hat{x}_n} & \frac{\partial x_1}{\partial \hat{y}_n} & \frac{\partial x_1}{\partial r_1} & \frac{\partial x_1}{\partial b_1} & \cdot & \frac{\partial x_1}{\partial r_m} & \frac{\partial x_1}{\partial b_m} \\
\frac{\partial y_1}{\partial \hat{u}} & \frac{\partial y_1}{\partial \hat{v}} & \frac{\partial y_1}{\partial \hat{\varphi}} & \frac{\partial y_1}{\partial \hat{x}} & \frac{\partial y_1}{\partial \hat{y}} & \frac{\partial y_1}{\partial \hat{\varphi}} & \frac{\partial y_1}{\partial \hat{x}_1} & \frac{\partial y_1}{\partial \hat{y}_1} & \cdot & \frac{\partial y_1}{\partial \hat{x}_n} & \frac{\partial y_1}{\partial \hat{y}_n} & \frac{\partial y_1}{\partial r_1} & \frac{\partial y_1}{\partial b_1} & \cdot & \frac{\partial y_1}{\partial r_m} & \frac{\partial y_1}{\partial b_m} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdot & \vdots & \vdots & \vdots & \vdots & \cdot & \vdots & \vdots \\
\frac{\partial x_m}{\partial \hat{u}} & \frac{\partial x_m}{\partial \hat{v}} & \frac{\partial x_m}{\partial \hat{\varphi}} & \frac{\partial x_m}{\partial \hat{x}} & \frac{\partial x_m}{\partial \hat{y}} & \frac{\partial x_m}{\partial \hat{\varphi}} & \frac{\partial x_m}{\partial \hat{x}_1} & \frac{\partial x_m}{\partial \hat{y}_1} & \cdot & \frac{\partial x_m}{\partial \hat{x}_n} & \frac{\partial x_m}{\partial \hat{y}_n} & \frac{\partial x_m}{\partial r_1} & \frac{\partial x_m}{\partial b_1} & \cdot & \frac{\partial x_m}{\partial r_m} & \frac{\partial x_m}{\partial b_m} \\
\frac{\partial y_m}{\partial \hat{u}} & \frac{\partial y_m}{\partial \hat{v}} & \frac{\partial y_m}{\partial \hat{\varphi}} & \frac{\partial y_m}{\partial \hat{x}} & \frac{\partial y_m}{\partial \hat{y}} & \frac{\partial y_m}{\partial \hat{\varphi}} & \frac{\partial y_m}{\partial \hat{x}_1} & \frac{\partial y_m}{\partial \hat{y}_1} & \cdot & \frac{\partial y_m}{\partial \hat{x}_n} & \frac{\partial y_m}{\partial \hat{y}_n} & \frac{\partial y_m}{\partial r_1} & \frac{\partial y_m}{\partial b_1} & \cdot & \frac{\partial y_m}{\partial r_m} & \frac{\partial y_m}{\partial b_m}
\end{array}\right]$$

Note that the entries of $\underline{g}$ are given by x_i and y_i as defined in (2.10), and they correspond to the new features observed in the most recent measurement step. The Jacobian of (2.13) contains the partial derivatives of every entry in the vector $\underline{g}$ with respect to the estimate of every state in the aggregate state vector (2.2) (including features already on the map). The number of features on the map prior to this initialization step is denoted by *n*, and the number of features being initialized is denoted by *m*. As it turns out, many of the entries of this initialization Jacobian are zero, since (2.10) is not a function of any of the vehicle velocity states or current feature states. Below is the initialization Jacobian with the values of all partial derivatives included:

$$\nabla g = \left[\begin{array}{cccccc|cc|cc} 0 & 0 & 0 & 1 & 0 & -r_1 \sin(b_1+\hat{\varphi}) & 0 & 0 \; . \; 0 & 0 & \cos(b_1+\hat{\varphi}) & -r_1 \sin(b_1+\hat{\varphi}) & . & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & r_1 \cos(b_1+\hat{\varphi}) & 0 & 0 \; . \; 0 & 0 & \sin(b_1+\hat{\varphi}) & r_1 \cos(b_1+\hat{\varphi}) & . & 0 & 0 \\ . & . & . & . & . & . & . & . \; . \; . & . & . & . & . & . & . \\ 0 & 0 & 0 & 1 & 0 & -r_m \sin(b_m+\hat{\varphi}) & 0 & 0 \; . \; 0 & 0 & 0 & 0 & . & \cos(b_m+\hat{\varphi}) & -r_m \sin(b_m+\hat{\varphi}) \\ 0 & 0 & 0 & 0 & 1 & r_m \cos(b_m+\hat{\varphi}) & 0 & 0 \; . \; 0 & 0 & 0 & 0 & . & \sin(b_m+\hat{\varphi}) & r_m \cos(b_m+\hat{\varphi}) \end{array}\right] \tag{2.14}$$

Once the initialization Jacobian is obtained, the *a priori* covariance matrix is updated to include the newly observed features:

$$P_{k+1|k} = \nabla g_{k+1} P^*_{k+1|k} \nabla g^T_{k+1} \tag{2.15}$$

A discrete time index is now assigned to $\underline{g}$ to emphasize that it varies in time from measurement step to measurement step (and when no new features are sighted, $\underline{g}$ is null). The newly initialized error covariance matrix yielded by (2.15) can now be used to compute the Kalman filter gain that operates on the newest measurement (this measurement vector contains the newly sighted features). This feature initialization procedure and also a special procedure for delayed feature initialization (useful when the feature extraction algorithm returns occasional false alarms) are described in detail by Williams and Durrant-Whyte [47].

2.2 Feature Detection Algorithm

Although the EKF can achieve simultaneous localization and mapping given range-and-bearing measurements of hull features, extracting these measurements from a DIDSON frame requires an image processing algorithm. Each DIDSON frame contains an image matrix, and each entry of the matrix, which consists of an intensity value, corresponds to a vehicle-relative range and bearing identified by the row and column of the entry. If a color spectrum is assigned to the intensity values in the image matrix, the image will appear similar to the example in Figure 2-2, which contains one of the mine-shaped training targets used on the USS Saratoga.

The goal of the algorithm is to identify these mine-shaped training targets and approximate their location by designating a single entry of the image matrix to represent each target. The range and bearing values associated with this entry are approximated as the vehicle-relative range and bearing of the training target. To identify the targets,

knowledge of their intensity signature in DIDSON imagery is used, specifically the high intensity returned by the target itself and the low intensity returned by the target's "shadow". The shadow represents a region from which no acoustic beams were returned, which is due to the obstruction created by the target. The significant contrast between the shadow and the surrounding image data, especially the contrast between the shadow and the target itself, can be exploited to produce an algorithm which runs fast enough to allow mapping to be executed in real-time.

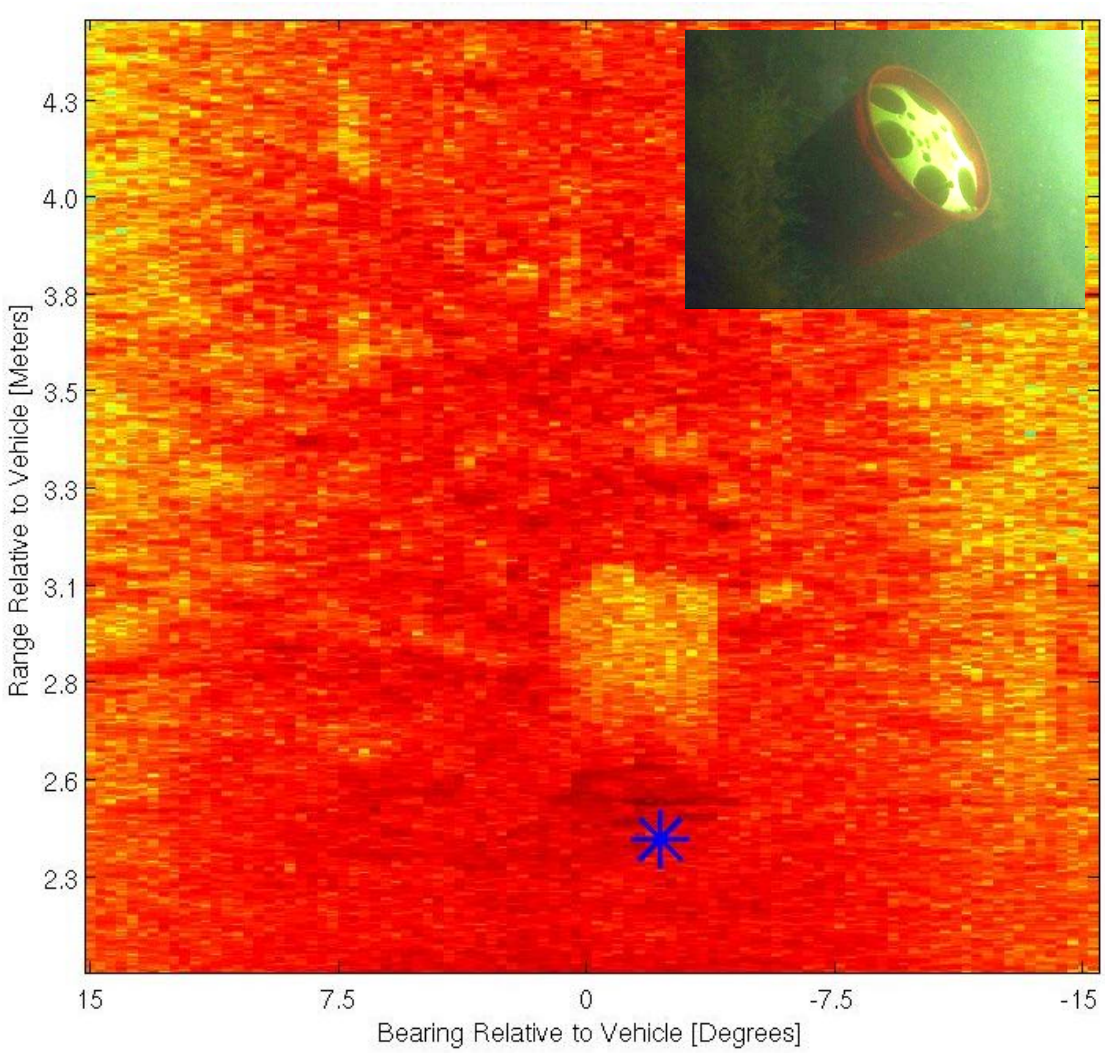

Figure 2-2: Raw DIDSON data from the USS Saratoga plotted in MATLAB, with a range of yellow and red colors associated with the intensity values of the image matrix. The vehicle-relative range and bearing corresponding to each matrix entry is indicated on the axes. The location of the mine-shaped training target is marked with a blue asterisk. Accompanying the DIDSON data is a photograph of a target planted on the hull of the Saratoga.

First, a rectangular section of the image matrix is selected for review. The algorithm begins by choosing a section in the upper left corner of the image matrix (with the matrix orientation identical to that displayed in Figure 2-2). The section is sized with the intent of surrounding a training target shadow with some room left around the edges. A smaller rectangular section is selected from the center of the large section and set aside separately (selected with the intention of falling completely within the shadow). The entries that comprise the small section are replaced with zeros inside the large section, and all of the non-zero values are stored (hopefully these values correspond to the border around the shadow). The mean intensity value of the small section is subtracted from the mean intensity value of the border values from the large section, and this difference is stored, along with the location of the image section that was just evaluated. What this algorithm obtains is a mean contrast in intensity between a small rectangular section of the image matrix and the ring that immediately surrounds it. This process is repeated iteratively, and the next large section is chosen slightly to the right of the first. Once the right edge of the image matrix is reached, the algorithm moves back to the left side, and shifts slightly beneath the location of the section that was evaluated first. Eventually the entire image is covered, and the vertical and horizontal step size between matrix sections evaluated can be varied to tune the speed and accuracy of the algorithm.

Once an entire image matrix has been evaluated, all intensity difference values gathered from the matrix are normalized by the root mean square of this set of values. This way, a similar intensity threshold can be used for images of different mean intensity. This is important since the mean intensity of an image is likely to vary as the robot is perturbed. Small perturbations can influence the projection of DIDSON beams onto the hull and alter the mean intensity of the image. After some experimental tuning, an intensity threshold is set for the normalized intensity difference values, and in any image where an intensity difference value exceeds the threshold, the maximum value within the image is declared to correspond to the location of a training target. The rectangular section of the image matrix which produced the winning value is revisited, and the matrix entry corresponding to the front and center of this matrix section, likely to be the front of the shadow where the actual target is located, is chosen to correspond to the range and bearing of the target.

This algorithm must be tuned according to the orientation of the DIDSON relative to the hull. If the DIDSON is pitched so that it views the portion of the hull beneath the vehicle, the rectangular shadows will be located beneath the targets. If the DIDSON is pitched so that it views the portion of the hull above the vehicle, the rectangular shadows will be located above the targets. For the experiments performed at AUVfest, the latter situation was the case, and the bottom center entry of the winning image matrix was selected as the location of the training target. The feature extraction algorithm is the major computational burden of the HAUV mapping algorithm, but is nonetheless capable of allowing real-time localization and mapping to run at about 3 Hz.

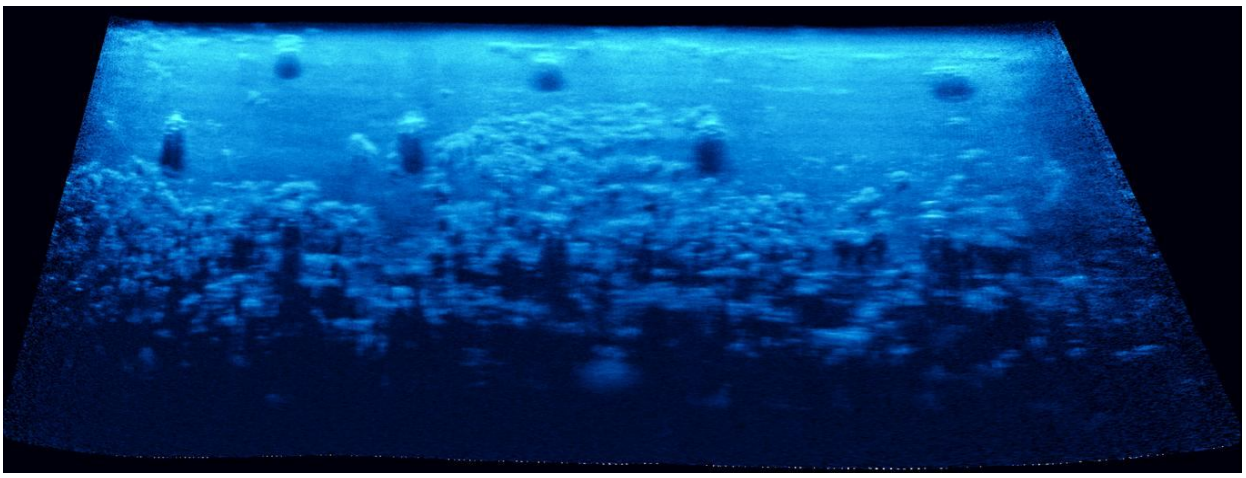

Figure 2-3: A mosaic of DIDSON frames displaying the set of mine-shaped training targets planted on the USS Saratoga for the AUVfest2008 real-time mapping exercise. Because the retired vessel resides at NUWC year-round, a significant amount of marine growth is present on the hull and visible in this image. Because the bottom row of three targets is almost completely obstructed by marine growth, only the upper six were used in the mapping exercise. Mosaic image provided courtesy of AcousticView (http://www.acousticview.com).

2.3 Experimental Results

Real-time localization and mapping was performed on three surveys of the Saratoga. Each survey covered a section of the hull approximately 12m by 2m in size, on the region of the ship indicated in Figure 2-1. Nine of the mine-shaped training targets were mounted on this section of the hull; a sonar mosaic displaying the layout of the targets underwater is given by Figure 2-3. Biofouling on the hull hindered the visibility of three targets, and so only six were used in the real-time localization and mapping exercise. Because the horizontal spacing of the targets is larger than their vertical spacing, a weight of two was

chosen for W_y of (2.11) and a weight of one was chosen for W_x. This amplified the vertical separation of the features to aid the data association process. Other parameters used in the experiment are listed in table 2-1.

Table 2.1: Parameters Used for Real-Time EKF-SLAM at AUVfest 2008

DIDSON Viewing Window Parameters:	Min Visible Range: 2m	Max Visible Range: 7m	Min/Max Heading: ± 15 degrees
Feature Extraction and Data Association Parameters:	Feature Range Threshold for Data Association: 1.8 m	Normalized DIDSON intensity difference threshold for feature detection: 5	
Vehicle Process Noise Parameters:	Process noise variance, surge: 0.01 $(m/s^2)^2$	Process noise variance, sway: 0.01 $(m/s^2)^2$	Process noise variance, roll: 10^{-12} $(rad/s^2)^2$
Odometry Sensor Noise Parameters:	Sensor noise variance, surge odometry: 10^{-4} $(m/s)^2$	Sensor noise variance, sway odometry: 10^{-6} $(m/s)^2$	Sensor noise variance, roll odometry: 0.04 $(rad/s)^2$
Feature Detection Noise Parameters:	Sensor noise variance, vehicle-relative range: 0.1 $(m)^2$	Sensor noise variance, vehicle-relative bearing: 0.1 $(rad)^2$	

Figure 2-4 demonstrates the performance of the feature extractor during the AUVfest 2008 experiments. The blue asterisks mark the feature extractor's predictions of where the training targets are located, and in these areas it is clear that the index used by the feature extractor was much higher in the region of the target than in other parts of the image. The feature extractor was tuned conservatively, which yielded very few false alarms (i.e., declaring a feature to be a training target when it is not) at the expense of fewer target sightings. The most frequently-sighted targets were observed thirty to forty times during one survey, while the least frequently-sighted targets were observed only two or three times. Although localization was performed concurrently with the mapping process during the hull surveys, the sparsity of feature sightings required the localization process to depend heavily on vehicle odometry from the DVL and depth sensor.

Results from the three hull surveys are displayed in Figures 2-5, 2-6, and 2-7. The EKF-estimated vehicle trajectory is plotted alongside the dead-reckoned trajectory computed internally by the HAUV, and it can be observed that the two trajectory estimates are quite similar. The maps constructed by the algorithm are also included in these figures, with red asterisks marking the estimated feature locations. Ninety-five percent confidence

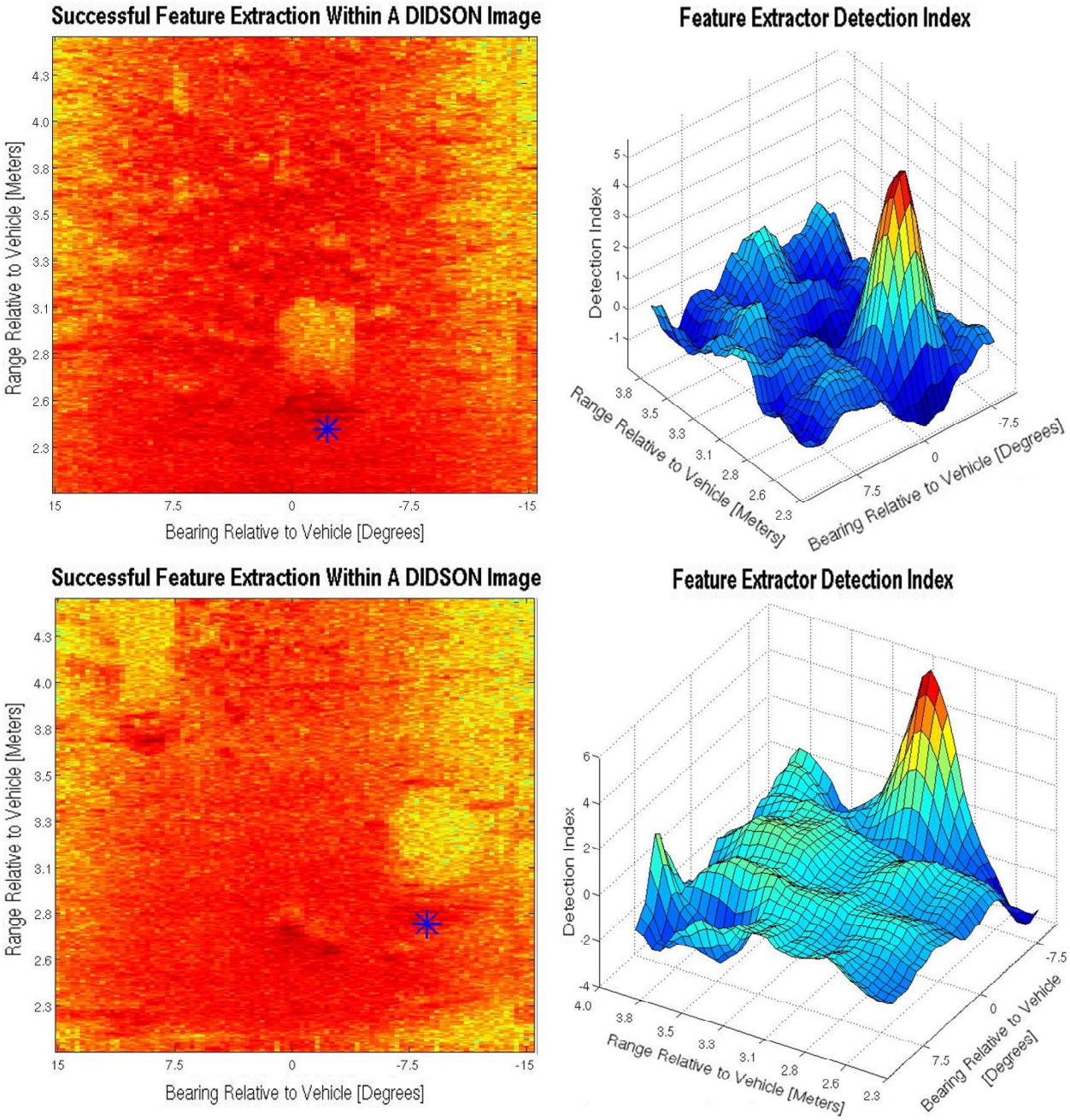

Figure 2-4: Performance of real-time feature extractor is demonstrated using two example DIDSON frames. The raw data is presented at left, and the feature extractor detection index for each rectangular quadrant of the image that was processed is displayed at right. Areas where features were identified (indicated by the blue asterisk) correspond to high peaks in the feature detection index.

ellipses derived from the error covariance matrix are plotted around each feature on the map. Features with small ellipses were sighted many times, yielding higher confidence in the feature estimate. In the cases of features sighted thirty to forty times, the filter's confidence in the feature estimate yielded an ellipse of about 0.5m in diameter. Features

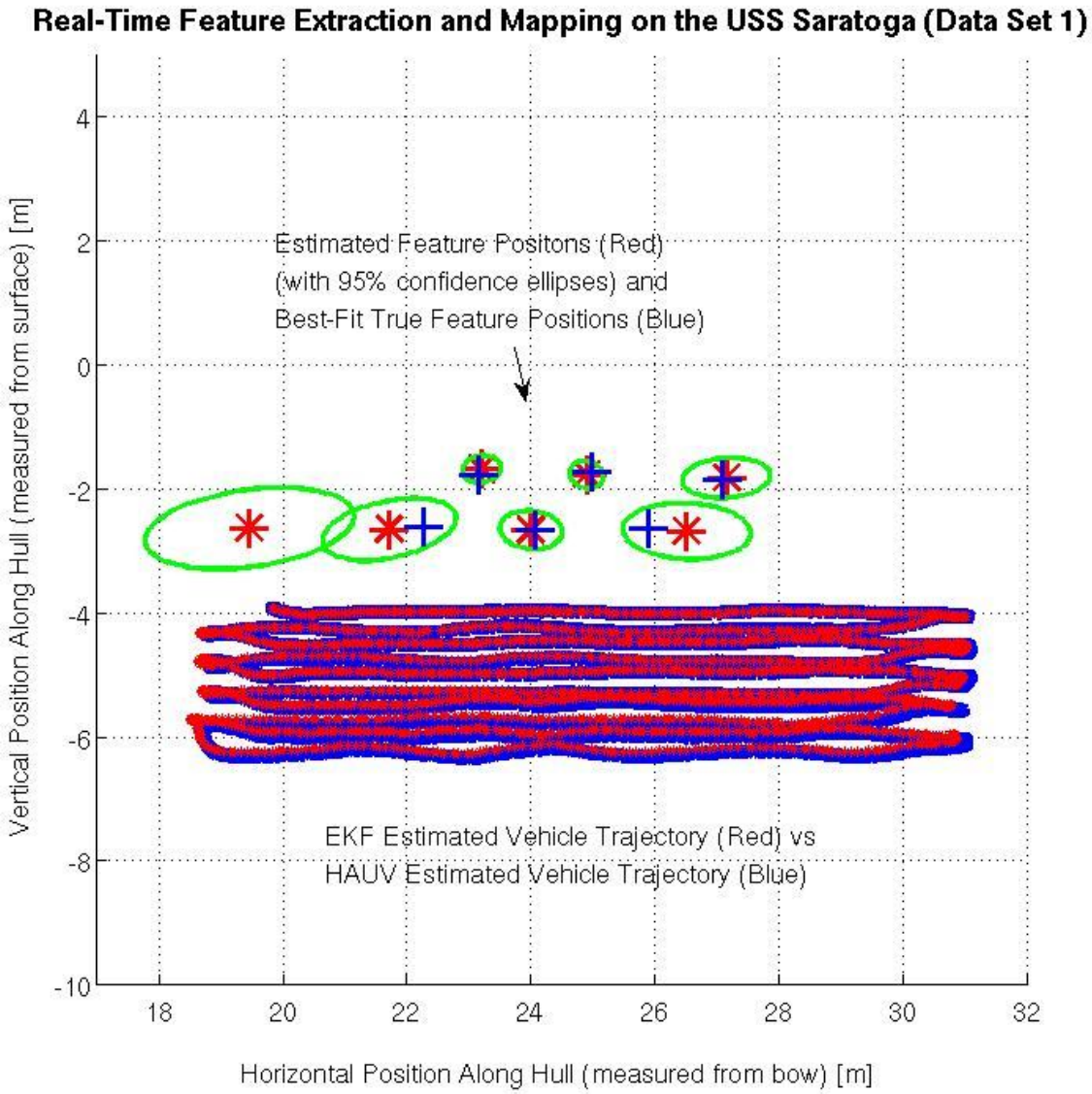

Figure 2-5: The first of three data sets produced from surveying the USS Saratoga. One false alarm was obtained during this survey, corresponding to the feature with the large error ellipse (which was sighted only once).

with the largest ellipses were sighted only a handful of times, with confidence ellipses as large as 2m in size. During each of the first two trials, the mapping algorithm picked up a single false alarm. This was due to the fact that certain marine growth features on the hull were very similar in shape to the training targets. Because the confidence ellipses associated with these erroneous features were large in size (about 4m in width), it was clear from viewing the map that these features should be discarded. The true locations of the training targets were approximated by associating six points with target locations pictured in the sonar mosaic of Figure 2-3. The absolute location of these six points was

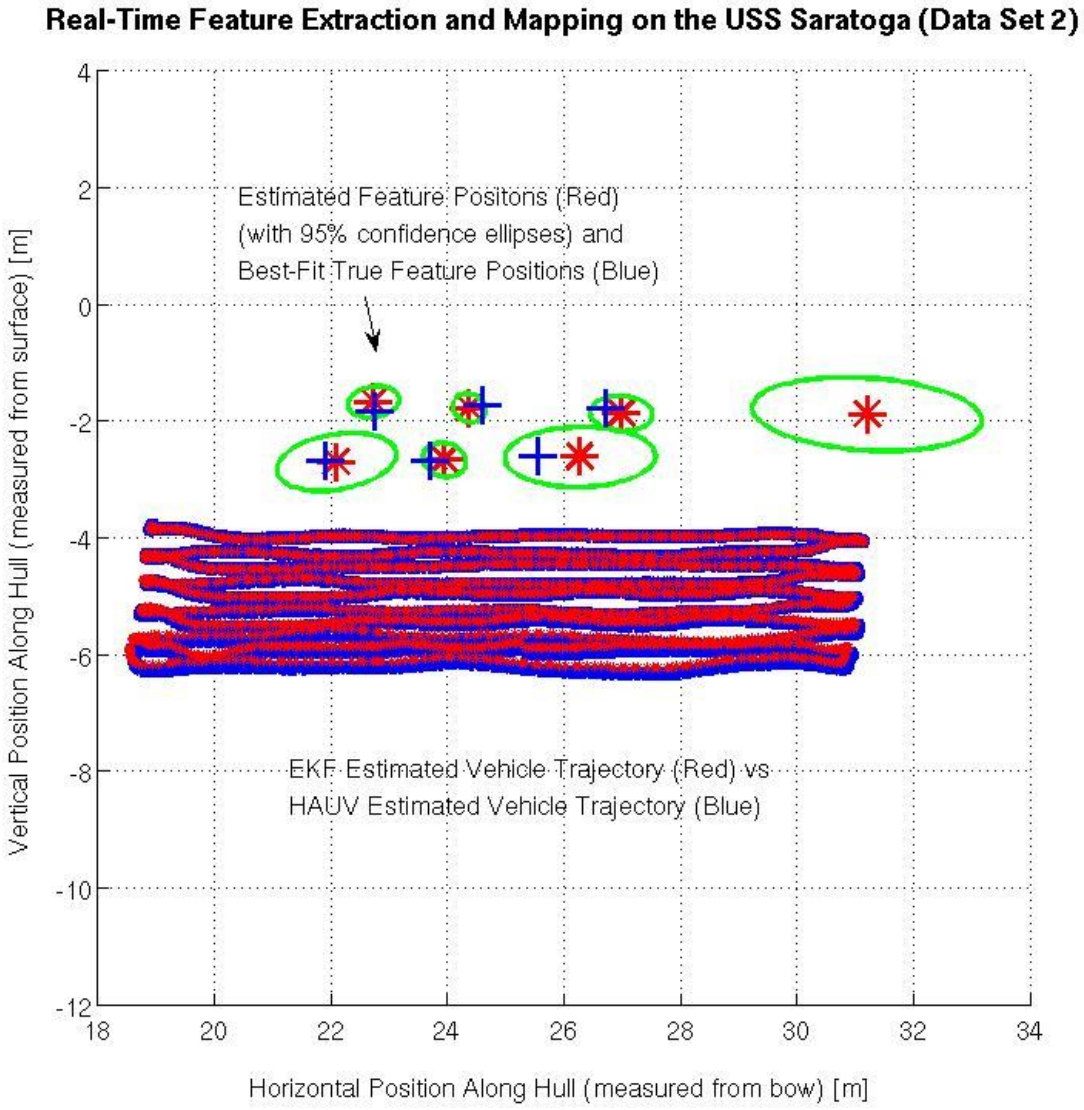

Figure 2-6: The second of three data sets produced from surveying the USS Saratoga. Similar to data set 1, one false alarm was obtained during this survey, corresponding to the feature with the large error ellipse (which was sighted only once).

not known, but their relative locations were, and so these six points were fit to the mapping algorithm's final feature position estimates using the 2-D iterative closest point algorithm. This algorithm produced the rotation and translation that would yield the closest fit between the feature estimates and the approximate true feature locations. When this fit was performed, all six features fell within the boundaries of the mapping algorithm's ninety-five percent confidence ellipses. The successful completion of this real-time mapping exercise demonstrated the potential of feature-based navigation to aid in mine detection and autonomous ship hull inspection.

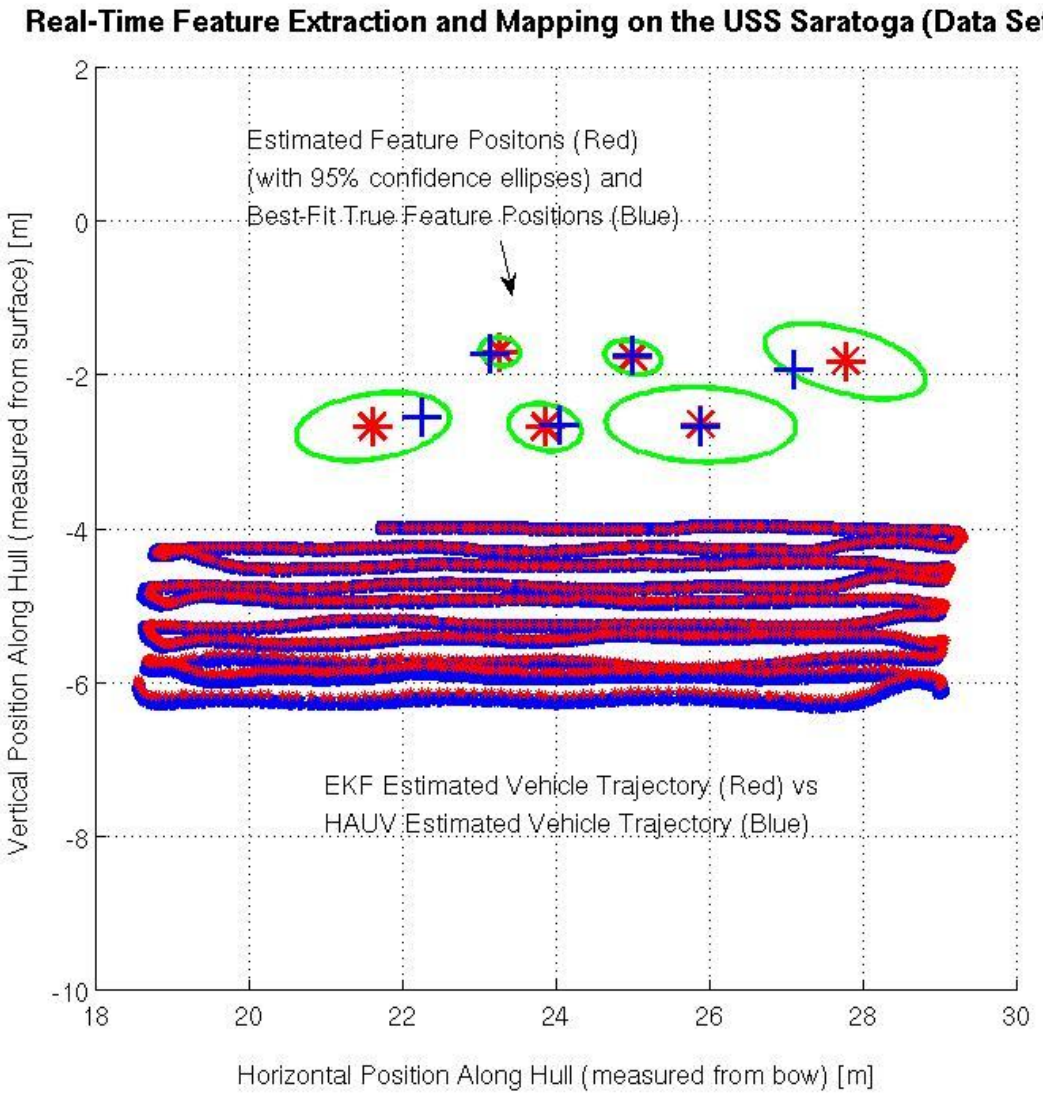

Figure 2-7: The third of three data sets produced from surveying the USS Saratoga. The width of the survey was reduced and as a result no false alarms were obtained.

Chapter 3

Unperturbed Marine Vehicle Stability Analysis

Having demonstrated the ability to identify features and maintain a map in real-time using imaging sonar data, the focus is now shifted to closing the feedback loop on a process that uses range and bearing measurements of point features for estimation. A more detailed planar marine vehicle model is introduced in this section, which interacts with both an estimator and controller. The model is linearized about a nominal vehicle survey trajectory, allowing the closed-loop system to be expressed using a linear time-varying state space framework. A numerical test is formulated that predicts the stability of the vehicle for a given layout of features and a given nominal trajectory among those features. This numerical test is a new contribution to the limited set of stability tools for linear time-varying systems, and permits a quick evaluation of whether the controller selected for a linear time-varying system is stable. Although vehicle the model is formulated to accommodate and overcome random disturbances, the stability analysis doesn't explicitly include a description of model inaccuracies caused by perturbation from the nominal trajectory. This will be addressed in chapter 4.

3.1 Marine Vehicle Model

The dynamics of a holonomic marine vehicle operating in a 2-D plane are described in discrete time by (3.1) with a forward Euler rule:

$$\begin{bmatrix} u_{k+1} \\ v_{k+1} \\ \dot{\varphi}_{k+1} \\ x_{k+1} \\ y_{k+1} \\ \varphi_{k+1} \end{bmatrix} = \begin{bmatrix} u_k + \Delta T\, b_u(u_k)/m \\ v_k + \Delta T\, b_v(v_k)/m \\ \dot{\varphi}_k + \Delta T\, b_{\dot{\varphi}}(\dot{\varphi}_k)/J \\ x_k + \Delta T(u_k \cos\varphi_k - v_k \sin\varphi_k) \\ y_k + \Delta T(u_k \sin\varphi_k + v_k \cos\varphi_k) \\ \varphi_k + \Delta T\dot{\varphi}_k \end{bmatrix} + \begin{bmatrix} I_{3x3} \\ 0_{3x3} \end{bmatrix} \begin{bmatrix} \Delta T \frac{U_1}{m} \\ \Delta T \frac{U_2}{m} \\ \Delta T \frac{U_3}{J} \end{bmatrix} + \begin{bmatrix} I_{3x3} \\ 0_{3x3} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \tag{3.1}$$

$$\underline{x}_{v|k+1} = f(\underline{x}_{v|k}) + B\underline{u}_k + \Gamma \underline{w}_k$$

The body-referenced forward velocity, sway velocity, and yaw rate are described by u, v, and φ, respectively, and x, y, and $\dot{\varphi}$ represent the horizontal, vertical, and angular position of the vehicle in the inertial plane. Hydrodynamic drag b is expressed as a function of velocity in each degree of freedom, and vehicle mass and rotational inertia are described by m and J. Surge, sway, and yaw commands are applied to the channels U_1, U_2, and U_3, respectively, and process noise w_i, which is zero mean Gaussian white noise with diagonal covariance matrix Q, is also applied to each channel. Throughout the analysis and experiments to follow, this model specifically describes the holonomic platform pictured in Figure 3-1, whose time constant of linear motion is approximately two seconds, and whose time constant of angular motion is approximately one second (although in some simulations a double-integrator model simplification will be made). This platform, hereafter referred to as "the raft", was designed with the intention of creating a scaled-down mockup of the HAUV, approximating hull-relative navigation by restricting the vehicle's motion to three planar degrees of freedom. Range and bearing measurement of features in the plane is enabled by a Hokuyo-URG laser range finder mounted on top of the platform. Holonomic actuation is enabled by four bilge pumps mounted on the underside of the platform. Thrusts produced by these pumps can be combined to generate force and torque of any desired directionality in the plane. The design and fabrication of the raft by Michael Kokko are detailed in his Master's Thesis [48].

3.1.1 Aggregate State Vector

A convenient way to permit KF-based SLAM to reconstruct the vehicle state using the

measurement of point features is the addition of quasi-states representing the horizontal and vertical positions of the features in the plane, a concept introduced in Chapter 2. As before, it will be assumed that the quasi-states are permanently fixed in space and have no dynamics. The aggregate state vector appears as follows:

$$\underline{x}_k = [\ \underline{x}_{v|k} \quad x_1 \quad y_1 \quad x_2 \quad y_2 \quad \ldots \quad x_n \quad y_n\ ]^T \tag{3.2}$$

The vehicle states at time k are contained within $\underline{x}_{v|k}$. The contents of this state vector permit the vehicle-relative range and bearing measurements of each feature to be assembled. Unlike the aggregate state vector introduced in Chapter 2, however, the state vector $\underline{x}_k$ will remain of constant size for the entire duration of the algorithm. To perform a stability analysis the features to be measured must be known in advance, and so it will be assumed that a prior map is available and that all features will be observed at each and every measurement step. Within these guidelines two separate scenarios will be considered, map refinement and map exploitation, which were introduced in Chapter 1. Map refinement will permit the feature estimates to vary and map exploitation holds them fixed, reducing the estimation problem to one of localization only.

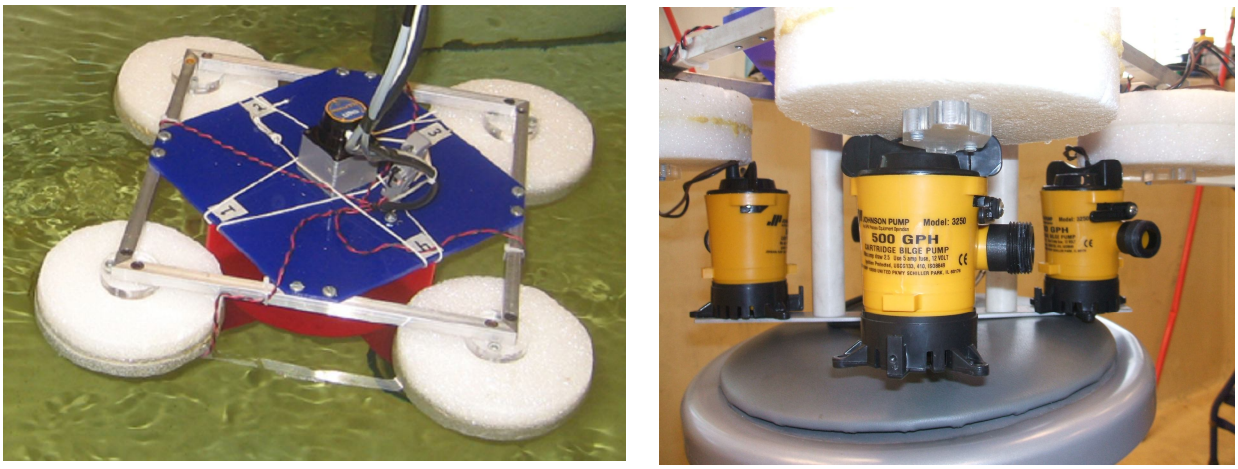

Figure 3-1: The model used in this stability analysis describes a holonomic floating platform with three degrees of freedom, a scaled down mockup of a ship hull inspection vehicle. The platform is equipped with a Hokuyo-URG laser range finder and four bilge pumps mounted on the foam pontoons which act as thrusters.

3.1.2 Nominal Trajectory

First a nominal trajectory is generated for the vehicle to send it to a desired waypoint from its starting position in the plane in a specified amount of time. This is posed initially

independently of any navigation considerations. An open-loop input trajectory delivers a nominal command at each time step, and a closed-loop control correction is used to counteract disturbances. The open-loop trajectory is found by solving a finite time optimal control problem with a quadratic cost function, which is given by (3.3):

$$J(t) = \frac{1}{2}(\underline{x}(T) - \underline{r}(T))^T S(\underline{x}(T) - \underline{r}(T)) + \frac{1}{2}\int_t^T (\underline{x}^T Q\underline{x} + \underline{u}^T R\underline{u})dt \tag{3.3}$$

In this cost function, T represents the final time, $\underline{r}(T)$ represents the desired reference state of the system at the final time T, and S, Q, and R are square, positive semi-definite matrices (R is the only one which must also be positive definite) that contain the penalties associated with the system's final configuration, the system's state at any time, and the system's control input at any time, respectively. The goal of this optimal control problem is to find the sequence of inputs $\underline{u}$ that will minimize J given an initial configuration, a desired reference, and a desired time interval across which the trajectory must be executed. Employing the maximum principle, the following system of differential equations must be solved:

$$\begin{aligned} \dot{\underline{x}} &= g(\underline{x},\underline{u}) = f(\underline{x}) + B\underline{u} \\ -\dot{\underline{\lambda}} &= \frac{\partial \underline{g}}{\partial \underline{x}}^T \underline{\lambda} + Q\underline{x} = \frac{\partial \underline{f}}{\partial \underline{x}}^T \underline{\lambda} + Q\underline{x} \\ 0 &= R\underline{u} + \frac{\partial \underline{g}}{\partial \underline{u}}^T \underline{\lambda} = R\underline{u} + B\underline{\lambda} \end{aligned} \tag{3.4}$$

The function $f(\underline{x})$ represents the nonlinear state transition relationships in (3.1). The vector $\underline{\lambda}$ contains LaGrange multipliers which are used to enforce the system dynamics in the cost function (3.3), these are also known as costates. Because the Jacobian of $g(\underline{x},\underline{u})$ taken with respect to the inputs $\underline{u}$ is not itself a function of $\underline{u}$ (the inputs are related to the system dynamics through a linear time-invariant input matrix B), the third equation of (3.4) can be solved for $\underline{u}$ and substituted into the first equation. This permits (3.4) to be reduced to a single nonlinear ordinary differential equation (if $\underline{x}$ and $\underline{\lambda}$ are placed in the same state vector). This reduced equation and its boundary conditions are given by (3.5):

$$\begin{bmatrix} \dot{\underline{x}} \\ \dot{\underline{\lambda}} \end{bmatrix} = \begin{bmatrix} f(x) + B(-R^{-1}B\underline{\lambda}) \\ -\dfrac{\partial f}{\partial \underline{x}}^{T} \ \underline{\lambda} - Q\underline{x} \end{bmatrix}$$

$$\underline{x}(t_0) = \underline{x}_0 \tag{3.5}$$

$$\underline{\lambda}(T) = S(\underline{x}(T) - \underline{r}(T))$$

Since there are boundary conditions at both the initial and final time, this is by definition a two-point boundary value problem. In this analysis it is solved using MATLAB's *bvp4c* function, which can solve ordinary differential equations formulated using the structure of (3.5). Because this function requires the problem to be posed in continuous time, equations (3.3) to (3.5) have been formulated in continuous time. The formal derivation of (3.4) from (3.3) can be achieved using variational calculus and is explained in detail by Bryson [49] and Lewis [50].

3.1.3 Feedback Control

Once the open-loop input trajectory is formulated, a feedback controller is designed to keep the vehicle on the nominal trajectory. The controller operates linearly on any perceived error in the vehicle state, and errors will exist if the vehicle is not at the precise position and velocity dictated by the nominal trajectory for the current instant in time. Combining the open and closed-loop controls, the system and measurement dynamics is given by:

$$\begin{aligned} \underline{x}_{k+1} &= f(\underline{x}_k) + B(\underline{u}_{OL\,k} - G_k N \delta \hat{\underline{x}}_k) + \Gamma \underline{w}_k \\ \underline{z}_{k+1} &= h(\underline{x}_{k+1}) + \underline{v}_{k+1} \end{aligned} \tag{3.6}$$

The term $\delta \hat{\underline{x}}_k$ represents the deviation of the state estimate from the nominal state trajectory, used as an error signal for the controller. The nonlinear functions *f(x)* and *h(x)* are used to represent, respectively, the state transition relationships in (3.1) and the nonlinear measurement of range and bearing relative to each of the features. The sensor noise term $\underline{v}_k$ represents zero mean Gaussian white noise with diagonal covariance matrix *R*. G_k is a time-varying matrix of feedback control gains, *N* is a 6 by 6+*2n* stripping matrix needed to extract the vehicle states from the state vector, discarding the 2*n* quasi-states for the purposes of control. Because the feature states have no dynamics, the lower 2*n* rows of

f(x) are the identity and the lower 2n rows of *B* and Γ are populated with zeros.

G_k is computed optimally using a quadratic cost function similar in appearance to (3.3). The only difference is that there is no longer a reference state $\underline{r}$ to which the system is being driven, it is assumed instead that the system is being driven to zero (i.e., a regulator problem). Since the controller acts on the vehicle's deviation from the nominal trajectory, $\delta\underline{\hat{x}}_k$, a state of zero corresponds to the vehicle being located on the nominal trajectory. This also allows the vehicle model of (3.1) to be linearized about the nominal trajectory and to be approximated as a linear time-varying system. The state transition Jacobian is used to approximate the system deviation from the nominal trajectory, and is given by (3.7) (note that the hydrodynamic damping is now assumed to be a linear function of velocity). Note that the Jacobian is a square 6+2*n* by 6+2*n* matrix that operates on the aggregate state vector (3.2). Because the system dynamics are linearly related to the system states, computation of the optimal control action is simplified. The time-varying

$$F(\underline{x}_k) = \begin{bmatrix} 1-\frac{b_u}{m}\Delta T & 0 & 0 & 0 & 0 & 0 & 0 & . & 0 \\ 0 & 1-\frac{b_v}{m}\Delta T & 0 & 0 & 0 & 0 & 0 & . & 0 \\ 0 & 0 & 1-\frac{b_\varphi}{J}\Delta T & 0 & 0 & 0 & 0 & . & 0 \\ \cos(\varphi_k)\Delta T & -\sin(\varphi_k)\Delta T & 0 & 1 & 0 & (-u_k\sin(\varphi_k)-v_k\cos(\varphi_k))\Delta T & 0 & . & 0 \\ \sin(\varphi_k)\Delta T & \cos(\varphi_k)\Delta T & 0 & 0 & 1 & (u_k\cos(\varphi_k)-v_k\sin(\varphi_k))\Delta T & 0 & . & 0 \\ 0 & 0 & \Delta T & 0 & 0 & 1 & 0 & . & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & . & 0 \\ . & . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & . & 1 \end{bmatrix} \tag{3.7}$$

optimal feedback gain matrix can be computed using the discrete matrix Riccati equation. This equation is solved backwards in time recursively, beginning with the final configuration of the vehicle along the nominal trajectory. Each step along the nominal trajectory yields a feedback gain matrix G_k which corresponds to the nominal system configuration at the particular instant in time *k*. The feedback gain matrix G_k is derived from the Riccati equation in the following manner:

$$\begin{aligned} S_k &= F(\bar{\underline{x}}_k)^T[S_{k+1} - S_{k+1}B(B^T S_{k+1}B + R)^{-1}B^T S_{k+1}]F(\bar{\underline{x}}_k) + Q \\ G_k &= (B^T S_{k+1}B + R)^{-1}B^T S_{k+1}F(\bar{\underline{x}}_k) \end{aligned} \tag{3.8}$$

The matrices Q and R are penalty matrices identical to those of (3.3) (not to be confused with the system noise covariance matrices Q and R). The matrix S of (3.3), used to penalize the final system configuration, is used as the very first S_{k+1} to initialize (3.8). Of course, the same penalty matrices S, Q, and R do not need to be used for both the open-loop and closed-loop control. In this analysis, different penalty matrices are used for the two controllers. All of the matrices used are diagonal, where each entry along the diagonal penalizes growth of its respective state or input. An iterative tuning process is begun by choosing a maximum desired value of the state or input, and initially setting the penalty on the state to the negative second power of the desired maximum value. After observing the system performance in simulation, the diagonal penalty values of S, Q, and R can be adjusted to obtain the desired system performance. This systematic method of tuning S, Q, and R, as well as the derivation of (3.8) from a quadratic cost function, are detailed in a more recent work by Bryson [51].

3.1.4 Linearized Kalman Filter

The use of a nominal vehicle trajectory permits a linearized Kalman filter to serve as the estimator for vehicle localization. This strategy allows the vehicle to move to any desired location in the plane, as long as an approximate layout of features is known in advance. It is also assumed that feature association can be performed successfully, and that ΔT between measurements is constant. These assumptions will allow vehicle pose estimation and map refinement to occur using a precomputed set of gains and Jacobians. The estimation equation is written in terms of deviation from the nominal trajectory $\bar{x}_k$:

$$\begin{aligned} \delta\hat{\underline{x}}_{k+1} &= \delta\hat{\underline{x}}_{k+1|k} + K_{k+1}[\delta\underline{z}_{k+1} - H(\bar{x}_{k+1})\delta\hat{\underline{x}}_{k+1|k}] \\ \delta\hat{\underline{x}}_{k+1|k} &= F(\bar{x}_k)\delta\hat{\underline{x}}_k - BG_k N\delta\hat{\underline{x}}_k \end{aligned} \tag{3.9}$$

The nonlinear state transition function $f(x)$ and the measurement function $h(x)$ of (3.6) are now replaced by $H(\bar{x}_k)$ and $F(\bar{x}_k)$, the corresponding Jacobians, which are linearized about the nominal trajectory at each time step and have been defined respectively in (2.4) and (3.7). The term $\delta\underline{z}_k$ represents the deviation of the measurement from the deterministic measurement along the nominal trajectory. K_k is the time-varying

Kalman gain, which is computed in advance along each step of the nominal trajectory. The LKF estimation algorithm is similar to the EKF algorithm described in Chapter 2, except the Jacobians used to propagate the error covariance matrix are linearized about the nominal trajectory instead of being linearized about the most recent *a priori* estimate. Another difference from the EKF is that Jacobians also appear in the equations which propagate the state estimate, as can be seen in (3.9). The final difference is in the quantity that is estimated, which is the deviation from the nominal trajectory rather than the complete trajectory. To obtain a complete state estimate, the nominal trajectory can be added to the deviation estimate produced by (3.9).

Feature initialization occurs according to a procedure similar to that used in Chapter 2. Instead of initializing features one at a time, for the case of map refinement the entire *a priori* map is initialized at once, producing the initial estimation error covariance matrix. Feature initialization is begun with a preparatory error covariance matrix similar to (2.12), which is given by (3.10):

$$P_0^* = \begin{bmatrix} P_{v0} & 0 \\ 0 & R_c \end{bmatrix} \tag{3.10}$$

This preparatory matrix contains an initial vehicle error covariance matrix, which is a diagonal matrix whose nonzero entries are variances chosen for each vehicle position and velocity state derived from the presumed accuracy of the initial guess for that state. Since no feature covariances have been added yet, R_c is a map confidence covariance matrix, a diagonal matrix which contains a range and bearing measurement variance that correspond to the confidence in the location of each feature on the *a priori* map. If the *a priori* map to be used for map refinement is known to be perfect, then sub millimeter and sub degree confidence can be expressed for every feature on the map; if the *a priori* map is likely to contain errors, then larger variances can be used for map initialization. This preparatory matrix is propagated through the heuristic feature position estimation equations defined in (2.10), which estimate feature locations using range and bearing measurements of features and the initial vehicle position. The range and bearing measurements used here are the deterministic measurements from the very first measurement step of the nominal trajectory, and the feature initialization Jacobian of

(2.14) is consequently computed and the initial error covariance matrix is obtained as follows:

$$P_0 = \nabla g_0 P_0^* \nabla g_0^T \tag{3.11}$$

Although an EKF is likely to yield better estimation in the presence of perturbations than an LKF, the EKF's nonlinearity and dependence on the vehicle's noise-influenced trajectory do not allow linear matrix computation or computation in advance of the vehicle's deployment. Thus the LKF will serve as the estimator for the remainder of this analysis at the risk of inaccuracy in the presence of large perturbations and with the benefit of enabling a more descriptive stability analysis. Despite this decision, the ultimate goal of this analysis is to develop a design procedure for ship hull survey algorithms which use an EKF rather than an LKF. It is expected, although not strictly proven, that the system configuration which yields the most stable and robust LKF-enabled closed-loop vehicle will also yield the most stable and robust EKF-enabled closed-loop vehicle.

3.1.5 Aggregate Transition Matrix

Thus far the only linearization approximations have been those which are called for specifically by the linearized Kalman filter. To enable a stability analysis, it will be further assumed that dynamics of the true physical plant are well approximated by the state transition Jacobian, and that the true measurement process is also well approximated by the measurement Jacobian. Simplification of (3.6) and (3.9) yields the following compact formulation:

$$\begin{bmatrix} \delta \underline{x}_{k+1} \\ \delta \underline{\tilde{x}}_{k+1} \end{bmatrix} = \begin{bmatrix} F(\bar{x}_k) - BG_k N & -BG_k N \\ 0 & F(\bar{x}_k) - E_k \end{bmatrix} \begin{bmatrix} \delta \underline{x}_k \\ \delta \underline{\tilde{x}}_k \end{bmatrix}$$

$$E_k = K_{k+1} H(\bar{x}_{k+1}) F(\bar{x}_k) \tag{3.12}$$

$$\delta \underline{\tilde{x}}_k = \delta \underline{x}_k - \delta \underline{\hat{x}}_k$$

The upper half of the state vector contains deviations from the nominal trajectory, rather than the full states, and the lower half of the state vector contains the estimation error. Equation (3.12) provides us with an equilibrium point of zero throughout the system's operation, and it also captures the complete closed-loop system dynamics and estimation

error dynamics in the structure of a linear time-varying system. This is a structure which lends itself to Lyapunov stability analysis, as will be demonstrated in the sections to follow. A comprehensive tutorial on linearized systems of this structure, which combine an open-loop nominal trajectory with an LKF estimator and a quadratic cost optimal feedback controller, is presented by Athans [52]. As noted previously, we pursue first a stability analysis of (3.12) without considering the effects of geometry errors.

3.2 Stability of Linear Time-Varying Systems

A few stability definitions will now be presented. A system equilibrium point is *stable in the sense of Lyapunov*, or *Lyapunov stable*, if for any region of specified size surrounding the point, it is possible to specify a second region from within which the system may start and is guaranteed remain within the boundaries of the first region for all time. A system equilibrium point is *asymptotically stable* if it is possible to specify a region from within which the system may start and will always return to the equilibrium point in the limit as time approaches infinity. *Exponential stability* is achieved when this asymptotic convergence may be described by a decaying exponential function of time. An equilibrium point is *globally stable* if the region from within which the system may be initialized and achieve one of the above stability guarantees consists of the entire state space, and an equilibrium point is *uniformly stable* if the stability guarantee holds for any choice of initial time k_0. A linear time-varying system can never be asymptotically stable without also being exponentially stable. A more detailed presentation of these definitions and further treatment of the topic of stability in the context of dynamic systems and control is presented by Slotine [53].

Before the stability of (3.12) is assessed, two stability theorems are laid out. These theorems are defined for use with discrete systems by Willems [54].

> **Theorem 3.1.** The null solution of (3.12) is stable in the sense of Lyapunov if and only if there exists a bound M, for any k_0, such that the following inequality holds for all $k \geq k_0$:
>
> $$\left\| \Phi(k, k_0) \right\| \leq M$$

$\Phi(k,k_0)$ is the transition matrix which propagates the system to time k from time k_0. If M can be taken independently of k_0, then the solution is uniformly stable in the sense of Lyapunov.

Theorem 3.2. The null solution of (3.12) is asymptotically stable if and only if the conditions of Theorem 3.1 for stability in the sense of Lyapunov are satisfied, and:

$$\lim_{k=\infty} \left\| \Phi(k, k_0) \right\| = 0$$

The solution is uniformly asymptotically stable if the above is satisfied and the bound M of Theorem 3.1 can be taken independently of k_0.

The norm used in Theorems 3.1 and 3.2 is the Euclidean or spectral norm, equivalent to the largest singular value of the transition matrix. Using the notation of Theorems 3.1 and 3.2, the state transition matrix of (3.12) would be expressed as $\Phi(k+1,k)$, as it propagates the closed-loop vehicle system from the state at time k to the subsequent state at k+1 . By multiplying the successive state transition matrices of (3.12), the transition matrix from any k_0 to any time k along the nominal trajectory can be computed. Because the transition matrix of (3.12) is linearized about a nominal trajectory, it is clear that a certain distance from the nominal trajectory the linearization will fail to capture the system dynamics accurately. For this reason a guarantee as strong as global stability cannot be achieved for the closed-loop marine vehicle system, and so uniform Lyapunov and asymptotic stability in the local region of the nominal trajectory will be pursued.

3.2.1 Trajectories Designed for Stability Analysis

Although the framework described in Section 3.1 is designed for a nominal trajectory that connects two waypoints, this scenario can be augmented for analysis at infinite time (permitting use of Theorem 3.2). In the analysis to follow, two periodic paths among a series of waypoints will be considered, one designed for simulation and another simplified trajectory designed for experiment. The trajectory designed for simulation, and the

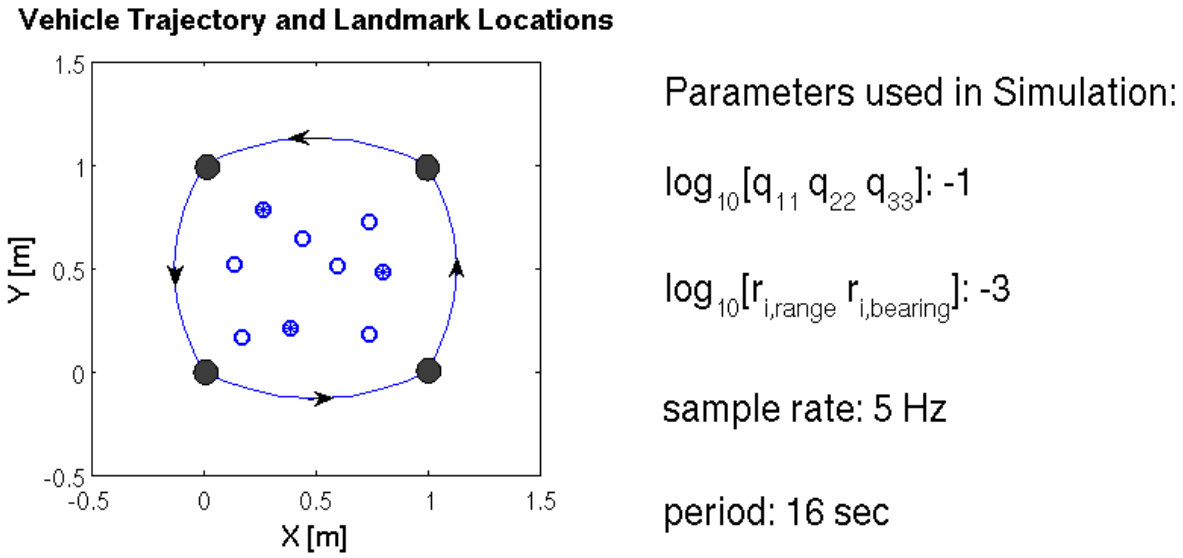

Figure 3-2: A periodic vehicle trajectory designed for simulation is depicted, along with the configuration of landmarks used by the vehicle for estimation. This trajectory is employed in simulations using all nine features and also a subset of three features. The three-feature subset is indicated by the features marked with an asterisk. The nominal changes in heading commanded across this trajectory are indicated by the angle φ drawn at each waypoint. Parameters that are used in the vehicle simulation are listed.

corresponding parameters used in simulation, are given in Figure 3-2. This trajectory uses a marine vehicle model with a double integrator physical plant that assumes unit masses and inertias. The vehicle begins at the inertial frame origin with an initial heading of 45° from the inertial *x*-axis, it proceeds to a second waypoint where its heading is 135° from the x-axis, followed by two more waypoints where another 45° is added to the heading, until it reaches the initial configuration where the trajectory began. The vehicle then proceeds in reverse along the same path until reaching the original waypoint at the origin. This entire procedure comprises one cycle of the trajectory. The specific path followed between waypoints is computed using the optimal control strategy described in Section 3.1. The trajectory is designed to allow the vehicle to pass through a wide range of its state space in which the linearized system Jacobians take on a wide range of values. The features used by the vehicle for navigation are also depicted in Figure 3-2, and were generated at random to populate the square-shaped area enclosed by the trajectory. This trajectory will hereafter be referred to as the *four-point double integrator trajectory*.

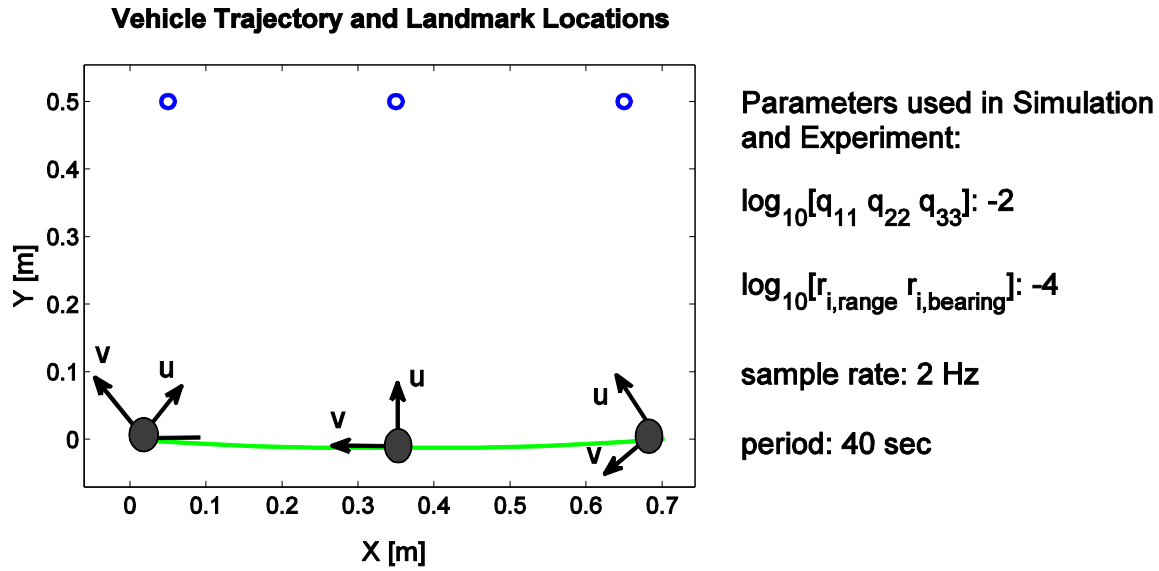

Figure 3-3: The periodic vehicle trajectory used for raft experiments is depicted, along with the configuration of three landmarks used for raft localization. The nominal changes in heading commanded across this trajectory are indicated by the body-fixed coordinates *u* and *v* and inertial angular coordinate φ. Parameters used in both experiment and simulation of this trajectory are listed.

Although the four-point double integrator trajectory should reveal much about the stability of a localization, mapping, and control process, it must be simplified for experimental implementation. The raft, a tethered platform with an external power source, will encounter difficulties maneuvering around the field of features and sighting all features on the randomly-generated map at once. A simplified path for use in experiment contains only two of the waypoints from the four-point double integrator trajectory, which are depicted in Figure 3-3. A periodic path between the two waypoints will be executed by the raft, with the specific trajectory between waypoints computed once again using optimal control. One period consists of a forward trip along the path, followed by a trip in reverse back to the origin. The double integrator assumption is abandoned and linear damping parameters are now used in the vehicle model to capture the hydrodynamic drag forces. Inertial parameters indicative of the raft's true mass and rotational inertial are used in the model as well. The configuration of three features along a horizontal line was chosen so that one feature will never block the line of sight to another feature. This is not a problem for simulation, but for experiment it is necessary that all features can be physically sighted by the raft's laser at each measurement step. This trajectory will hereafter be referred to as

the *two-point experimental trajectory*. This trajectory will be analyzed briefly in the final section to lay the groundwork for the raft experiments of Chapters 4 and 5.

3.3 Evaluating the Transition Matrix Norm

In this section the transition matrix norm of the holonomic marine vehicle described by (3.12) will be evaluated along both of the nominal trajectories identified in Section 3.2, for cases of both map refinement and map exploitation. Theorems 3.1 and 3.2 will be applied to the transition matrix norm to assess stability.

3.3.1 Map Refinement

The first estimation scenario considered is one in which the vehicle uses point features for localization and can simultaneously adjust its estimates of the feature locations (i.e., it can refine the *a priori* map as it repeatedly observes the features). For the case of the four-point double integrator trajectory, Figure 3-4 displays the values of the transition matrix norm for the $\Phi(k,k_0)$ that transit from every initial time k_0 to every final time k along two cycles of the nominal trajectory, for cases of three and nine features. A prominent attribute of the 3D surface formed by plotting the transition matrix norm is the periodic spiked ridge encountered among the first few time steps. This preliminary increase in the value of the norm after being propagated for a few discrete sampling instants is comparable to overshoot, the growth of the system states before settling to a final value. Beyond this ridge the norm converges to a periodic surface, and it does so independent of the value of k_0. The value of the norm does not reach zero, but it remains bounded and does not exceed the values of the initial "overshoot". This permits the conclusion that the transition matrix norm can be assigned a bound M which it will not surpass, and that this bound can be taken independent of k_0. Hence, Theorem 3.1 is satisfied and we may conclude the nominal trajectory investigated is uniformly stable in the sense of Lyapunov. This is the strongest stability guarantee we can obtain for a system that includes the feature quasi-states, which allow map refinement to occur as the features are measured. It also confirms the intuition that a system cannot drive both the vehicle and feature state estimates to zero if the map

used for pre-computing the LKF filter gains and Jacobians is in need of refinement.

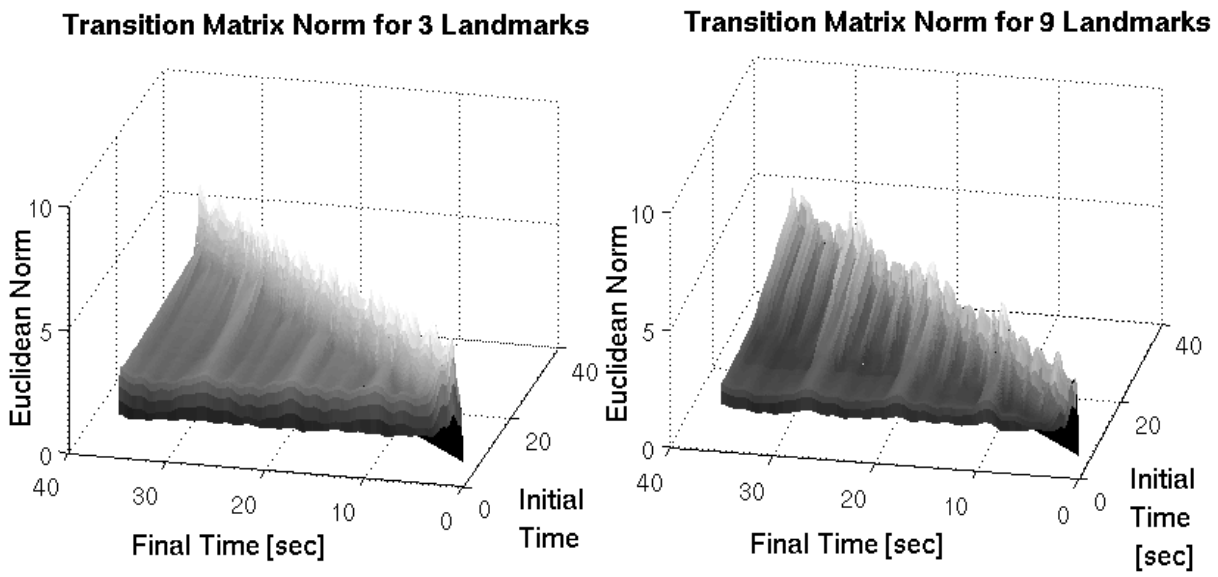

Figure 3-4: The transition matrix norm is plotted for the cases of both three and nine landmarks using the four-point double integrator trajectory with map refinement. The plots encompass two complete cycles of the trajectory and use the parameters identified in Figure 3-2.

A few differences are noticeable between the three-feature and nine-feature plots in Figure 3-4. The three-feature plot has a larger apparent overshoot than the nine-feature case, and it converges to a steady-state periodic surface that is larger in magnitude than that of the nine-feature case. Although these features are not proof of superior performance, examination of the system eigenvalues in each case offers further support that the nine-feature case yields improved estimation. Although this is a time-varying system and the system's poles change location over time, the poles of the estimator for the nine-feature case always remain closer to the origin of the *z*-plane than those of the three-feature case, and consequently the nine-feature system should exhibit faster estimation error dynamics.

3.3.2 Map Exploitation

In search of a stronger stability guarantee for the closed-loop marine vehicle, map exploitation is considered next, in which case the *a priori* map is assumed to contain the correct feature locations and only the vehicle states are estimated (i.e., localization only).

The plots of Figure 3-5 show the transition matrix norm for the $\Phi(k,k_0)$ that transit from every initial time k_0 to every final time k along two cycles of the four-point double integrator nominal trajectory. Unlike the cases of map refinement, the norm converges to a value of zero, and it does so irrespective of initial time k_0. This means that not only may we identify a bound M for the norm according to Theorem 3.1, but we may also apply Theorem 3.2 since the norm of $\Phi(k, k_0)$ approaches zero as k approaches infinity, independent of the choice of k_0. Thus, map exploitation is capable of achieving uniform asymptotic stability, since the vehicle and its state estimates can be driven to zero simultaneously if the map is accurate.

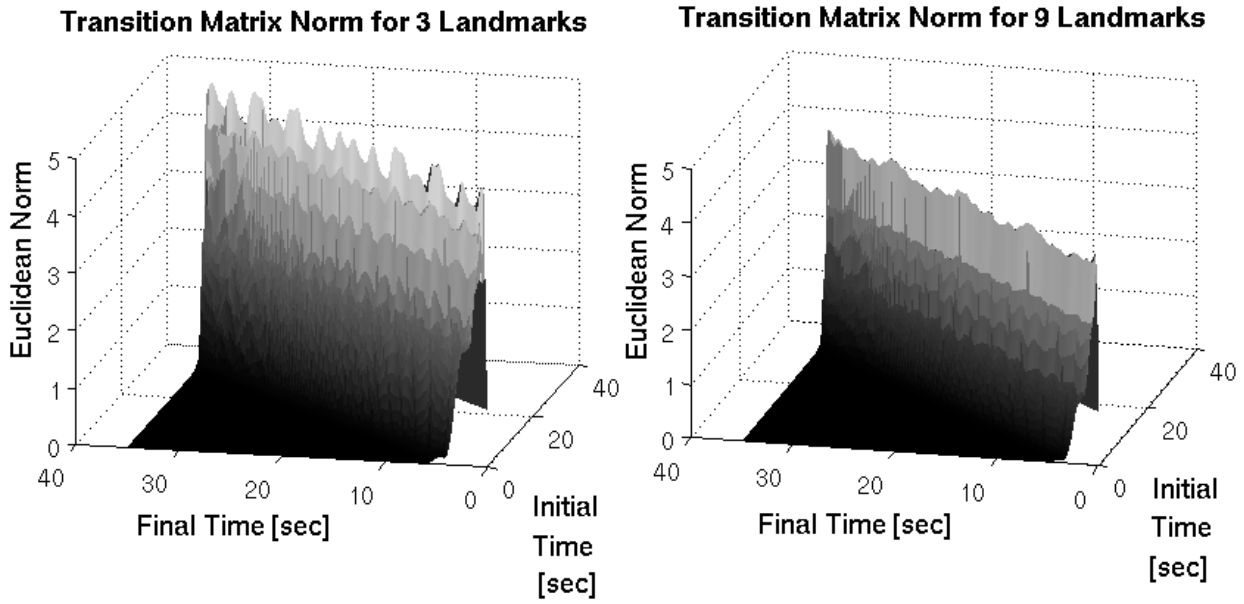

Figure 3-5: The transition matrix norm is plotted for the cases of both three and nine landmarks using the four-point double integrator trajectory with map exploitation. The plots encompass two complete cycles of the trajectory and use the parameters identified in Figure 3-2.

Once again, there are identifiable differences between the three-feature and nine-feature cases. The three feature transition matrix norm exhibits a larger overshoot, and it also converges to zero more slowly than the nine-feature case. Because the systems depicted in Figure 3-5 are asymptotically stable, we can draw more tangible conclusions from these plots about the behavior of the system state. As the transition matrix norm converges to zero, it is also necessary for the system state to converge to zero. The fact that the norm converges to zero faster in the nine-feature case confirms that the state will also

converge faster, and thus the closed-loop dynamics of the nine-feature system are faster. As both configurations use the same controller, it is once again in the estimation error dynamics where the difference in speed emerges.

3.3.3 Estimating the Upward Gain Margin

The conditions for stability presented in Theorems 3.1 and 3.2 are both necessary and sufficient, which means that if the transition matrix norm for a system is clearly unbounded (i.e., it grows without bound as time increases), the system under consideration is unstable. Only stable feature-based systems have been discussed so far, but in this section the controller gains designed for a stable system will be uniformly amplified until the onset of instability is observed in the behavior of the transition matrix norm. Three different estimation strategies are compared for the four-point double integrator trajectory; map refinement for three features, map exploitation for three features, and a measurement process which can directly observe the states x, y, and φ. To ensure that this is a fair comparison, the nominal controller gains and the Kalman filter tuning parameters are set equal in all three cases (the LKF parameters remain unchanged from those indicated in Figure 3-2). For the direct measurement process, this entails setting the sensor noise variance of the x, y, and φ measurements equal to the variance of the vehicle-relative range and bearing measurements used by the four-point double integrator trajectory. The penalty matrices S, Q, and R used to generate the feedback control gains for the four-point double integrator trajectory are given by Table 3.1.

Table 3.1: Penalty Matrices Used for Generating Optimal Feedback Control Gains Along the Four-Point Double Integrator Trajectory

State Penalty Matrix Q	$Q = diag\left[\frac{1}{.25^2} \quad \frac{1}{.25^2} \quad \frac{1}{.25^2} \quad \frac{1}{.01^2} \quad \frac{1}{.01^2} \quad \frac{1}{.01^2}\right]$
Input Penalty Matrix R	$R = diag\left[\frac{1}{.05^2} \quad \frac{1}{.05^2} \quad \frac{1}{.05^2}\right]$
Terminal State Penalty Matrix S	$S = diag\left[10^{12} \quad 10^{12} \quad 10^{12} \quad 10^{12} \quad 10^{12} \quad 10^{12}\right]$

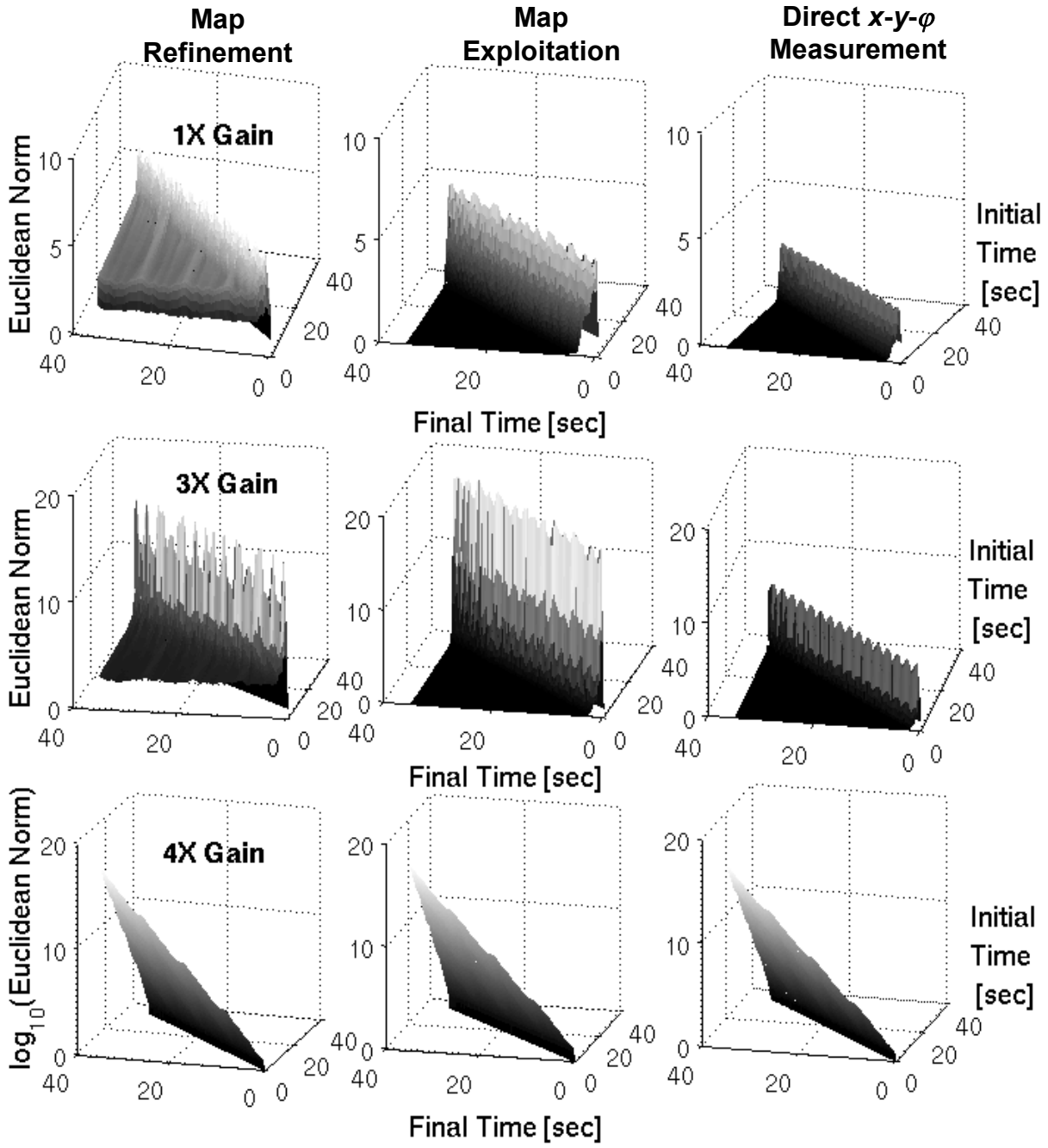

Figure 3-6: The transition matrix norm is compared for three-feature map refinement, three-feature map exploitation, and direct measurement of x, y, and φ, each for two complete cycles of the four-point double integrator trajectory. The first row corresponds to the nominal feedback gains produced by the penalty matrices of Table 3.1, the second and third rows correspond to gains which have been uniformly amplified by factors of three and four, respectively.

Figure 3-6 contains plots of the transition matrix norm for two complete cycles of the nominal trajectory for each of the three estimation scenarios. In the top row, the nominal feedback control gains are used, in the middle row, the gains have been multiplied by a factor of three, and in the bottom row, the gains have been multiplied

by a factor of four. The three systems remain stable using a factor of three (although each exhibits larger overshoot), but a factor of four is sufficient to render all systems unstable, as $\Phi(k,k_0)$ is unbounded and grows rapidly with increasing k in all three plots in the bottom row of Figure 3-6. Although the x-y-φ measurement system exhibits less overshoot than the two feature-based systems, its upward gain margin is similar to the feature-based systems, indicating that use of a feature-based estimation process does not reduce the gain margin of the closed-loop system. Just as the norm of $\Phi(k,k_0)$ was used to examine stability in the previous sections, it can also be used to check for instability and gauge the approximate gain margins of the system.

Although this method of examining the norm of $\Phi(k,k_0)$ predicts instability due to an ill-conditioned controller, there are some aspects of the localization estimator which can pass through unnoticed when ill-conditioned (such as a map whose geometry will not allow successful localization). For this reason, we must also consider the effect of perturbations on the system, which are needed to bring about the failure of the filter due to certain aspects of filter conditioning. This issue will be discussed in Chapter 4.

3.4 Simulated Marine Vehicle Time Response

Thus far system stability has been investigated using the transition matrix only, and the impact of initial conditions on system response hasn't been addressed. To examine the effect of initial conditions and filter tuning choices on the dynamic response of the marine vehicle, a few time responses along the four-point double integrator trajectory are simulated in the absence of process and sensor noise. In reality a marine vehicle will never be located at the precise pose where its controller and estimator are initialized, and errors in the a *priori* map are also likely to exist. It is important to ensure that integrated localization, mapping, and control is robust to discrepancies of this nature if it is to be used for autonomous ship hull inspection.

Every time response investigated here is initialized with an error either in vehicle position or in the location of features on the *a priori* map. For the cases of erroneous pose initialization, the filter is initialized with the vehicle at zero velocity, located at the inertial

plane origin, with a heading of 45°. In reality the vehicle is offset 0.1m in *x* and in *y* from this location, with the same 45° heading. For cases of erroneous map initialization, the filter is initialized with a map in which the features are rotated 30° about their centroid from the true feature locations. The true map is comprised of the three-feature subset introduced in Figure 3-2. Two sets of filter tuning parameters are also varied among the simulations; initial confidence in vehicle pose (the initial vehicle error covariance matrix) and initial map confidence (the map confidence matrix used to initialize the error covariance matrix). A high initial vehicle pose confidence corresponds to a diagonal covariance matrix with entries equal to 10^{-2} in magnitude (used in all other simulations in this analysis), and a low vehicle pose confidence corresponds to entries of magnitude one. Specifically, this formulates a comparison between Gaussian estimation error distributions with standard deviations of a tenth-meter, tenth-radian, tenth-meter-per-second and tenth radian-per-second (respective to each of the states in the state vector) versus a standard deviation of one meter, one radian, and so on. A high initial map confidence corresponds to a diagonal map covariance matrix with entries equal to 10^{-4} in magnitude, and a low map confidence corresponds to entries of 10^{-2} in magnitude. These magnitudes describe the assumed estimation error distributions of the initial range and bearing measurements, which possess units of meters and radians respectively. The system configurations described here are summarized in Table 3.2. In the figures to follow, high or low confidence will be indicated by an upward or downward-pointing arrow.

Table 3.2: Parameters Used for Analyzing Closed-Loop Time Response Sensitivity to Erroneous Initial Conditions

Initialization Options:	Poor Pose Initialization: Offset of 0.1m in *x* and in *y*	Poor Map Initialization: A priori map is rotated 30°
Vehicle Confidence Options:	High Initial Vehicle Confidence: $P_0 = diag\begin{bmatrix}10^{-2} & 10^{-2} & 10^{-2} & 10^{-2} & 10^{-2} & 10^{-2}\end{bmatrix}$	Low Initial Vehicle Confidence: $P_0 = diag\begin{bmatrix}10^{-1} & 10^{-1} & 10^{-1} & 10^{-1} & 10^{-1} & 10^{-1}\end{bmatrix}$
Map Confidence Options:	High Initial Map Confidence: $R_C = diag\begin{bmatrix}10^{-4} & 10^{-4} & 10^{-4} & 10^{-4} & 10^{-4} & 10^{-4}\end{bmatrix}$	Low Initial Map Confidence: $R_C = diag\begin{bmatrix}10^{-2} & 10^{-2} & 10^{-2} & 10^{-2} & 10^{-2} & 10^{-2}\end{bmatrix}$

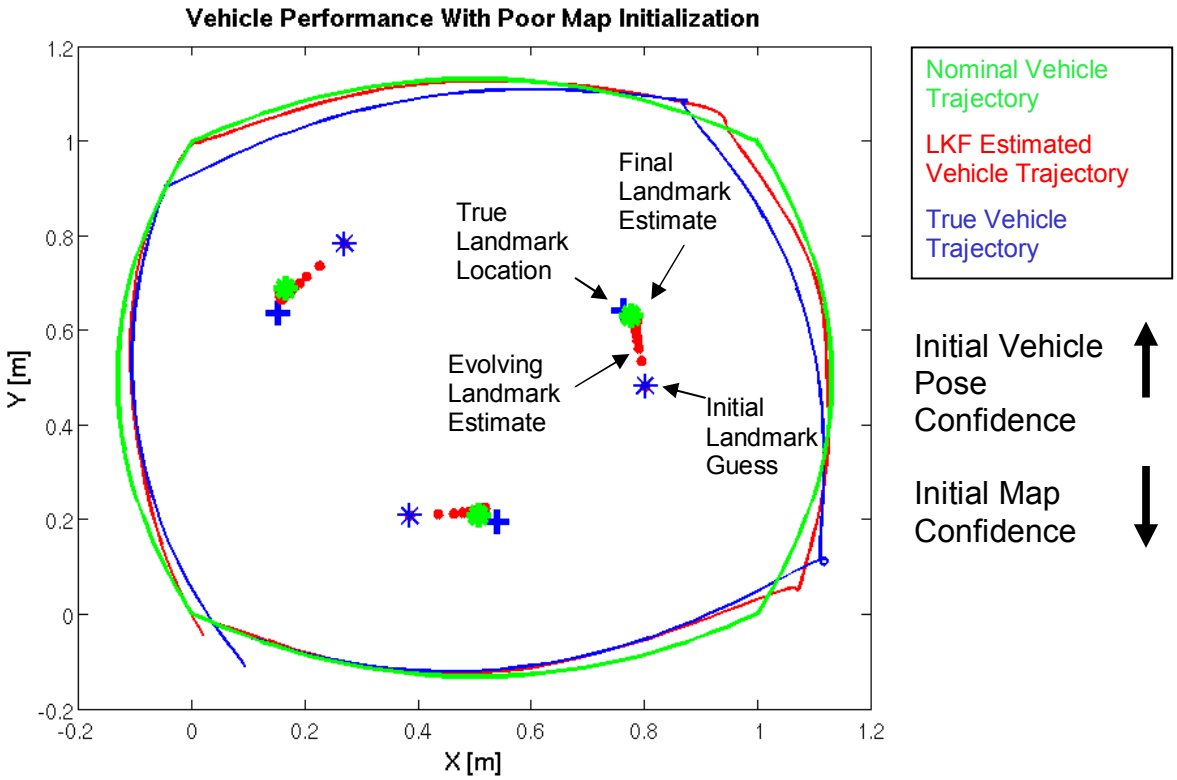

Figure 3-7: Time response for a half-cycle of the four-point double integrator trajectory for erroneous map initialization plotted in the *x*-*y* plane, for the case of map refinement. High confidence is expressed in the initial vehicle pose and consequently the feature estimates gradually approach the true feature locations.

For the case of map refinement, Figure 3-7 demonstrates that a high initial pose confidence and low initial map confidence will allow a vehicle with erroneous map initialization to recover; the feature estimates propagate toward the true feature locations as the features are repeatedly observed. Figure 3-8 demonstrates that varying initial pose confidence will influence the ability of a vehicle with erroneous pose initialization to recover. With high initial pose confidence, the blame is placed on the map and the feature estimates converge to enforce the errors in the vehicle pose estimate. These erroneous estimates prevent the vehicle from being driven along the nominal trajectory. Low initial pose confidence allows the pose estimation error to be corrected, with estimates quickly converging to the true vehicle pose as the vehicle is driven correctly along the nominal trajectory, although the feature estimates remain in error. For the case of map exploitation, Figure 3-9 shows that even if initial pose confidence is high, the estimator will correct the pose estimation error and the vehicle will converge to the nominal trajectory. This is because the feature locations are fixed in this localization-only scenario, which can prove

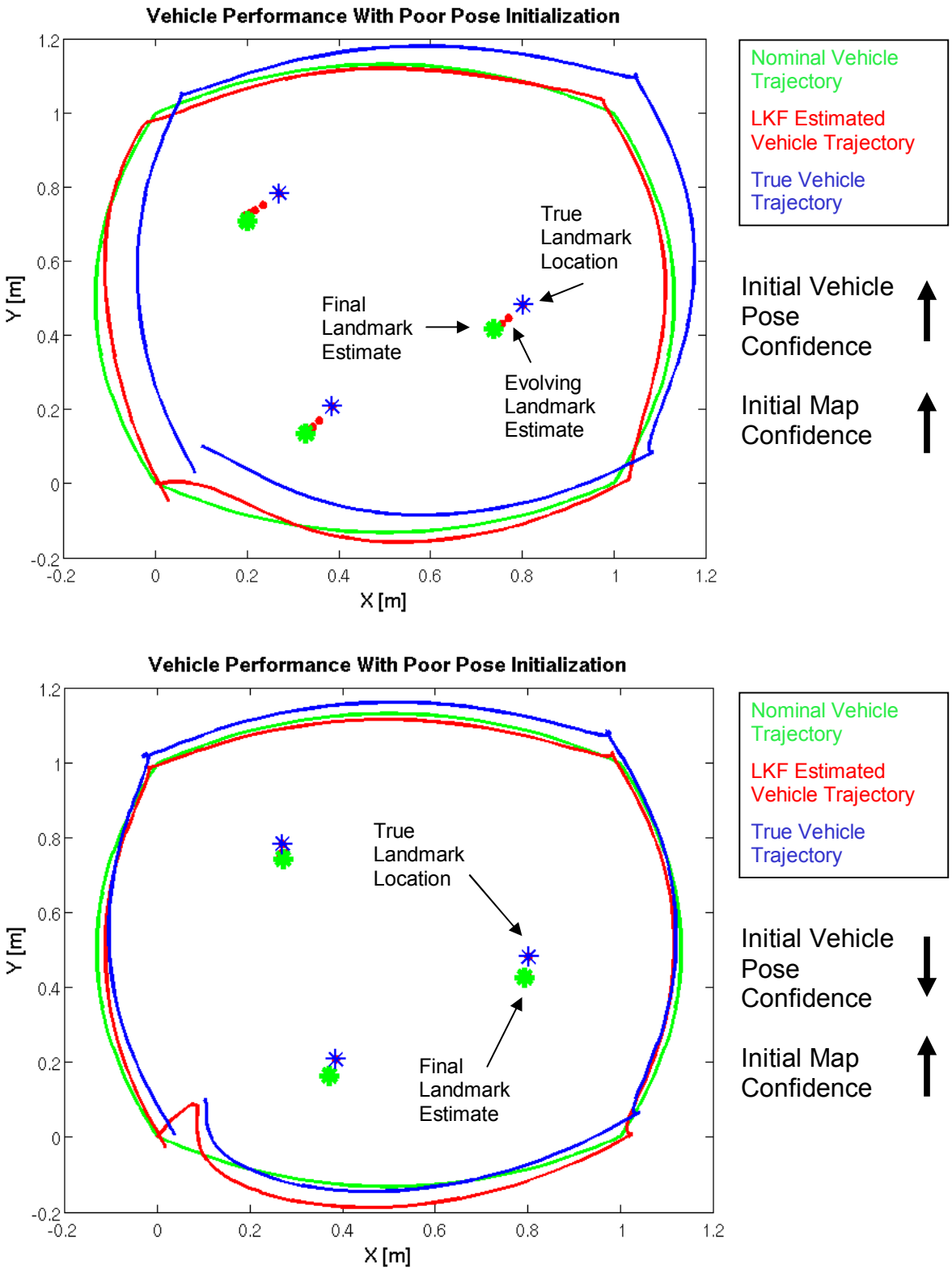

Figure 3-8: Time response for a half-cycle of the four-point double integrator trajectory for erroneous pose initialization plotted in the *x*-*y* plane, for the case of map refinement. Varying confidence is expressed in the initial vehicle pose and the pose estimate's ability to converge to the true pose depends on this.

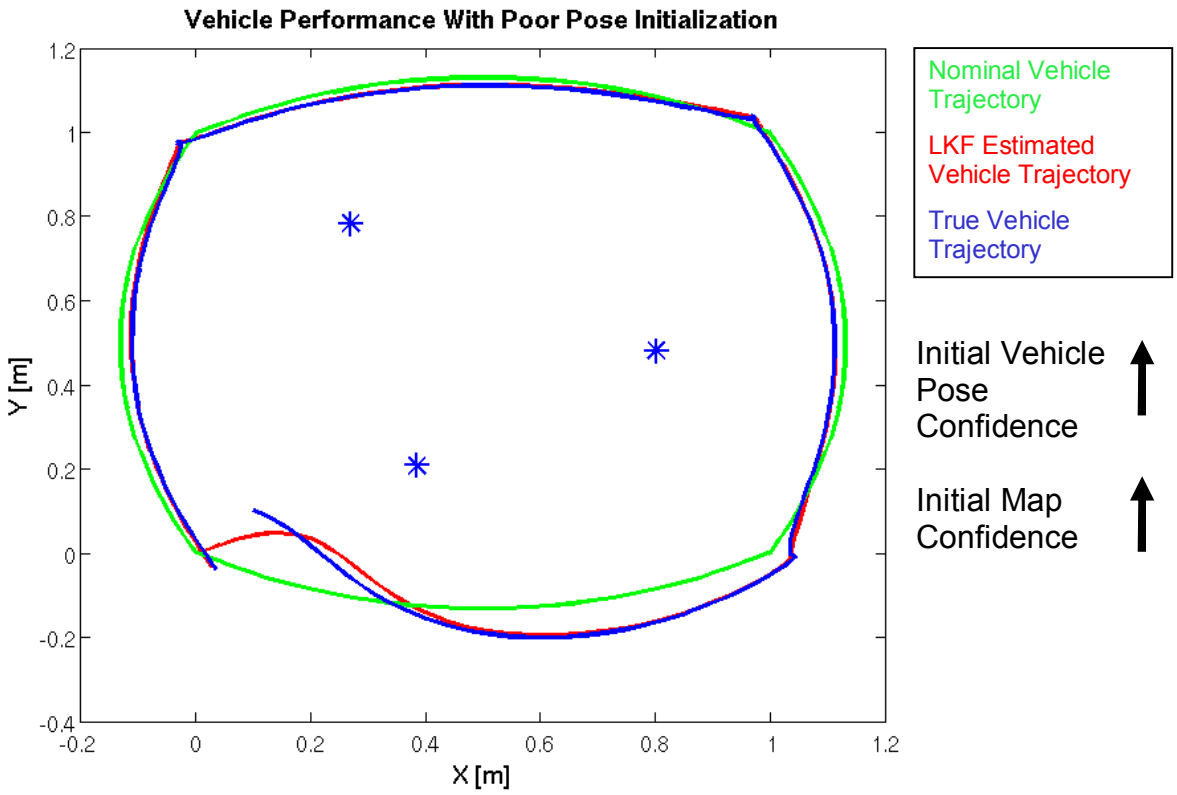

Figure 3-9: Time response for a half-cycle of the four-point double integrator trajectory for erroneous pose initialization plotted in the *x*-*y* plane, for the case of map exploitation. High confidence is expressed in the map and initial vehicle pose and pose estimate succeeds in converging to the true vehicle pose.

more robust than map refinement if the *a priori* map is known to be accurate.

To complement this assessment of simulated marine vehicle time responses, another important question to ask in considering real-time implementation of integrated localization, mapping, and control is whether the estimates produced by the LKF are comparable to those produced by an EKF when the vehicle is subjected to process noise that displaces it from the nominal trajectory. Figure 3-10 shows simulation results which support the assertion that LKF estimation is a good predictor of EKF performance, even in the presence of perturbations. Even though the LKF precomputed error covariance is ignorant of the perturbations encountered, it still provides an accurate prediction of the EKF perturbation-influenced error covariance. In addition, LKF estimation error largely remains within a single standard deviation of the EKF's noise-dependent estimation error distribution, and so LKF predictions should prove a useful tool for designing EKF surveys.

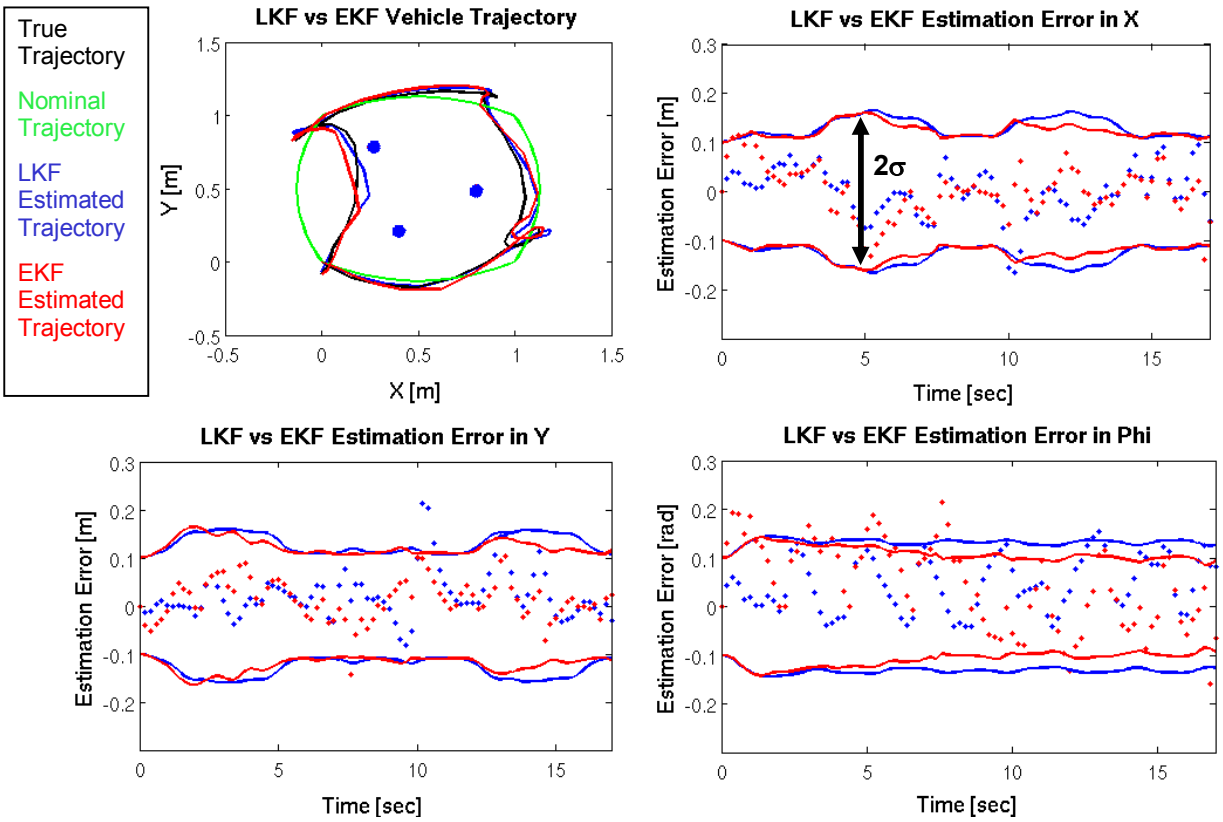

Figure 3-10: Time response and covariance comparison for a half-cycle of the four-point double integrator trajectory is plotted in the case of map exploitation, with process noise included. LKF pose estimation, estimation error, and precomputed error covariance is compared with EKF noise-dependent pose estimation, estimation error, and error covariance. The error covariance is plotted using bounds of 2σ.

3.5 A Bounding Tube Stability Test

The techniques presented in Section 3.3 serve to indicate when a closed-loop marine vehicle is stable or unstable, but little insight is provided on the characteristics of system convergence without simulating a vehicle time response. In particular, for cases of map exploitation where the vehicle can achieve asymptotic stability, it is desirable to describe the speed with which the system converges to its equilibrium point. A special feature of the eigenstructure of a system carrying out map exploitation will allow exactly such a description to be formulated.

As described earlier, the map exploitation framework uses a state vector containing the six vehicle states only, since the feature locations on the *a priori* map are assumed to be trustworthy. The map refinement framework, on the other hand, has two states in its state vector for each feature on the map, and even though the feature state estimates can change,

the feature states themselves are static. Since the feature positions can't be influenced by the system's controller, the state transition matrix of the map refinement framework possesses eigenvalues of value one corresponding to each of the static feature states. Because the state vector of the map exploitation framework contains only the vehicle states, and all of these states can be influenced by the system's controller, the map exploitation framework can maintain all eigenvalues within the unit circle. A system in which all eigenvalues lie within the unit circle is known as a Hurwitz system, and a variety of techniques for describing the convergence rate of linear time-varying Hurwitz systems have been developed [55], [56].

A recent result from Mullhaupt, Buccieri, and Bonvin [57] documents a numerical sufficiency test for asymptotic stability of linear time-varying Hurwitz systems which at each time step defines an ellipsoid in which the system is guaranteed to stay. A Lyapunov function of the form:

$$V(t) = x(t)^T P_i x(t) \tag{3.13}$$

is computed at each time step and applies from time t_i to t_{i+1}. The algorithm may be summarized as follows:

1) Approximate the maximum value of (3.13):

Compute λ, the largest eigenvalue of P_i, then:

$$\beta_{i-1}(t) = \max(\, eig\,(A^T(t)P_{i-1} + P_{i-1}A(t)))$$

$$\hat{V}_i = V_{i-1} \exp(\frac{1}{\lambda}\int_{t_{i-1}}^{t_i} \beta_{i-1}(\tau)d\tau) \tag{3.14}$$

2) Define (3.13) for the current time step:

Find the Lyapunov function by solving for P_i:

$$-I = A(t_i)^T P_i + P_i A(t_i) \tag{3.15}$$

3) Compute the maximum value of (3.13):

Solve the following constrained optimization problem:

$$V_i = \max_{\upsilon}\ \upsilon^T P_i \upsilon$$

$$0 = \upsilon^T P_{i-1} \upsilon - \hat{V}_i \tag{3.16}$$

β_t in (3.14) must be chosen so it is at all times negative. The maximum value of the Lyapunov function V_i is used both as the sufficiency criterion for asymptotic stability and also to define an ellipsoid in which the system is guaranteed to be found from time t_i to t_{i+1}. If an ordered subsequence V_j can be extracted from the sequence of V_i, and each member of V_j is monotonically decreasing relative to the respective member of V_{j-1}, then the system is uniformly asymptotically stable. In addition, if x_0 is chosen so that $x_0^T P_0 x_0 \leq V_0$, then $x(t)^T P_i x(t) \leq V_i$ for all t from t_i to t_{i+1}. Mullhaupt, Buccieri, and Bonvin supply proof of these guarantees in their statement of the algorithm.

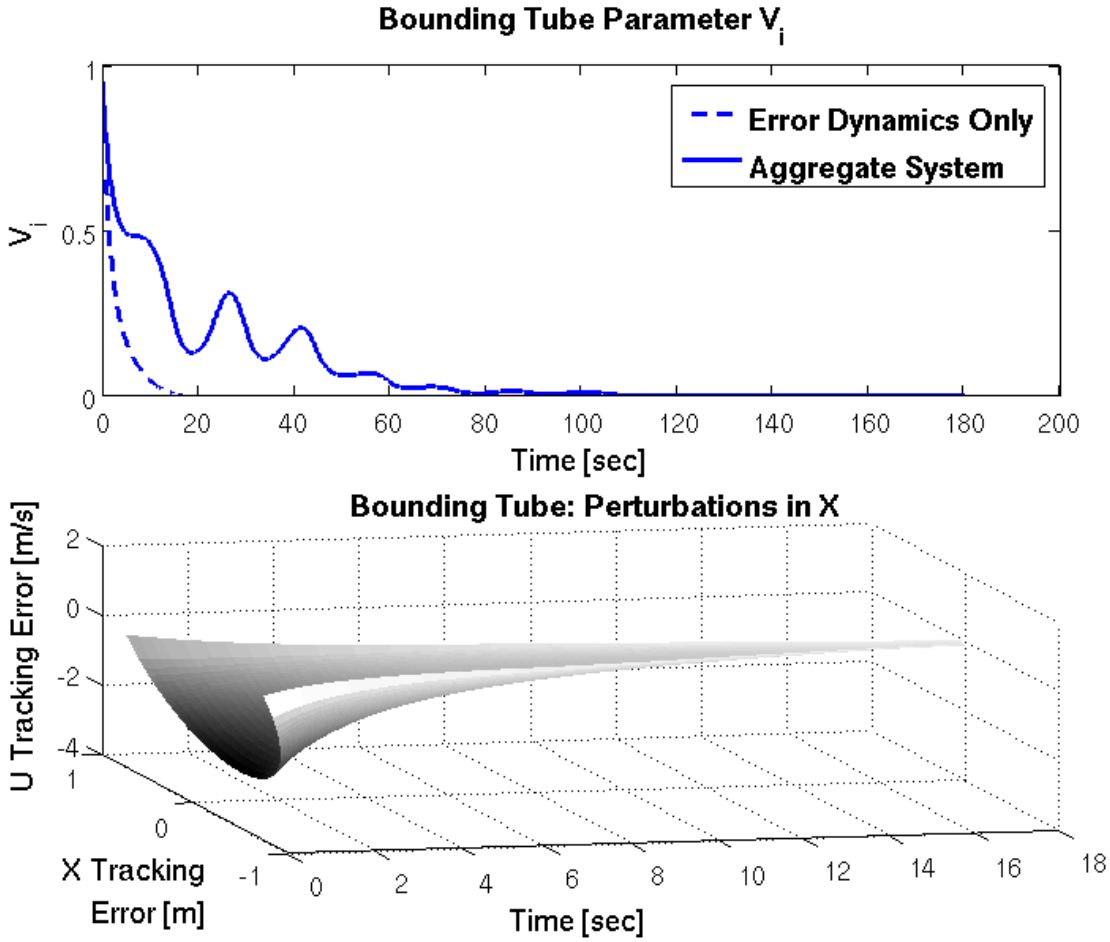

Figure 3-11: The bounding tube parameter V_i is shown for the four-point double integrator trajectory using the three-feature subset for map exploitaiton. A bounding tube for the system's estimation error in the surge direction is shown. The period of the four-point double integrator trajectory is slowed to 2 min, since faster trajectories failed to satisfy the sufficiency test.

Figure 3-11 shows the application of this sufficiency test to the four-point double integrator trajectory, which qualifies as uniformly asymptotically stable for the indicated

choice of parameters. The estimation error in the surge direction is explored using a bounding tube for u and x while the estimation error of the other four vehicle states is held at zero. The analysis isolates position and velocity in a single degree of freedom so that the bounding ellipsoid simplifies to a 2-D ellipse that varies in size along the nominal trajectory. Any estimation errors in u and x that begin within this bounding tube are guaranteed to stay within the tube. This test yields a quantitative bound for the stability of the map exploitation process, although unlike the transition matrix norm criteria, it is a sufficiency test only, and confirms stability only when a system is sufficiently slowly time-varying. Hence, the period of 16 seconds used in all other simulations of the four-point double integrator trajectory must be slowed to 120 seconds for the test to succeed. To demonstrate the sensitivity of this sufficiency test, the period of the vehicle simulation used in Figure 3-11 is decreased from 120 to 96 seconds, requiring a faster completion of the four-point double integrator nominal trajectory, and the bounding tube test fails to guarantee uniform asymptotic stability, with V_i diverging in Figure 3-12.

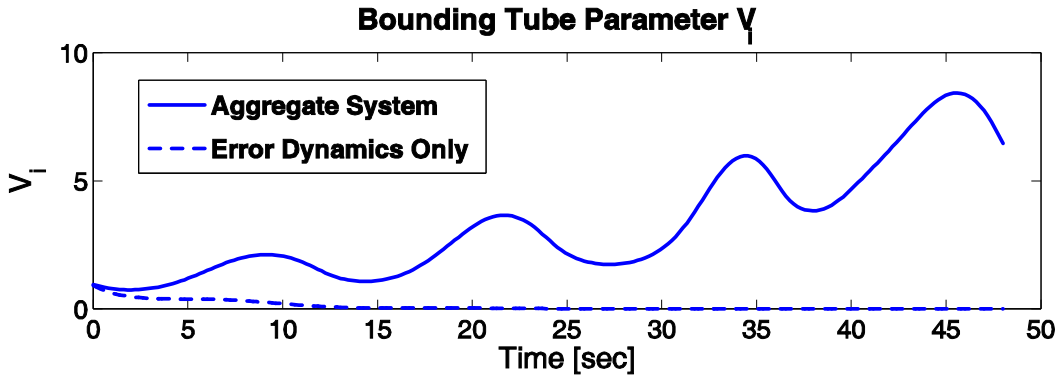

Figure 3-12: The bounding tube parameter V_i is shown for the four-point double integrator trajectory explored in Figure 3-12, with the period adjusted to 96 seconds to bring about failure of the sufficiency test.

Although a set of guaranteed boundaries on system behavior make this an appealing stability test, the requirements on the slowly-varying nature of the system to which the test is applied are limiting. Since future plans for autonomous ship hull inspection involve fast execution of aggressive survey trajectories, other techniques are pursued in greater depth for describing the stability and robustness of the integrated localization and control process.

3.6 Preparing for Experimentation

Looking ahead to the experimental implementation of integrated localization, mapping, and control on the raft platform, the two-point experimental trajectory introduced in Section 3.2 (Figure 3-3) is now considered. Rather than using double integrators, experimentally derived mass and damping parameters are now substituted into the vehicle model, and the vehicle is assigned a more conservative nominal trajectory designed for experiment. To predict the stability of the raft system in advance of any experiments, the transition matrix norm of the two-point experimental trajectory is computed and plotted along two cycles of the nominal trajectory for both map refinement and map exploitation, given by Figure 3-13. In addition, the feedback control gains are amplified to approximate the system's upward gain margin. The values of the model parameters and the feedback controller penalty matrices used for the two-point experimental trajectory are summarized in table 3.3.

Table 3.3: Penalty Matrices Used for Generating Optimal Feedback Control Gains Along the Two-Point Experimental Trajectory, and Raft Model Parameters

Raft Model Parameters:	Mass: 3.8 kg	Rotational Inertia: .08 kg m^2	Linear Damping: 2.2 Ns/m	Rotational Damping: .06 Nms/rad
State Penalty Matrix Q	$Q = diag\left[\frac{1}{.25^2} \quad \frac{1}{.25^2} \quad \frac{1}{.25^2} \quad \frac{1}{.01^2} \quad \frac{1}{.01^2} \quad \frac{1}{.01^2}\right]$			
Input Penalty Matrix R	$R = diag\left[\frac{1}{.03^2} \quad \frac{1}{.008^2} \quad \frac{1}{.003^2}\right]$			
Terminal State Penalty Matrix S	$S = diag\left[10^{12} \quad 10^{12} \quad 10^{12} \quad 10^{12} \quad 10^{12} \quad 10^{12}\right]$			

The upper left plot of Figure 3-13 displays, for the case of map refinement, the values of the transition matrix norm for the $\Phi(k,k_0)$ that transit from every initial time k_0 to every final time k along two cycles of the two-point experimental trajectory. The now-familiar qualities of an overshoot followed by convergence to a periodic surface can be observed for this trajectory. This plot permits the conclusion that the transition matrix norm can be assigned a bound M which it will not surpass, and that this bound can be taken independent of k_0. Since Theorem 3.1 is satisfied, uniform stability in the sense of Lyapunov is achieved for this map refinement scenario. In addition, the upper right plot of

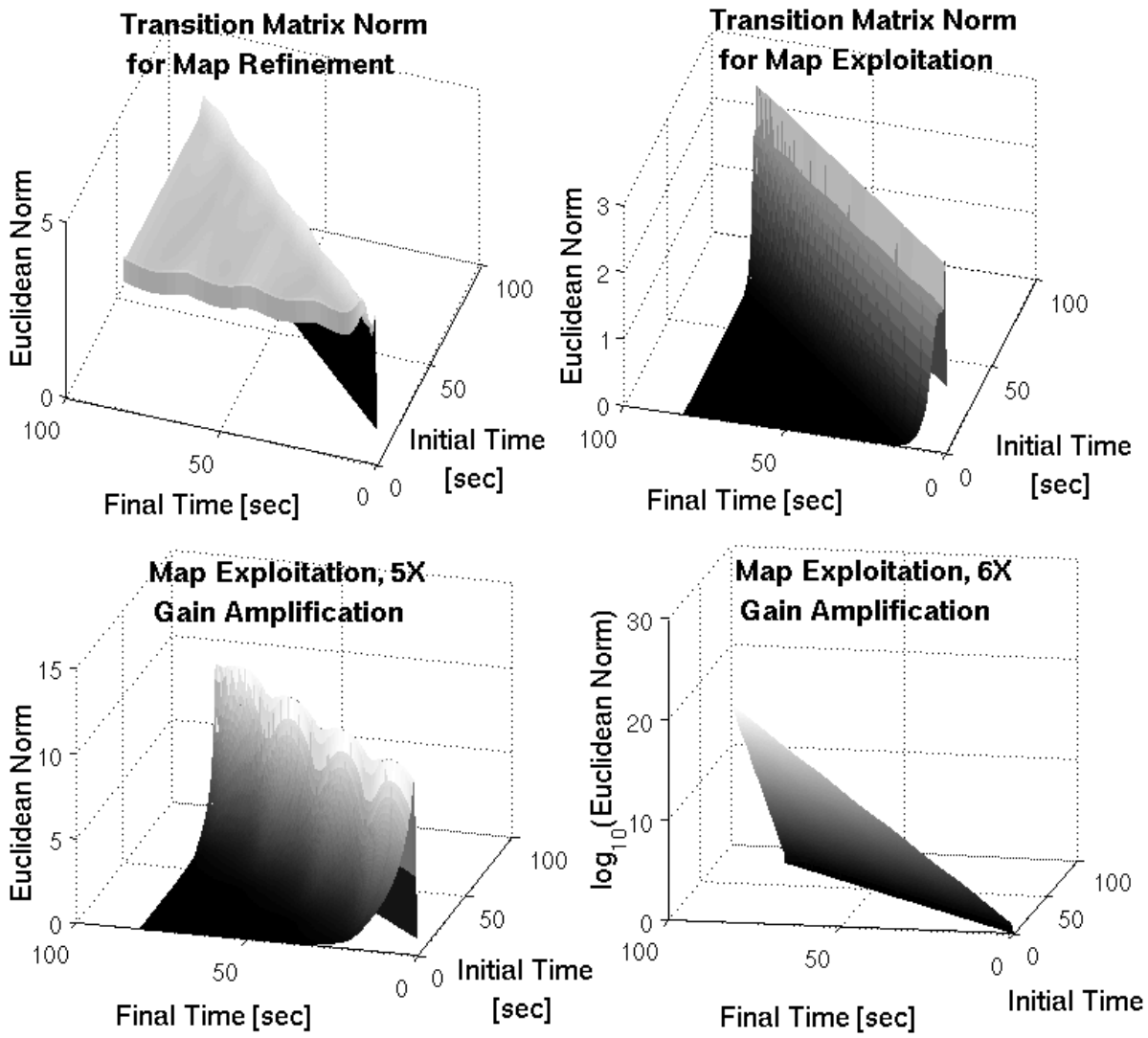

Figure 3-13: The transition matrix norm is plotted for the two-point experimental trajectory. Cases of both map refinement and map exploitation are considered. The plots encompass two complete cycles of the nominal trajectory, using the parameters identified in Figure 3-3.

Figure 3-13, which displays the norm for the case of map exploitation, demonstrates convergence to zero which occurs independently of k_0. Since Theorem 3.2 is satisfied, uniform asymptotic stability is achieved for this map refinement scenario. The bottom of Figure 3-13 displays transition matrix norm plots for a map exploitation case in which the nominal controller gains are amplified by factors of five and six, respectively. The system remains stable using a factor of five, but a factor of six is sufficient to render the system unstable, as $\Phi(k,k_0)$ is unbounded and grows rapidly with increasing k.

In the absence of perturbations, this system appears stable and robust, although

perturbations from the nominal trajectory must be considered explicitly before drawing this conclusion. The stability tools introduced in this chapter which rely on numerical computation of the closed-loop transition matrix norm, in addition to comprising a new technique for the evaluation of linear time-varying systems, will prove pivotal in enabling an analysis of perturbations that will identify ill-conditioned estimator configurations.

Chapter 4

Perturbed Marine Vehicle Stability Analysis

Analysis of the transition matrix of a marine vehicle performing integrated localization, mapping, and control has produced a numerical stability test that indicates whether or not the closed-loop system is stable in the absence of perturbations. It was shown in Section 3.3 that instability due to an ill-conditioned feedback controller is successfully predicted by this test. Although it is important to ensure the stability of the controller, analysis of the transition matrix norm will not always detect an ill-conditioned estimator. In particular, the method of Chapter 3 fails to consider the estimation challenges created by displacing the vehicle from the nominal trajectory. Once the vehicle is displaced from the nominal trajectory, it must successfully localize while facing linearization inaccuracies that worsen with the magnitude of the displacement. In addition, when the vehicle is displaced the deterministic series of measurements anticipated along the nominal trajectory will not arrive as expected, and the vehicle must use the surrounding features to infer how far it has been displaced. The difficulty of this task will vary depending on the layout of features on the *a priori* map.

A robustness performance metric is defined in this chapter which considers the effect of such perturbations on stability, and a method is introduced which permits a comparison of relative robustness among different system configurations. A specific example is presented in which the performance metric is used to assess the variation in robustness resulting from a variation in the geometric pattern of features on the map. This

chapter closes with a consideration of how this performance metric may be used to plan a path among a set of features that is optimized for robustness against perturbations.

4.1 The Perturbation Matrix

To understand the system's behavior in the presence of perturbations we must consider how the governing equations change when the vehicle is displaced from the nominal trajectory. Because a displacement from the nominal trajectory renders the linearization of the nominal state transition Jacobian F and measurement Jacobian H incorrect, correction terms ΔF and ΔH are needed to express the true location of the vehicle. Despite this, the need for correction is unknown to the estimator. Using these correction terms the propagation of the state and the estimate appears as follows:

$$\begin{aligned}
\delta \underline{x}_{k+1} &= (F_k + \Delta F_k)\delta \underline{x}_k - BG_k N\delta \hat{\underline{x}}_k + \Gamma \underline{w} \\
\delta \hat{\underline{x}}_{k+1} &= \delta \hat{\underline{x}}_{k+1|k} + K_{k+1}[\delta \underline{z}_{k+1} - H_{k+1}\delta \hat{\underline{x}}_{k+1|k}] \\
\delta \hat{\underline{x}}_{k+1|k} &= (F_k - BG_k N)\delta \hat{\underline{x}}_k \\
\delta \underline{z}_{k+1} &= (H_{k+1} + \Delta H_{k+1})\delta \underline{x}_{k+1} + \underline{\nu}_{k+1}
\end{aligned} \tag{4.1}$$

These equations can be manipulated to produce an aggregate system whose state vector is identical to that of (3.12), and all terms containing the correction matrices ΔF and ΔH can be collected in an aggregate perturbation matrix ΔA_k, where ΔA_k is given by (4.2):

$$\Delta A_k = \begin{bmatrix} \Delta F_k & 0 \\ [\Delta F_k - K_{k+1}(\Delta H_{k+1}(F_k + \Delta F_k + BG_k N) + H_{k+1}\Delta F_k)] & -K_{k+1}\Delta H_{k+1}BG_k N \end{bmatrix} \tag{4.2}$$

The aggregate perturbation matrix is combined with the state transition matrix of equation (3.12) as well as the process and measurement noise to express the perturbed system equations as follows:

$$\begin{bmatrix} \delta \underline{x}_{k+1} \\ \delta \tilde{\underline{x}}_{k+1} \end{bmatrix} = \begin{bmatrix} A_k \end{bmatrix} \begin{bmatrix} \delta \underline{x}_k \\ \delta \tilde{\underline{x}}_k \end{bmatrix} + \begin{bmatrix} \Delta A_k \end{bmatrix} \begin{bmatrix} \delta \underline{x}_k \\ \delta \tilde{\underline{x}}_k \end{bmatrix} + \begin{bmatrix} \Gamma & 0 \\ [I - K_{k+1}(H_{k+1} + \Delta H_{k+1})]\Gamma & K_{k+1} \end{bmatrix} \begin{bmatrix} \underline{w}_k \\ \underline{\nu}_k \end{bmatrix} \tag{4.3}$$

The state transition matrix of equation (3.12) is represented here by A_k. The solution of this difference equation for the system state at a specific time k can be found by multiplying successive transition matrices A_k, as was performed in Chapter 3. The notation $\Phi(k,k_0)$ will

once again be used to represent the transition matrix which propagates the system state from time k_0 to time k. Given the notation of (4.3), $\Phi(k,k_0)$ would be computed by multiplying all A_k from A_0 to A_{k-1}. This solution of (4.3) is given below by (4.4):

$$\begin{bmatrix} \delta \underline{x}_k \\ \delta \underline{\tilde{x}}_k \end{bmatrix} = \begin{bmatrix} \Phi(k,k_0) \end{bmatrix} \begin{bmatrix} \delta \underline{x}_{k_0} \\ \delta \underline{\tilde{x}}_{k_0} \end{bmatrix} + \sum_{i=k_0}^{k-1+k_0} \begin{bmatrix} \Phi(k-1-i,k_0) \end{bmatrix} \begin{bmatrix} \Delta A_i \end{bmatrix} \begin{bmatrix} \delta \underline{x}_i \\ \delta \underline{\tilde{x}}_i \end{bmatrix} \tag{4.4}$$
$$+ \sum_{i=k_0}^{k-1+k_0} \begin{bmatrix} \Phi(k-1-i,k_0) \end{bmatrix} \begin{bmatrix} \Gamma & 0 \\ [I - K_{i+1}(H_{i+1} + \Delta H_{i+1})]\Gamma & K_{i+1} \end{bmatrix} \begin{bmatrix} \underline{w}_i \\ \underline{v}_i \end{bmatrix}$$

By taking the Euclidean norm of both sides of this equation, (4.4) is transformed into an inequality that bounds the size of the aggregate state vector.

$$\left\| \begin{bmatrix} \delta \underline{x}_k \\ \delta \underline{\tilde{x}}_k \end{bmatrix} \right\| \le \left\| \begin{bmatrix} \Phi(k,k_0) \end{bmatrix} \right\| \left\| \begin{bmatrix} \delta \underline{x}_{k_0} \\ \delta \underline{\tilde{x}}_{k_0} \end{bmatrix} \right\| + \sum_{i=k_0}^{k-1+k_0} \left\| \begin{bmatrix} \Phi(k-1-i,k_0) \end{bmatrix} \right\| \left\| \begin{bmatrix} \Delta A_i \end{bmatrix} \right\| \left\| \begin{bmatrix} \delta \underline{x}_i \\ \delta \underline{\tilde{x}}_i \end{bmatrix} \right\|$$
$$+ \sum_{i=k_0}^{k-1+k_0} \left\| \begin{bmatrix} \Phi(k-1-i,k_0) \end{bmatrix} \right\| \left\| \begin{bmatrix} \Gamma & 0 \\ [I - K_{i+1}(H_{i+1} + \Delta H_{i+1})]\Gamma & K_{i+1} \end{bmatrix} \right\| \left\| \begin{bmatrix} \underline{w}_i \\ \underline{v}_i \end{bmatrix} \right\| \tag{4.5}$$

If the right side of (4.5) converges to zero as k approaches infinity, then the left side of the inequality is forced to converge in turn, and it may be concluded that (4.3) is asymptotically stable. If this convergence holds for any value of k_0, then (4.3) is uniformly asymptotically stable. An analysis by Chen and Dong in 1988 presented a condition required for the convergence of (4.5) and the consequent stability of (4.3) that depends on the size of the perturbation matrix ΔA [58].

Theorem 4.1. The null solution of equation (4.3) is uniformly asymptotically stable if two conditions are satisfied. First, the system must be uniformly asymptotically stable in the absence of perturbations, indicated by the following:

$$\left\| \Phi(k,k_0) \right\| \le m r^k \qquad (m > 0,\ 0 < r < 1)$$

This means that the Euclidean norm of the state transition matrix must be bounded by a discrete exponential with parameters m and r, and that this bound holds independently of k_0. Second, for a series of perturbation matrices ΔA_k, the following must also hold for all k:

$$\left\|\Delta A_k\right\| \leq \frac{1-r}{m}$$

If both conditions are satisfied, then the system will remain uniformly asymptotically stable in the presence of plant perturbations ΔA_k.

A detailed proof of Theorem 4.1 is presented by Chen and Peng [59]. Convergence of (4.5) is proven by applying the Bellman-Gronwall lemma, which allows an integral inequality to be expressed in terms of an exponential whose convergence may be assessed directly. A presentation of Theorem 4.1 for continuous time systems is given by Weinmann [60].

The conservative nature of Theorem 4.1 requires that the unperturbed system under consideration must be uniformly asymptotically stable, which excludes cases of map refinement from evaluation since these cases achieve Lyapunov stability at best. Consequently, map exploitation is the only control and estimation framework considered for the remainder of this analysis. A method is presented for choosing the map exploitation system configuration best equipped to handle a given perturbation of size ΔA_k , but it is also remains our goal to find the map refinement system configuration that is best equipped for perturbations. It is presently conjectured that the system configuration which is best-suited for robustness in cases of map exploitation will also be best-suited for robustness in cases of map refinement.

4.2 A Robustness Performance Metric

One challenge in implementing Theorem 4.1 is choosing appropriate perturbation matrices ΔA_k. In this analysis a displacement of constant magnitude is iteratively imposed on the vehicle at each sampling instant of the nominal trajectory. Because the inequalities of Theorem 4.1 must hold for all k, we need only find the single ΔA_k for a given displacement magnitude which yields the largest norm over the entire nominal trajectory, and if the r and m parameters match this norm we are guaranteed stability for the entire trajectory. The displacements are applied in one degree of freedom only, and robustness in each degree of freedom is considered separately. An example of how the displacements would be applied to our experimental nominal trajectory is given by Figure 4-1.

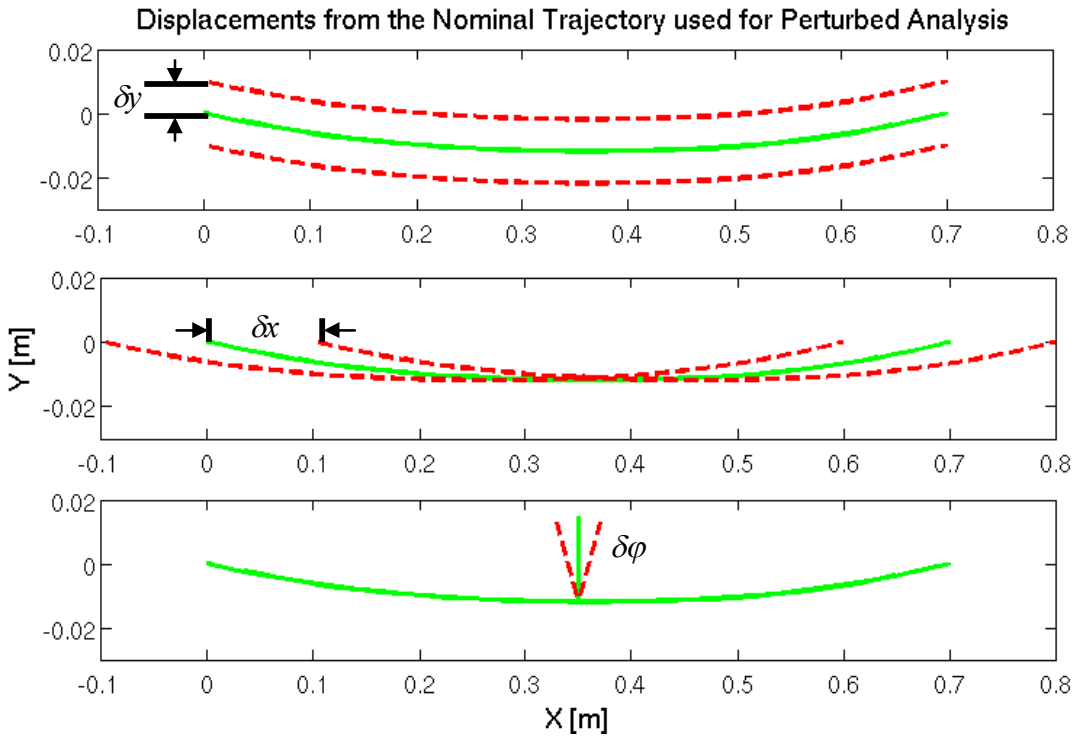

Figure 4-1: A visualization of displacements δy, δx, and $\delta \varphi$ imposed on the two-point experimental nominal trajectory. Once the displacement size is selected, each displacement is added and also subtracted from the nominal trajectory at each sampling instant. The resulting ΔA from each displacement is stored, and the single largest $||\Delta A||$ in each degree of freedom is selected for the application of Theorem 4.1.

The parameters r and m are obtained by bounding the surface plot of the $\Phi(k,k_0)$ Euclidean norm with an exponential function that the transition matrix norm will never exceed. As one can observe from the plots of Chapter 3, plotting the transition matrix norm from every choice of initial time k_0 to every choice of final time k along the nominal trajectory forms a 3-D surface, and the behavior of this surface must be captured by a 2-D exponential function. To achieve this, the value of k_0 which yields the single worst-behaved curve (i.e., the curve which contains the single largest value of the norm) between $\Phi(k_0,k_0)$ and $\Phi(k,k_0)$ is extracted, the curve is shifted so that k_0 is set to a value of one, and an iterative algorithm is employed to efficiently fit an exponential function of the form mr^k to this curve while exceeding it in value for all k. First a least squares fit is performed on the curve of the $\Phi(k,k_0)$ norm from k_0 to the final value of k, and then a series of exponential functions of the form mr^k are computed for a discrete range of values of r and m spanning from the least squares values to values that far exceed them. Each function is evaluated at

all values of k to determine whether the exponential function exceeds the transition matrix norm in value for the entire range of k, and the exponential functions that fail to achieve this are thrown out. Finally, the integral over k of each exponential function that successfully bounds the transition matrix norm is evaluated and the one with the smallest integral is chosen as the most efficient bounding curve. The values of r and m of this winning curve are applied to Theorem 4.1. Finding the most efficient bounding curve is important because smaller values of r and m are more likely to satisfy Theorem 4.1.

By comparing the parameters r and m of the exponential bounding curve with the norm of the largest ΔA_k, it is apparent when the system is guaranteed asymptotically stable. In general, the maximum perturbation magnitude for which Theorem 4.1 guarantees stability may be smaller than the maximum perturbation for which stability is attainable, since this is a sufficiency test only, and our algorithm chooses the worst-case ΔA_k over the entire nominal trajectory. Despite this, by comparing the stability guarantees obtained for different system configurations, the relative robustness of two trajectories, maps, or otherwise can be compared. In this manner the right side of the Theorem 4.1 perturbation inequality is used as a robustness performance metric.

4.2.1 Simulated Robustness Predictions

As an illustrative example, the effect of feature spacing on vehicle robustness can be investigated using this performance metric. For the simulations and experimental results to be presented, the experimental trajectory introduced in Figure 3-3 is used (along with the model parameters given in Figure 3-3 and Table 3.3), and the spacing of the three features on the *a priori* map is varied incrementally. Starting with three features condensed to a single point and gradually separating them until they achieve the configuration shown in Figure 3-3, the maximum perturbation magnitude for which stability is guaranteed can be computed for each feature configuration. This is demonstrated in Figure 4-2, in which the maximum value of $||\Delta A||$ is plotted for varying configurations of features, for varying perturbation magnitudes, and separately in each degree of freedom. The perturbation norm is compared with the r and m parameters of the exponential fit to $\Phi(k,k_0)$ for each configuration.

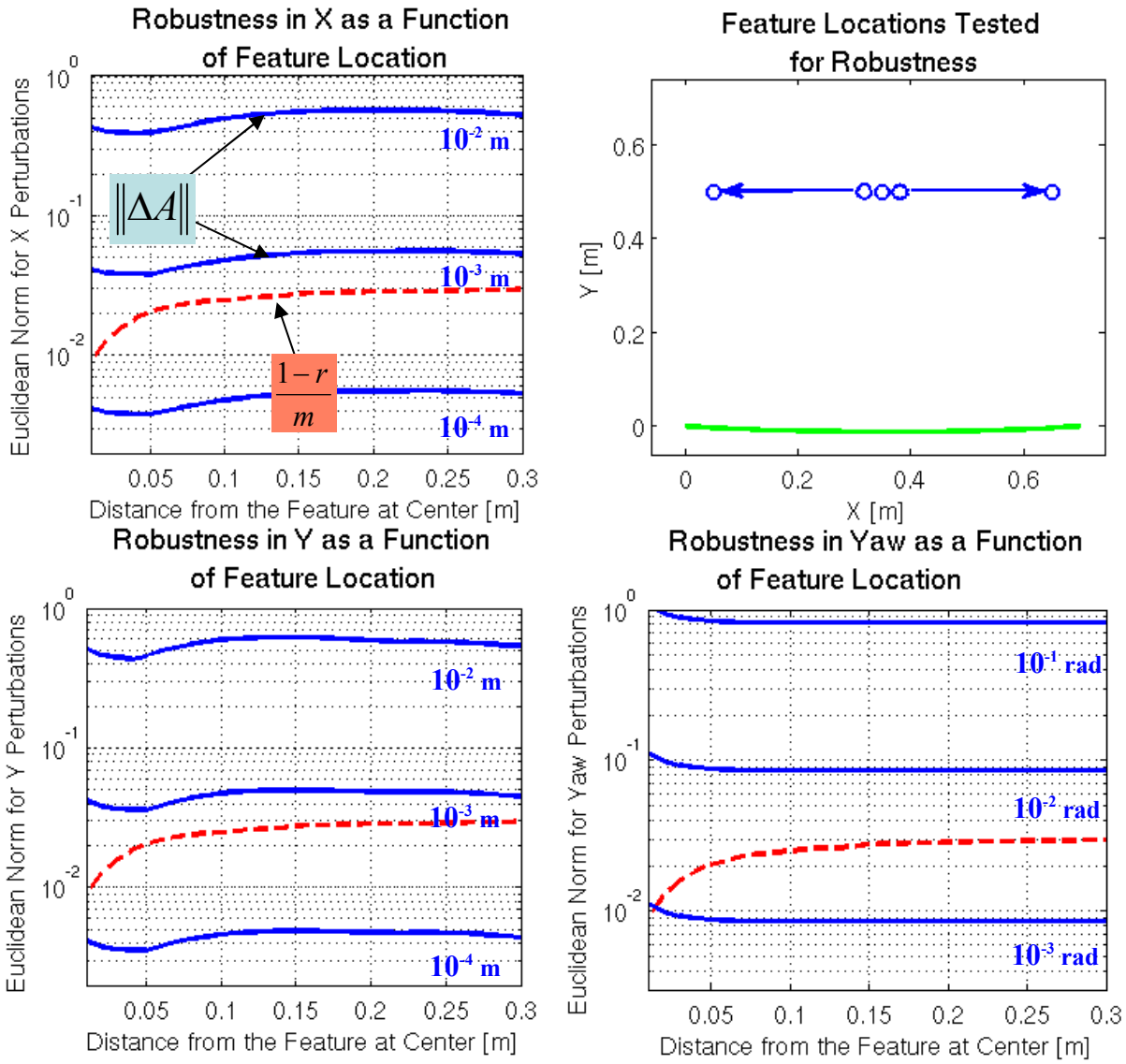

Figure 4-2: The perturbation matrix norm is plotted in each degree of freedom as a function of feature spacing for perturbations of incrementally varying magnitude (captured on the solid blue lines). Alongside each set of perturbation matrix curves is plotted the exponential bounding curve performance metric (the dashed red line), indicating the maximum perturbation size for which each feature layout is asymptotically stable. The smallest feature spacing considered in this plot is 0.01m, since a spacing of 0m yields a performance metric of zero, which cannot be expressed on the above logarithmic scale.

To offer an example of what we may conclude from these plots, the bottom right plot of Figure 4-2 shows that the vehicle is guaranteed stable against angular perturbations of order 10^{-3} radians for a feature spacing of 0.1m, since the performance metric (the red dashed curve) exceeds the norm for perturbations of this magnitude (the nearest solid blue curve). This guarantee cannot be made for a feature spacing of 0.01m, since the worst-case norm for perturbations of magnitude 10^{-3} radians exceeds the robustness performance

metric for this feature configuration. Hence, a vehicle observing features with a spacing of 0.1m is more robust than if it were observing features with a spacing of 0.01m, since it is guaranteed stable against larger-sized perturbations. The minimum-sized perturbation for which the marine vehicle is guaranteed stable would correspond to the value of $||\Delta A||$ that coincides exactly with the value of the robustness performance metric. Since it is not computationally feasible to plot $||\Delta A||$ curves for all possible perturbation sizes, plotting curves for a few values of $||\Delta A||$ is sufficient to understand the perturbation sizes for which a given system configuration is stable. In the plots of Figure 4-2 corresponding to perturbations in surge and sway, it is clear that none of our map configurations yield a system that could be guaranteed stable against perturbations of 10^{-3}m in size, but there are configurations whose performance metric comes a lot closer than others to the $||\Delta A||$ curve for 10^{-3}m-sized perturbations. The shape of the blue $||\Delta A||$ curves conveys clearly that there is a perturbation size between 10^{-4}m and 10^{-3}m for which features spread 0.3m apart can be guaranteed stable and features spread 0.01m apart cannot. By choosing a benchmark perturbation size and comparing the distance between $||\Delta A||$ and the robustness performance metric for different system configurations, the relative robustness of any two configurations can be compared.

In general, Figure 4-2 permits the conclusion that maps with a wider spacing between features are guaranteed stable against larger-sized perturbations, although the performance metric exhibits asymptotic behavior and there is a point beyond which no significant gains in robustness are achieved by spreading the features further. This point can be identified more clearly by choosing a perturbation size whose $||\Delta A||$ curve bounds the robustness performance metric, and normalizing the $||\Delta A||$ curve by the robustness performance metric. This was performed for the results of Figure 4-2, and the middle $||\Delta A||$ curve pictured in each plot was normalized by the performance metric curve for each degree of freedom. The results are displayed in Figure 4-3. A "knee" appears in each curve of this normalized performance ratio which confirms the previous intuition that beyond a spacing of approximately 0.05m between features, no significant gains are made in robustness.

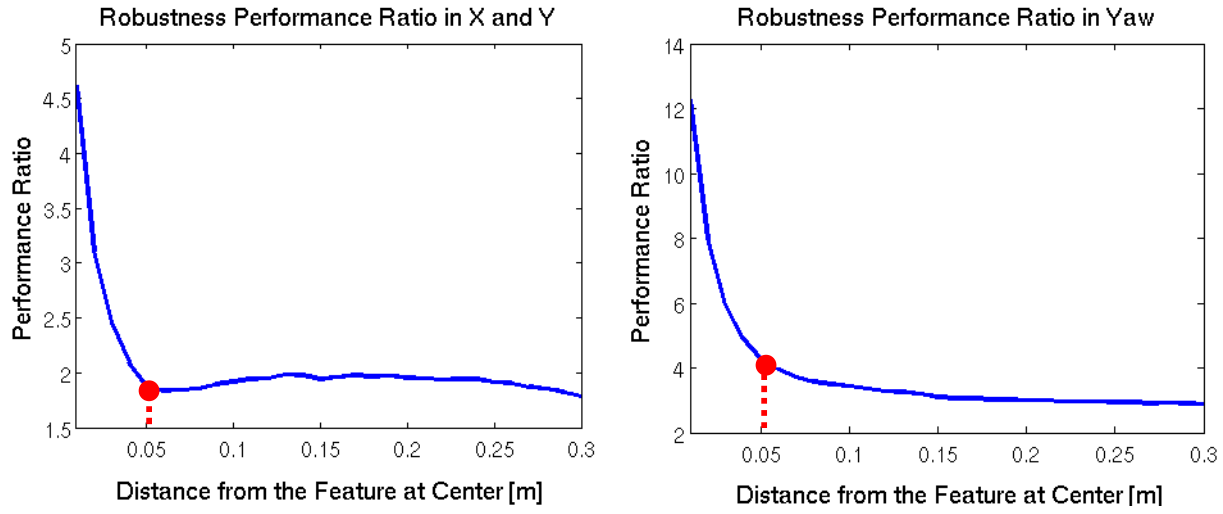

Figure 4-3: The ratio of perturbation norm $||\Delta A||$ to the robustness performance metric is plotted as a function of feature spacing using the data of Figure 5-2. The curve is identical in appearance in x and in y. A "knee" becomes evident after which the curve exhibits approximately asymptotic behavior, after which no significant gains in robustness are made by spreading the features further.

Although it may be an intuitive result that a vehicle performing localization and control is more robust if the features are spread further apart, Theorem 4.1 provides a framework through which this result may be formally proven. This will prove useful in more complex scenarios where it is unclear intuitively which of several map configurations can achieve the best stability guarantee. Before applying this method to a more complex scenario, the complete algorithm for making a robustness comparison among different system configurations is given by Algorithm 4-1.

4.3 Experimental Results

The predictions of Figure 4-2 were verified experimentally using the holonomic raft pictured in Figure 3-1. This vehicle carried out map exploitation and attempted to execute the two-point experimental nominal trajectory of Figure 3-3 using feature spacings of 0.3m, 0.05m, and 0m. These spacings were chosen since 0.3m is near the asymptotic upper limit of the performance metric in Figure 4-2, the performance metric for 0.05m is about 50% smaller than this asymptotic limit, and 0m has a performance metric of zero, since $\Phi(k,k_0)$ does not converge to zero when all three features share the same location (and so this case is never guaranteed asymptotically stable).

Algorithm 4-1: The Robustness Comparison Algorithm

1: Choose a system parameter p whose influence on robustness is to be analyzed.
2: **for** all values of p **do**
3: **for** all perturbation magnitudes δ **do**
4: **for** each degree of freedom f **do**
5: **for** each sampling instant along the nominal trajectory **do**
6: Displace the vehicle by positive and negative δ in f.
7: Compute the measurement Jacobian H in each location.
8: Subtract the nominal measurement Jacobian to obtain ΔH.
9: Compute the state transition Jacobian F in each location.
10: Subtract the nominal state transition Jacobian to obtain ΔF.
11: Compute ΔA according to (5.2).
12: Compute and store $||\Delta A||$, the Euclidean norm of ΔA.
13: **end for**
14: Store the maximum $||\Delta A||$ found along the nominal trajectory.
15: **end for**
16: **end for**
17: Compute $||\Phi(k,k_0)||$ from all k_0 to all k along a few cycles of the nominal trajectory.
18: Find the k_0 whose curve from k_0 to k yields the largest value of $||\Phi(k,k_0)||$.
19: Apply least squares to fit an exponential of the form mr^k to this curve.
20: **for** values of m from m_{lsq} to m_{max} **do**
21: **for** values of r from r_{lsq} to 1 **do**
22: **if** mr^k bounds the $||\Phi(k,k_0)||$ curve everywhere **then**
23: Compute and store the integral of mr^k.
24: **else then**
25: Throw out m and r.
26: **end if**
27: **end for**
28: **end for**
29: Find the minimum mr^k integral that was stored, this is the winning exponential.
30: Compute and store $(1-r)/m$ for this exponential function.
31: **end for**
32: Compare the worst-case perturbation norms $||\Delta A||$ for each p, f, and d with the robustness performance metric $(1-r)/m$ in all p and f.

Figure 4-4 demonstrates that the 0.3m spacing yielded the most effective closed-loop vehicle, as the raft adhered closely to the nominal trajectory and the estimator adhered closely to the vehicle's true location. The small errors that did occur in following the nominal trajectory were due to perturbations in the water tank testing environment and interference from the vehicle tether. The 0.05m spacing caused significant errors in the estimator, which at times remained closer to the nominal trajectory than to the

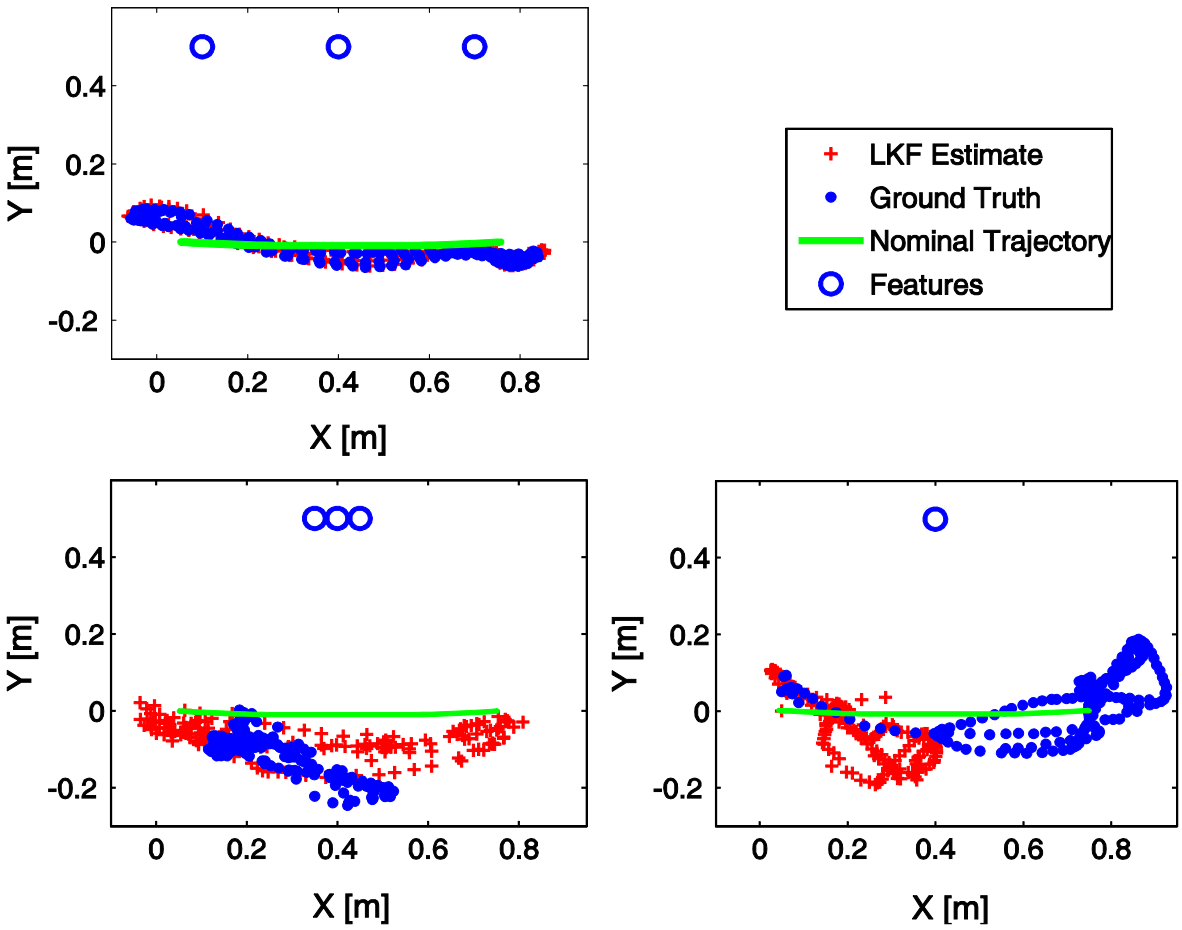

Figure 4-4: The raft platform was driven along the two-point experimental nominal trajectory for three different configurations of three point features. The feature locations, nominal trajectory, vehicle position estimates, and vehicle position ground truth are plotted for two complete cycles of the nominal trajectory.

vehicle's true location and thus the vehicle failed to execute the entirety of the nominal trajectory. In the 0m spacing case (which effectively consists of a single feature on the map) the estimator failed completely and the vehicle was driven off of the nominal trajectory. Figure 4-5 isolates performance in the angular degree of freedom and displays filter estimation error alongside the displacement of the raft from the nominal trajectory. Here it becomes clear that in the 0.05m and 0m cases, instability results, as the estimator failed to converge to the true location of the vehicle. If the estimator cannot accurately represent the true state of the vehicle, then the controller will issue inappropriate commands and the closed-loop system will fail to execute the nominal trajectory. Although Theorem 4.1 only provides a sufficiency condition and doesn't explicitly predict vehicle instability, in this case it called for selection of the map best-suited for localization and

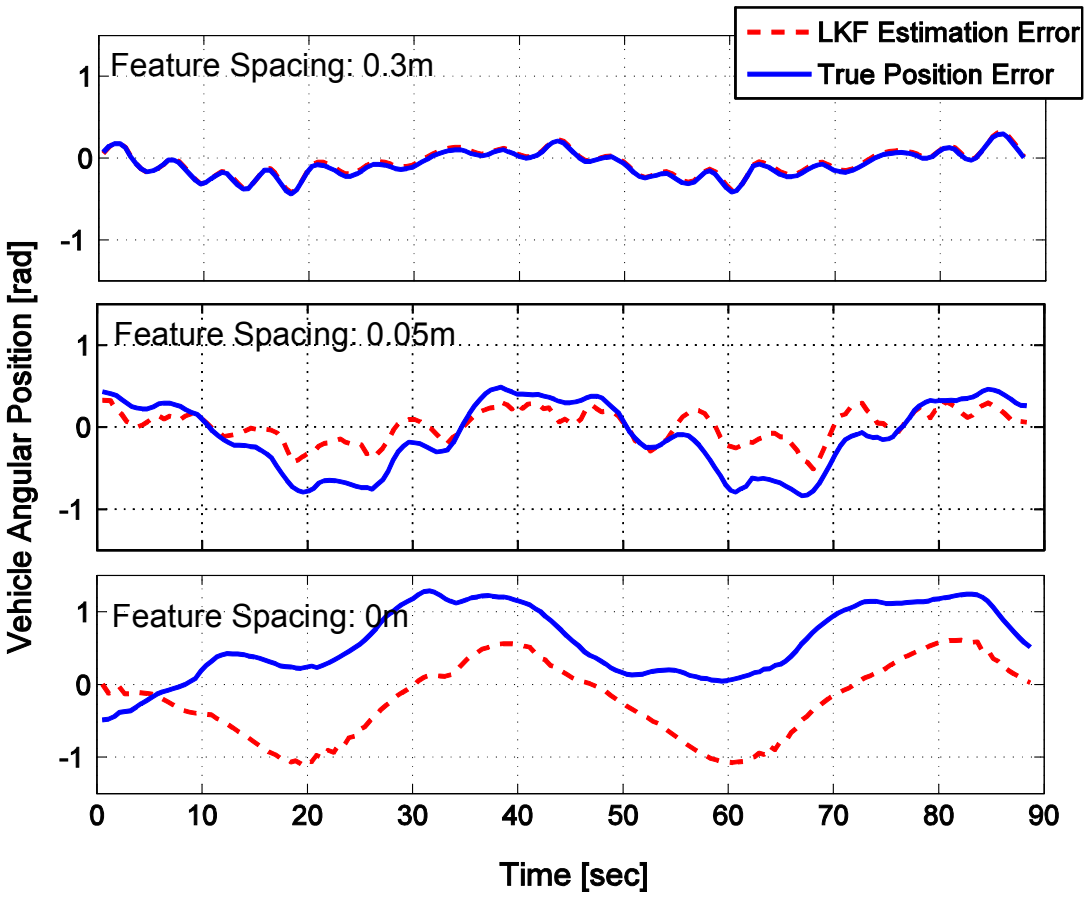

Figure 4-5: For the same three experimental trajectories displayed in Figure 4-2, estimator performance in the angular degree of freedom is isolated and estimation error is plotted alongside vehicle displacement from the nominal trajectory. All plots display data from two complete cycles of the nominal trajectory.

closed-loop control, avoiding less robust feature configurations which ultimately brought about system instability. Of course, in most autonomous vehicle scenarios it most likely the case that the map cannot be varied and it is instead the shape of the trajectory which is easiest to design for robustness considerations. Consequently, this method of robustness comparison is next extended to the problem of choosing the nominal trajectory that is most robust.

4.4 Application to a Real Ship Hull Inspection Scenario

Looking ahead to implementation of the robustness comparison algorithm on the HAUV itself, the robustness performance metric was used to investigate which of two candidate trajectories is a more robust ship hull survey trajectory. The HAUV hull survey trajectories

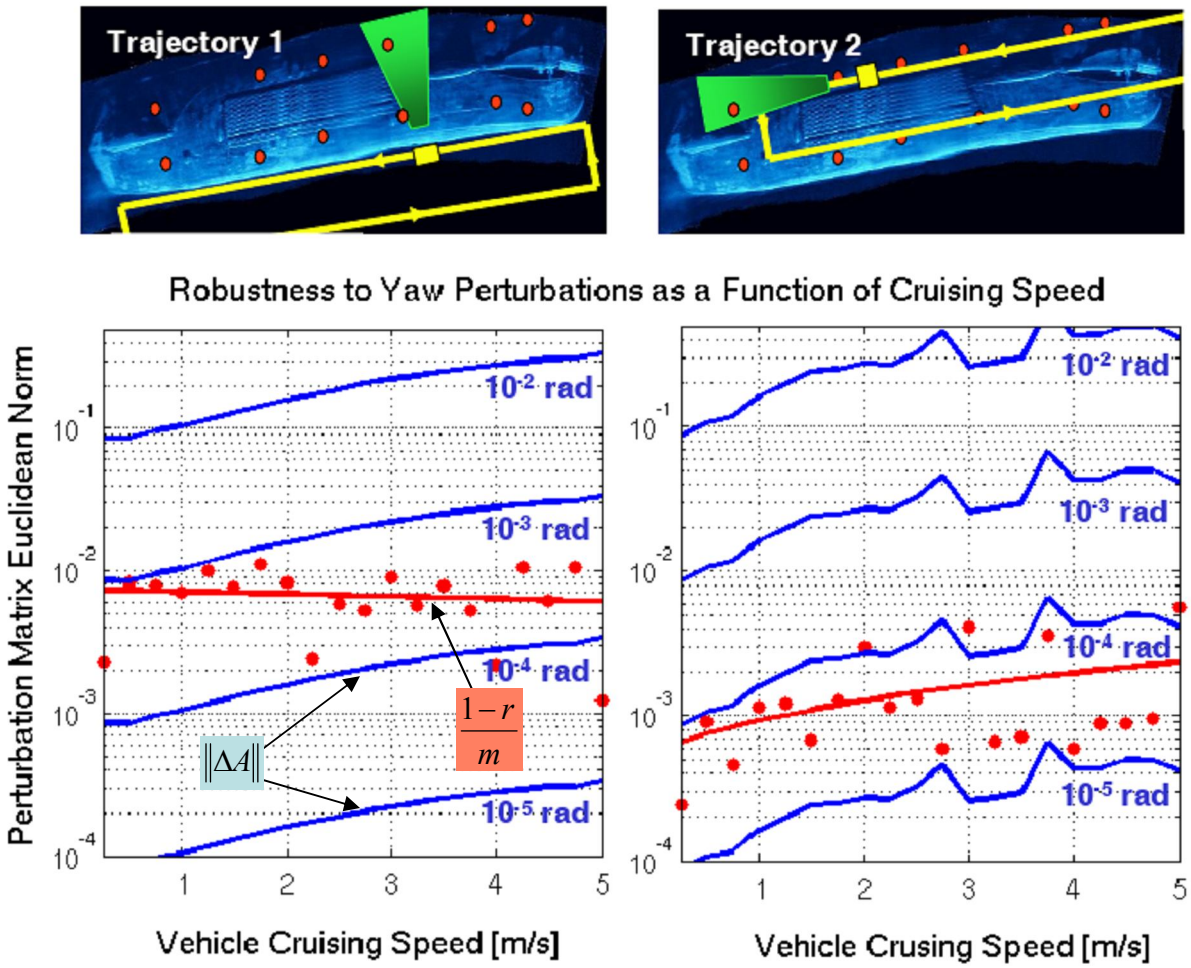

Figure 4-6: The robustness performance metric is used to evaluate two HAUV candidate trajectories. The perturbation matrix norm is plotted in the angular degree of freedom as a function of vehicle cruising speed for angular disturbances of incrementally varying magnitude (the solid blue lines). Alongside each set of perturbation matrix curves is plotted the exponential bounding curve performance metric, indicating the maximum perturbation size for which each hull survey trajectory is asymptotically stable (expressed as a series of red points with a best-fit line).

depicted in Figure 4-6 were selected heuristically based on simplicity and apparent hull coverage, and both have been used frequently in vehicle experiments performed on the hull that is pictured. The King Triton is a US Coast Guard Inland Buoy Tender that is approximately 30m in length and 7m in width, and the twelve zinc anodes mounted on the hull are easily recognized in imaging sonar data, making them ideal map features. For the two trajectories evaluated in Figure 4-6, the vehicle is only permitted to observe features that fall within a viewing window sized to reflect the viewing window of the DIDSON imaging sonar. Trajectory 1 keeps the viewing window oriented width-wise along the hull and Trajectory 2 keeps the viewing window oriented lengthwise along the hull.

The relative robustness of these trajectories is challenging to predict since the vehicle viewing window is limited in size and only a subset of the features will be observed at each measurement step. If the cruising speed of the vehicle is varied, the frequency at which features will be observed will vary in turn. The robustness performance metric was computed along each nominal trajectory for a variety of cruising speeds, and the perturbation matrix norm $||\Delta A||$ was computed for a variety of perturbation magnitudes in each degree of freedom. The dynamic model used to compute the transition matrix and perturbation matrix was identical in structure to that used for the raft, but experimentally derived inertial and damping parameters for the HAUV were used instead of raft parameters. These parameters along with tuning parameters used for the HAUV linearized Kalman filter are given in Table 4.1. An additional change from previous simulations is that odometry in each degree of freedom was added among the measurement capabilities since there are short instances along the hull survey trajectories where no features are visible.

Figure 4-6 contains plots showing the variation in $||\Delta A||$ and the variation in the robustness performance metric in the angular degree of freedom as the cruising speed of the vehicle is varied along each nominal trajectory (results in the other two degrees of freedom were similar in appearance). Unlike the clean and smooth results of Figure 4-2, the robustness performance metric tends to jump around in value as the vehicle cruising speed is varied. This is because the vehicle observes an entirely different sequence of features at each cruising speed, and the varying system configurations are not as tightly linked as they were for the raft scenario explored in Figure 4-2. To obtain an idea of the general behavior of the robustness performance metric for each trajectory, a least squares fit was performed on the data and the resulting best-fit line is plotted alongside the raw data. Using this best-fit line as an approximation for the behavior of the robustness performance metric, the plots demonstrate that Trajectory 1 is guaranteed to tolerate perturbations approximately an order of magnitude greater than those for which Trajectory 2 is equipped. For almost all nominal cruising speeds Trajectory 1 is guaranteed stable against perturbations of 10^{-4} radians in size, while Trajectory 2 achieves this guarantee for very few cruising speeds. Hence, the performance metric would indicate that

Trajectory 1 is the robustness-optimal choice from among the two candidates, as it is guaranteed to yield asymptotic stability for larger-sized perturbations than Trajectory 2.

Table 4.1: Parameters used for HAUV Robustness Simulation

DIDSON Viewing Window Parameters:	Min Visible Range: 2m	Max Visible Range: 7m	Min/Max Heading: $\pm$ 15 degrees
Discretization:	Sampling Interval: 0.5 s		
Vehicle Process Noise Parameters:	Process noise variance, surge: 0.1 $(m/s^2)^2$	Process noise variance, sway: 0.1 $(m/s^2)^2$	Process noise variance, roll: 0.1 $(rad/s^2)^2$
Odometry Sensor Noise Parameters:	Sensor noise variance, surge odometry: 0.1 $(m/s)^2$	Sensor noise variance, sway odometry: 0.1 $(m/s)^2$	Sensor noise variance, roll odometry: 0.1 $(rad/s)^2$
Feature Detection Noise Parameters:	Sensor noise variance, vehicle-relative range: 0.01 $(m)^2$	Sensor noise variance, vehicle-relative bearing: 0.01 $(rad)^2$	
HAUV Model Parameters:	Mass: 113 kg	Rotational Inertia: 8 kg m^2	Linear Damping: 55 Ns/m	Rotational Damping: 3 Nms/rad
Initial Vehicle Error Covariance:	$P_0 = diag \begin{bmatrix} 10^{-1} & 10^{-1} & 10^{-1} & 10^{-1} & 10^{-1} & 10^{-1} \end{bmatrix}$			
Initial Map Confidence:	$R_C = diag \begin{bmatrix} 10^{-4} & 10^{-4} & 10^{-4} & 10^{-4} & 10^{-4} & 10^{-4} \end{bmatrix}$			
State Penalty Matrix Q	$Q = diag \begin{bmatrix} \frac{1}{.25^2} & \frac{1}{.25^2} & \frac{1}{.25^2} & \frac{1}{.001^2} & \frac{1}{.001^2} & \frac{1}{.001^2} \end{bmatrix}$			
Input Penalty Matrix R	$R = diag \begin{bmatrix} \frac{1}{.05^2} & \frac{1}{.05^2} & \frac{1}{.05^2} \end{bmatrix}$			
Terminal State Penalty Matrix S	$S = diag \begin{bmatrix} 10^{12} & 10^{12} & 10^{12} & 10^{12} & 10^{12} & 10^{12} \end{bmatrix}$			

The evaluation of a real ship hull inspection scenario demonstrates the versatility of the robustness comparison algorithm presented in this chapter. Systems with a wide variety of constraints can be accommodated, such as the addition (or lack of) odometry, a sensor footprint of highly specific geometry, and even a time-varying number of incoming measurements. The only hard requirement of the robustness comparison algorithm is that the closed-loop system must be expressed within a linear time-varying state space framework and must be capable of achieving asymptotic stability in the absence of perturbations.

Although the most robust trajectory was selected from among the two candidates evaluated, there is of course the possibility that there are trajectories better-suited for robustness than those evaluated here. Consequently, the challenge of robustness-optimal motion planning is addressed in the next chapter. Rather than heuristically choosing a series of candidate trajectories and evaluating their relative robustness, a trajectory is designed and optimized for robustness simultaneously, using the robustness performance metric to evaluate the trajectories that result from different control choices and making these control choices accordingly.

Chapter 5

Robustness-Optimal Motion Planning

The process of finding the most robust closed-loop system configuration for a holonomic marine vehicle is next combined with a globally optimal kinodynamic motion planning algorithm. As introduced in Chapter 4, a system configuration is designated the most robust configuration if it achieves a guarantee of asymptotic stability against a larger sized-perturbation than other competing system configurations. Rather than heuristically choosing a handful of system configurations and evaluating their relative closed-loop robustness, as was done in Chapter 4, a modified A* algorithm is used to select the most robust trajectory from among all trajectories in the vehicle state space that connect a given start waypoint and a given goal waypoint. The robustness-optimal trajectories selected by the kinodynamic planning algorithm are adapted for experimental implementation on the raft and compared with A* trajectories that do not consider the robustness performance metric.

5.1 Augmented A* Algorithm

As described in Chapter 1, the A* algorithm finds the optimal path between a start node and a goal node on a graph given a cost function that consists of a cost-to-come, which is the cost incurred along the path that has been traveled from the start node, and a cost-to-go, which approximates the cost to be incurred by traveling to the goal node. This standard A* cost function is given by (5.1):

$$C_{total} = C_{come} + C_{go} \tag{5.1}$$

The simplest implementations of A* plan on graphs which represent the 2-D spatial plane. Although in many cases the ship hull inspection problem can be idealized as a planning problem in the 2-D plane, spatial planning completely neglects the marine vehicle's dynamic model, hence neglecting the control effort and time required to move the vehicle from one location to another along a ship hull. Even though a straight-line trajectory between the start and the goal might be the most efficient in terms of path length, effort, and time, a purely spatial path does not inform the vehicle operator of how to command the vehicle from the start to the goal. For this reason, the approach of kinodynamic planning is adopted and A* is employed to search the vehicle's control input space.

5.1.1 Kinodynamic Planning in Control Input Space

By planning in the space of possible vehicle control actions, a trajectory between a start configuration and goal configuration may be found which rewards efficient use of control effort rather than simply choosing the shortest spatial path. Because searching the entire continuous space of possible control inputs is computationally infeasible, a set of nine discrete input choices is selected. Once issued, each input command must be applied for a fixed time interval before the next command may be delivered. To further improve the computational feasibility of searching the control input space, the vehicle is not allowed to make any adjustments to its heading, and can actuate only in surge and sway. The cost associated with each input choice is heuristically formulated to penalize the expenditure of energy and of time. Figure 5-1 displays the nine discrete control input choices and their associated costs tabulated in a spatial grid. In this grid, a positive surge thrust is located one unit to the right from the center, and a negative surge thrust is located one unit to the left from the center. A positive sway thrust is located one unit up from center, and a negative sway thrust is one unit down. Applying positive thrust in surge and sway corresponds to the grid's upper right corner, and so on. The location of these input choices on the nine-by-nine spatial grid corresponds approximately to the energy required to implement them. Costs are assigned to the eight outermost input choices based on the Euclidean distance of each grid entry from the center. The center square is assigned a non-zero cost so the vehicle does not find sitting idle to be the cheapest action. In addition, this

cost is less than the cost of single-actuator thrust so the vehicle will opt to coast along without using additional thrust once it is close to the goal, instead of hitting the goal at top speed.

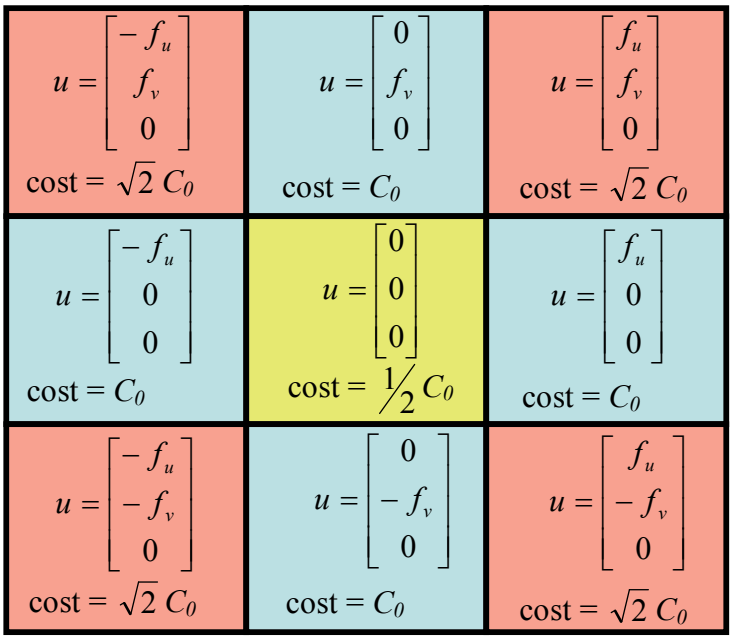

Figure 5-1: The discrete input choices used for implementation with A* are displayed in a spatial grid. The distance of each choice from the center corresponds to the relative energy required to enact its input thrust command. Input choices with equivalent cost are assigned the same color.

5.1.2 Tree Graph

An A* algorithm which plans in the vehicle's input space demands a special type of graph structure. Unlike taking steps in a spatial grid, the ordering of the input commands applied to the vehicle is not commutative, and so applying a surge thrust, followed by a sway thrust will not leave the vehicle in the same state as applying a sway thrust, followed by a surge thrust. The vehicle state matters because the start and goal nodes of the planning algorithm correspond to vehicle poses in the 2-D plane, and even though the planning occurs in the input space, the resulting vehicle state after each input thrust must be stored for comparison with the goal configuration. Each path between nodes represents a series of input commands which has defined a completely unique location for the vehicle, and encoded in each node is not only an input command, but a vehicle state which was produced by applying this series of commands. As a result, the propagation of the graph

from a given node may only radiate outward from the node and can never wind back to previously visited nodes as would be the case for 2-D spatial planning.

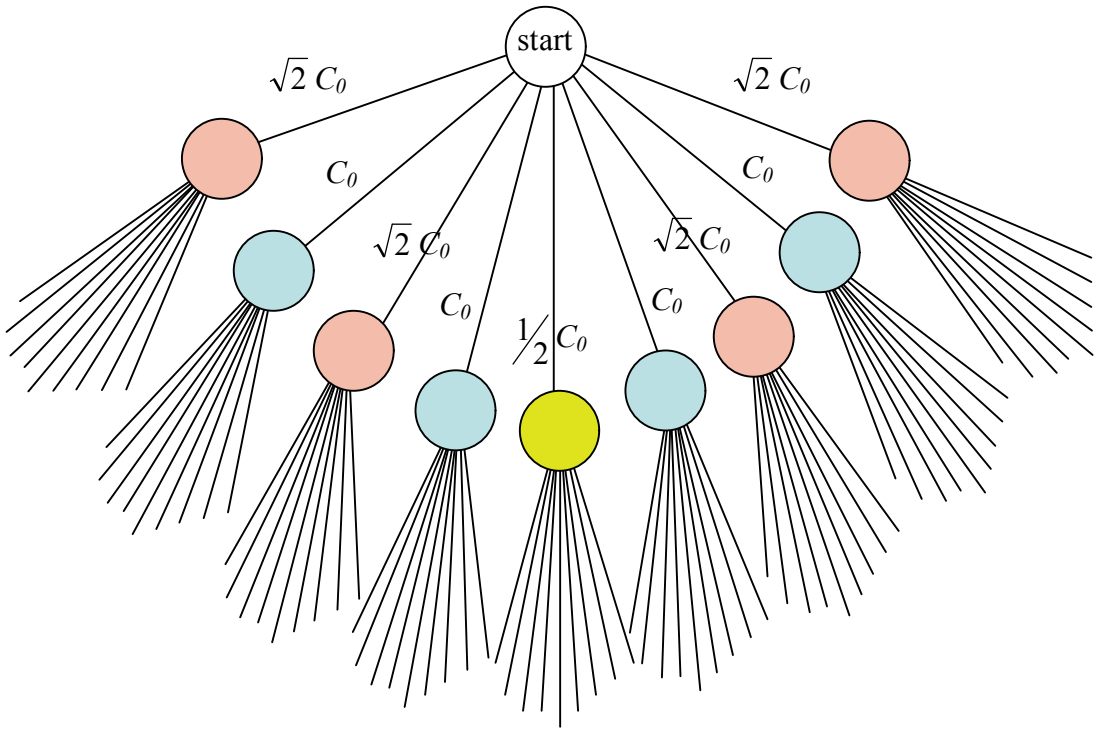

Figure 5-2: A diagram representing the evolution of the A* tree graph. Each node connected to the start node represents one of the nine input commands from the input grid of Figure 5-1. Nine edges then sprout from each of these nine nodes, connecting to all possible subsequent input commands. The cost-to-come associated with each graph connection is listed.

This requirement necessitates the use of a tree graph, which at the addition of each node creates a branch that splits off from the rest of the graph and may only propagate by growing in length and dividing into additional branches. These branches may only give rise to new nodes and may not rejoin any of the nodes which already exist on the graph. Figure 5-2 demonstrates pictorially the process of planning over a tree graph for the holonomic marine vehicle under consideration. The colors of the nodes in this graph correspond to the grid entries of Figure 5-1. From the start configuration, each of the nine possible input commands is considered, and the application of nine possible commands after the first command is considered next, causing the tree to expand rapidly into many diverging branches, each of which represents a unique combination of input commands. This graph structure was adapted from a similar strategy used by Greytak for a nonholonomic marine vehicle [61]. Because many hundreds of command sequence possibilities can be generated

in planning over a relatively small space, the search must be informed by a carefully chosen cost-to-go function that aids in directing the evolution of the tree to the goal configuration.

5.1.3 Cost-to-Go Heuristic

To ensure that the A* algorithm converges quickly to the goal, an admissible heuristic must be employed which accurately approximates the cost required to reach the goal without over-estimating the true minimum cost of travel to the goal node from the node under consideration. To achieve this, three simplifying assumptions are made. First, all hydrodynamic drag forces are neglected. It is assumed that the marine vehicle behaves as a double integrator in each degree of freedom. Less input thrust is required to move a double integrator from one location to another than to move an equivalent mass with any amount of hydrodynamic drag, and so this assumption will aid in producing an underestimate. The second simplifying assumption is that regardless of the vehicle pose at the node being evaluated, it is assumed that a thruster is pointing directly at the goal so that the double-thruster cost will never be applied in generating the cost-to-go. The only input cost that will be used is the cost associated with running a single thruster. The third assumption projects the vehicle velocity onto a vector connecting the vehicle's current position and the goal, assuming that the vehicle has no velocity that is not aligned with the goal. The only thrust required by the heuristic will be thrust directed at the goal. The resulting procedure for computing the cost-to-go is given by (5.2):

$$t_{goal} = \frac{-V_0 + \sqrt{V_0^2 + 2\frac{F_0}{m}d}}{\frac{F_0}{m}} \tag{5.2}$$

$$C_{go} = floor(t_{goal} / T_{plan}) * C_0$$

This procedure directly computes the time required to reach the goal from the current vehicle position, t_{goal}. This closed-form solution is possible because of the double integrator assumption. V_0 represents the net velocity in the direction of the goal, which is positive when the vehicle is moving toward the goal and negative when the vehicle is moving away from the goal. The distance between the current position and the goal is d, and F_0

represents the fixed amount of thrust that may be applied by a single thruster. The mass of the vehicle is m, T_{plan} is the fixed time interval over which each input command is applied, the "floor" function rounds its argument down to the nearest integer, and C_0 is the baseline unit of cost introduced in Figure 5-1. It has not been proven that this procedure yields a cost-to-go heuristic that is always admissible, but it has thus far yielded satisfactory planning results and fast algorithm convergence. The only possibility left to chance by this heuristic is that it is sometimes the case that thrusting, followed by coasting, is truly the most cost-efficient behavior. Since computing the exact time to turn off the thrusters and begin coasting in the computation of C_{go} is yet another optimization problem, the simplifying assumptions described above are used instead to offset any unanticipated possibility of getting to the goal at a lower cost by coasting.

5.1.4 Robustness-Augmented Cost Function

The final and most unique feature of this kinodynamic A* algorithm is the consideration of robustness against perturbations to the marine vehicle. In addition to the cost-to-come and the cost-to-go, the cost function used in this analysis includes a term which evaluates the robustness of the path from the start node to the current node by applying the performance metric introduced in Chapter 4. This new cost is concerned only with the specific path traveled so far, and so it is an additional cost-to-come. The complete cost function is given by (5.3):

$$C_{total} = C_{come} + C_{go} + C_{robust,\ come} \tag{5.3}$$

$$C_{robust,\ come} = W * \log_{10}(\|\Delta A\|_{\max} - \frac{1-r}{m})$$

The robustness component of the cost function consists of a tunable weight W multiplying the base ten logarithm of the difference between a perturbation matrix norm and an exponential performance metric obtained from the vehicle transition matrix. The robustness parameters are obtained by designating the trajectory under consideration between the goal node and the current node to be the vehicle's nominal trajectory. A series of LKF and feedback control gains are computed for this nominal trajectory so that a closed loop transition matrix may be formulated for the system. The perturbation matrix is

computed according to (4.2) for a benchmark perturbation size selected for the entire duration of the algorithm. The perturbation is applied to the vehicle in positive and negative x and y at each sampling instant of the trajectory, and the ΔA matrix with the largest Euclidean norm over the entire nominal trajectory, over all perturbations considered, is selected for the robustness cost function. A large perturbation size is selected so that ΔA is always greater than the robustness performance metric and the argument of the logarithm in (5.3) is always positive. The exponential parameters r and m, which were first introduced in Theorem 4.1, are obtained by bounding the marine vehicle's transition matrix norm along the A*-formulated nominal trajectory with an exponential function. In this procedure, the norm of $\Phi(k,k_0)$ from every instant k_0 to every instant k along the entire nominal trajectory is no longer considered. Instead r and m are obtained by fitting an exponential function to $||\Phi(k,k_0)||$ for a single k_0 which corresponds to the start node. Rather than $\Phi(k,k_0)$ producing a 3-D surface as was the case in Chapters 3 and 4, it is now a simple 2-D curve that is considered, motivated primarily by a desire for computational efficiency. Many hundreds of nominal trajectories might be evaluated over the course of the A* algorithm, and computing the norm of $\Phi(k,k_0)$ for every possible k and k_0 is an expensive task.

The implication of considering only a single k_0 is that, as per Theorem 3.2, a guarantee of uniform asymptotic stability is no longer sought, but merely asymptotic stability instead. Theorem 4.1 can now only be applied to investigate stability for the specific k_0 chosen, and not for all k_0. This is viewed as an acceptable compromise since the vehicle will presumably be initialized at the start node, and not at another location along the path computed by A*. If the vehicle is initialized at the start node, then the cost function of (5.3) will be a valid predictor of robustness.

A* Algorithm

The ultimate goal of this A* algorithm is to achieve a planning procedure which guides the vehicle from the designated start waypoint to the designated goal waypoint, but does so in a manner which may divert from the most energy-efficient path in order to gain

robustness against perturbations. As the unique features of the algorithm have each been presented in detail, the algorithm in its entirety is now given by Algorithm 5-1. This algorithm is next applied to the raft vehicle model and used to examine the impact of the robustness weight W of (5.3) and the choice of start node and goal node on the resulting optimal path for a given set of features.

Algorithm 5-1: The Robustness-Augmented A* Algorithm

1: Add the start node to the queue Q.
2: **while** $Q > 0$ **do**
3: Choose the node n from Q with the lowest cost.
4: **for** each of the nine input command choices u **do**
5: Propagate the system state at node n by applying u for time T.
6: Create a node n' adjacent to n corresponding to the new system state.
7: Compute the cost-to-come, C_{come}, by adding the cost of u to the C_{come} of node n.
8: Compute the cost-to-go, C_{go}, using (5.2).
9: Compute $C_{robust,\ come}$ of the path from the start node to n' using (5.3).
10: The total cost of n' is $C_{total} = C_{come} + C_{go} + C_{robust,\ come}$.
11: **end for**
12: Add n to the list of dead nodes, D, and remove n from Q.
13: **for** all nodes d in D **do**
14: **if** d is the goal **then**
15: Add d to the list of goal-reaching paths, G, and remove from D.
16: **end if**
17: **end for**
18: Find the node in G with the minimum cost, g_{min}.
19: Find the node in Q with the minimum cost, q_{min}.
20: **if** the cost of q_{min} is greater than the cost of g_{min} **then**
21: $Q = 0$, algorithm will terminate.
22: **else then**
23: **for** all nodes q in Q **do**
24: **if** cost of q is greater than cost of g_{min} **then**
25: Remove q from Q and add to D.
26: **end if**
27: **end for**
28: **end if**
29: **end while**

5.2 Performance of Robustness-Augmented A*

A map of three collinear features is selected to test the augmented A* algorithm and to allow experimental implementation on the raft. A similar set of features was used in the experiments of Chapter 4, although in the analysis and experiments of this chapter the spacing between features remains fixed at 0.1m and the vertical position of the features is varied. Figure 5-3 displays the three-feature map, the start and goal waypoints, and the trajectories produced by A* with and without a robustness cost for three different separation distances between the features and the waypoints.

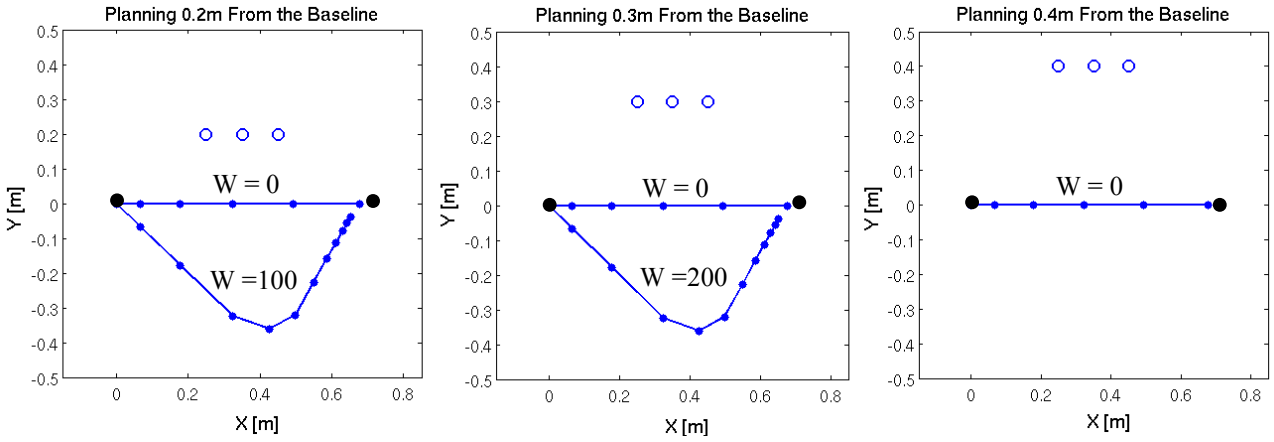

Figure 5-3: Results from regular and robustness-augmented A* when the start and goal waypoints are separated by a horizontal distance of 0.7m. Three cases with different vertical distances from the three-feature baseline are considered. A robustness weight of 100 is required to produce the curved trajectory for the 0.2m case, a weight of 200 is required to produce the curved trajectory for the 0.3m case, and no weight is capable of producing a curved trajectory in the 0.4m case. The points plotted represent the location of the vehicle at each sampling instant along the A*-planned path. The start and goal waypoints are plotted in black.

In the absence of a robustness cost (achieved by setting the weight W from (5.2) to zero), A* yielded a horizontal straight-line trajectory between the two waypoints. This makes intuitive sense as it achieves the shortest spatial distance between the two points and it can be achieved through the cheapest and most basic application of thrust. In each case where *W* was set to zero the algorithm used two consecutive applications of the surge thruster to reach the goal. By setting the robustness weight to larger values, a curved trajectory eventually emerged which departed from the energy-efficient path to stay in more robust locations relative to the features. In all locations where the curved path emerged, it consisted of a simultaneous application of thrust in positive surge and negative

sway, followed by an application of positive thrust in sway, after which the vehicle refrains from using thrusters and coasts to the goal for the remainder of the trajectory. Relative to the 0.2m vertical separation between the waypoints and the three-feature baseline, the 0.3m separation required twice the weighting factor to force the vehicle away from the regular A* path, and no weight was found which could force the trajectory with 0.4m separation to curve its path. These results suggest that as the three-feature baseline is moved closer to the designated waypoints, the regular A* trajectory becomes less and less robust.

The three-feature horizontal baseline was selected because of the anticipated challenges of performing accurate estimation on the far left or right side of this map. Due to the small differences in relative bearing between the vehicle each of the features, it was expected that a perturbation in this region could easily throw off the entire measurement Jacobian. As numerical computation revealed, the measurement Jacobian is significantly more sensitive to perturbations along the horizontal straight-line trajectories than to perturbations along the robustness-optimal curved trajectories for the 0.2m and 0.3m separation distances. The part of the trajectory with the greatest sensitivity, though, was the portion near the center of the three-feature baseline. An examination of the perturbation norm $||\Delta A||_{max}$ and of the robustness performance metric revealed that the greatest sensitivity is present in the perturbation norm, and it is contributed largely by the error ΔH in the nominal measurement Jacobian when the vehicle is perturbed. ΔH represents the difference between the nominal measurement Jacobian used by the LKF for localization and a Jacobian linearized about the vehicle's perturbed location. This error in the measurement Jacobian is displayed graphically in Figure 5-4, which illustrates the size of ΔH for perturbations of 0.1m in x and in y, respectively, for each of the A*-computed nominal trajectories that is plotted in Figure 5-3. The size of the ΔH matrix is represented by computing the Euclidean norm. The location of each bar in Figure 5-4 corresponds approximately to the location of the respective point in x along the A*-computed nominal trajectory. Table 5.1 provides a tabulated summary of the measurement Jacobian error and also the robustness performance parameters for all of the trajectories displayed in Figure 5-3.

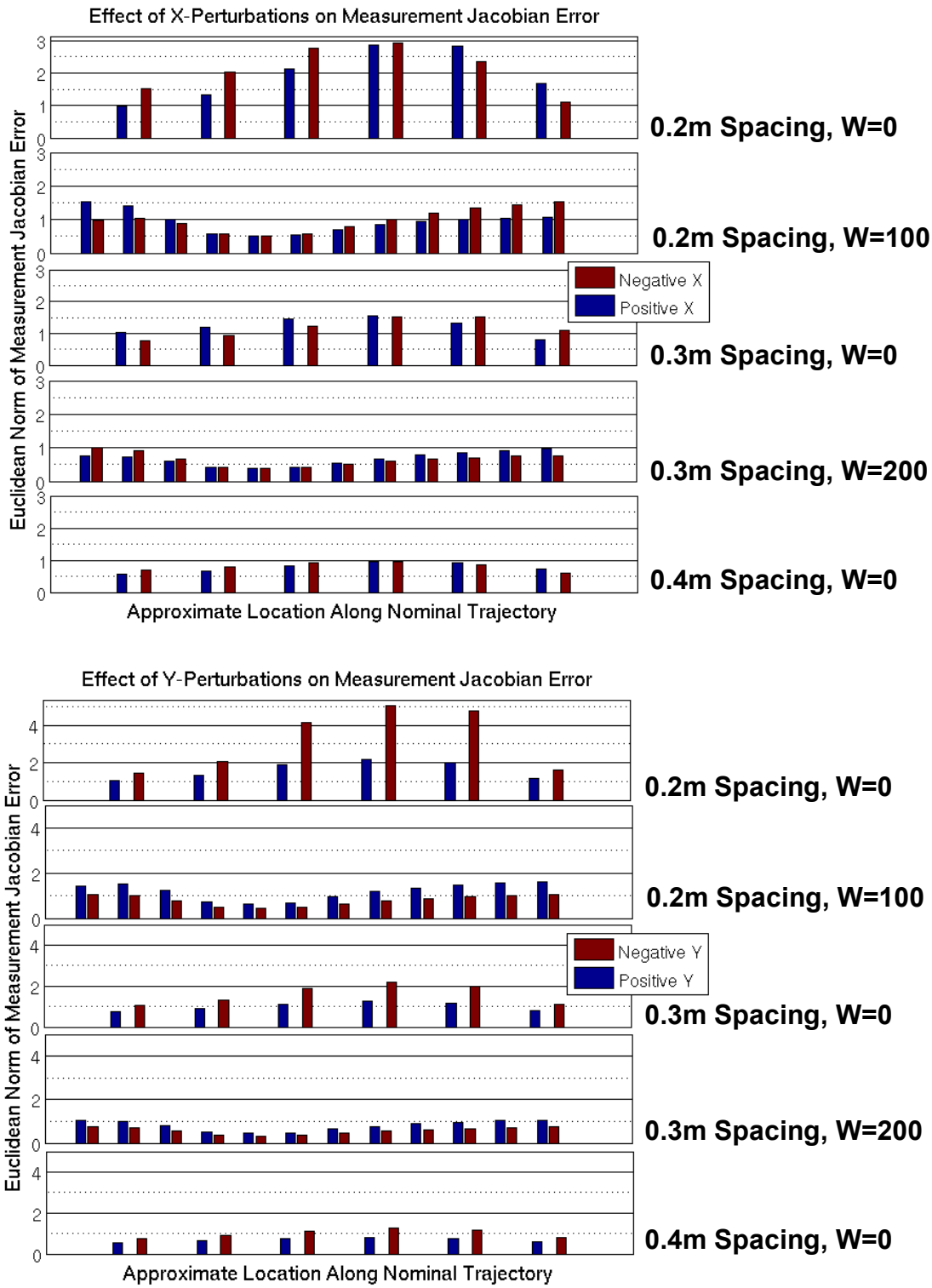

Figure 5-4: The Euclidean norm of the Jacobian error ΔH induced by perturbations of positive and negative 0.1m is plotted separately in x and in y for each of the five A*-computed trajectories displayed in Figure 5-3. The worst Jacobian errors occur along the horizontal straight-line trajectories that are close to the three-feature baseline.

Table 5.1: Robustness Parameters for the A* Trajectories of Figure 5-3

Spacing	Weight	$C_{robust,\ come}$	$\|\|\Delta A\|\|_{max}$	$\frac{(1-r)}{m}$	Mean $\|\|\Delta H\|\|$ In X	Mean $\|\|\Delta H\|\|$ in Y	Maximum $\|\|\Delta H\|\|$ in X	Maximum $\|\|\Delta H\|\|$ in Y
0.2m	0	120	4.05	0.0319	2.04	2.38	2.92	5.02
0.2m	100	50	1.81	0.0315	0.96	1.00	1.53	1.59
0.3m	0	47	1.74	0.0320	1.20	1.29	1.56	2.19
0.3m	200	1	1.04	0.0309	0.66	0.68	1.01	1.06
0.4m	0	0	0.94	0.0321	0.79	0.83	0.95	1.24

Note: All robustness costs, $C_{robust,\ come}$, are computed according to (5.3) using a weight W=200 to allow fair comparison among the results.

Both Figure 5-4 and Table 5.1 demonstrate that the measurement Jacobian is especially sensitive to perturbations from the 0.2m horizontal straight-line trajectory which was computed with a robustness weight W of zero. It was second-most sensitive to perturbations from the horizontal trajectory spaced 0.3m from the baseline. Both robustness-augmented curved trajectories yield significant improvements in the sensitivity of the measurement Jacobian, and also the overall robustness cost assigned to the trajectory. In particular, for the 0.3m spacing case the robustness-augmented trajectory accumulates a robustness cost-to-come of 1, while the regular A* trajectory accumulates a robustness cost of 47. From table 5.1 it can be observed that the changes in nominal trajectory had almost no effect on the robustness performance metric computed from exponential parameters m and r, which is nearly the same for all trajectories. All of the sensitivity to perturbations was exhibited in the perturbation norm $||\Delta A||_{max}$, and in particular can be traced to the measurement Jacobian.

It must be noted that the trajectories found using the robustness-augmented A* algorithm may not truly represent the most robust trajectories in the entire state space, but they are optimal given the constraints of discrete time and the discrete input space. It can be noticed in Figure 5-3 that none of the A*-planned trajectories precisely reach the goal waypoint, and this is because any trajectory that arrives within 0.05m of the goal in both x and y is declared to have reached the goal. This is one of several discretization choices which must be made to implement this algorithm efficiently, and these parameters, in addition to the Kalman filter tuning parameters needed for the robustness evaluation and

the various cost and weighting parameter choices for the raft vehicle model, are summarized in Table 5.2.

Table 5.2: Parameters Used in Robustness-Augmented A* Simulation and Experiment

Discretization Parameters:	Sampling Time Interval: 0.5 sec	Force Application Time Interval: 1 sec	Magnitude of Force Application: 0.2 N
Cost and Weighting Parameters:	Base Unit of Cost C_0 (as defined in Fig. 5-1): 10	Benchmark Perturbation size used for ΔA: 0.1m	
Margin of Error for Arrivals at the Goal:	Margin of error in x: 0.05 m	Margin of error in y: 0.05 m	
Vehicle Process Noise Parameters:	Process noise variance, surge: 0.1 $(m/s^2)^2$	Process noise variance, sway: 0.1 $(m/s^2)^2$	Process noise variance, roll: 0.1 $(rad/s^2)^2$
Feature Detection Noise Parameters:	Sensor noise variance, vehicle-relative range: 10^{-4} $(m)^2$	Sensor noise variance, vehicle-relative bearing: 10^{-4} $(rad)^2$	

5.3 Experimental Results

The A* trajectories displayed in Figure 5-3 were adapted for experimental implementation on the raft platform. Although A* planned these trajectories to consume minimum time and energy, they must be interpolated and made less aggressive for successful use in experiment. Because the raft frequently encounters disturbances in its testing tank and bias forces from its tethered power cable, it is unlikely that an application of the precise forces used in Figure 5-3 will reach the goal in the amount of time dictated by the A* simulation. In addition, it is not only desired that the vehicle reach the goal a single time, but that the vehicle execute a periodic path between the start waypoint and goal waypoint used by A*. A periodic trajectory that endures for several cycles will provide stronger evidence as to whether robustness-augmented A* has planned trajectories which achieve significant gains in robustness. And so, the shape of the trajectories planned by A* is preserved, but the period of time and sequence of inputs over which they occur is adjusted. Because the main contributors to instability in this holonomic vehicle navigation problem are perturbation-sensitive spatial configurations, the basic spatial layout of the A* trajectory is the main product sought from the planning algorithm.

The first series of raft experiments interpolated the trajectories of Figure 5-3 to fit a time interval of 20 seconds for travel between the two waypoints. This interpolated trajectory was then used as the nominal trajectory for a linearized Kalman filter, using the same tuning parameters listed in Table 5.2. Rather than interpolate the sequence of force inputs, the nominal input was set to zero for the entire trajectory and the feedback controller was relied upon to propagate the vehicle from waypoint to waypoint along the interpolated A* trajectory. First, the trajectories with zero robustness cost were explored to identify any instabilities that may result from vulnerability to perturbations. Figure 5-5 displays the experimental data gathered for the standard horizontally-directed A* trajectory with vertical spacings of 0.2m, 0.3m, and 0.4m from the three-feature baseline. The 0.3m and 0.4m cases were successfully implemented using an LKF and were robust to errors in initialization (i.e., they didn't have to be initialized from the exact start waypoint). The 0.2m case, on the other hand, consistently exhibited the onset of instability only a few seconds after initialization.

The curved trajectory computed by robustness-augmented A* was attempted next using an LKF, and this trajectory, for both the 0.2m and 0.3m case, exhibited instability. This curved trajectory was difficult for the vehicle to adhere to closely because of the tether forces exerted on the raft. It is believed that the raft was not capable of adhering closely enough to the nominal trajectory for the LKF algorithm to succeed. The unstable results for the 0.3m and 0.4m case are displayed in Figure 5-6.

Because of the failure encountered in implementing the robustness-augmented A* trajectories using an LKF, an EKF was implemented next, and the failed trajectories were interpolated further to allow 25 seconds of transit time between the start and goal waypoints. It is reasonable to expect an LKF algorithm to fail if the vehicle is perturbed or displaced significantly from its nominal trajectory, but an EKF algorithm is linearized locally about the current state estimate and should not fail along a robust nominal trajectory. First, the EKF was applied to the simple horizontal trajectory with a vertical spacing of 0.2m, which had exhibited failure using an LKF in Figure 5-5. Although the 0.2m trajectory occasionally succeeded, it also encountered instability in numerous trials. Figure 5-7 displays three representative cases of the 0.2m horizontal trajectory, one which

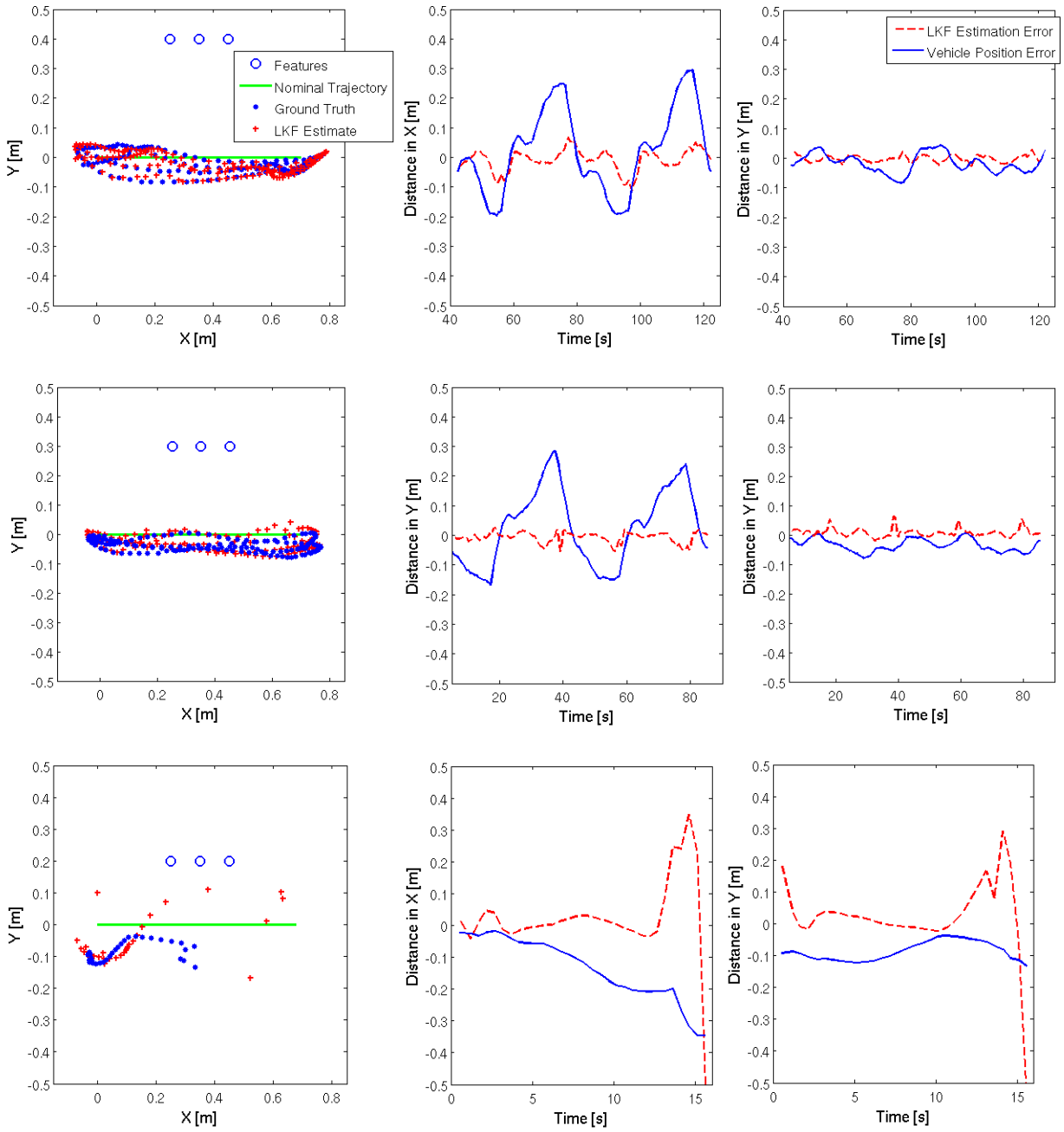

Figure 5-5: Two complete cycles are plotted of a periodic trajectory between two waypoints adapted from the standard A* algorithm (in the case of instability fewer cycles are plotted). The same horizontal trajectory is attempted in each case, with vertical spacing between the trajectory and the features varying from 0.2m to 0.4m. Using an LKF, the 0.3m and 0.4m cases are stable and the 0.2m case is unstable. Estimation error in x and in y is plotted, alongside the error in vehicle position from the nominal trajectory.

succeeded and two which encountered instability. Even with an EKF, this trajectory was particularly sensitive to the location from which it was initialized. When far enough away from the exact starting waypoint (0.1m to 0.2m seemed to be sufficient), this trajectory led

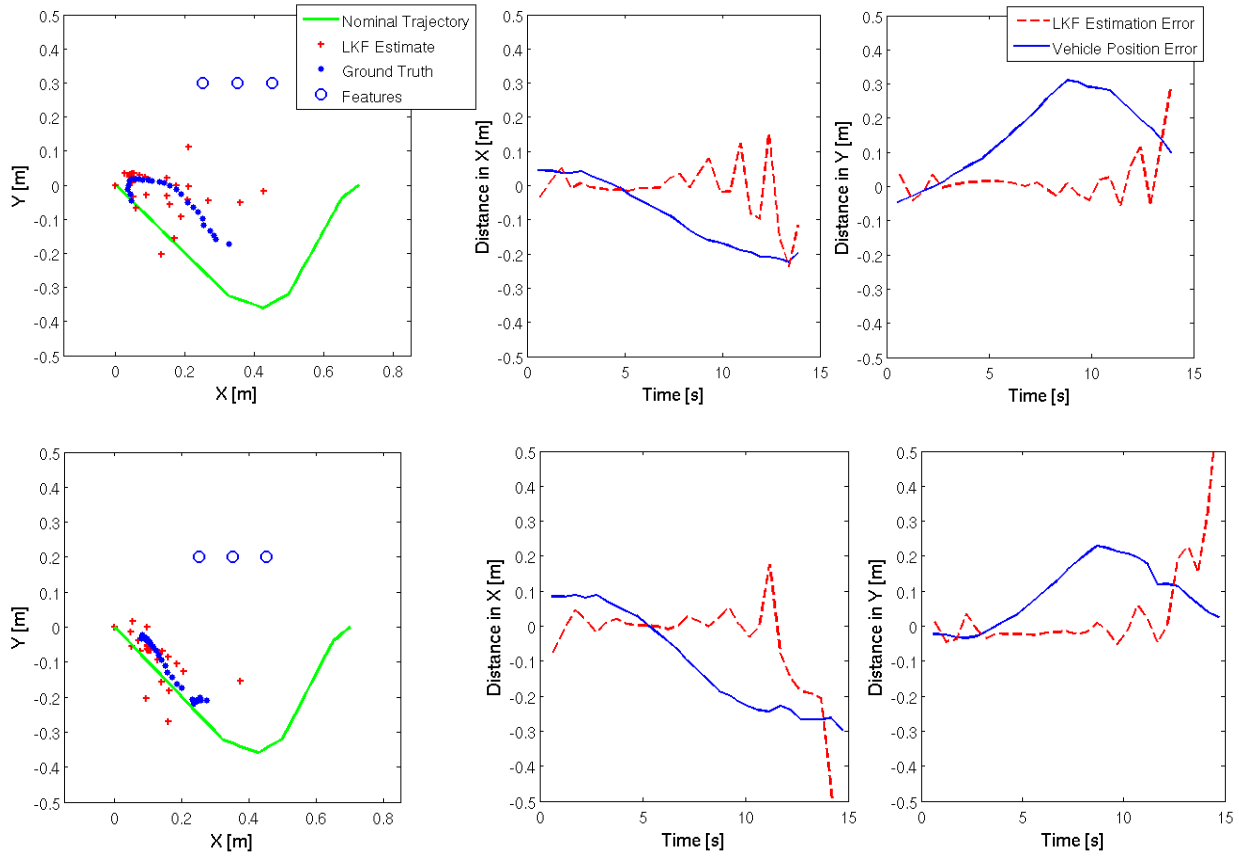

Figure 5-6: The onset of instability is plotted for a periodic trajectory between two waypoints adapted from the robustness-augmented A* algorithm. The same trajectory is attempted in each case, with vertical spacing between the trajectory and the features varying from 0.2m to 0.3m. Using an LKF, both the 0.2m and 0.3m cases were found to be unstable. Estimation error in x and in y is plotted, alongside the error in vehicle position from the nominal trajectory.

to instability.

Fortunately, when the EKF was applied to the two curved trajectories obtained from robustness-augmented A*, the closed-loop system remained stable. Results for both 0.2m and 0.3m spacing from the three-feature baseline are displayed in Figure 5-8. Although the raft does not adhere precisely to the nominal trajectory, the vehicle travels successfully between the two waypoints and there is little if any estimation error. Errors on the order of one or two tenths of a meter did not lead to sporadic instability as they did for the 0.2m horizontal trajectory. This indicates that in the case of 0.2m spacing from the three-feature baseline, the robustness-augmented A* algorithm succeeded in choosing a more robust trajectory than the standard A* algorithm. Although the robustness-augmented A* algorithm also recommended a curved trajectory for spacing of 0.3m from the baseline, no

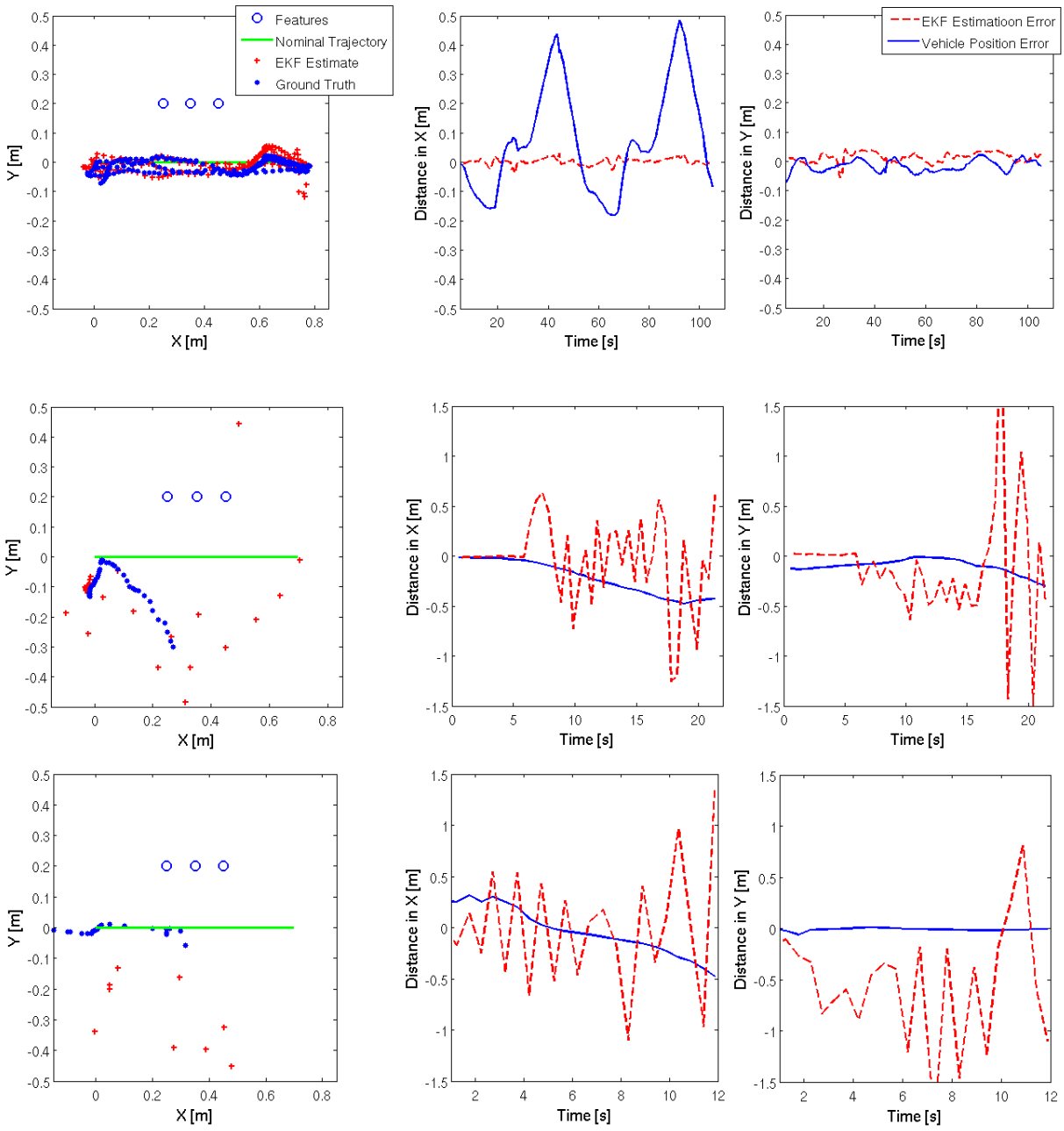

Figure 5-7: A periodic horizontal trajectory between two waypoints adapted from the standard A* algorithm is plotted for a vertical spacing of 0.2m from the three-feature baseline. The three sets of plots represent three typical time responses encountered, which included the frequent onset of instability. EKF Estimation error in x and in y is plotted, alongside the error in vehicle position from the nominal trajectory.

tangible gain in robustness was made apparent by this experiment. And, as predicted by the A* results of Figure 5-3, for a spacing of 0.4m from the baseline the simple horizontal trajectory was sufficiently robust without the addition of any curvature. For this particular series of experiments a robustness weight of 100 would have been sufficient to ensure

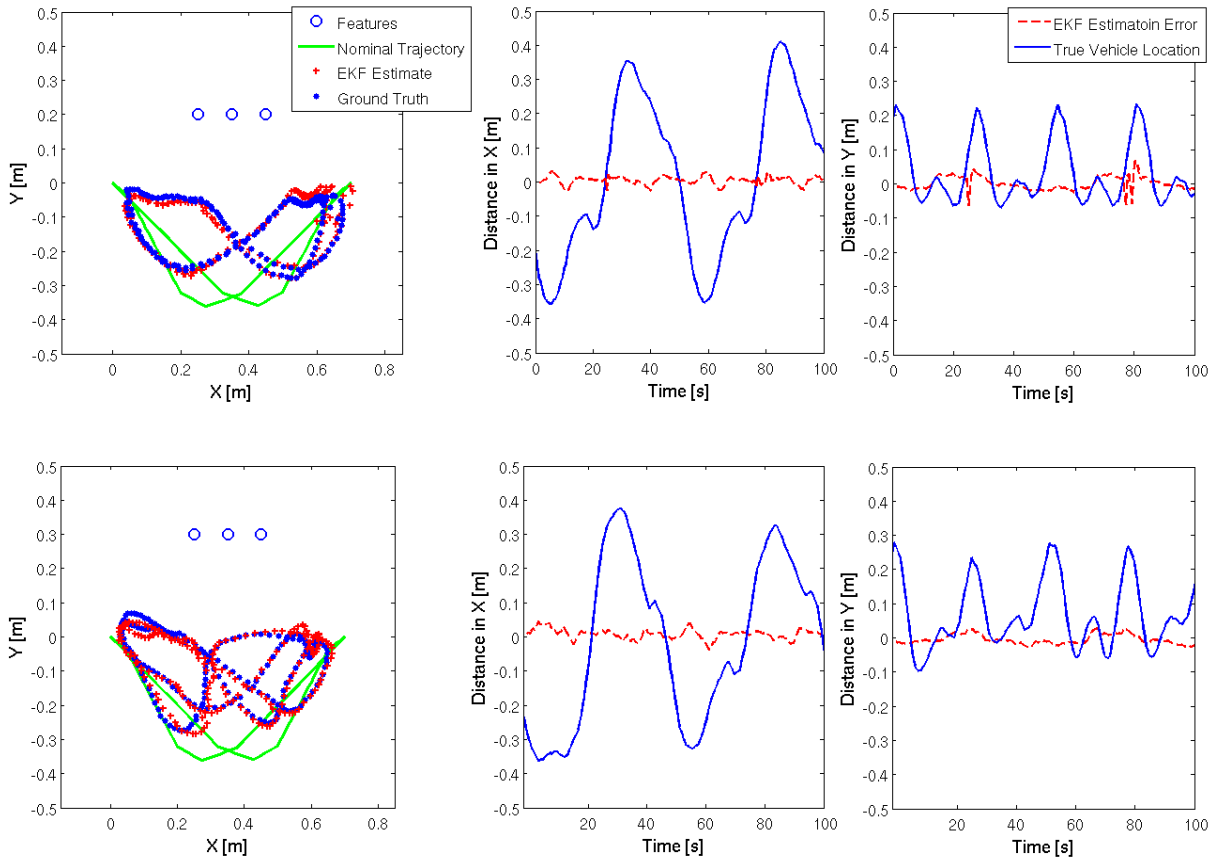

Figure 5-8: Two complete cycles are plotted of a periodic trajectory between two waypoints adapted from the robustness-augmented A* algorithm. The same curved trajectory is attempted in each case, with vertical spacing between the trajectory and the features varying from 0.2m to 0.3m. Using an EKF, both the 0.2m and 0.3m cases are found to be stable. Estimation error in x and in y is plotted, alongside the error in vehicle position from the nominal trajectory.

closed-loop vehicle stability, since this value of the robustness weight W only recommended a curved trajectory for the 0.2m case. For the raft vehicle platform, these experiments served as a tuning procedure which aided the selection of appropriate costs and weights for the robustness-augmented A* algorithm.

This method has demonstrated that only after a thorough search of an autonomous vehicle's state space can it be concluded that the vehicle is safe from the pitfalls of perturbation-sensitive configurations. A simple map of three collinear features was used to simply experimental implementation, but this technique can be extended to path-planning for any system that is capable of achieving asymptotic stability in the absence of perturbations and whose dynamics can be expressed within a linear time-varying state space framework. For an autonomous ship hull survey, where perturbations are more than

likely to occur, tempering a search for an energy and time-efficient path with a robustness cost will help the survey vehicle avoid instabilities such as those encountered by the raft in Figure 5-7.

Chapter 6

Conclusion

This thesis has introduced a series of algorithms designed to evaluate the stability and robustness of a holonomic marine vehicle carrying out an integrated localization, mapping, and control process using an *a priori* map. Despite the specific nature of the application, the algorithms are applicable to any autonomous vehicle control and estimation scenario whose closed-loop dynamics may be expressed within the framework of a linear time-varying state space model.

6.1 Summary

In Chapter 2, the real-time map building capabilities of a holonomic marine vehicle are demonstrated as a preliminary step toward autonomous ship hull inspection. An EKF is used to localize the HAUV and construct a map of mine-shaped training targets which are extracted from DIDSON sonar imagery. The successful performance of this task in real-time at a reasonable bandwidth demonstrates the potential of feature-based navigation methods for successful assimilation with high-fidelity feedback control.

In Chapter 3 a model is introduced which integrates a map refinement estimation process with feedback control to achieve a linear time-varying closed-loop system which is linearized about a nominal vehicle survey trajectory. The stability of the vehicle is investigated by computing the norm of the state transition matrix, $\Phi(k,k_0)$. Analysis of $||\Phi(k,k_0)||$ reveals that map refinement (i.e., localization and mapping) can achieve Lyapunov stability at best, while map exploitation, which is strictly localization, can achieve asymptotic stability. It is then shown that $\Phi(k,k_0)$ can be used as a necessary and sufficient

indicator of stability in inspecting the vehicle's controller. An examination of the approximate upward gain margin demonstrated comparable sensitivity among a map exploitation, map refinement, and direct x-y-φ position measurement process.

In Chapter 4 we consider errors in the linear time-varying closed-loop vehicle model that are introduced by perturbations from the nominal trajectory. A performance metric is derived from a well-established sufficiency condition for asymptotic stability in the presence of perturbations. This metric enables the evaluation of the relative robustness among different system configurations. The performance metric allows us to discern the impact of subtle aspects of filter conditioning, such as the geometric configuration of map features, on the robustness of the marine vehicle. The use of this technique yields a procedure for evaluating which of several candidate trajectories is best-equipped to tolerate perturbations, and it is applicable to complex estimation scenarios which may involve multiple sensors and map features which drop in and out of view.

Finally, selection of a sufficiently robust path from among all goal-reaching paths in the vehicle state space is achieved using the A* graph search algorithm in Chapter 5. Considering the robustness performance metric in the A* cost function causes the planning algorithm to divert from the most energy-efficient trajectory if there is a significant gain to be made in robustness against perturbations to the vehicle. An algorithm of this nature can aid the design of vehicle survey trajectories by enforcing careful maneuvering around sensitive areas of the map and by softening the impact of perturbation-induced linearization errors.

Although the framework for analyzing stability and robustness has used the LKF to allow advance computation along a nominal vehicle survey trajectory, it is recommended that an EKF be used for experimental implementation of vehicle surveys. The LKF framework is intended to serve as a tool to inform design, and upon designing a trajectory that is optimally suited for robustness against perturbations using an LKF, experimental implementation with an EKF may provide an additional safeguard against instability due to displacement from the nominal survey trajectory.

6.2 Future Work

Several of the topics explored in this thesis warrant future investigation. One important remaining task is to apply the robustness methods of Chapters 4 and 5 to systems which cannot achieve asymptotic stability, and achieve Lyapunov stability at best. It has not yet been proven that the most robust trajectory for map exploitation, which can be selected using the robustness performance metric of Chapter 4, is also the most robust trajectory for map refinement.

In addition, a method of robustness evaluation which can determine the system configuration best-equipped against perturbations without being informed of a specific perturbation size is highly desirable. It seems intuitive that the most robust trajectory should be robust irrespective of the size of the perturbation, and a result of this nature would greatly simplify the implementation of the robustness evaluation algorithms presented in Chapters 4 and 5. An achievement of this nature would also simplify the task of making robustness comparisons between different systems.

With respect to the problem of planning a robust path between two waypoints, there are many desirable additions to the A* cost function. For the application of ship hull inspection, achieving one hundred percent coverage of the survey area is a critical objective, and these coverage requirements must be combined with the goal-reaching and robustness costs. The information gained by achieving wide coverage can also be tempered by a cost assigned to estimation uncertainty, which will ensure that the autonomous vehicle acts to reduce its estimation error variance whenever possible. The relationship between error variance reduction and robustness optimization must be explored and understood, and any overlap among the two motion planning objectives must be identified.

From a broader perspective, the ability to analyze the interaction of a feedback control process with a large-scale localization and mapping process is also a desirable objective. At the current time estimation algorithms which do not use an *a priori* map, but instead build a map from scratch, cannot be incorporated into the linear system stability analysis presented here. In addition, the Kalman filter framework grows computationally infeasible as maps accumulate hundreds of features, and so an estimation framework is

sought which can accommodate localization and mapping on a large scale and still permit an analysis of the interaction between the estimation and control processes and identification of the associated stability margins.

Despite the need for continued investigation, this study is a first step toward understanding the interaction between feedback control and feature-based estimation and managing the robustness of the integrated process. If feature-based estimation algorithms are to be used successfully in the closed-loop surveying of marine environments, an understanding of how stability and robustness may be achieved and managed is of paramount importance.

Bibliography

[1] F. Hover, J. Vaganay, M. Elkins, S. Wilcox, V. Polidoro, J. Morash, R. Damus, and S. Desset, "A Vehicle System for Autonomous Relative Survey of In-Water Ships," *Marine Technology Society Journal,* vol. 41(2), 2007, pp. 44-55.

[2] J. Vaganay, M. Elkins, S. Wilcox, F. Hover, R. Damus, S. Desset, J. Morash, and V. Polidoro, "Ship Hull Inspection By Hull-Relative Navigation and Control," *Proc. IEEE OCEANS Conf.,* Washington, D.C., 2005, pp. 761-766.

[3] J. Vaganay, M. Elkins, D. Esposito, W. O'Halloran, F. Hover, and M. Kokko, "Ship Hull Inspection with the HAUV: US Navy and NATO Demonstrations Results," *Proc. IEEE OCEANS Conf.,* Boston, 2006, pp. 1-6.

[4] E. Belcher, H. Dinh, D. Lynn, and T. Laughlin, "Beamforming and Imaging with Acoustic Lenses in Small, High Frequency Sonars," *Proc. IEEE OCEANS Conf.,* Seattle, 1999, pp. 1495-1499.

[5] E. Belcher, B. Matsuyama, and G. Trimble, "Object Identification with Acoustic Lenses," *Proc. IEEE OCEANS Conf.,* Honolulu, 2001, pp. 6-11.

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics,* Cambridge, MA: The MIT Press, 2005.

[7] S. LaValle, *Planning Algorithms,* Cambridge, UK: Cambridge University Press, 2006.

[8] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion,* Cambridge, MA: The MIT Press, 2005.

[9] A. Gelb, ed., *Applied Optimal Estimation*, Cambridge, MA: The MIT Press, 1984.

[10] R.C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Int. J. Robotics Research*, vol. 5(4), 1986, pp. 56-68.

[11] J. Leonard and H.F. Durrant-Whyte, "Mobile Robot Localization by Tracking Geometric Beacons," *IEEE Trans. on Robotics and Automation*, vol. 7(3), 1991, pp. 376-382.

[12] F. Lu and E. Milios, "Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans," *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* Seattle, 1994, pp. 935-938.

[13] P. Besl and N. McKay, "A Method for Registration of 3-D Shapes," *IEEE Trans on Pattern Analysis and Machine Intelligence,* 14(2), 1992, pp. 239-256.

[14] R. Madhavan, G. Dissanayake, and H. Durrant-Whyte, "Map-Building and Map-Based Localization in an Underground Mine by Statistical Pattern Matching," *Proc. IEEE Int. Conf. on Pattern Recognition,* Brisbane, 1998, pp. 1744-1746.

[15] R. Madhavan and H. Durrant-Whyte, "2D Map-Building and Localization in Outdoor Environments," *Journal of Robotic Systems,* vol. 22(1), 2005, pp. 45-63.

[16] C. Roman and H. Singh, "Improved Vehicle-Based Multibeam Bathymetry Using Sub-Maps and SLAM," *Proc. IEEE Int. Conf. on Intelligent Robots and Systems,* Edmonton, 2005, pp. 3662-3669.

[17] B. Schiele and J. Crowley, "Comparison of Position Estimation Techniques Using Occupancy Grids," *Proc. IEEE Int. Conf. on Robotics and Automation,* San Diego, 1994, pp. 1628-1634.

[18] R. Simmons and S. Koenig, "Probabilistic Robot Navigation in Partially Observable Environments," *Proc. Int. Joint Conf. on Artificial Intelligence,* Montreal, 1995, pp. 1080-1087.

[19] A. Cassandra, L. Kaelbling, and J. Kurien, "Acting Under Uncertainty: Discrete Bayesian Models for Mobile-Robot Navigation," *Proc. IEEE Int. Conf. on Intelligent Robots and Systems,* Osaka, 1996, pp. 963-972.

[20] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo Localization for Mobile Robots," *Proc. IEEE Int. Conf. on Robotics and Automation,* Detroit, 1999, pp. 1322-1328.

[21] R. Smith, M. Self, and P. Cheeseman, "Estimating Uncertain Spatial Relationships in Robotics," I. Cox and G. Wilfon, eds., *Autonomous Robot Vehicles*, New York: Springer-Verlag, 1990, pp. 167-193.

[22] M. Csorba, *Simultaneous Localisation and Map Building,* PhD Thesis, University of Oxford, 1997.

[23] G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A Solution to the Simultaneous Localization and Map Building (SLAM) Problem," *IEEE Trans. on Robotics and Automation*, vol. 17(3), 2001, pp. 229-241.

[24] J. Leonard and H. Feder, "A Computationally Efficient Method for Large-Scale Concurrent Mapping and Localization," *Proc. Ninth Int. Symposium On Robotics Research,* Salt Lake City, 1999, pp. 169-176.

[25] F. Lu and E. Milios, "Globally Consistent Range Scan Alignment for Environment Mapping," *Autonomous Robots,* vol. 4(4), 1997, pp. 333-349.

[26] J. Folkesson and H. Christensen, "Graphical SLAM – A Self-Correcting Map," *Proc. IEEE Int. Conf. on Robotics and Automation,* New Orleans, 2004, pp. 383-390.

[27] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durant-Whyte, "Simultaneous Localization and Mapping with Sparse Extended Information Filters," *Int. J. Robotics Research,* Vol. 23(7-8), 2004, pp. 693-716.

[28] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," *Proc. AAIA Nat. Conf. on Artificial Intelligence,* Edmonton, 2002, pp. 593-598.

[29] H. Choset, "Coverage for Robotics – A Survey of Recent Results," *Annals of Mathematics and Artificial Intelligence,* Vol. 31, 2001, pp. 113-126.

[30] H. Feder, J. Leonard, and C. Smith, "Adaptive Mobile Robot Navigation and Mapping," *Int. J. Robotics Research,* vol. 18(7), 1999, pp. 650-668.

[31] S. Huang, N. Kwok, G. Dissanayake, Q. Ha, and G. Fang, "Multi-Step Look-Ahead Trajectory Planning in SLAM: Possibility and Necessity," *Proc. IEEE Int. Conf. on Robotics and Automation,* Barcelona, 2005, pp. 1091-1096.

[32] A. Makarenko, S. Williams, F. Bourgoult, and H. Durrant-Whyte, "An Experiment in Integrated Exploration," *Proc. IEEE Int. Conf. on Intelligent Robots and Systems,* Lausanne, Switzerland, 2002, pp.534-539.

[33] P. Newman, M. Bosse, and J. Leonard, "Autonomous Feature-Based Exploration," *Proc. IEEE Int. Conf. on Robotics and Automation,* Taipei, 2003, pp. 1234-1240.

[34] T. Kollar and N. Roy, "Trajectory Optimization Using Reinforcement Learning for Map Exploration," *Int. J. Robotics Research,* vol. 27(2), 2008, pp. 175-196.

[35] T. Vidal-Calleja, J. Andrade-Cetto, and A. Sanfeliu, "Estimator Stability Analysis in SLAM," *Proc. 5th IFAC/EURON Symp. on Intelligent Autonomous Vehicles*, Lisbon, July 2004.

[36] J. Andrade-Cetto and A. Sanfeliu, "The Effects of Partial Observability in SLAM," *Proc. IEEE Int. Conf. on Robotics and Automation*, New Orleans, 2004, pp. 397-402.

[37] F. Hover, "Stability of Double-Integrator Plants Controlled using Real-Time SLAM Maps," *Proc. IEEE Int. Conf. on Robotics and Automation.*, Pasadena, 2008, pp. 637-642.

[38] P. Corke, "Mobile Robot Navigation as a Planar Visual Servoing Problem," R. Jarvis and A. Zelinsky, eds., *Robotics Research: The Tenth International Symposium,* Heidelberg: Springer-Verlag, 2003, pp. 361-371.

[39] E. Malis, "Stability Analysis of Invariant Visual Servoing and Robutness to Parametric Uncertainties," A. Bicchi, H. Christensen, and D. Prattichizzo, eds., *Control Problems in Robotics,* Heidelberg: Springer-Verlag, 2003, pp. 265-280.

[40] F. Chaumette, "Potential Problems of Stability and Convergence in Image-Based and Position-Based Visual Servoing," D. Kriegman, G. Hager, and A Morse, eds., *The Confluence of Vision and Control,* Heidelberg: Springer-Verlag, 1998, pp. 66-78.

[41] L. Deng, F. Janabi-Sharafi, W. Wilson, "Stability and Robustness of Visual Servoing Methods," *Proc. IEEE Int. Conf. on Robotics and Automation,* Washington, D.C., 2002, pp. 1604-1609.

[42] E. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik,* vol. 1(1), 1959, pp. 269-271.

[43] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. On Systems Science and Cybernetics,* vol. 4(2), 1968, pp. 100-107.

[44] R. Dechter and J. Pearl, "Generalized Best-First Search Strategies and the Optimality of A*", J. *Assoc. Comput. Mach., vol. 32(3), 1985, pp. 505-536.*

[45] J. Canny, J. Reif, B. Donald, and P. Xavier, "On the Complexity of Kinodynamic Planning," *29th Annual Symp. On Foundations of Computer Science,* White Plains, NY, 1988, pp. 306-316.

[46] E. Frazzoli, "Maneuver-Based Motion Planning and Coordination for Single and Multiple UAVs," *AIAA 1st Technical Conf. and Workshop on Unmanned Aerospace Vehicles,* Portsmouth, VA, 2002.

[47] S. Williams, H. Durrant-Whyte, and G. Dissanayake, "Constrained Initialization of the Simultaneous Localization and Mapping Algorithm," *Int. J. Robotics Research,* vol. 22(7-8), 2003, pp. 541-564.

[48] M. Kokko, *Range-Based Navigation of AUVs Operating Near Ship Hulls,* M.S. Thesis, Massachusetts Institute of Technology, 2007.

[49] A. Bryson, *Applied Optimal Control: Optimization, Estimation, and Control,* Washington: Hemisphere Pub. Corp., 1975.

[50] F. Lewis and V. Syrmos, *Optimal Control*, New York: John Wiley and Sons, 1995.

[51] A. Bryson, *Applied Linear Optimal Control: Examples and Algorithms,* Cambridge, UK: Cambridge University Press, 2002.

[52] M. Athans, "The Role and Use of the Stochastic Linear-Quadratic-Gaussian Problem in Control System Design," *IEEE Trans. On Automatic Control,* vol. AC-16(6), 1971, pp. 529-552.

[53] J.J. Slotine, *Applied Nonlinear Control,* Englewood Cliffs, NJ: Prentice Hall, 1991.

[54] J.C. Willems, *Stability Theory of Dynamical Systems*, London: Thomas Nelson and Sons ltd., 1970.

[55] A. Ilchmann, D.H. Owens, and D. Prätzel-Wolters, "Sufficient conditions for stability of linear time-varying systems," *Systems and Control Letters*, vol. 9, 1987, pp. 157-163.

[56] F. Amato, G. Celentano, F. Garofalo, "New sufficient conditions for the stability of slowly varying linear systems," *IEEE Trans. Automatic Control*, vol. 38(9), 1993, pp. 1409-1411.

[57] P. Mullhaupt, D. Buccieri, and D. Bonvin, "A numerical sufficiency test for the asymptotic stability of linear time-varying systems," *Automatica*, vol. 43(4), 2007, pp. 631-638.

[58] B. Chen and T. Dong, "Robust stability analysis of Kalman-Bucy Filter under parametric and noise uncertainties"*, Int. J. Control*, 48, 1988, pp. 2189-2199.

[59] A. Weinmann, *Uncertain Models and Robust Control*, New York: Springer-Verlag/Wien, 1991.

[60] G. Chen, ed., *Approximate Kalman Filtering*, Singapore: World Scientific, 1993.

[61] M. Greytak, *Autonomous Learning and Control for Marine Vehicles,* PhD Thesis, Massachusetts Institute of Technology, 2009.