BOUNDED-ERROR VISION-BASED NAVIGATION OF AUTONOMOUS UNDERWATER VEHICLES

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Stephen D. Fleischer

May 2000

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

Stephen M. Rock
Department of Aeronautics and Astronautics
(Principal Adviser)

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

Carlo Tomasi
Department of Computer Science

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

Claire J. Tomlin
Department of Aeronautics and Astronautics

Approved for the University Committee on Graduate Studies:

To My Family

Abstract

Underwater vehicles are being used extensively to explore the ocean depths, but to enhance their utility for marine scientists and other end-users, new navigation capabilities must be developed. To determine vehicle location for the purpose of navigation, vision-based mosaicking is a promising new technology with several inherent benefits: it is inexpensive, utilizes any existing camera on-board the vehicle, and does not require extensive set-up or calibration. Other alternatives for navigation exist, but these have significant limitations in the underwater environment: the Global Positioning System (GPS) cannot penetrate the ocean surface; acoustic positioning systems require a transponder net to be constructed before a particular area can be explored; and sonar systems are often bulky and exhibit poor accuracy, depending on the cost of the system.

While vision sensing is an attractive option, it also has its challenges. The vehicle must always be within visual range of the ocean floor to make motion measurements, so this limits navigation to the near-bottom environment. Current techniques for vision-based navigation are inaccurate and unreliable because they are based on dead reckoning. In dead reckoning, the absolute vehicle position is estimated by integrating the vision-based relative motion measurements along the vehicle path. Even though the relative motion measurements are precise, the error in absolute position is unbounded because of the small errors that accrue over the length of the vehicle path. This situation results in unbounded navigational errors, and it is the focus of this work.

The purpose of this thesis is to develop a map-based approach to visual navigation. The composite-image mosaics are used as reference maps for navigation. Once the map

is constructed, the live image from the vehicle can be compared directly to the goal location within the map. Since the relative displacement between the current and desired vehicle locations can be measured directly in map coordinates, the navigational error will be bounded, regardless of the accuracy of absolute position estimation with respect to some global coordinate system.

In the map-based approach, the internal consistency of the mosaic map is a major issue. Since the alignment of images within a mosaic is susceptible to the propagation of errors similar to dead reckoning, crossover points (where the image chain comprising the mosaic loops back upon itself) are used to provide additional mosaic alignment information. The new measurement is used to re-align the mosaic map through a smoothing process that globally minimizes the alignment errors in the mosaic.

Theoretical and experimental results of this work are presented. The accuracy of the vision sensor was quantified on the Space Frame, a precision gantry platform capable of controlling a camera head in 3-DOF within its workspace. The complete navigation system was demonstrated both on the Oceanographic Technologies Testbed for Engineering Research (OTTER) autonomous underwater vehicle (AUV) in the test tank and on the Ventana remotely-operated vehicle (ROV) in the ocean.

This work has been performed under a joint effort between the Aerospace Robotics Laboratory (ARL) at Stanford University and the Monterey Bay Aquarium Research Institute (MBARI).

Acknowledgments

First and foremost, I wish to thank my advisor, Professor Stephen M. Rock, for the years of guidance that have shaped my research, for his optimistic attitude that has made the work a pleasure, and for the freedom he has given me to enjoy my tenure at Stanford to the fullest. The advice I have received from our discussions over the years will serve me well in the career and life ahead of me. To Professor Robert H. Cannon, Jr., I owe a debt of gratitude for bringing together such a talented collection of people to form the ARL and continually replenishing the lab with students of the highest caliber; I have enjoyed the challenge set forth by my peers everyday to maintain the standards that you have set. I also wish to thank all of the members of my oral defense and dissertation reading committees for their insightful questions and advice: Professors Per K. Enge, Thomas W. Kenny, Carlo Tomasi, and Claire J. Tomlin.

I would like to thank Michael J. Lee for his development of the ARL/MBARI joint program and his commitment to teaching us students a little bit about the real world, and thanks go to Dick Burton for his uncanny ability to somehow fix and upgrade OTTER whenever and however needed, despite the vehicle's frequent attempts at complete breakdown; their devotion to the OTTER project is much appreciated. While some may argue that trips on the Pt. Lobos to work with Ventana are "unsettling" experiences, thanks go to the Ventana pilots and Pt. Lobos crew for enabling my successes in the open ocean and for making the whole process fun (in between swells). In particular, I wish to thank T.C. Dawe, pilot extraordinaire, for his continued enthusiasm in our projects and for his willingness to accommodate every need of my experiments.

Funding for my research has come from several sources: the Department of Defense (DoD) National Defense Science and Engineering Graduate (NDSEG) Fellowship program, the Packard Foundation, and MBARI. I greatly appreciate the support provided to me by these organizations; the success of my work would not have been possible without their help.

Thank you to all of my fellow students in ARL, who have provided me with a challenging and fun place to work, although I'll never understand how most of them work those crazy 9-to-5 hours. In particular, the students in the ARL/MBARI program, both past and present, deserve my special thanks for the intellectual and technical discussions on the long commutes to Monterey (where we got most of our work done), for pulling those late-nighters to prepare for the many all-important demos, and for the camaraderie that comes from sharing in common experiences: Rick Marks (who gave me the idea for my thesis and an excellent foundation from which to build), Howard Wang, Tim McLain, Kortney Leabourne, Andreas Huster, Jason Rife, and Jeff Ota.

My heartfelt thanks go out to all of those people without whom, well, I would have graduated years ago. Thanks to Rick and Nordberg for teaching me what I never learned at Princeton. To all of the alumni and friends of Rains 14H, I have appreciated the distractions more than you know: the parties, the barbecues, and the nights in downtown Palo Alto have been memorable. For providing me with the greatest single distraction from my thesis, I must thank the Stanford Ultimate Frisbee teams. I especially want to thank the Men's B-team for letting an old guy join in the fun for the past four years.

Special thanks to the boys of 14H: Goof, Mike, Karl, and Chahz, for their many years of friendship. The endless games of DOOM, tennis, disc, hacky-sack, and foosball, the trips to Jamba Juice, the dinners at Taxi's, and more recently their constant encouragement for me to finish the thesis have meant a lot to me. I also wish to thank Rick for the wine, cigars, cards, dice, and other depravity over the years, including our Master's year at Stanford, his departure to the big city, and his eventual return as an M.B.A. (to find me still at Stanford).

I would like to thank all of my friends back home for their support and encouragement. In particular, thanks to the Celebreight crew for the many summers on the Outer Banks,

where I could relax and leave my problems on the West Coast. I especially want to thank Cheryl, for convincing me to join her on crazy adventures across the country and around the world, whenever I needed a break from life on The Farm.

Most importantly, I wish to thank my entire family for their continual love and support, and for never questioning my statement that graduation was "just two years" away. Thank you to my grandmother, who always asked how things were going, waiting patiently until she could call me "Doc". To Jeff and Leslie, I hope you're proud of your big brother, because I'm going to need some of that money you've been making while I've been in school forever. Finally, Mom and Dad, I couldn't have made it to the end of this long road without you. Thank you for your love and encouragement—it has made all the difference.

Contents

List of Tables

List of Figures

List of Symbols

This list summarizes the symbols used in this thesis:

x	scalar variable
${}^{A}\mathbf{p}_{BC}$	position vector from point B to C, expressed in frame A
${}^{A}_{B}R$	rotation matrix that describes frame B relative to frame A
${}^{A}_{B}\mathbf{q}$	Z:Y:X body-fixed Euler angles that describe frame B relative to frame A
$\mathrm{E}[x]$ or $\mathrm{E}[\mathbf{x}]$	expectation of the random scalar x or random vector $\mathbf{x}$
$\bar{x}$ or $\bar{\mathbf{x}}$	mean of the random scalar x or random vector $\mathbf{x}$
$\mathrm{Var}[x]$	variance of the random scalar x
$\mathrm{Cov}[\mathbf{x}]$	covariance matrix of the random vector $\mathbf{x}$
$\mathrm{Cov}[x, y]$	covariance of the random scalars x and y
$\mathrm{Cov}[\mathbf{x}, \mathbf{y}]$	covariance of the random vectors $\mathbf{x}$ and $\mathbf{y}$
$\mathcal{N}[x, X]$	Gaussian (normal) distribution with mean x and variance X

List of Acronyms

This list summarizes the acronyms used in this thesis:

ABE	Autonomous Benthic Explorer
ADCP	Acoustic Doppler Current Profiler
ARL	Aerospace Robotics Laboratory
AUV	Autonomous Underwater Vehicle
AVP	Advanced Vision Processor
CASE	Computer-Aided Software Environment
CCD	Charge-Coupled Device
CTD	Conductivity-Temperature-Density
DOF	Degree(s) of Freedom
FAU	Florida Atlantic University
FOV	Field Of View
GPS	Global Positioning System
GUI	Graphical User Interface
HDTV	High-Definition Television
ICS	Industrial Computer Source
IMU	Inertial Measurement Unit
INU	Inertial Navigation Unit
ISE	International Submarine Engineering
LoG	Laplacian-of-Gaussian

MBARI	Monterey Bay Aquarium Research Institute
MFC	Microsoft Foundation Classes
MIT	Massachusetts Institute of Technology
NASA	National Aeronautics and Space Administration
NDDS	Network Data Delivery Service
OTTER	Oceanographic Technologies Testbed for Engineering Research
PD	Proportional-Derivative
PID	Proportional-Integral-Derivative
REMUS	Remote Environmental Monitoring UnitS
ROV	Remotely-Operated Vehicle
RTI	Real-Time Innovations
SHARPS	Super High Accuracy Ranging and Positioning System
SLoG	Signum of Laplacian-of-Gaussian
USF	University of Southern Florida
UUV	Unmanned Underwater Vehicle
WHOI	Woods Hole Oceanographic Institute

Chapter 1

Introduction

This dissertation describes Bounded-Error Vision-Based Navigation of Autonomous Underwater Vehicles. This research was conducted under the auspices of a joint program between the Stanford ARL and MBARI. The ultimate goal of this research is to enable real-time, visual-based navigation for mobile robotic platforms.

1.1 Motivation

The purpose of this work is to fulfill a need for new sensing and control technologies for the remote exploration of unknown, unstructured environments. Mobile robots are being used extensively in areas that are unsuitable for human exploration. To optimize the performance and utility of these mechanical devices, higher levels of intelligence and autonomy must be incorporated, and more sophisticated sensing algorithms must be used to provide the detailed, accurate input data that the machines need to decide on the proper course of action.

This research is motivated specifically by the development of unmanned underwater vehicles (UUV's) for ocean exploration. Both classes of UUV's will be impacted by this work: ROV's, that require a human pilot in the control loop; and AUV's, that have the ability to perform high-level tasks without user intervention. Marine scientists utilize UUV's

extensively to explore the ocean environment, and the effectiveness of exploration missions would be greatly improved by enhanced vehicle capabilities.

In particular, the capability of autonomous navigation is a common requirement for a wide variety of typical UUV exploration missions, and its development is the focus of this dissertation. Autonomous navigation benefits both ROV's and AUV's.

Over the past decade, ROV's have been the workhorses of the marine science community (Section 1.2.1). ROV operations are dependent on a skilled, experienced pilot to control the vehicle, while the scientist provides mission-level directives to the pilot. During the execution of ROV exploration missions, the pilot relies primarily on live video feeds from on-board cameras to control the vehicle. The integration of these data into a world model of the ocean environment is performed entirely within the mind of the pilot. This task, in combination with the responsibility for low-level vehicle control, often results in extremely high pilot workloads. An autonomous navigation capability would significantly reduce this workload for pilots during many types of ROV exploration missions.

For instance, population determination for a field of brachiopods (i.e. sea clams) on the ocean floor is a typical and recurring mission for ROV's such as the MBARI-owned Ventana, and it would be greatly facilitated by a navigational pilot aid. To perform this mission, the ROV pilot performs a video survey of the brachiopod field, while the scientist determines the population count directly from the video, either online or offline (Figure 1.1). Providing complete coverage with minimum overlap and avoiding missed or duplicate counts is a tedious and difficult task for the pilot-scientist team. With a video-based autonomous navigation aid, the ROV pilot could direct the vehicle to create a map of the area while following a regular coverage pattern. The map then could provide a much better representation of the information than the raw video for the scientist to perform the population count.

AUV's have been the focus of recent developments in marine exploration, since human-in-the-loop control is not always achievable (Section 1.2.1). Currently, most AUV missions have been primarily engineering demonstrations to prove new technologies and perform

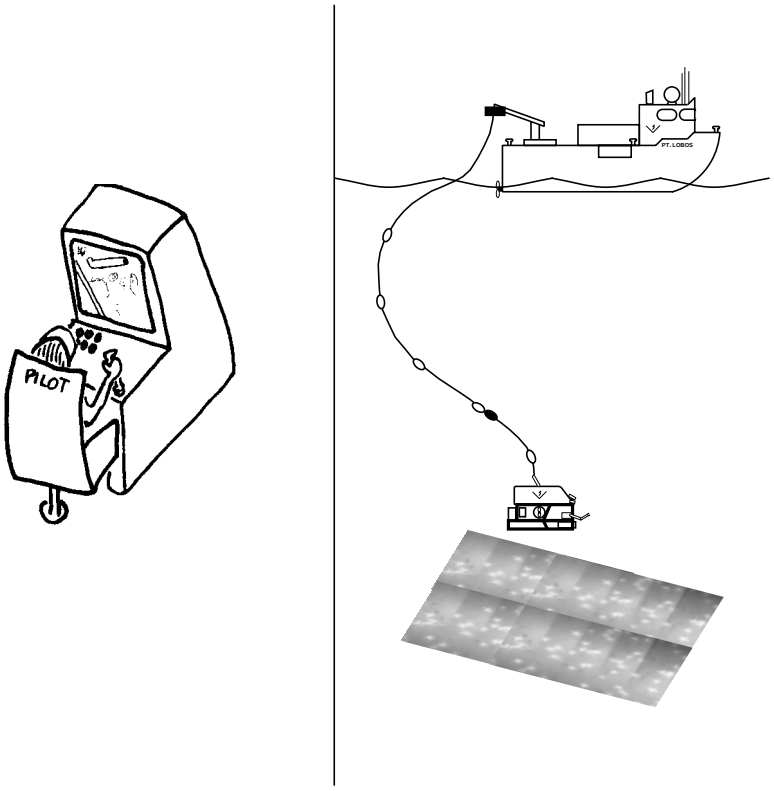

Figure 1.1: **Typical ROV Mission**

To perform a video survey of the brachiopod field, the pilot controls the ROV directly to follow the desired survey pattern. Pilot commands are sent from the console on-board the support ship to the ROV through its tether, while live video and other data are received from the ROV through the tether and displayed to the pilot.

specific tasks. To perform complex science-oriented missions in the absence of human intervention, new capabilities must be developed. Thus, the benefit of autonomous navigation is even more profound for AUV than ROV missions; a navigational capability enables AUV exploration missions that could not be accomplished otherwise.

As an example, the National Aeronautics and Space Administration (NASA) has proposed a fully autonomous mission to visit Europa, one of the moons of Jupiter, and search its oceans for evidence of primordial life. The core task of this mission requires an AUV to locate hot vents at the bottom of the Europa oceans, collect relevant science data, and relay the data back to Earth—all without any human intervention (Figure 1.2). The first

phase of the task, searching for the ocean-floor hot vents, requires the AUV to navigate itself within a completely unknown environment.

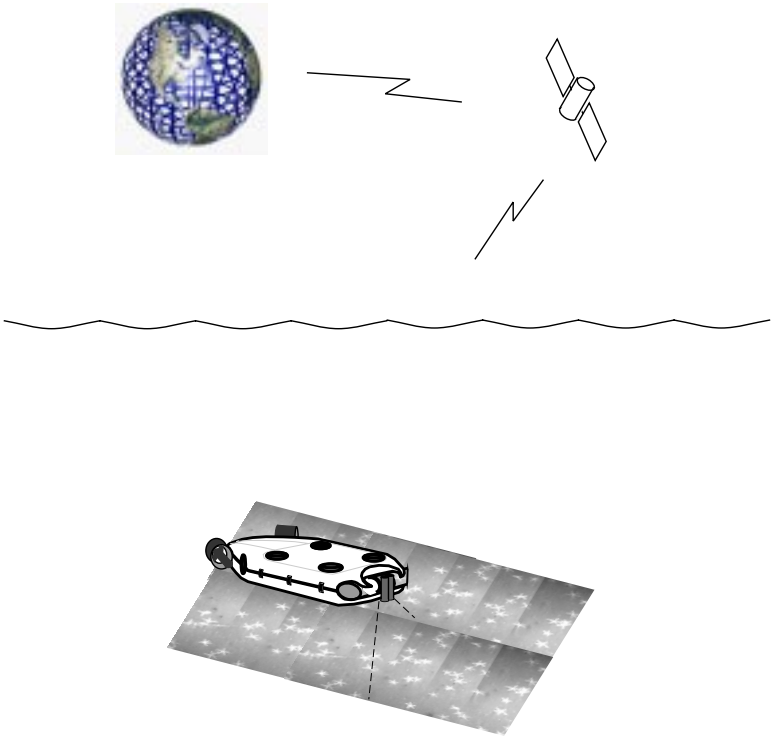

Figure 1.2: **Typical AUV Mission**

The first phase of the proposed NASA mission to Europa is to search for hot vents on the ocean floor. The underwater robot must be fully autonomous, since online human intervention during mission execution is impossible over the slow communications link to Earth.

To solve the problem of UUV navigation, this thesis proposes to utilize vision-based sensing techniques to create composite images, or mosaics, of the ocean floor, for use as reference maps in the navigation process. A mosaic is created in real-time by aligning successive images from an on-board camera to form a composite image of the scene, as the vehicle travels along its path. A typical underwater mosaic depicted in Figure 1.3 illustrates how the mosaicking process creates the composite image by snapping new images and adding them to the existing mosaic as needed to ensure a minimum overlap between adjacent images. The mosaic map is the primary benefit of the visual sensing process, and it is a more compact representation of the visual data than the live video feed coming directly

from the on-board camera. In this approach, the vehicle is able to navigate itself using the mosaic map as it is being constructed—no *a priori* map or map construction stage is needed. Given the vehicle altitude above the ocean floor (e.g. from an altimeter) and camera field of view, the actual area covered by the mosaic map (and individual images) is known, so navigation in real world coordinates (e.g. meters) is possible.

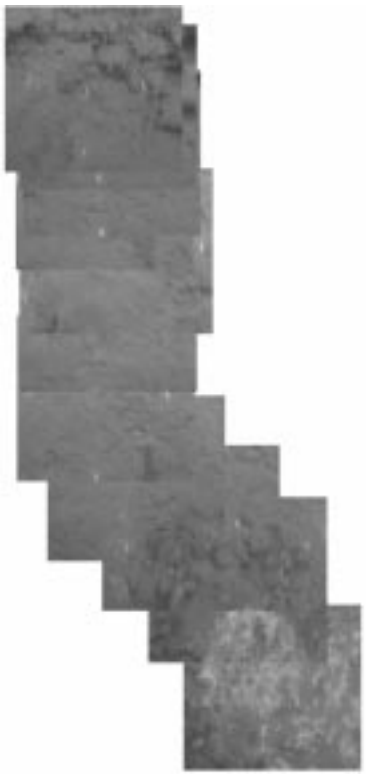

Figure 1.3: **Typical Mosaic of an Underwater Scene**

This mosaic was created in real-time from eleven individual images of the ocean floor.

Vision sensing for UUV control has opened up new possibilities in task automation for both AUV's and ROV's, as illustrated in Figure 1.4. All of these tasks utilize the same underlying visual sensing technology to detect vehicle or object motion. While object tracking is generally performed in the midwater regime, station keeping and mosaicking are near-bottom tasks, and navigation is a generalization of both station keeping and mosaicking. The tasks will be discussed in detail in Section 1.2.3. Independent of its role in UUV

navigation and control, vision is more beneficial as the primary sensor than other viable options for many missions, (Section 1.2.2) because visual information is inherently intuitive to humans in displaying scientific information.

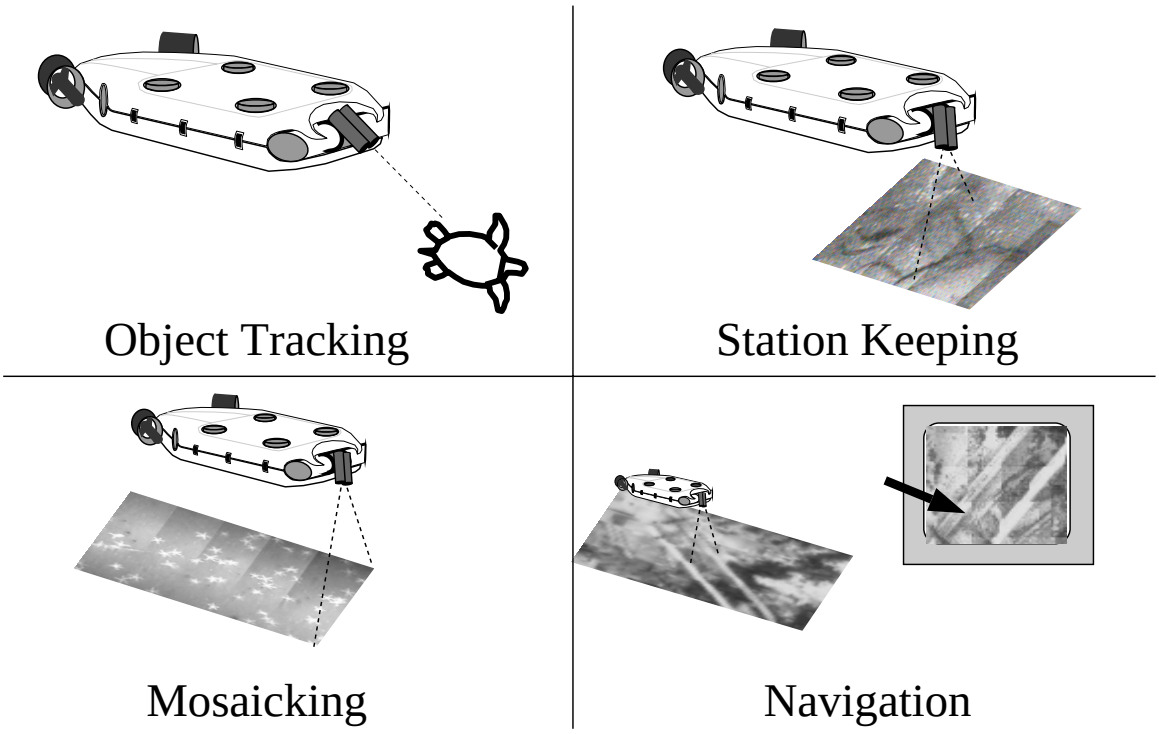

Figure 1.4: **Vision-Based Robotic Control Tasks**

For each of these tasks, the feedback data used to control the vehicle is extracted from the live camera video.

However, vision sensing, specifically mosaicking for navigation, is not without its challenges. The use of cameras for guidance and control limits the robot to motions that are within visual range of the ocean floor (or target surface of interest). While an on-board camera is a passive sensor (i.e. the sensed energy is not transmitted by the sensor), the need to carry artificial lights results in an active sensing system that may disturb light-sensitive creatures or cause other problems, depending on the application.

More importantly, the current state-of-the-art in vision-based navigation is essentially a form of dead reckoning [7]. In land-based vehicles, a count of wheel rotations is used to determine how far the vehicle has traveled. UUV's can utilize an analogous method by

counting images. In creating a mosaic, a series of images are taken from a video stream and aligned with each other to form a chain of images along the vehicle path. When a new image is about to be added to a mosaic, it must be properly aligned with the last image in the chain of images comprising the mosaic. To accomplish this, the two images are compared, and the displacement vector between the two image centers is calculated. For this research, the origin of absolute, or global, coordinates, is defined to be the location of the initial image in the mosaic.[1] Therefore, to estimate the current absolute vehicle position, it would be possible to compute the total distance traveled by summing the image displacement measurements along the image chain. This position estimation technique is known as *dead reckoning*.

The dead reckoning approach to vehicle position estimation has a fundamental problem: the unbounded propagation of errors on vehicle position over time. This random-walk effect is due to the accumulation of image alignment errors as the length of the mosaic increases (Figure 1.5). Although the error between consecutive images is small because the image correlation measurements are accurate to within a few pixels, the error in placing the most recent image relative to the starting point (i.e. the absolute origin) increases without bound as more images are added to the mosaic.

This unbounded error propagation restricts the use of vision as a navigational sensor, because dead-reckoned vehicle position estimation errors lead to potentially unbounded navigation errors. The *navigation error* is defined as the difference between the user's specified goal location and the vehicle's current location, and it includes both the sensing error in measuring the relative displacement between the desired and current vehicle locations, and the control error in regulating the relative displacement to zero:

$$\text{navigation error} \quad = \quad \text{sensing error} + \text{control error} \tag{1.1}$$

[1] To navigate in a worldwide inertial coordinate system that is common to all mosaics and even other sensors, the initial image location with respect to the inertial frame must be known *a priori*.

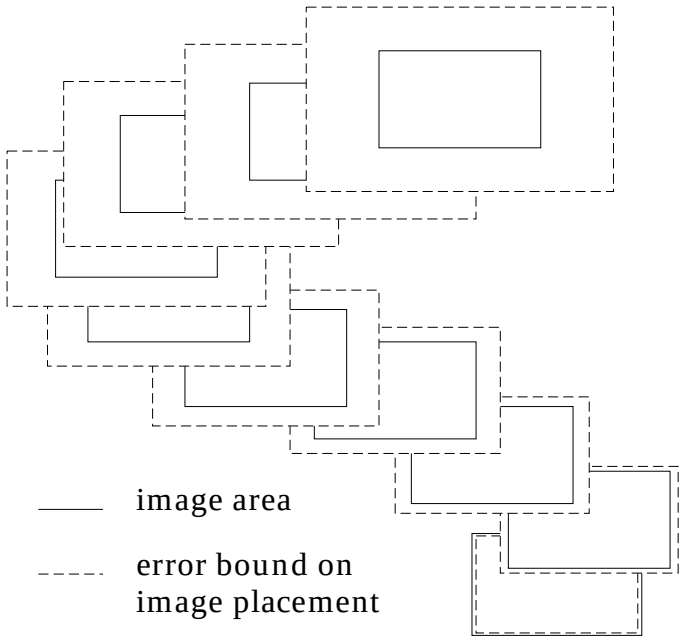

Figure 1.5: **Error Propagation in Image Chain**

As the length of the image chain comprising the mosaic increases, the error bound in placing the last image relative to the initial image (i.e. the origin) continues to grow according to a random walk.

In the dead reckoning approach, the goal location is specified in global coordinates, so the sensing error is equivalent to the error in estimating the vehicle position with respect to the global frame, G. As shown in Figure 1.6, the dead-reckoned sensing error, $\mathbf{e}_{sensing}$, equals the difference between the estimated vehicle global position, $\mathbf{r}_{est}$, and its true location, $\mathbf{r}_{truth}$. Thus, unbounded dead-reckoned errors on the current position estimates result in unbounded navigation errors. The control error component depends on the control system and vehicle dynamics, and it will always be bounded for a robust, well-designed controller.

This dissertation proposes visual map-based navigation as a solution to overcome the challenges of dead-reckoned navigation. In this novel approach, the mosaic is used as a reference map for navigation to solve the problem of unbounded navigation error (Figure 1.7). The current image from the on-board camera is compared directly to the overlapping section of the mosaic, in order to localize the vehicle relative to the map. Since the desired goal location is visually specified by clicking on a point in the mosaic, the sensing error

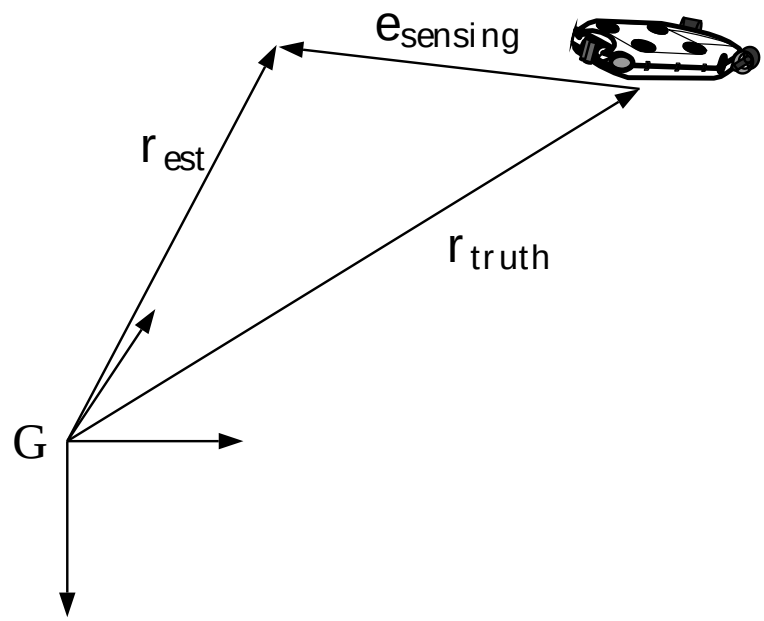

Figure 1.6: **Dead-Reckoned Sensing Error**

G is the origin of the global reference frame. $\mathbf{r}_{est}$ and $\mathbf{r}_{truth}$ represent the estimated and true vehicle global position vectors, respectively. $\mathbf{e}_{sensing}$ is the sensing error component of the dead-reckoned navigation error.

component of Equation 1.1 is equivelent to the error in estimating ${}^{map}\mathbf{r}_{veh}$, the vehicle position relative to the map. ${}^{map}\mathbf{r}_{veh}$ is accurate to within the error of a single image alignment measurement, regardless of the error in estimating $\mathbf{r}_{map}$, the map coordinates relative to the global frame, G. As a result, the sensing error component of the navigation error is on the order of a few centimeters for a typical vehicle altitude of a meter above the ocean floor, and the navigation error remains bounded at all times. If global vehicle position estimation is required for other applications, the map-based approach produces bounded-error position estimates, since (as it will be shown) it is possible to generate mosaic maps that have bounded errors with respect to the global frame.

While the map-based approach to navigation is superior in concept to dead reckoning, several issues must be addressed. For instance, techniques to localize the current image within the entire mosaic map must be developed, since the estimation process is no longer

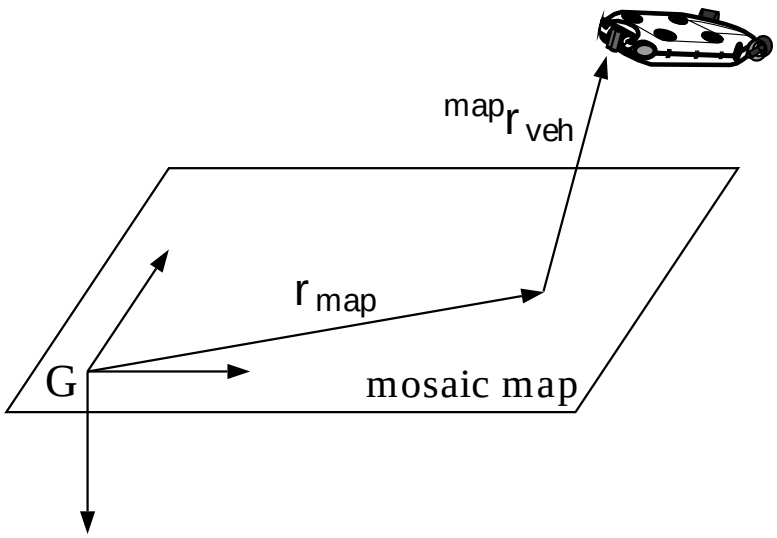

Figure 1.7: **Map-Based Sensing Error**

G is the origin of the global reference frame, located at the center of the initial image in the mosaic. $\mathbf{r}_{map}$ *represents the map location in global coordinates, and* ${}^{map}\mathbf{r}_{veh}$ *represents the current position of the vehicle with respect to the mosaic map. The sum of these two quantities equals the current estimate of global vehicle position.*

a simple image comparison between the current image and the previous image in the image chain. Furthermore, a running estimate of vehicle position relative to the map could be computed to narrow the search within the map.

The fundamental issue to be addressed is the internal consistency of the mosaic map. As the mosaic grows, errors in alignment of the most recent image with respect to the rest of the mosaic accumulate in the same manner as dead reckoning. Thus, if the image chain were to loop back upon itself, there could be significant mis-alignment at the crossover point. This could lead to the situation where the same point of interest is duplicated in two different locations in the mosaic. The need for map self-consistency also raises the question of how to estimate the errors in image alignment, since these errors must be known in order to re-align the mosaic properly. To minimize the map inconsistencies, a method will be developed that registers each of the images with respect to an absolute coordinate system and performs a global mosaic re-alignment. A side effect of this approach is that the estimation error in absolute position, not just the navigation error, for every point in the mosaic is bounded.

This enables tasks such as the fusion of mosaic data with dissimilar sensors, or the direct measurement of distances within the mosaic.

This thesis will explore these issues and solve fundamental problems to enable bounded-error, visual map-based navigation for AUV's and ROV's.

1.2 Background

This section provides a background on the state-of-the-art of UUV technologies, underwater navigation systems, and vision-based control for UUV's.

1.2.1 State-of-the-Art in UUV Technology

The current generation of UUV's is capable of performing a wide variety of missions in the deep ocean environment. While a particular vehicle is usually designed with a specific subset of tasks in mind, the set of all possible tasks that can be accomplished currently by underwater robots can be grouped into general classes, several of which are listed here:

- Mapping
- Science Sensor Data Collection
- Instrument/Equipment Placement and Servicing
- Biological and Geological Sample Collection
- Inspection of Man-Made Structures
- Pipeline/Cable Tracking and Inspection

UUV's can be classified into two general categories: ROV's and AUV's. For the most part, the above tasks are accomplished only by ROV's, since they have the advantages of a human in the control loop, directing the vehicle at all levels in the control hierarchy. In the marine science community, ROV's are a major tool for gathering scientific data. For instance, the MBARI-owned Ventana ROV performs daily missions in the Monterey Canyon,

and it has a wide range of optional sensors and interchangeable toolsleds to carry out a whole spectrum of tasks typically required by the scientists. The Jason/Medea vehicle, developed by Woods Hole Oceanographic Institute (WHOI), is primarily a real-time optical imaging platform. It takes a novel approach to achieving high-precision ocean surveys: Jason is a separate, highly maneuverable ROV that is deployed from the Medea ROV platform. While Medea performs wide-area surveys, Jason enables precise inspection and survey of relatively small areas.

While the benefit of human experience enables ROV capabilities that are not possible currently on AUV systems, new automation technologies would greatly ease the operator workload and allow the human pilot to concentrate on higher level aspects of the ROV task and mission performance. The development of Tiburon, a next-generation ROV designed and built by MBARI, is an example of this being done. The design of Tiburon simplifies the integration of novel technologies and sensors into the on-board vehicle systems.

In particular, an autonomous navigation pilot aid would ease the burden of station keeping and near-bottom navigation, which is an intensive and tedious task for ROV pilots. During this research, the navigation system was implemented on the MBARI-owned Ventana ROV (Figure 1.8) for ocean demonstrations.

The current generation of AUV's is capable of a much smaller number of missions than ROV's in real ocean environments. The AUV missions that have been demonstrated successfully represent impressive accomplishments, given the harsh and often unpredictable ocean conditions that must be handled in the absence of human intervention. A list of several AUV systems is presented in Table 1.1. Currently, transit-type vehicles have exhibited the most success in accomplishing science-oriented ocean missions. Odyssey, Autonomous Benthic Explorer (ABE), and Remote Environmental Monitoring UnitS (REMUS) all perform deep-ocean sampling surveys using a variety of science sensors, and Ocean Voyager II performs a similar survey mission in coastal waters. To navigate autonomously, these AUV's utilize a combination of short and long baseline acoustic transponders. Theseus was designed with a particular commercial mission in mind, namely, laying fiber-optic cable

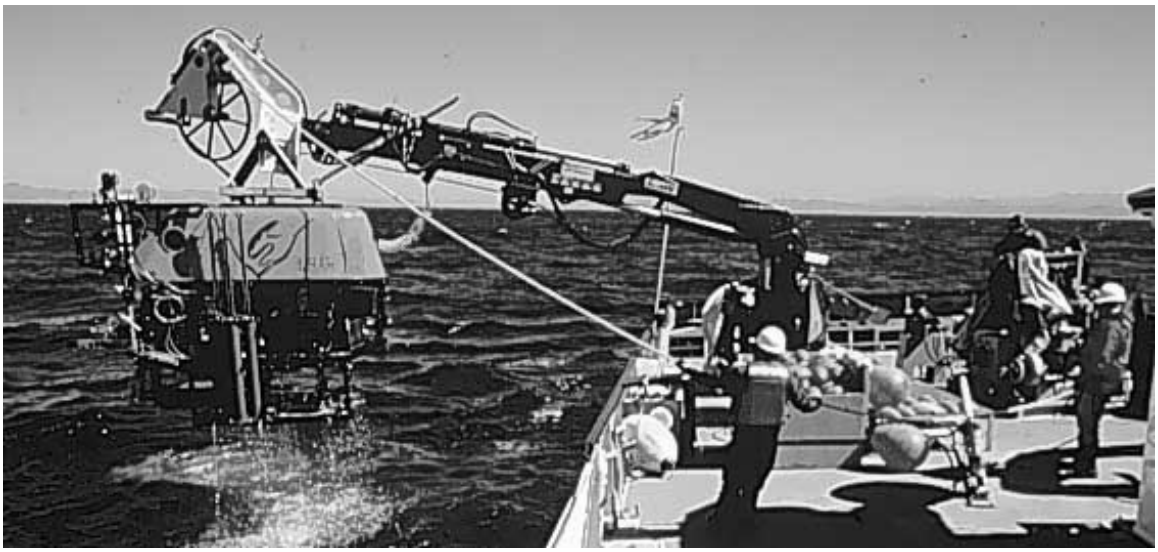

Figure 1.8: **Ventana ROV**

under the Arctic ice. It uses an Inertial Navigation Unit (INU) with Doppler sonar for navigation in transit.

Hover-type AUV's that can interact with their environment require greater intelligence and capabilities than transit-type AUV's, so they are just beginning to reach the open ocean. ABE and REMUS are capable of both hover and docking in deep ocean conditions. Hover-capable AUV's have been used to demonstrate more advanced missions in the controlled environment of the test tank. Much of the work on both the Phoenix and OTTER AUV's focuses on achieving precise, robust vehicle control and navigation using less traditional sensors, such as vision, sonar, or GPS. Specifically, the OTTER AUV (Figure 1.9) was used to demonstrate the vision-based navigation results of this dissertation.

1.2.2 State-of-the-Art in Underwater Position Sensing for Navigation

All of the above ROV/AUV missions require some method to sense vehicle position, in order to enable some form of navigation or station keeping. Several sensor options are available for the underwater environment that provide the UUV with knowledge of its position. As

AUV Robot	Research Organization	Tasks	Test Environment
Theseus	ISE	laying fiber-optic cable under Arctic ice	ocean
Odyssey	MIT	sampling surveys with water quality sensors, side-scan sonar, water-current profiler	ocean
ABE	WHOI	magnetometer surveys of lava flow; docking	ocean
REMUS	WHOI	Acoustic Doppler Current Profiler (ADCP), Conductivity-Temperature-Density (CTD), and side-scan sonar surveys; docking	ocean
Ocean Voyager II	FAU/USF	coastal ocean survey for bottom classification and albedo measurement	ocean
Phoenix	NPS	experiments in AUV control architectures, navigation	test tank/ ocean
OTTER	ARL at Stanford	vision-based station keeping, navigation; object search and retrieval; autonomous manipulation	test tank

Table 1.1: **AUV Robots and Missions**

Figure 1.9: **OTTER AUV**

described in this section, every sensor has particular benefits and drawbacks that make it particularly well-suited to certain tasks. For instance, acoustic transponder networks are suitable for repeated visits to the same site, while inertial sensors may be sufficient for short-duration transits in the midwater environment. For science-oriented missions devoted to near-bottom ocean exploration, cameras are already on-board the vehicle, so vision is a natural choice for high-resolution, low-cost position measurements.

Vision Using various computer vision algorithms, relative position measurements can be extracted from live video imagery. Given the high resolutions of digital imaging, measurement accuracies on the order of millimeters can be achieved. However, this method is limited to regimes where the object or terrain of interest is within both the field of view and visual range of the camera. Also, the need for artificial lights often increases the expense and power consumption of the robot. While relative measurements are quite accurate, integrating these measurements to estimate absolute positions leads to error propagation problems that are the central challenge of this thesis.

Another advantage of this method is that visual information is inherently intuitive to the user. Most underwater vehicles are already equipped with on-board cameras

and many are primarily vision-based, simply because their design goal is to provide humans with a view to the underwater world. For instance, the extensive array of camera and lighting equipment aboard the Ventana ROV provides the pilot with the visual feedback required to control the robot, and it is used by the mission scientist to plan the robot's next task. While other sensors exist on-board Ventana, the video monitors provide the primary interface for both pilot and scientist.

Sonar The propagation of sound waves through water and their reflections off solid objects is a well-studied problem, and as a result, a wide variety of devices has been created to take advantage of these properties. These devices are known collectively as sonar. Sonar equipment has been used to perform a host of different tasks on underwater robots, such as tracking of objects in midwater, determination of altitude above the ocean floor, and multi-image surveys of the marine terrain. While this sensor has been highly successful, the raw sonar image data often must undergo significant processing before they can be presented to the pilot or scientist in a usable format. Furthermore, sonar is an active sensor, since the sensed energy is originally emitted by the device itself; this may not be appropriate for some applications, such as military surveillance or tracking of sound-sensitive animals.

Acoustic Transponder Networks By placing sonar-based transmission and reception devices around a site of interest, precise 3-D position sensing can be achieved within the volume enclosed by the network. The network only tracks objects equipped with a transponder, so natural or unanticipated objects are invisible from the point of view of the network. This type of sensor is ideal for repeated excursions to the same area, although surveying the location of each transponder with respect to the others is often a difficult task. Furthermore, investigation of unexplored regions of the ocean is impossible, since *a priori* transponder setup is required for this method.

Inertial Sensors Sensors such as gyroscopes, accelerometers, inclinometers, and compasses can be used to measure angular rates, linear accelerations, and attitude of the underwater robot. Through the technique of dead reckoning, the vehicle position

can be inferred by integrating the inertial measurements. Unfortunately, dead reckoning is only accurate for short time durations unless one is willing to incur extremely large expenses in acquiring high-precision equipment; since the measurement noise is integrated along with the signal, the errors on position accumulate quite quickly. Consequently, external reset mechanisms are required.

GPS The array of satellites known as GPS is used extensively to provide precise positioning data for land-based and air-based vehicles. Unfortunately, GPS signals do not penetrate the surface of water, so its use in the underwater regime is severely limited. One approach taken in recent work is to equip an AUV with GPS and bring it to the surface at periodic intervals. The GPS signal can then be used to reset drift errors in position that arose from the integration of inertial sensor measurements. Similarly, several organizations have successfully integrated long or short/ultra-short baseline acoustic positioning networks with differential GPS to perform such tasks as autonomous station keeping and biological sample collection [8, 25]. While this technique shows excellent promise for AUV navigation, it does not provide the scientist with any visual or topographical map of the ocean floor. Also, it relies on the previous setup of an acoustic network, so exploration of completely new environments is impossible.

The best choice for position sensing may differ among various vehicles and applications, so the decision must be tailored to meet the specific needs of the pilot or scientist. For UUV ocean exploration missions where vision is already being used as the primary science sensor, vision-based position sensing may be the best choice to enable vehicle navigation.

1.2.3 State-of-the-Art in Vision-Based Control

The decision to use vision-based position sensing enables control of the UUV in a variety of different modes, depending on the needs of the specific mission. The tasks depicted in Figure 1.4 represent the state-of-the-art in real-time, vision-based control of UUV's in unstructured, unknown environments. Using the same core functionality of texture-based

image correlation (Chapter 3), these tasks all perform vision-based control of the underwater robot, but they differ primarily in the type of vision data used for feedback.

Object Tracking

In this task, the camera(s) is pointed at an object of interest. In order to discern the object from the background, two different techniques may be used, possibly in conjunction with each other. Both of these object tracking methods were demonstrated in previous research by Richard Marks, as part of the ARL/MBARI joint program [13].

The first technique distinguishes the object by measuring its motion in the image plane relative to the background motion. This optical flow method enables the vehicle control system to maintain the object in the center of the camera field of view.

The second method is a stereo vision technique that compares two images taken simultaneously from a camera pair and calculates the range at a series of grid points in the images. Thus, the object can be tracked by taking advantage of the fact that it is closer to the vehicle than all background points. This enables full 3-D position control of the vehicle relative to the object.

Station Keeping

The robot's goal during the station keeping task is to hold station over a fixed point on the ocean floor. For instance, if the scientist wants to study a particular rock formation or sea star, the vehicle will hover over the point of interest and maintain that point in the center of the camera's field of view. Previously, this task had been achieved by several researchers using optical flow techniques to measure image motion [21, 22, 37]. However, this method resulted in poor performance, since the vehicle position estimates would drift over time.

To improve station keeping performance, a new method was devised. Using this method, an initial image of the scene is stored and becomes the reference image. Then, subsequent live images are compared to the reference image, and the motion away from station is measured. The pilot or automatic control system attempts to compensate for this motion and drive the vehicle back to station. Since all motion measurements are made relative

to the reference image, this method has the advantage that measurement errors do not accumulate. As a result, the vehicle will never drift off station over time. However, the magnitude of vehicle motions is limited to the reference image's field of view. Thus, large currents may push the vehicle too far off station, and the vehicle will not be able to recover itself. This newer technique has been demonstrated experimentally in both test tank and ocean environments [10, 15, 35].

Mosaicking

A visual mosaic of the ocean floor is created by aligning successive images from an on-board camera to form a composite image of the scene. Mosaicking can be considered to be an extension of the station keeping task, where vehicle motion is no longer limited to a single field of view. Since the live image is continuously aligned with the reference image in the station keeping task, it is possible to snap a new reference image at any time and place it next to the original reference image. Then, the new reference image can be used for comparison with live images. If this process is repeated whenever the live image approaches the edge of the most recent reference image, a visual map of the scene can be constructed as the vehicle moves to new goal positions. This mosaicking technique has the advantage that it allows planar vehicle translations of arbitrary size. Furthermore, while the accumulation of errors does occur, the drift increases only when a new image is snapped for the mosaic, not every time the live image is compared with the reference image.

The ARL/MBARI project has achieved interesting results in the area of constrained video mosaicking, in which a multiple-column mosaic is created by correlating the images in adjacent columns [17]. This effort has also produced impressive single-column mosaics of the sea floor using Ventana, the MBARI ROV. Related research has investigated the possibility of extending the concept of mosaicking to include 3-D motion estimation for vehicle control [36]. There has also been promising theoretical work to solve the occlusion problem when mosaicking terrain with significant altitude variations [6, 32]. While this research has been quite successful, the intensive computations required currently prohibit a real-time implementation of the necessary algorithms.

Navigation

Previous research has already demonstrated this task in the form of dead-reckoned navigation [7]. However, the specific goal of this research is to develop, implement, and demonstrate a bounded-error, map-based navigation system in a realistic underwater environment.

Robot navigation from video is an extension of the concept of mosaicking. The mosaicking task already provides a reference map that can be utilized by both robot and human. By comparing live images to the mosaic map, the robot can compute an estimate of vehicle position. An automatic control system then uses this estimate to control the vehicle to the desired location. In order to specify goal locations, the user is provided with an intuitive interface. This interface enables the pilot or scientist to point-and-click on any section of the mosaic that is being created, and the robot will drive itself to the new desired position.

1.3 Research Objectives

In order to develop and implement a complete vision sensing system, the research described in this thesis will meet the following specific objectives:

- Identify appropriate techniques for creating visual maps, tagging the maps with relevant position information, and updating the maps as new information becomes available.
- Quantify the noise characteristics of texture-based image correlation measurements.
- Develop new theory to maintain the internal consistency of the mosaic map through global re-alignment, despite significant noise on the image alignment measurements.
- Invent a novel method to provide bounded-error vehicle localization with respect to the mosaic map.
- Design a complete system structure to accommodate the various stages of mapping and position estimation.

- Verify the new vision-based sensing capabilities against truth measurements in the laboratory environment.

In order to demonstrate real-time navigation on several underwater vehicles, using the newly developed vision sensing system, the following objectives will be met:

- Design a complete system structure to integrate the vision sensor, vehicle control, and user interface components required for real-time vehicle navigation.
- Test the navigation system on underwater vehicles in both controlled and realistic environments.

The remaining chapters in this thesis document the entire research process, from concept to execution. They describe the above objectives in detail and explain how each objective was accomplished.

1.4 Reader's Guide

This chapter presents the goal of this research to be visual map-based, bounded-error navigation for UUV's, and it motivates this work by demonstrating its importance in fulfilling the need for remote scientific exploration of the ocean environment. Chapter 2 describes the approach to solving the problem of map-based navigation. Specifically, it outlines a series of tasks to be accomplished, and then it highlights the stages in the implementation of the vision sensing task. Chapter 3 describes the rationale behind the specific computer vision algorithms chosen for this work, and it provides a technical background on the relevant technologies.

Chapter 4 introduces the various software and hardware platforms used during the course of this research, including technical specifications where appropriate. Chapters 5–7 explain the three major components of the vision sensing system in technical detail, namely, the state estimator, crossover detection/correlation algorithm, and smoother. Furthermore, these three chapters provide experimental verification in the lab environment of the correctness and accuracy of these components. Chapter 8 presents the results of experimental

demonstrations of the complete navigation system on two different platforms: the OTTER AUV in the test tank, and the Ventana ROV in the Monterey Canyon deep ocean environment.

Chapter 9 summarizes the contributions of this research. In addition, it presents several possibilities for future research efforts that would extend the results of the current work in new directions.

Chapter 2

Approach Overview

The goal of this chapter is to describe the approach to visual map-based navigation in detail, highlight the relevant issues, and present the structure and components of the core task, vision sensing.

2.1 Introduction

In presenting the novel approach to solving the problem of UUV navigation taken in this work, it is useful first to define the concept of *navigation*, and then to introduce the unique features of the proposed system. For the purposes of this dissertation, navigation entails the ability to direct a vehicle to a specified location or along a specified path. Thus, it can be differentiated from vehicle control in the sense that the control system is merely a central component of a navigation system.

Navigation encompasses three aspects of the human-robot system: sensing, control, and location/path specification. A variety of options exist for each of these components, as explained below. They are depicted graphically in Figure 2.1, within the overall architecture for the proposed navigation system. As seen in the system diagram, the vision sensor component produces an output signal, y, which is differenced with the desired signal, r, to produce an error signal, e. The controller component attempts to zero this error signal by sending an appropriate control signal, u, to the plant (e.g. underwater vehicle). For this

work, the desired signal, r, is generated from a graphical user interface (GUI) presented to the pilot that serves as the location/path specification component.

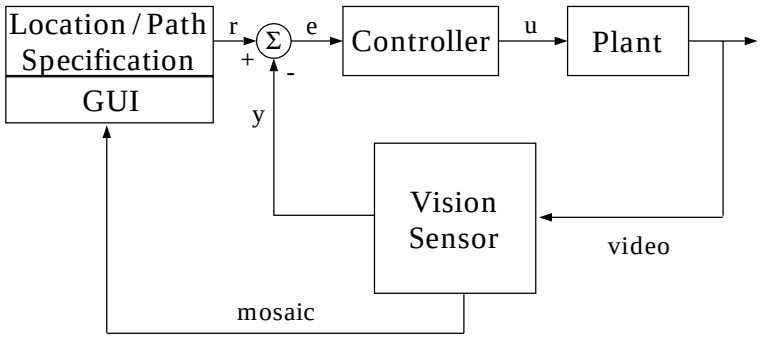

Figure 2.1: **Block Diagram for Navigation System**

The development of the vision sensor component for navigation is the technical focus of this dissertation. Section 1.2.2 has already discussed the sensing options and justified the use of vision. The issues in creating a map-based vision sensor are explored in Section 2.3. In response to these issues, the solution approach is presented in Section 2.4. To solve the problem of error propagation within the mosaic map and maintain its internal consistency, a new method for re-aligning the mosaic has been developed. The concept of the mosaic re-alignment method is to minimize image alignment error within the mosaics by taking advantage of loops in the image chain (Section 2.4.1). This concept is generalized into a three-stage estimation process (Section 2.4.2): a state estimator to estimate absolute image and vehicle positions from relative image displacement measurements, a crossover algorithm to handle loops in the mosaic, and a smoother to align optimally the images within the mosaic map. The fundamental contributions of this research are a direct result of the vision sensor development; these are briefly discussed in Section 2.5.

Control theory provides many options for robot control, but the chosen controller must be able to work with the chosen sensor, and it must be able to handle arbitrary reference inputs. The controller components used on each of the experimental vehicle systems will be described in Chapter 4.

The location/path specification component provides the control system with its reference inputs. For ROV's, it is usually some type of user interface; for most AUV's, a method for pre-programming mission specifications often is utilized, while some AUV's are directed through user interfaces. The specification component may include path planning algorithms and other modules, but this is not required. For the experimental work presented in this dissertation (Chapter 8), a GUI will be utilized to specify goal locations to the navigation system. The interface consists of the mosaic of the underwater scene, a marker to indicate the current vehicle position within the mosaic, and a marker to indicate the goal location for the vehicle (Figure 2.2). The mosaic and the current position markers are updated whenever new information is received from the vision sensor, and the user is able to use a mouse to point-and-click on any section of the mosaic (or an unexplored area) to specify a new goal location. The image-based goal specification is then translated into an absolute desired position, to be used by the vehicle control system.

Section 2.2 describes the unique features of the proposed visual map-based navigation system, while the remaining sections of this chapter delve into the task of creating the vision sensor.

2.2 Unique Features of the Navigation System

The work presented in this dissertation focuses on creating a vision-based sensing system for UUV navigation. This research distinguishes itself from other attempts at vehicle position sensing in the following ways:

- The vision sensor is capable of performing in *unknown, unstructured environments.* No previous knowledge of the scene is taken into account and no inherent structure or geometry is assumed, unlike other forms of navigation such as pipe-following methods.

- The proposed sensing solution is *map-based.* The local image displacement measurements are combined to form a global map of the environment. Specifically, a video mosaic is created for real-time navigational use by both the human and the robot.

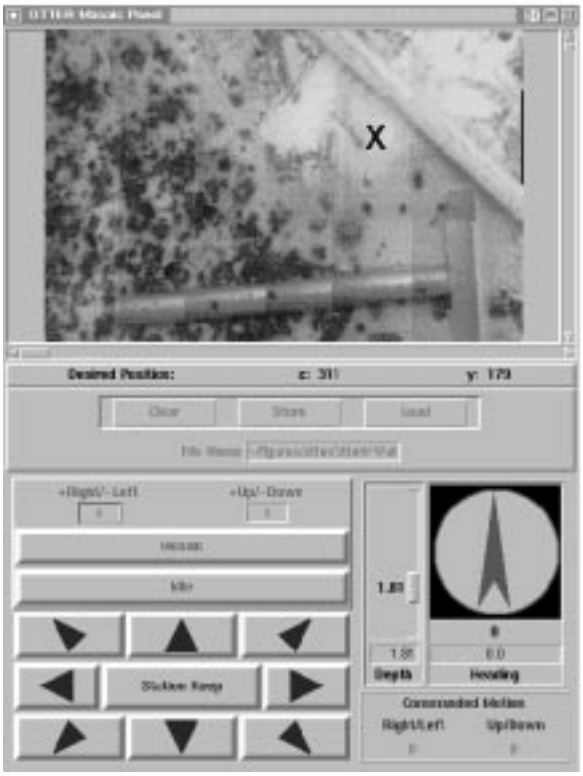

Figure 2.2: **Mosaic-Based Graphical User Interface**

To specify goal locations, the mosaic is presented to the user and updated as new information becomes available. The 'x' represents the current position of the vehicle relative to the mosaic map, and the 'o' represents the desired vehicle position.

- In contrast to dead reckoning methods, *bounded-error* navigation is achieved. By using a video mosaic as a navigational map, the sensing error in measuring the relative displacement between current and desired vehicle location is bounded, regardless of the amount of time elapsed, number of images comprising the mosaic map, distance traveled, or area covered. This leads to bounded navigation error, assuming the vehicle controller always remains stable. Furthermore, the error in determining absolute robot location is bounded for a scene of bounded size.

- All of the computations required for mapping and vehicle localization are performed in *real-time*. Therefore, the vision sensor can be integrated into a navigation system for operational underwater vehicles.

To demonstrate the results of this research, the novel vision sensing system is implemented on several experimental platforms and integrated with the robot control systems to enable visual map-based navigation. These demonstrations share the following characteristics:

- *Autonomous* navigation is performed. In other words, the vehicle control system, not a pilot, is responsible for controlling the robot along the desired path. Thus, the human is placed at a higher level within the system hierarchy.

- All of the computations required for sensing, control, and ultimately navigation, are performed in *real-time*. As a result, the user is no longer responsible for low-level, high-bandwidth interactions and instead can focus on higher-level tasks related to the mission.

2.3 Issues in Vision Sensing

The central challenge of this research is the creation of a visual map-based sensor for bounded-error UUV navigation. The problem can be broken down into three subordinate tasks: map creation, map re-alignment, and vehicle localization within the map. Since the vision sensor is intended for use on-board operational underwater vehicles, all three tasks must be designed to perform in real-time.

The following sections provide general discussions on the issues to be considered before deciding how to accomplish each of these three tasks. Once the relevant concepts have been presented, Section 2.4 is devoted to providing a detailed explanation of the solution approach.

2.3.1 Photographic Map Creation

The term *mapping* is an extremely broad concept that covers a myriad of environments, sensors, platforms, and computing algorithms. For instance, mapping may refer to sonar bathymetry of the ocean floor, satellite photography of the earth's surface, landmark-based digital elevation maps for missile guidance, or a host of other possible applications. If the map is to be included as part of a vehicle navigation system, there is a significant restriction on its design: the map must be accessible (and modifiable, for the case of dynamic mapping) in real-time. This requirement places a severe computational limit on the map-access and map-building algorithms, that precludes the use of many of the more sophisticated techniques of photogrammetry and related fields.

For this particular work, the map creation task must accomplish two goals:

- It must produce a macroscopic, high-resolution, multiple-image view of the environment. This composite-image view is a useful product in itself, independent of the map's utility for vehicle position sensing and navigation.
- It must provide a global reference map for bounded-error vehicle navigation. To accomplish this, the mosaic is augmented with absolute position information and compared to the live camera image in real-time.

Visual-Based Maps

In Chapter 1, the merits of using vision as the primary sensor for navigation in the near-bottom ocean environment were discussed. Vision sensing enables the creation of photographic maps, or *mosaics*. A visual mosaic of the ocean floor is created by aligning successive images from an on-board camera to form a composite image of the scene. Mosaics are particularly well-suited for the human side of the man-machine system. If a human were to be placed in an underwater environment, vision would be the primary perceptual mode for interpreting the scene; mosaics present this perceptual feedback to the user remotely. Therefore, the remainder of this discussion will be restricted to real-time video mosaicking in the underwater environment.

The approach taken in this work to enable the creation of photographic maps for navigation is based entirely upon the work of Richard Marks. In his thesis [16] and other published works [12, 13, 14, 15, 17], Marks has developed and implemented efficient, real-time techniques for underwater visual sensing, including object tracking, station keeping, and mosaicking. Chapter 3 describes in detail his relevant techniques, and it presents his work as the lowest level component in the mapping phase of the vision sensing system. Since the mosaics produced by this component will be utilized as reference maps for vehicle navigation, a method for updating the mosaic with new sensor data must be developed. This method will be introduced in Section 2.4.2 as part of the approach to creating a vision-based sensing system.

2.3.2 Map Expansion and Re-Alignment

This task addresses the issue of how mosaics can evolve over time as new information is received. The new sensor data can augment and/or conflict with the existing mosaic data, leading to internal inconsistencies in the mosaic map. To deal with these inconsistencies, the mosaic images are registered with respect to a global coordinate system. Once the mosaic is tied to an absolute frame of reference, minimizing internal inconsistencies becomes equivalent to minimizing the absolute errors on image placement within the mosaic.

The following discussion explores these issues briefly in relation to the simpler case of pre-existing static maps, and the methods for mosaic expansion and re-alignment are discussed in detail in Section 2.4.

Static vs. Dynamic Maps

The use of an *a priori* map aids the navigation process by providing a common frame of reference for both the human and the vehicle. For the human, the map provides an overall picture of the entire scene and enables selection of vehicle goal locations. The vehicle can use the same map, in conjunction with on-board sensors, to determine its location within the map. In order to ascertain its current position, the vehicle must correlate its current sensor readings with the data stored in the map.

While useful for navigation, this type of *a priori* map is static. In other words, it has no capability to evolve over time, as new information is collected by the vehicle sensors. By integrating an online mapping system with the vehicle sensor suite, it is possible to create dynamic maps that take advantage of the most recent information available. This dynamic mapping technique has two significant advantages:

Refinements in Map Accuracy For static maps, the accuracy of any position estimate is limited by the accuracy of the map data. However, new sensor data could provide additional information that possibly exceeds the map accuracy. In this case, either the new sensor data could replace a section of the map, or they could be combined with existing data in some fashion to improve the map accuracy. In turn, this would improve future vehicle position estimates that utilized that section of the dynamic map. This modification would be most beneficial when the vehicle is within the space already covered by the current map.

Expansion of Coverage Area A major goal of many navigation applications, particularly the underwater application that is central to this thesis, is the exploration of unknown environments. In order for navigation to be successful in this regime, it is essential that the map used for navigation evolve over time to include newly discovered areas. This is contrary to the definition of a static map. Thus, a method could be devised to append new sensor data to a dynamic map, such that the map grows in size whenever the vehicle travels beyond the current map borders.

As in the case of static maps, a method for fusing the current sensor data with the map-based data to localize the vehicle within the map must be developed. While similar to the static case, the problem becomes slightly more complex since the dynamic map is constantly growing and re-aligning itself.

2.3.3 Vehicle Localization

The proposed map-based approach to vision sensing greatly simplifies the task of vehicle position estimation. The construction and maintenance of a mosaic map reduces the problem to localizing the vehicle within the map. Specifically, a method must be devised to compare a live camera image from the vehicle to the dynamic mosaic map. However, it is too computationally intensive to correlate the live image against the entire composite-image mosaic. Instead, it is possible to augment the images within the mosaic with global position information, thereby enabling more efficient techniques for vehicle localization within the map. The following discussion introduces several technical concepts that are relevant to the problem of vehicle position estimation, while the solution approach is discussed in Section 2.4.2 within the context of the entire vision sensing system.

Local vs. Global

For the sake of clarity in the explanations to follow, it is important to make the distinction between *local* and *global* quantities. Within this thesis, the concepts *local* (or *relative*) and *global* (or *absolute*) are applied to two different mathematical constructs: frames and vectors. The term *global* is never used to refer to the idea of worldwide localization, as is the case for GPS.

When referring to frames of reference, defined by an origin and a set of unit vectors along its axes [4, pp. 23–24], a local frame refers to any frame that is allowed to translate or rotate arbitrarily with respect to inertial space. A local frame is not required to be moving, since local frames may also be fixed in inertial space. A global reference frame must be fixed in inertial space. Thus, there may be more than one global frame in a particular system. It is generally assumed that the global frame(s) has an origin coincident with some initial location, and its axes are aligned along physically meaningful directions (e.g. the direction of gravity, or the surface of the ocean floor).

From a geometric standpoint, vectors are independent from any reference frame. However, in practical use, vectors are usually associated with a specific frame, and its components are expressed in the coordinates and units of its parent frame. Thus, local and global vectors are simply vectors which are defined with respect to local or global frames, respectively.

Displacement vs. Position

A displacement vector typically refers to the difference between two position vectors. From a mathematical point of view, each vector is defined completely by its magnitude and direction. From a physical point of view, these vectors are slightly different. A displacement vector is a free vector, in that there is no specific location or point of application for this type of vector. A position vector is a bound vector, since by definition it starts at the origin of its parent frame. [5, p. 3] [4, pp. 20–21, 56–57]

However, this may lead to some confusion, since a position vector may be viewed as a displacement vector from the zero vector (origin) to the point in question. Similarly, a displacement vector in one frame may be considered as a position vector with respect to a frame whose origin coincides with the head of the displacement vector.

For use in this text, the terms *displacement* and *position* are defined exclusively with respect to a global frame. For instance, "local image displacement" refers to the difference between the global position vectors of two overlapping images (i.e. the relative distance between the images), even though "local" refers to the local frame of the first image, in which this same vector is considered to be a position vector.

State

Up until this point, only the sensing and estimation of image and vehicle positions have been considered. To be more precise, the intent of this task is to estimate the full image and vehicle states at all times. The concept of *state* in the context of this thesis is slightly different than the standard definition of vehicle position and velocity. *State* is defined to be a 6-DOF representation containing both position and orientation information. The position

is typically expressed in the form of a 3x1 vector from the origin of the relevant frame. The orientation can be expressed as a 3x1 vector of Euler angles, or alternatively, as a 3x3 rotation matrix.

Error vs. Variance

Before proceeding any further in this discussion, it is necessary to define the concept of measurement error precisely. In previous sections, *error* has referred to the vague concept of inaccuracies in estimating image or vehicle states. From this point onward in this document, the term *measured error*, or simply *error*, will refer specifically to the difference between a state's actual value (i.e. truth) and its measured value (i.e. estimate). Therefore, this term can only be applied to specific cases where these quantities could be extracted (at least in theory), such as during an individual experimental run.

On the other hand, it often will be necessary to predict or estimate what the measured errors of the vision sensor will be under typical operating conditions. These error estimates take the form of probability distributions, that define the likelihood of all possible error values. One can consider these distributions to be time averages over many data samples, or alternatively, ensemble averages over many experimental runs. This probabilistic interpretation leads to the concepts of *variance* and *predicted error bound*. The *variance* is a well-known term in probability theory that defines the width of a probability distribution around its mean. The *predicted error bound*, or simply *error bound*, is derived directly from the variance and defines an envelope of magnitudes in which the measured error probably will lie.

2.4 Approach to Vision Sensing

This section describes the solution approach to creating a visual map-based sensor for bounded-error UUV navigation, while accomplishing the tasks and addressing the issues presented in Section 2.3.

A map-based approach was chosen over dead reckoning because it enables bounded-error navigation. The construction and maintenance of an internally consistent mosaic map enables direct measurement of the relative displacement between the current and desired vehicle positions. Since the error on this single measurement is bounded, the navigation error remains bounded (assuming stable vehicle control).

Conceptually, navigation is performed with respect to the mosaic map coordinates, irrespective of any absolute external coordinate system. Working directly with the mosaic map is more appropriate: the goal locations are specified visually, the mosaic data (not global position data) are available directly from the camera sensor, and navigation error, not absolute position error, is the primary concern.

In practice, the mosaic images are registered with an absolute coordinate system in an effort to minimize map inconsistencies. The mosaic re-alignment method uses the absolute position errors to provide quantitative metrics for evaluating the map inconsistencies. This leads to the additional benefit that global position estimates are bounded for mosaics of bounded size; this guarantee cannot be made for dead-reckoned navigation.

The approach to visual map-based sensing is based on the mosaic re-alignment method that takes advantage of loops in the mosaic to minimize alignment errors; this core concept is explained in Section 2.4.1. The generalization of this idea to mosaics of arbitrary size and shape has been formalized into three stages for estimating image and vehicle states (Section 2.4.2). The three stages form the components of the vision sensor architecture. For complete technical details on each of the estimation stages, refer to Chapters 5–7.

2.4.1 Mosaic Re-Alignment Method

A technique must be devised to solve the error propagation problem depicted in Figure 1.5. Although this problem was originally encountered in dead reckoning, the map creation process is susceptible to this type of error propagation along the chain of images comprising the mosaic.

To reduce these errors, an external measurement of global position is needed, in order to reset the integration error. Accomplishing this reset can be considered to be a sensor fusion

problem for two dissimilar measurements, with two important distinctions. First of all, it is not sufficient simply to update the current global image and vehicle states and continue the mosaicking process. Because the dynamic mosaic is used as a correlation reference map for vehicle navigation, this improvement must be propagated back through the image chain, to improve the internal consistency of the mosaic. This will also have the beneficial side effect of improving the visual quality of the mosaic. Second, it is possible to obtain an external measurement of global position whenever the mosaic crosses back upon itself, by correlating the two images at the crossover point (Figure 2.3). Thus, the entire process can be completely encapsulated within the vision sensor component.

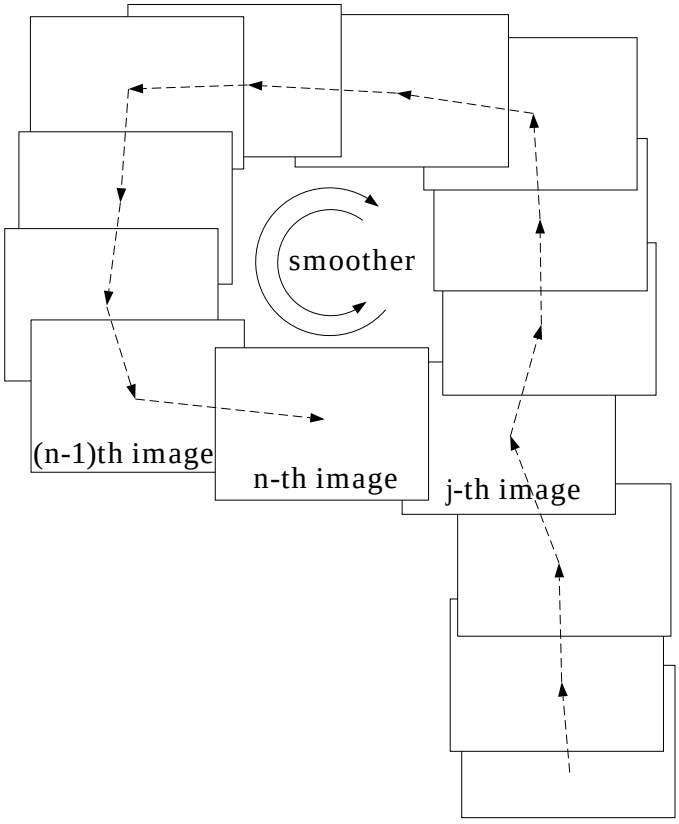

Figure 2.3: **Error Reduction in Image Chain**

By re-aligning the overlapping images when the image chain loops back upon itself and propagating the re-alignment around the loop, the errors in absolute image alignment are reduced. This improves the internal consistency of the mosaic map.

The key sources of information in this technique are the crossover points in the image chain, where the mosaic loops back upon itself. By correlating the nth image with the jth image as well as the $(n-1)th$ image (Figure 2.3), another measurement of the nth image global state is gained. Furthermore, this new measurement is more accurate, since the jth image occurs earlier in the image chain and thus its global state measurement has a smaller error. By isolating the measurements along the loop between image j and image n, we can propagate this additional information by applying a smoother to these measurements.

This concept is the basis for the estimation method outlined in Section 2.4.2. Furthermore, in subsequent chapters, this concept will be extended to take advantage of multiple loops in the image chain and the availability of a vehicle dynamic model.

2.4.2 Estimation Stages

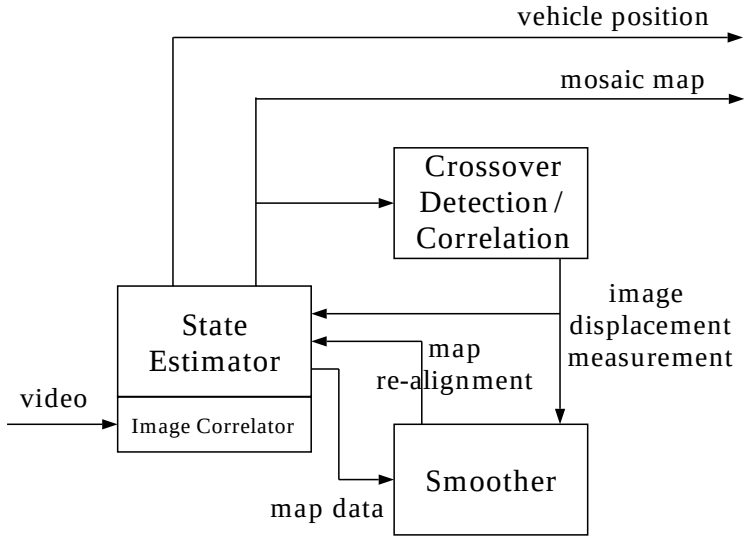

Figure 2.4: **Vision Sensor Block Diagram**

The vision sensor component in Figure 2.1 is responsible for creating a vision-based navigational map, maintaining the internal consistency of the map by implementing the crossover-based re-alignment method, and determining vehicle location within the map. To

accomplish these interrelated tasks, the vision sensor component consists of three stages (Figure 2.4):

- State Estimator
- Crossover Detection and Correlation
- Smoother

This section will explain briefly the function of the three stages and how data are passed among them. Each of these stages provides an incremental improvement in the accuracy of the mosaicking/estimation system. Figure 2.5 illustrates the predicted error bounds on the vehicle state estimates, after each of the stages has been applied. This figure is provided for conceptual purposes only; actual experimental data corresponding to these plots are shown in Chapters 5–7. These chapters also provide the technical details for each of the three estimation stages.

State Estimator

In order to estimate image and vehicle states, the transformation from local vision measurements to global states must be fully modeled. These transformations are used to align images within the mosaic and to localize the vehicle within the mosaic reference map. This primary stage provides both state estimates and the variances of these estimates for optimization by subsequent stages.

Crossover Detection and Correlation

By detecting when the mosaic crosses itself, and correlating the two images at each crossover point, the relative image displacement measurement can be used to derive an improved estimate of both image and vehicle current global position, thereby improving future position estimates. The crossover stage performs the first step in re-aligning the mosaic map to resolve measurement conflicts.

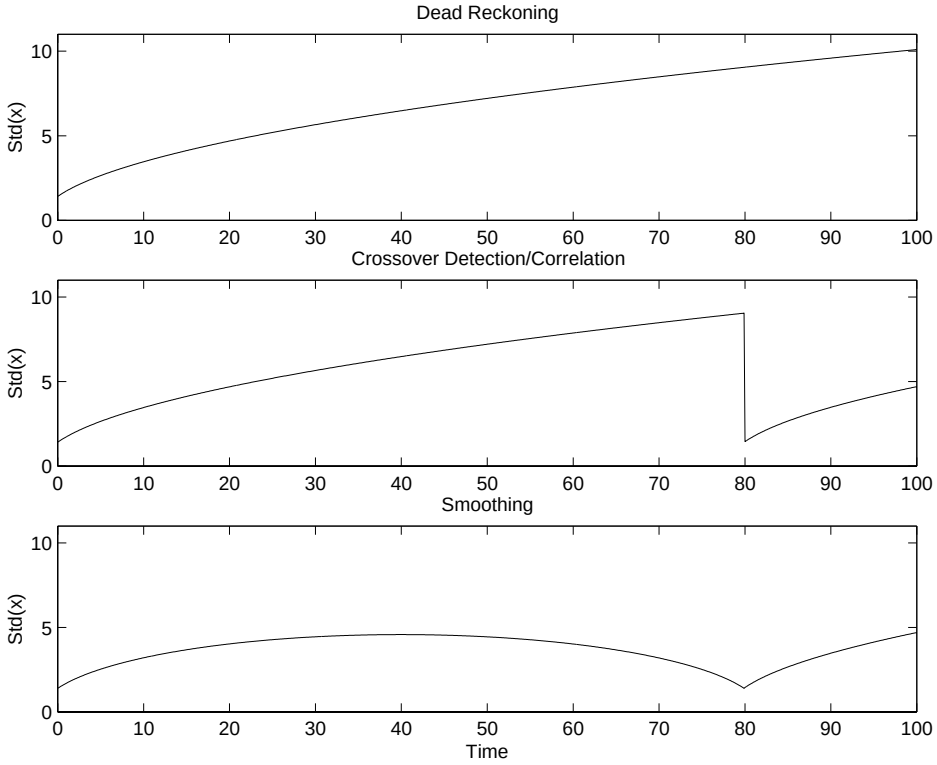

Figure 2.5: **Stages of Position Estimation**

These plots indicate the hypothetical error bounds for a single vehicle state component x, as the vehicle travels along a path containing a single loop. Alternatively, the plots could be interpreted as the absolute errors in image x position along the image chain, demonstrating the improvement in mosaic self-consistency. The horizontal axis is in arbitrary units of time, and the vertical axis is in arbitrary units of distance. The path crosses back upon itself at time 80, and each successive figure shows the affect of applying another estimation stage.

Smoother

By utilizing the image displacement measurements from the crossover stage, the mosaic can be re-aligned to minimize the measurement conflicts at every crossover point in the mosaic. To achieve this re-alignment, the smoothing stage performs a global optimization to minimize the errors in image placement within the mosaic. This refinement in global map accuracy results in better internal consistency of the mosaic map.

2.5 Contributions

In Section 1.3, the overall objectives of this research were discussed. In order to achieve these objectives, several technical innovations were required. These innovations form the fundamental contributions of this research. Specifically, the contributions detailed in this dissertation focus on the development of a complete real-time vision sensor system that performs the following novel functions:

- Estimation of the size of mis-alignment errors within a mosaic and the size of errors on vehicle state (position and orientation)
- Re-alignment of overlapping images to correct the mosaic
- Optimization of the location of all other images in the mosaic in order to accommodate this re-alignment and improve future vehicle state estimates

Chapters 5–7 have been dedicated to describing in technical detail the development behind each of these contributions.

2.6 Summary

In this chapter, the challenge of creating a real-time, bounded-error, visual map-based navigation system for underwater vehicles was presented. The issues relating to the major component of the system, namely, the map-based vision sensor, were presented in terms of three tasks: map creation, map expansion and re-alignment, and vehicle localization. A solution approach was designed to accomplish these tasks in three stages: the state estimator, crossover detection and correlation, and smoother. Technical details for these three stages, and the experiments that demonstrate their efficacy, will follow in subsequent chapters.

Chapter 3

Vision Sensing

This chapter provides a background for the low-level vision sensing techniques used in this research. Specifically, it justifies the use of the texture-based image correlation used in this work, and it explains the merits and drawbacks of other possible techniques. Although the contributions of this research are not in the area of low-level image processing, they are built upon the foundations explored in this chapter. Thus, a solid understanding of the computer vision algorithms may be useful to the reader.

3.1 Introduction

Before delving into the details of the mosaic creation and re-alignment methods in Chapters 5–7, a deeper understanding of the texture-based image correlation and mosaicking process is helpful. In order to define the scope of this chapter, Section 3.2 states that the general problem of computer vision as applied to UUV navigation is the recovery of a world model from images of the underwater scene. The mosaic serves as the world model, and it is created through the extraction of geometric information from image pairs, followed by the calculation of scene-relative state.

In order to choose among the various image correspondence methods and transformation models, the assumptions and constraints of the AUV navigation task must be considered. These restrictions are discussed in detail in Section 3.3, and based on these restrictions, a

texture-based image correspondence approach using a 2-D translation-only model is chosen. Section 3.4 explains each stage of this solution approach in some detail, in order to provide an in-depth background on the low-level computer vision technology utilized in this work.

For interested readers, Sections 3.5 and 3.6 present various alternative approaches that were considered for solving the problems of geometric information extraction and scene state calculation. In addition to describing each solution method, the relative merits and drawbacks of each method are discussed.

3.2 Problem

The goal of computer vision is "the automatic deduction of the structure and properties of a possibly dynamic three-dimensional world from either a single or multiple two-dimensional images of the world."[20, p. 4] This grand effort can be narrowed significantly when applied to the specific case of robot navigation. The scene, while unstructured, is considered static, and only the geometric properties, in contrast to material or lighting properties, of the scene are desired. Further assumptions will be made in Section 3.3.

Specifically, the objective of vision sensing for robot navigation is the recovery of camera motion and scene geometry from a set of sequential monocular images. This objective can be subdivided into two stages, namely, geometric image information extraction and scene-relative state calculation [16]. During the geometric image information phase, the relative geometry between a pair of images is determined by matching corresponding points in each image and calculating the parameters of an assumed geometric model. The scene-relative state calculation phase takes this relative geometry information and calculates both the image locations and the camera location within the scene.

3.3 Assumptions and Constraints

In deciding on the best approach for determining camera motion and scene geometry for real-time vision-based navigation of underwater vehicles, it is necessary to discriminate

among several options based on how well they perform under the particular constraints of this problem. For image correspondence, the specific nature of the scene determines which method is most applicable for finding correspondence points. To extract the desired geometric information, a simplified transformation model can be used if certain assumptions can be made about the scene geometry and camera motion.

In order to constrain the problem and enable computationally efficient methods for vision sensing, the following assumptions have been made, based on the scene properties and the capabilities of underwater vehicles:

- The region of operation is the near-bottom ocean floor environment. The underwater environment has several rather unique properties, and the next section will explain how these properties determine the proper image correspondence scheme to use.

- The scene is mostly static, and it consists entirely of an approximately 2-D planar surface within 3-D space. This assumption precludes the existence of large moving objects or a non-stationary background, although motion of very small objects relative to the field of view generally are ignored by the vision sensor. Furthermore, it reduces the required number of correspondence-pairs needed to solve for the transformation model parameters, since the computations can take advantage of the fact that all scene points are co-planar. The effect on the image registration of small 3-D terrain variations around the nominal 2-D plane will be discussed in the next section.

- Sequential images from a single camera are utilized for processing. This choice constrains the possible images sources and resultant geometric information that can be extracted. In other words, stereo vision techniques are not used as part of this research, so only optical flow or optical displacement information may be determined.

- Large motions of the underwater vehicle are only permitted in the two translational degrees of freedom corresponding to a single plane parallel to the terrain. This assumption is justified for any vehicle using an active control system to maintain its position and orientation. The image correlation assumes that rotations and range

changes around the nominal operating point are approximately zero. The effect of small rotations and range changes on the image registration will be discussed in the next section.

- The vision sensor is required to perform in real-time, on hardware with limited computational power.[1] As a result, computational efficiency is an important factor in determining which methods to use for image registration.

3.4 Solution

After considering the constraints particular to the problem of underwater vehicle navigation along ocean floor terrain, a set of methods has been chosen to handle the process of geometric image information extraction. The details of the texture-based image registration method using a translational transformation model are described in this section. In addition, an efficient pipeline-based implementation to perform these computations on every sampled image will be described. Finally, the process by which a mosaic is created in real-time using these methods will be explained in detail, since this provides the basis for our advances in mapping and state estimation.

3.4.1 Sub-Image Texture-Based Registration

In order to maximize the robustness of the measurements under arbitrary scene conditions, a texture-based registration method is utilized. Furthermore, in order to minimize computation, subsections of each image-pair are compared. The details of this registration method are presented in this section.

[1] As discussed in detail in Chapter 4, the computational engine used for this research was a dual-processor Pentium 133-Mhz system. Upgrades to this hardware would allow more complex algorithms to be utilized, thereby increasing the measurement accuracies and/or robustness.

Correspondence

In the texture-based correspondence method, the images are first convolved with a signum of Laplacian-of-Gaussian (SLoG) filter. The Laplacian-of-Gaussian (LoG) operator, also known as the Marr-Hildreth operator, recognizes rapid intensity variations and was originally used as part of filtering schemes for edge detection [18]. In conjunction with the signum operator, it has several unique properties that make it ideal for use in the underwater environment.

The Gaussian filter replaces each pixel in an image with a weighted average of it and its surrounding pixels. Convolution with the Gaussian kernel acts as a low-pass filter to smooth the images, thus reducing the effect of noise on the image. This is particularly useful for ocean floor imagery, since small particulate matter in the water, known as marine snow, often adds a significant noise component to each image.

The next phase is the Laplacian operator, which performs a spatial second derivative in two dimensions. It acts as a high-pass filter and has the effect of separating the image into regions of similar texture. When taken together, the LoG acts as a band-pass filter to reject image noise. The band frequency can be moved by adjusting the standard deviation parameter, σ, of the Gaussian filter.

The final stage of the filter is a signum function that thresholds the intensity values. Thus, it transforms the image from grayscale to black-and-white, greatly reducing the amount of information contained within the image. Furthermore, by thresholding the intensity, the image correspondence becomes largely insensitive to lighting variations, such as spotlight effects or shadows. These lighting variations are quite common underwater, since lighting must be provided artificially by spotlights on-board the vehicle.

Once each image has been filtered, the two images are correlated to establish a correspondence. Since the output of the SLoG filter contains binary pixel values, cross correlations (Equation 3.2) become sign correlations (Equation 3.3), significantly improving the computational efficiency of the image correspondence. To reduce the required computation further, the correlation stage does not compare the entire two images. Instead, a correlation window

is chosen in one image, and a search region is chosen in the second image. The correlation window is located at the center of the live image, and the search region is located within the reference image (see below for an explanation of the live and reference images). The image correspondence algorithm performs the sign correlation for every possible location of the correlation window within the search region. This produces a correlation surface, where every point on the surface corresponds to the sign correlation value at a particular x, y location of the correlation window within the search region. The highest peak on this surface is chosen as the best match location, and the x, y location of this peak represents the relative image motion.

Transformation Model

Based on the fact that the robot is actively controlled to remain within a plane parallel to the image scene, a 2-DOF translational transformation model is used to extract the relative geometry from the image correspondence measurements. Thus, the x, y pixel displacement measurements are converted simply to meters, based on the camera fields of view and the range.

Since the robot controller is not perfect and the ocean floor not perfectly flat, the rotation and range change of the vehicle will not be identically zero. Thus, the assumptions of the translational transformation model are violated routinely in practice, so it is important to understand the effects of small rotations or range changes on the image correspondence.

For a non-zero yaw, range change, or 3-D terrain variation away from the nominal, the correspondence location is shifted and the measurement confidence degrades. However, the shift in location can be removed if the correlation window in the live image is taken to be at the center of the image. Even if there are yaw and range changes in the presence of image translation, the correspondence of the center of the live image with the reference image will yield an accurate measurement, since rotation and scaling of an image shift every point in the image except the center.

The effect of non-zero roll or pitch can be handled differently. Since roll and pitch are equivalent to x,y translations to first order, they offset the correspondence location without

degrading the measurement confidence. This offset can be taken into account by measuring roll and pitch with an external sensor (e.g. inclinometer) and backing out the actual x,y translations when solving for camera position.

3.4.2 Image Processing Pipeline

For the purpose of vehicle navigation, the goal of this vision sensor is to measure image motion while minimizing measurement drift. Therefore, an optical displacement method will be used, which dictates the two image sources to be the live image and a previously stored reference image. To be able to compare non-adjacent images in the mosaic, it is also required that any image stored in the mosaic may be used as the new reference image for future computations.

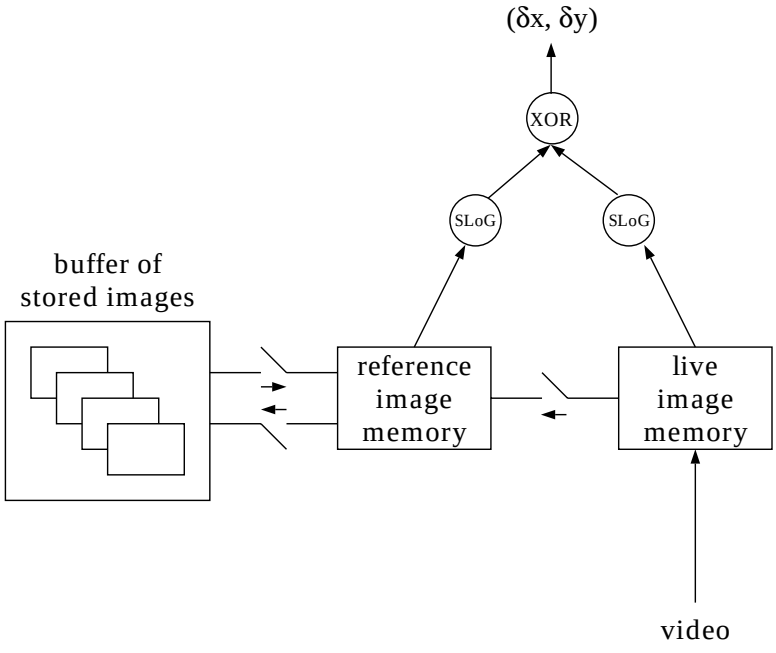

Figure 3.1: **Image Processing Pipeline**

To satisfy these constraints while performing the image registration computations efficiently, an image processing pipeline has been created, as depicted graphically in Figure 3.1.

To start a cycle, the camera video is digitized and fed into the live image memory. The image registration is then performed on the live and reference images, and the extracted displacement sent to the next stage of the vision sensor. This entire cycle is performed at the frame rate of the digitizer board, subject to computational constraints. For this research, the digitizer frame rate is 30 Hz, and the computational hardware allows the image processing pipeline to run at 10–30 Hz.

At any arbitrary time determined by the mosaicking process, a *snapshot* can be taken. First, the live image is copied into the reference image memory. As soon as this transfer occurs, this same image (now the new reference image) is copied into one of the empty slots in the buffer of stored images. Simultaneously, the image is added to the evolving mosaic by copying it over to mosaic storage. If a loop in the vehicle path occurs, any image from the buffer may be transferred back into the reference image memory and compared to the live image.

3.4.3 Mosaicking Process

Once the image processing pipeline has been established, the mosaicking process is relatively straightforward (Figure 3.2). Whenever a new reference image is snapped, it is added to the evolving mosaic. By using the last registration measurement, which compared the new reference (then live) image to the old reference image, the new snapshot can be precisely aligned in the mosaic. A new snapshot is taken whenever the overlap between the live image and reference image reaches a pre-specified minimum area. This ensures that there will always be sufficient overlap for image correspondence, and it produces a mosaic whose images are taken at regular spatial intervals. On the occasion that a correspondence measurement is deemed invalid because it falls below a given confidence threshold, a new snapshot is taken and the last valid measurement is used for alignment.

The advantage of this mosaicking process is that it enables dynamic mapping of the environment. New snapshots are added to the mosaic as they are received, thus enabling the mosaic to grow over time as more terrain is explored. Also, it is possible to incorporate redundant measurements to improve the map accuracy. If new alignment information between

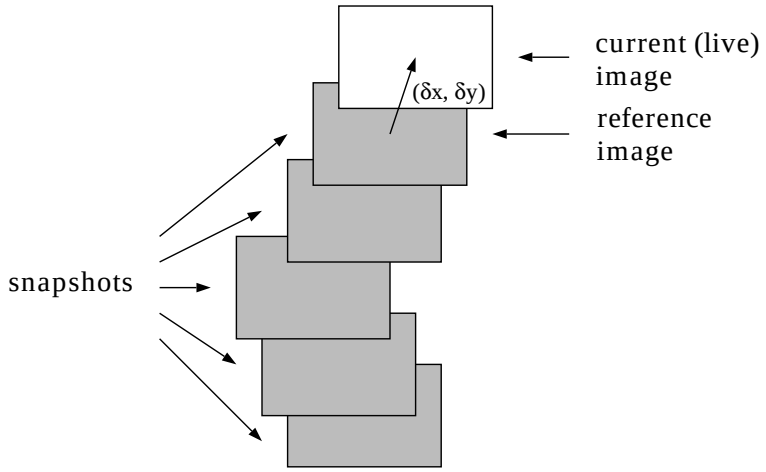

Figure 3.2: **Mosaicking Process**

any two images in the mosaic is received, the images can easily be shifted to accommodate the change.

3.5 Options for Geometric Image Information Extraction

The various methods described in this section all have a common purpose: to extract geometric information from the raw image data. Specifically, images are compared in pairs, and an attempt is made to register the two images such that points corresponding to the same physical point in 3-D space are aligned. This technique is known as image registration, and it is discussed in Section 3.5.1. The choice of images makes a significant difference in what type of geometric information is extracted; various image pair possibilities are presented in Section 3.5.2.

3.5.1 Image Registration

As stated above, image registration involves the alignment of a pair of images. In order to accomplish this, points in one image must be found to correspond with points in the other image. Finding these point correspondence pairs automatically is not a simple task. The

geometric or intensity properties of corresponding points may be different in each image, or a corresponding point may be absent altogether. This could be due to image noise, or it could result from the fact that 3-D objects look different when viewed from different perspectives, or it may be caused by occlusion. Several methods have been presented in the literature to try to determine the correspondence pairs in a robust fashion.

In order to make use of these correspondence pairs, a geometric transformation model must be established between the two images. In other words, given constraints on camera motion, scene geometry, and required accuracy, a simplified model can be assumed for the geometric transformation between the two images. The parameters of this model can be calculated using one or more point correspondence pairs. Due to the imperfect nature of the image point correspondence methods, a best-fit method can be devised to obtain the best estimate of the model parameters.

Correspondence Establishment

To solve the correspondence problem, the first task is to determine what information will be the basis of determination for matching. Several options have been used with much success; each one tends to be well-suited for some applications in terms of both accuracy and robustness, while it displays poor performance on other applications.

Feature-Based Correspondence As the name implies, this method matches recognizable features in both images. An example of this is edge-based correspondence. Standard edge detection algorithms exist, such as the methods developed by Canny [3] and Marr and Hildreth [18], that filter the images to produce robust results, so the challenge is then to discern which one of many (if any) edges in the second image match a given edge in the first image. Similarly, corner-based correspondence methods utilize edge intersection points to register images [30]. Feature-based correspondence works especially well in man-made environments where real features, such as edges, are clearly defined. However, natural scenes often have few if any readily distinguishable features; the locations of the weak features that are found tend to be noisy and ill-defined, and they are much more difficult to match

with feature locations in the other image. Computationally, efficient methods do exist for this correspondence technique.

Intensity-Based Correspondence This is perhaps the most straightforward class of techniques for establishing correspondences. In this method, a region in the first image is compared to a region of the same size in the second image, by comparing the individual intensities of corresponding pixels and summing over the entire region. Since the intensity values are integers with a finite range of values, the exact method of pixel comparison varies.

Two of the most popular techniques are the *sum of squared differences* and the *cross-correlation.*[20, p. 227] The sum of squared differences is calculated as follows:

$$\mathrm{SSD}(\Delta x, \Delta y) \equiv \sum_{i=1}^{m} \sum_{j=1}^{n} [I_0(i,j) - I_1(i - \Delta x, j - \Delta y)]^2 \tag{3.1}$$

where $I_0(i,j)$ and $I_1(i,j)$ represent the two images, the comparison region size is m x n, and $(\Delta x, \Delta y)$ is the disparity between the locations of the two matched regions. The cross-correlation technique uses a slightly different method for intensity comparison:

$$\mathrm{CC}(\Delta x, \Delta y) \equiv \sum_{i=1}^{m} \sum_{j=1}^{n} I_0(i,j) I_1(i - \Delta x, j - \Delta y) \tag{3.2}$$

These techniques are useful for a wider class of input images than the feature-based techniques, since they do not take advantage of any underlying image structure. However, since these techniques rely on intensity comparisons, they are sensitive to lighting variations or noise on the intensity values. Furthermore, each intensity comparison is an integer multiplication, so the required computation increases rapidly for large regions.

Texture-Based Correspondence This method combines elements of both the feature-based and intensity-based techniques. Originally used for stereo image registration, it was first suggested by Nishihara [23, 24]. The correspondence computation is defined by Equation 3.2 in the same manner as for cross-correlation. However, before performing this computation, both images are filtered using a SLoG filter. As discussed earlier in this chapter,

this filter reduces the intensity value at each pixel to a binary 0 or 1. Thus, the multiplication in Equation 3.2 is reduced to a logical comparison. This technique is appropriately called a sign correlation:

$$\mathrm{SC}(\Delta x, \Delta y) \equiv \sum_{i=1}^{m} \sum_{j=1}^{n} XOR(\mathrm{sgn}[\nabla^2 G] * I_0(i,j), \mathrm{sgn}[\nabla^2 G] * I_1(i - \Delta x, j - \Delta y)) \tag{3.3}$$

For this reason, the texture-based correspondence is computationally efficient. Furthermore, due to the effects of the SLoG filter, the correspondence results are robust and largely insensitive to lighting variations.

Correspondence-Pair Transformation Models

Once one or more correspondence-pairs have been established, the relative geometry between the two images, or more generally, among a set of images, must be determined. In the most general case of arbitrary 3-D scenes and 6-DOF camera motions, the transformation results in highly complex, nonlinear equations that are difficult to solve. By taking advantage of constraints or imposing restrictions on the scene geometry or camera motion, simplified transformation models can be used. These simplified models, while potentially less accurate in capturing the true nature of the system, are much more straightforward computationally. For the case of robot navigation, a detailed 3-D model of the scene is not required, so linear models are quite advantageous in satisfying real-time requirements.

The options below are not meant to be an exhaustive list; recent work has produced exciting results using variants and extensions of these transformations. In particular, recent advances in technology for the creation of panoramic mosaics have included the use of manifold projections [26], projective transformations [31], and a combination of local and global alignment techniques [28]. These novel mosaicking techniques are excellent candidates for extending current underwater mosaicking, provided real-time constraints can be met eventually.

Perspective Transformation This method uses a perspective projection [20, pp. 33–38] to model the geometry of a pinhole camera, as illustrated in Figure 3.3. While this is a very general camera model,[2] the equations for relating image points to 3-D scene locations are nonlinear. Furthermore, due to the large number of unknown parameters, many correspondence-pairs are needed to produce a set of equations that are equal in number to the number of unknowns. Thus, the computations required to produce multiple correspondence-pairs and solve a nonlinear set of equations often preclude the use of this method in real-time applications, such as robot navigation. A recent paper by Bell [1] provides an excellent explanation of image correlation techniques using perspective projection.

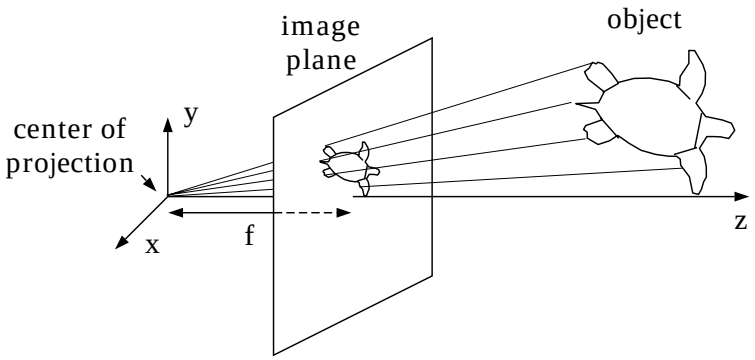

Figure 3.3: **Perspective Projection**

In this type of projection, f is the focal length of the camera, and the image size depends on the range from the center of projection to the object.

Affine Transformation This model assumes that the 3-D scene relief is small compared to the distance from the camera to the target. Also, it assumes the correspondence points in the image are located sufficiently close to the optical axis of the camera (usually the center of the image); this is essentially a restriction on the size of the camera field of view. Under these assumptions, the orthographic projection [20, pp. 38–39] is a valid linear approximation to the perspective projection (Figure 3.4). An affine transformation

[2]More general models exist that take into account the effect of lenses in image formation. Often, these nonlinear models for lens aberrations and the resulting image distortion are included as part of a separate camera calibration step. [20, pp. 41–48]

between a pair of images maps points in one image to corresponding points in the other image. Under the assumption of orthographic projection, the affine transformation is able to perform an identical linear transformation for every point-to-point mapping. The affine transformation is fully defined by 6 parameters. These parameters can be obtained through linear estimation. The affine model only determines scene locations up to a scale factor. In other words, scene locations are only known in image units (e.g. pixels); the conversion from image units to scene units (e.g. meters) is unknown. If needed, this scale factor usually is determined rather easily from alternate sensors and/or camera calibration. Early work in affine-based image registration by Irani-Peleg [9] and scene structure/camera motion recovery by Tomasi-Kanade [33] have demonstrated the utility of this type of method. Given the current advances in computational power, the affine transformation method has become increasingly attractive for the robust estimation of 6-DOF camera motions and 3-D scene geometry in real-time.

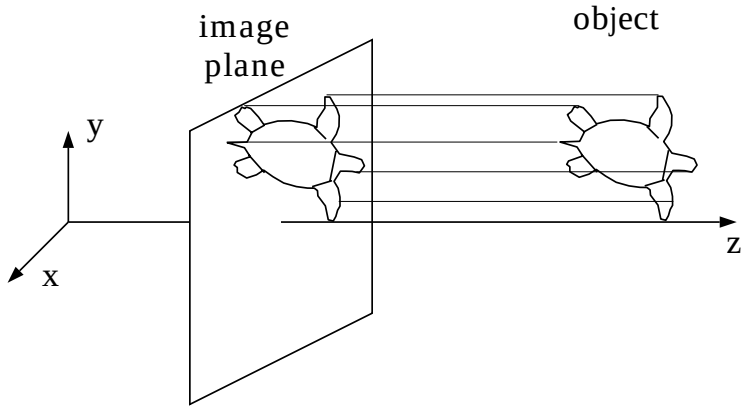

Figure 3.4: **Orthographic Projection**

In this type of projection, the image size is independent of the range to the object.

Translational Transformation To simplify the task of estimating camera motion further, the translational transformation method has been developed. In addition to utilizing an orthographic projection, this method assumes the entire scene consists of a plane perpendicular to the optical axis of the camera. In addition, it assumes the camera motion is

restricted to translations within a plane parallel to the scene; no rotation or range variation is allowed. Finally, it is assumed that the range from the camera to the scene is known, since this is constant under the given constraints and unobservable from any possible pair of images. Thus, the only goal of this registration is to determine the planar translations of the camera, since the rest of the geometry is already known. As with the affine model, the planar translations are only determined up to a scale factor. Again, this scale factor is usually determined rather easily from alternate sensors and/or camera calibration. Since only two parameters (i.e. δx, δy camera translations) fully define this translational transformation, one correspondence-pair provides the two equations needed to solve for the unknowns. More correspondence-pairs could be used to form a more robust least-squares estimate of the two parameters, at the cost of increased computational effort. This model has been designed with the specific case of real-time robot navigation in mind; it takes advantage of the inherent system constraints to determine only the required quantities, expending as little computation time as possible.

3.5.2 Image Sources

Once a pair of images has been registered and the relevant parameters computed, the resulting geometric information can be used to determine the camera state relative to the scene. However, the exact nature of the information extracted in the image registration depends on the sources of the two images used in the computation. The methods for extracting geometric information can be grouped into three classes:

Stereo In a stereo vision setup, two synchronized cameras are pointed towards the same scene at all times. Thus, each camera provides an input image to the registration phase. Since the two images are collected at precisely the same time, one can assume there is no intervening camera motion. By measuring the disparity between corresponding points in the two images, the 3-D location of the scene at that point (relative to the current camera location) can be calculated.

Optical Flow The goal of this method is to measure image motion between subsequent images from a single camera. The two image inputs are from the same camera, but they are separated in time by the digitization sample time.[3] During this time, the camera may move a small amount, and this change is measured through the image registration phase. This computation is performed at every sample time to compute the image motion. In essence, this measurement is the discrete equivalent of an image velocity. However, if the ultimate goal is to measure image displacements, integrating optical flow will result in measurement drift. That is, in order to calculate image motion, the optical flow measurements are summed. Therefore, errors on the optical flow measurements will continue to accumulate over time. Even if the measurement errors are zero-mean, the error variance will increase in random-walk fashion. This is completely analogous to the dead-reckoned error one encounters when integrating wheel rotations or inertial measurements on land-based mobile robots. For the case of optical flow, this measurement drift is a function of time.

Optical Displacement This method also strives to measure the image motions of a single camera. However, it uses different image sources to alleviate the measurement drift problem described above. Instead of comparing the latest two images from the camera, a single camera image is stored and then compared to the current image at every sample time. As a result, the image displacement is measured directly, so measurement errors do not accumulate over time. This setup has the drawback that the displacement magnitudes are limited by the field of view of the camera. In other words, the two images must overlap in order to make this measurement. To overcome this problem, a new image can be snapped and stored as the reference image whenever the image displacement approaches the edges of the camera field of view. Since the new reference image has already been aligned with the original reference image, these two images form a composite image, or mosaic, of the scene. This improved

[3]Depending on the application, the sample rate varies by orders of magnitude. For the experimental hardware presented in this thesis, the sample rate is between 10 Hz and 30 Hz, which is equivalent to a sample time on the order of 0.1 seconds.

method determines image motion by summing the image displacements between previously stored images in the mosaic. Thus, while the measurements are not summed at the digitization sample rate, they are still summed for large image motions. For optical displacement, the error drift is a function of distance traveled. While this is an improvement over optical flow, the reduction of this error drift is a fundamental contribution of the research discussed in this dissertation.

3.6 Scene-relative State Calculation

After choosing the proper image sources, computing the image registration, and extracting the desired geometric information, the next step is to utilize this information to determine the camera state relative to the static scene. The approach to this problem depends on several factors: the specific image correspondence and registration methods, the exact nature of the geometric image information, the sensor configuration, and the relevant physical and mathematical assumptions. As such, it is impossible to derive a general method for solving this problem. The development of scene-relative state equations for a specific real-time robot navigation application is one of the primary technical objectives of this thesis, and it is accomplished within the state estimator stage of the vision sensor (Chapter 5).

3.7 Summary

In order to understand fully the contributions made by this research, an adequate background in computer vision is useful. This chapter provides the background, decomposing the problem of determining scene geometry and camera motion from vision into several distinct stages. After presenting the properties of the specific application of underwater vehicle navigation, particular solution methods were chosen for each of these stages and explained in detail. These low-level computer vision methods are the foundation upon which the contributions of this research are built, and their intrinsic properties affect the development of higher level innovations for robot navigation. A variety of other possible methods were

cited to compute the desired information, and the benefits and drawbacks of each method were described.

Chapter 4

Experimental Systems

This chapter provides technical descriptions for each of the experimental systems utilized during the course of this research.

4.1 Introduction

Before proceeding to the theoretical development, implementation, and verification of the vision sensing system, and the demonstration of UUV navigation using this novel vision sensor, the experimental systems that serve as testbed platforms for this work are presented. The core system utilized for all experiments is the navigation software and host hardware, as described in Section 4.2. The next three sections, Sections 4.3–4.5, are devoted to providing technical specifications for each of the three robotic platforms used for testing: the Space Frame, the OTTER AUV, and the Ventana ROV.

4.2 Navigation Software

The navigation software is a hierarchical implementation of the algorithms and functionality required to perform the tasks of vision sensing and robot navigation. It is designed to be a highly flexible and re-configurable component that can be integrated into several different types of hardware platforms. To enforce both the external interfaces to hardware

and internal interfaces among sub-components, and to enable simultaneous execution of multiple functional blocks, this software was written as an object-oriented, multi-threaded application.

Specifically, the code was written in Microsoft Visual C++ 6.0 using the Microsoft Foundation Classes (MFC) library, under the Windows NT 4.0 operating system. The host hardware for this sensing and navigation application is a dual Pentium PC, running at 133 MHz. Live video from a camera input is captured using a Matrox Meteor digitizer board, at frame rates of up to 30 Hz and 24-bit color image resolutions of up to 512 x 480 pixels. In addition, the PC has ethernet and serial communication ports to exchange data with other computers. The video input and bi-directional network ports are the only connections to external hardware.

The software hierarchy is divided into two levels. The lower level is responsible for creating and executing the image processing pipeline to perform real-time image correlations. These local image displacement measurements are then passed to the higher level of the hierarchy. The role of the higher level is to perform the simultaneous tasks of mapping, vehicle state estimation, and navigation. The following sections describe the implementation of each of these levels in the hierarchy.

4.2.1 Advanced Vision Processor (AVP) Library

The lower level of the software hierarchy is implemented as a software library known as AVP. The AVP library was written by Rick Marks while an engineer at Teleos Research. While AVP can perform many functions, including object tracking and stereo ranging, its role within the navigation software is to provide the image registration capabilities described in Sections 3.4.1–3.4.2. Thus, AVP creates an image processing pipeline that is capable of correlating the live camera image with a stored reference image. In addition, the reference image can be stored in a buffer for later retrieval and comparison. Essentially, AVP is a software implementation of the work originally performed by Marks on specialized hardware

for his thesis research [16]. To reduce the computational requirements and satisfy the real-time constraints of the vision sensor, the maximum resolution of the digitizer board is not utilized: the AVP input images are 8-bit grayscale, with a resolution of 256 x 240 pixels.

4.2.2 Sensor 0.7 Application

The higher level of the hierarchy takes the form of a multi-threaded application called Sensor (the latest version is 0.7).[1] Each thread in the application performs a distinct, well-defined task that can execute at a sample rate independent of the other threads. Thread synchronization and data exchange are performed through shared memory guarded by mutual exclusion semaphores, remote procedure calls, and message-passing. Figure 4.1 graphically depicts all threads in the Sensor 0.7 application and the interactions among them, and the following sections explain the role of each thread.

AVP Engine Thread

As seen in Figure 4.1, the AVP Engine Thread is the central thread in the application. This computation engine interfaces directly with the AVP library through function calls to obtain image registration measurements, and it communicates with other threads to receive external updates from sensors on-board the vehicle. It performs real-time calculations at speeds of 10–30 Hz, where the digitization frame rate is 30 Hz. The computations are divided into functional components that are executed in sequence during every calculation cycle. The interconnection of components is illustrated in the data flow diagram of Figure 4.2.

The AVP Engine Thread is an implementation of the vision sensing system, and it can be interfaced with other threads to create new applications. For this particular research it was combined with interface and communication threads to enable a navigation application, but

[1] This application is called Sensor because it was originally designed as the vision sensing system. Since then, the application has grown around this core functionality to include additional capabilities required for robot navigation.

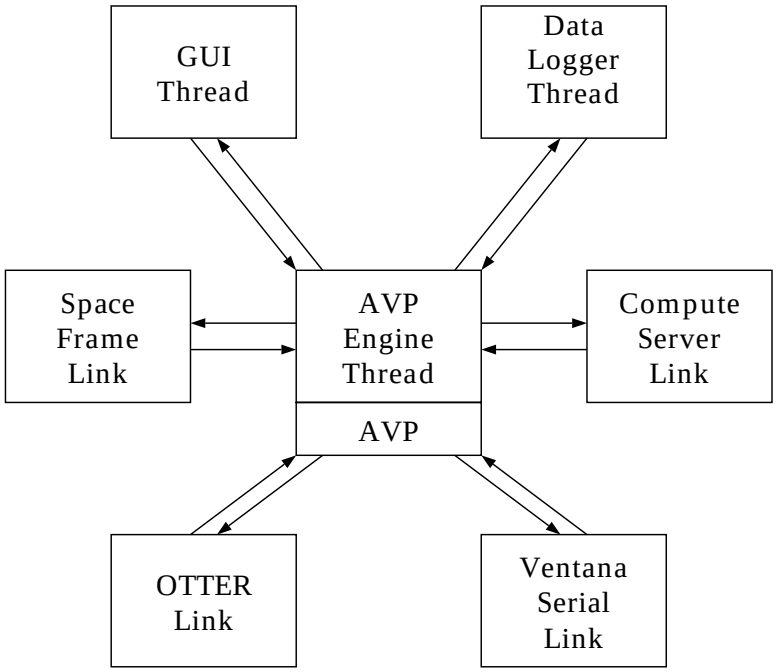

Figure 4.1: **Thread Diagram for Sensor 0.7 Application**

it is an independent entity whose utility is not limited to AUV navigation. Additional components were implemented within this thread to perform navigation functions in addition to vision sensing, as shown in the block diagram of Figure 4.2.

GUI Thread

The GUI Thread provides an image-based interface for the purpose of vehicle navigation. Specifically, it presents the dynamic mosaic to the user in a scrollable window, with an 'x' overlay to indicate the estimated current vehicle position within the mosaic, and an 'o' overlay to indicate the goal position. The user is able to point-and-click at a new location within (or outside of) the mosaic to specify a new goal location. These data are then sent to the AVP Engine Thread to control the vehicle to its new desired location.

In addition to the mosaic interface, the GUI thread provides a series of menus and dialog boxes to manage both application execution and mosaic file storage. One of these menus

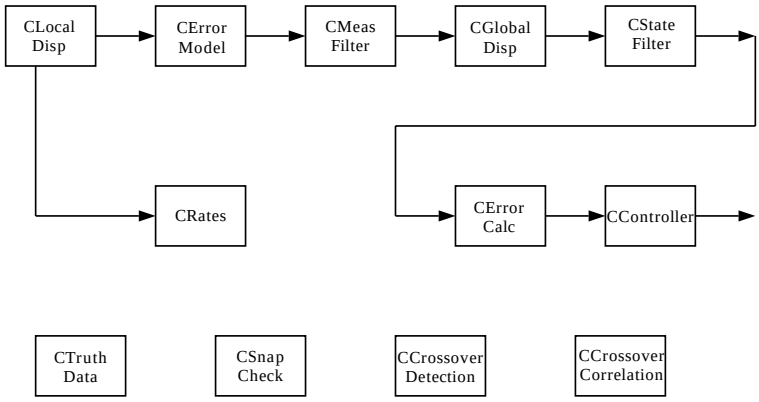

Figure 4.2: **Data Flow Diagram for AVP Engine Thread**

enables the user to switch the application among idle, passive sensing, and active navigation modes. Within each dialog box, graphical controls exist to modify relevant parameters for a specific aspect of the navigation application.

Since the GUI is not as time-critical a task as real-time vehicle sensing and control, the GUI Thread is run at a lower priority than the core AVP Engine Thread. Since each thread executes at an independent sample rate, the GUI Thread can slow down to yield computational power to more urgent tasks if the processor becomes overloaded.

Communications Link Threads

The communications link threads are a set of threads responsible for exchanging data with external hardware or software systems. For a particular experimental setup, each of these threads may be active or inactive, depending on whether a link to the given device is utilized. The roles of the various communications link threads are discussed in the following paragraphs.

ComputeServerLink This thread is enabled whenever bounded-error navigation is required. It connects via AVPNet to a MATLAB-based smoother program that performs the optimal estimation computations for mosaic re-alignment. The smoother program executes

a MATLAB engine remotely on a Solaris UNIX compute server. AVPNet is a simple library written to create a two-way point-to-point connection between two programs over ethernet using the Windows Sockets API (Applications Programming Interface).

SpaceFrameLink (FlightTableLink)[2] When experiments are performed on the Space Frame, this thread connects to a network node running on a UNIX machine via AVPNet. This network node then passes the data along to the Space Frame processor using the Network Data Delivery Service (NDDS), a low-level, high-bandwidth, peer-to-peer networking service developed by Real-Time Innovations (RTI) for real-time communications. Sensor data and truth measurements are received from the Space Frame, and desired position data are sent by the application through the SpaceFrameLink.

OtterLink For experiments on OTTER, the OtterLink connects to a network node running on a UNIX machine via AVPNet, which passes the data to OTTER's on-board processor using NDDS. Since OTTER is an AUV, an automatic control system is executed by the on-board processor. Thus, data from on-board sensors are received by the application, and both vision sensor data and desired position data are sent back to the OTTER vehicle.

VentanaSerialLink Since no ethernet connection is available to the Ventana ROV, network communication is accomplished over a serial line. The role of the VentanaSerialLink is to provide a bi-directional serial connection directly to the Ventana ship-side processor. Since Ventana is an ROV, it is not equipped with a complete automatic control system. Thus, control computations are performed within the Sensor 0.7 application. Sensor data are received from Ventana over the serial connection, and thruster commands are sent back to the vehicle.

[2]The Flight Table was a previous name for the experimental apparatus now known as the Space Frame. In the actual Sensor 0.7 code, all references are made to the Flight Table, not the Space Frame.

Data Logger Thread

The role of this thread is to record any relevant data in real-time for later analysis. During each cycle of this thread, data are accessed from AVPEngineThread and saved to disk. The Data Logger Thread has the capability to record both synchronous and asynchronous data in real-time. Since the data logging facility is an independent thread from the primary computations, it can run at a different sample rate so AVPEngineThread can maintain a constant time interval between cycles. However, if possible, these two threads run at the same rate, so every iteration of the computations is collected.

4.3 Space Frame

Figure 4.3: **Space Frame**

The Space Frame is a precision gantry platform capable of controlling its camera head in 3-DOF within its workspace.[3] The system consists of three independently controllable axes in the x, y, and z translational directions, as depicted in the photograph of Figure 4.3. Its workspace is 2.9 m long, 1.1 m wide, and 0.9 m high. Because of the extremely high accuracy in both sensing and controlling each axis, the Space Frame can be utilized as a truth sensor to test the vision sensing system independently, in the absence of any control errors. The following sections provide detailed specifications for each of the components within the Space Frame architecture.

4.3.1 Sensors

To sense the 3-D position of the camera, the motor for each axis is equipped with a high-resolution motor encoder. Each encoder can measure motor rotations with a resolution of 4096 counts per revolution. Once the encoder counts have been reset at a known "home" location (sensed by limit switches), the encoder precision results in a position sensing accuracy of less than 1 mm across the entire workspace. Each encoder is connected to the controller board through an A/D channel.

4.3.2 Actuators

The motors used to actuate the translational axes are Parker Compumotor 606 brushless motors. Each motor is capable of producing 1.57 KW of continuous power output, and this size provides more than enough control authority to position the camera head quickly and accurately. To control these motors, the controller board outputs signals on D/A channels to power amplifiers, that are directly connected to the motors.

4.3.3 Computer System

The computer system that manages the Space Frame uses a VME-based architecture: it consists of a series of hardware boards in a card cage connected via a VME backplane.

[3]The Space Frame is designed to control the camera head in all 6-DOF, but the rotational control has not yet been implemented. For this work, only the three translational axes were utilized.

The major tasks that must be performed by the system are to control the Space Frame in real-time, accept positioning commands directly from the user or a user program, and communicate with off-board programs.

Hardware

To interface with the sensors and actuators and perform the controller functions, the card cage contains a DMC-1300 board produced by Galil. This board is a combination A/D, D/A, and controller that is capable of simultaneously controlling the three translational axes of the Space Frame.

A Motorola MVME-167 processor card with a 68040 processor manages the operation of the entire computer system. Its primary role is to enable the user to direct the actions of the controller board, either interactively or through programs executed on-board the processor.

To facilitate communications with other computers, an ethernet card is installed on the VME backplane. Thus, the MVME-167 can exchange data with remote programs, such as the PC-based navigation software and the UNIX-based compute server.

Software

The MVME-167 runs the VxWorks real-time operating system. The ControlShell design software written by RTI is used to create component-based user applications. These applications are then executed on the MVME-167 in the ControlShell real-time environment that runs on top of the VxWorks operating system.

Control System

All control functions are performed on the DMC-1300 controller board. To control the Space Frame in real-time, this board implements a Proportional-Integral-Derivative (PID) controller for each axis that executes at a sample rate of 1 kHz. While high-rate PID controllers are adequate for the purposes of this research, the design architecture does provide the capability to bypass the DMC-1300 board and control the camera location with applications written in ControlShell.

Communications

Programs executed on the MVME-167 can utilize NDDS, the peer-to-peer networking service, to communicate with off-board processors over ethernet. In order to communicate with the Sensor 0.7 application, an intermediate network node was executed on a Solaris workstation to translate NDDS data into AVPNet data.

4.4 OTTER

OTTER is an AUV jointly constructed by ARL students and MBARI technicians (Figure 4.4). The vehicle is utilized exclusively in the test tank, to explore new technologies that will enhance the capabilities of marine scientists. The design strategy behind the OTTER prototype vehicle is to create a core platform that can be easily interfaced with these new technologies. Several Ph.D. research projects already have been performed on OTTER, including work in control architectures [34], underwater vision sensing [16], and underwater manipulation [19].

Figure 4.4: **OTTER AUV**

For this particular research, OTTER communicates with the PC running the navigation software, such that on-board sensor measurements are received by the PC, and control error signals are fed directly into the OTTER control system. Essentially, the GUI and vision

sensing system components of the navigation software replace existing components in the OTTER software structure.

The physical structure of OTTER consists mainly of three cylindrical pressure housings welded to a steel frame. The main pressure housing measures 1.25 m long and 36 cm in diameter, and it contains the two computer card cages, power circuitry, and several sensors. The two battery housings lie on either side of the main housing, and each one measures 1.25 m long and 12 cm in diameter. The entire structure measures 2.4 m x 0.9 m x 0.5 m, and it is enclosed in a fiberglass shell. The dry mass of the vehicle is 150 kg, and its estimated wet mass[4] is approximately 500 kg. Vehicle propulsion is accomplished through eight ducted thrusters, including two main drive thrusters at the rear of the vehicle, and six maneuvering thrusters that penetrate the fiberglass shell at various locations. To facilitate testing, OTTER is attached to the topside environment via a tether. This tether is used to provide power to the vehicle, support ethernet communications, and return live video from the on-board cameras. The communications portion of the tether is connected to a topside computer that serves as a communications relay to broadcast the data on the topside network.

Figure 4.5 shows the core structure of OTTER that enables the demonstration of vision-based navigation in the test tank. The following sections describe the various components shown in the diagram. For a complete explanation of the design and construction of the OTTER robot, refer to the dissertation research performed by Howard Wang [34].

4.4.1 Sensors

OTTER contains an extensive sensor suite for measuring the current vehicle position and orientation. The following list briefly describes the sensors relevant to the navigation task:

Cameras Two Pulnix TM-840N scientific low-light charge-coupled device (CCD) cameras are mounted on the front of the vehicle and point vertically downward towards the

[4] The wet mass of the vehicle includes the entrained water that moves along with the vehicle and effectively adds to the vehicle inertia.

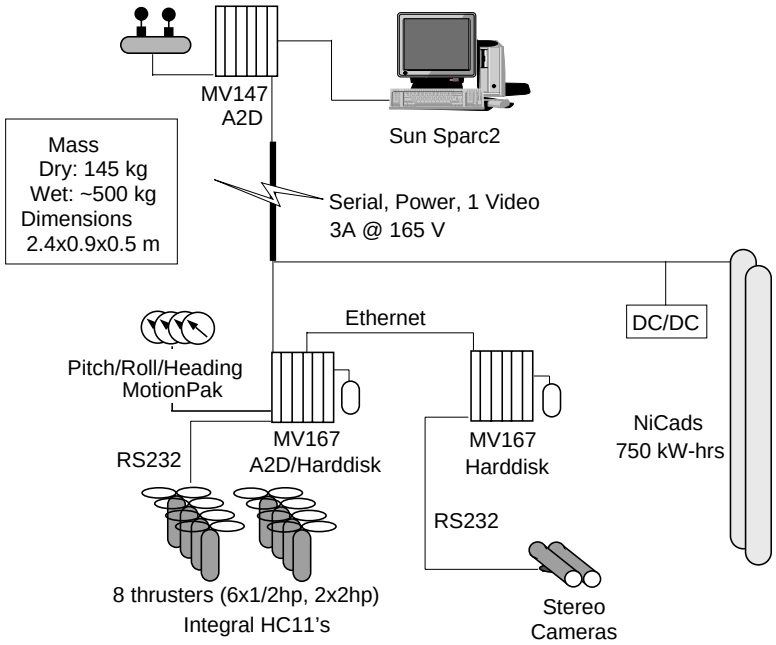

Figure 4.5: **OTTER Architecture**

test tank floor. Each camera provides interlaced video at a frame rate of 30 Hz. The images are grayscale, with a resolution of 800 x 490. The field of view (FOV) of each camera is 48° in air; since the index of refraction is different in water than air, the FOV in water is approximately 35°. While the camera pair is genlock-capable for stereo imaging, only one camera is utilized by the vision sensing system for this work.

Inclinometer This sensor is essentially a two-axis pendulum that measures the pitch and roll of the vehicle. Assuming that the vehicle is not accelerating, the hanging pendulum will always align with the direction of gravity, so the inclination of the pendulum base (i.e. the vehicle) can be measured relative to the pendulum. The measurement range of the inclinometer is approximately ±45° in both pitch and roll.

Fluxgate Compass The KVH Industries ROV 1000 fluxgate compass determines vehicle heading by measuring magnetic flux. Measurements are taken at 10 Hz, with a signal

resolution of 0.1°. While electromagnetic noise within the main housing can corrupt the signal, the heading measurement is accurate to approximately 1°.

MotionPak Inertial Measurement Unit (IMU) This sensor package from Systron-Donner contains three orthogonal servo accelerometers for measuring translational accelerations, and three orthogonal solid-state gyros for measuring angular rates. The accelerometer measurements are not used in this research, although they could be fused with the image correlator, depth sensor, and/or Super High Accuracy Ranging and Positioning System (SHARPS) measurements to improve the vehicle position estimates. The angular rate measurements are integrated and then combined with the inclinometer and compass measurements to improve the estimates of vehicle rotation.

Depth Sensor Since the hydrostatic pressure at any point is proportional to the depth below the surface of the water, the depth can be determined indirectly by measuring the pressure. Thus, the depth sensor is simply a pressure transducer that measures the hydrostatic pressure at a sample rate of 10 Hz.

SHARPS Acoustic Positioning System SHARPS, produced by Marquest Group, Inc., is a long-baseline acoustic positioning system for vehicle tracking. The system consists of a network of three acoustic transceivers that are mounted at fixed locations in the test tank. Within the volume enclosed by this network, a fourth transceiver mounted on OTTER can be tracked by measuring the ranges between it and the three fixed transceivers. Using these range measurements and the network geometry, the 3-D position of the vehicle can be estimated to within 2 cm, at a rate of 10 Hz. The SHARPS system is not used during the navigation demonstration for this work, since the image correlator measures x, y displacements, and the altitude above the test tank floor can be more accurately inferred from the depth sensor.

4.4.2 Actuators

OTTER can be actuated in all 6-DOF with its eight bi-directional propeller thrusters. Each thruster has a brushless DC, variable-reluctance motor that is governed by an HC11 microcontroller. The HC11 achieves closed-loop velocity control at a sample rate of 1 kHz. The two main drive thrusters at the rear of the vehicle are 2.0 hp motors. The six remaining thrusters are 0.5 hp motors; four of them are mounted vertically at locations fore/aft and port/starboard of the vehicle center of mass, and the other two mounted laterally fore and aft of the vehicle center.

4.4.3 Power

The vehicle is equipped with on-board NiCad rechargeable batteries that are capable of providing 750 W-hrs of energy. In practice, power is provided through the tether; the batteries are trickle-charged from the tether, and they act as buffers in the power system to enable high peak power outputs to the thrusters.

The 165 V tether power input is then passed through DC-DC converters to produce ± 5, 12, and 15 V outputs and electrically isolate the on-board electronics from the tether. A bank of power relays controls the distribution of power to the following electronic subsystems: cameras, thruster electronics, thruster power, MotionPak, battery relays, and auxiliary VME cage.

4.4.4 Computer System

The computer system for OTTER consists of two VME cages inside the main pressure housing. For this research, only the main cage is utilized. In addition to on-board computing, there is a topside real-time VME cage that acts as a communications relay, and a Sun workstation that runs the engineering interface for OTTER. The following sections provide a more detailed description of the entire computing system.

Hardware

Within the main VME cage on-board OTTER, there resides a Motorola MVME-167, a 68040-based single board computer. This processor performs all real-time computations necessary for OTTER management and control. To interface to the other on-board systems, the cage also contains a Xycom XVME-500 board that has sixteen 12-bit A/D inputs for sensor signals and eight digital I/O connections to control the power relays. An ethernet interface is also included to enable communication with topside computers over the tether.

The topside real-time computer is a Motorola MVME-147, based on the 68030 processor. This machine acts as a communications relay between OTTER and the topside computer network, and it receives input data from the SHARPS system for broadcast on both subnets.

Software

Both the on-board and topside VME processor cards run the VxWorks real-time operating system. The software applications for these systems were created using the ControlShell Computer Aided Software Environment (CASE) tool, and then executed on each of these computers within the ControlShell real-time environment running on top of VxWorks.

Control System

The control system is implemented entirely within the ControlShell application executing on-board OTTER. This application includes multiple threads of execution, running at independent sample rates. The low-level control loop executes at a sample rate of 100 Hz, while other loops run at slower sample rates.

Communications

The main VME cage is connected to the HC11 micro-controller processors on each thruster via two RS232 serial networks. Each network daisy-chains four thrusters together in a full-duplex serial loop. Communication over these networks follows a custom protocol that runs at 31.25 kBaud.

To communicate with the topside real-time computer, the main OTTER processor utilizes NDDS over the tether's ethernet line. This computer connects to the Sun workstation and navigation software PC through a separate ethernet subnetwork, and it also utilizes NDDS. To allow the PC to communicate with the other computers, the Sun workstation runs a network node that converts the NDDS protocol into AVPNet protocol.

4.5 Ventana

The Ventana ROV, as shown in Figure 4.6, was built by Internal Submarine Engineering (ISE) for MBARI in 1988. Since then, it has performed more than a thousand marine science missions in the Monterey Canyon. During the course of this research, the use of Ventana has enabled the first-ever demonstration of vision-based, bounded-error navigation on an operational vehicle in the deep ocean environment.

In order to provide the flexibility required to perform missions for scientists, researchers, and engineers from diverse fields, Ventana is capable of incorporating a wide variety of data collection equipment and science sensor packages. The vehicle usually has two manipulators mounted on the front, and its chassis is designed to support a number of toolsleds designed for specific tasks such as core drilling or biological sample collection. In its standard configuration, it has a dry weight of 5150 lbs. The relevant components of the Ventana system are discussed in subsequent sections.

4.5.1 Point Lobos Support Ship

The daily operations of Ventana are dependent upon the support ship Pt. Lobos (Figure 4.7). The deck of the Pt. Lobos is equipped with a crane for ROV deployment and retrieval, and a tether management system to control the 2100 m tether that connects Ventana to Pt. Lobos.

The control room in the hold of Pt. Lobos is the center of operations for the Ventana ROV. This room is dominated by video and computer monitors that display live video feeds and status information from the vehicle. The pilot's chair includes joystick and slider

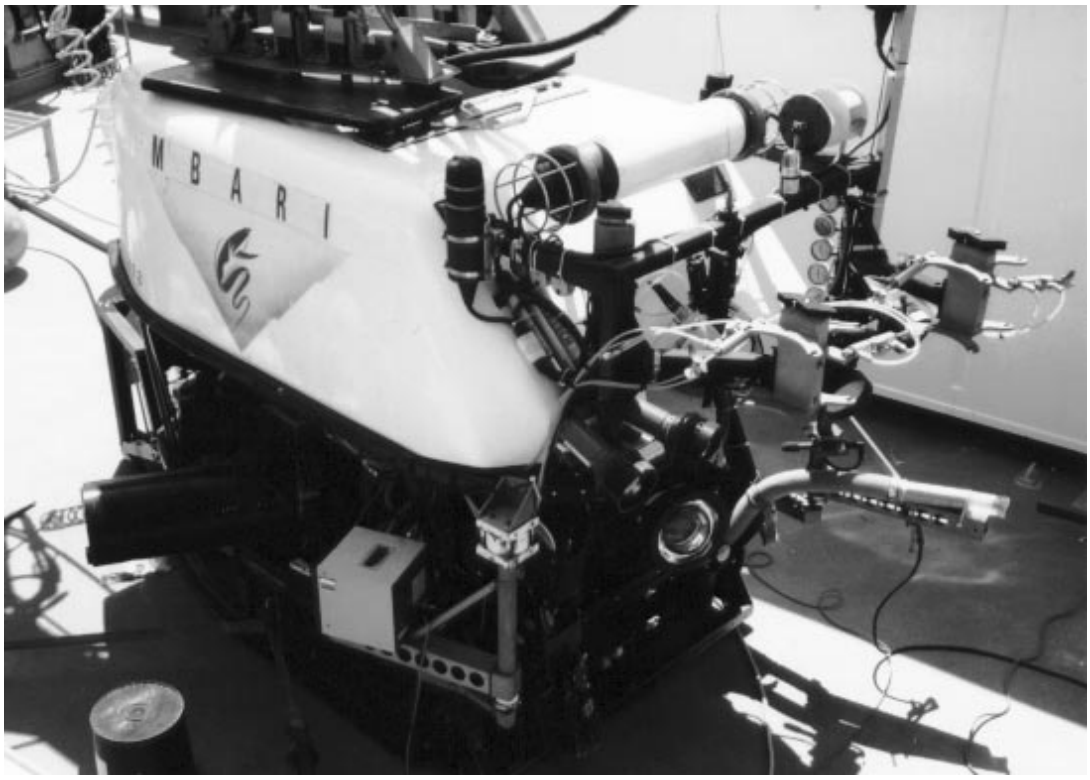

Figure 4.6: **Ventana ROV**

controls to drive the vehicle and a touch screen interface to manage the various vehicle functions. In addition, the control room contains a control panel for the main camera and a master manipulator used to control the slave manipulator on-board Ventana. For this research, the navigation software PC resided in the control room and was connected to the real-time shipside computer through a serial connection.

4.5.2 Sensors

While there is a vast array of standard and optional sensors that are available for Ventana, the sensors described below are the ones needed to perform the navigation demonstration:

Cameras The main camera is a Sony DXC3000 mounted on a 3-axis pan/tilt system at the front of the vehicle. The system is capable of panning from $-33°$ to $+44°$ and

Figure 4.7: **Point Lobos**

titling from $-105°$ to $+45°$, and it is equipped with sensors on the pan/tilt motors. The camera video is fed into the image correlator, and the encoder data are used by the vision sensing system. Near the end of testing on Ventana, a new Sony High-Definition Television (HDTV) camera was installed in place of the DXC3000, so this camera was used as the primary input to the vision sensor. In addition to the camera equipment, on-board lights are required to illuminate the underwater scene at the typical dive depths of approximately 1000 m for this work.

Compass The compass on-board Ventana is a D604 Humphrey north-seeking gyroscope, and it provides the vision sensing system with heading information.

Inclinometers Two Sperry Accustar inclinometers measure the pitch and roll of the vehicle to within $\pm 1°$. Since the vehicle is passively stable, no actuation is needed

for pitch and roll to maintain the vehicle around its nominal operating point. However, the inclinometers are still needed to measure small perturbations that affect the transformations performed by the vision sensor.

Altimeter In order to provide the vision sensor with ranging information to within ± 1 cm, a sonar altimeter is mounted on the main camera's pan/tilt unit, aligned with the optical axis of the camera. In this configuration, the altimeter measurement is guaranteed to be orthogonal to the image displacement measurements, thereby enabling 3-D position sensing regardless of the camera orientation relative to the vehicle or ocean floor.

4.5.3 Actuators

Ventana is actuated using six hydraulic thrusters. The thrusters are mounted symmetrically in pairs around the vehicle center of mass. One pair of Rexroth A2F-16 thrusters is mounted longitudinally at the rear of the vehicle to control fore/aft motions, a pair of Volvo F11 thrusters is mounted laterally to control heading and port/starboard translations, and another Volvo pair is mounted vertically to control depth. Each thruster has a servo valve manifold to control the 2900 psi @ 10 gpm hydraulic fluid flow.

4.5.4 Computer System

The Ventana computer system primarily consists of the on-board real-time computer and the shipside real-time computer. The on-board computer is responsible for maintaining vehicle status information, collecting data from the on-board sensors, and sending thruster commands received from the pilot. The shipside computer provides an I/O interface between the multitude of display and control components in the control room and the ROV.

Hardware

The on-board processing is accomplished with a Gespac real-time embedded computer driven by the Motorola 68000 processor. A 100 MHz, 486DX2 Industrial Computer Source (ICS) machine serves as the shipside real-time computer.

Software

Both the Gespac and ICS processors run CSP control software in the DOS operating system. The CSP software was developed at ISE and tailored to meet the needs of the Ventana ROV.

Control System

Since Ventana is a remotely-operated vehicle, no automatic control system exists on-board the vehicle, but the Gespac computer receives joystick commands and sends these to the thrusters at a sample rate of 10 Hz. To implement autonomous navigation on Ventana for this research, a 3-DOF control system was implemented in the navigation software to control the vehicle position. During every iteration of the 10 Hz control loop, the calculated control commands were summed with the pilot joystick commands and sent to the vehicle. Thus, automatic control could be attempted while the pilot still maintained sufficient control authority to rescue the vehicle if problems with the vision sensing/control system caused instability.

Communications

All communications among processors and sensors are achieved over RS232 serial lines. The navigation software PC was connected to the shipside computer in the same manner. This connection was bi-directional, so the PC could receive a serial string containing the relevant sensor data, and it then could transmit thruster commands through the same connection.

4.6 Summary

This chapter describes the four experimental systems that are relevant to this work. The vision sensing and navigation software is the implementation of the fundamental contributions outlined in Chapter 2, and it represents the core product of this research. The Space Frame provides truth measurements in the laboratory, to validate the theoretical developments behind the vision sensing system. OTTER serves as a testbed for testing and demonstrating the complete navigation system in a controlled underwater environment, namely, the test tank. Ventana enables the first-ever operational demonstration of vision-based, bounded-error navigation for an underwater robot in a deep ocean environment.

The technical descriptions presented in this chapter provide information that pertain to discussions throughout the remainder of this dissertation. The testing performed on these experimental systems yielded quantitative data that supported the contributions of this research, and subsequent chapters will describe these experiments in detail.

Chapter 5

State Estimator

5.1 Introduction

The state estimator is the first of three stages in the implementation of the vision sensor (Figure 2.4). The goal of the state estimator is to compute estimates of the current image state, current vehicle state, and associated variances on these estimates, and to store the image state data as part of the mosaic map. Even though navigation is performed directly in mosaic map coordinates, absolute image positions and errors serve as "consistency metrics" for the map re-alignment methods in subsequent chapters. In essence, this component of the vision sensor serves as a world model of the entire system, including the creation and maintenance of the mosaic map representing the ocean floor and vehicle localization relative to this reference map. The data flow through the sub-components of the state estimator is depicted in Figure 5.1.

The kinematic model (Section 5.3) is responsible for transforming the sensor measurements and measurement variances into the image and vehicle state estimates and state variances. The image and vehicle state estimates are based entirely on the following sensor measurement inputs; no dynamic vehicle model is assumed:

- An *image correlator* measures local displacements in the x and y directions of the image plane.

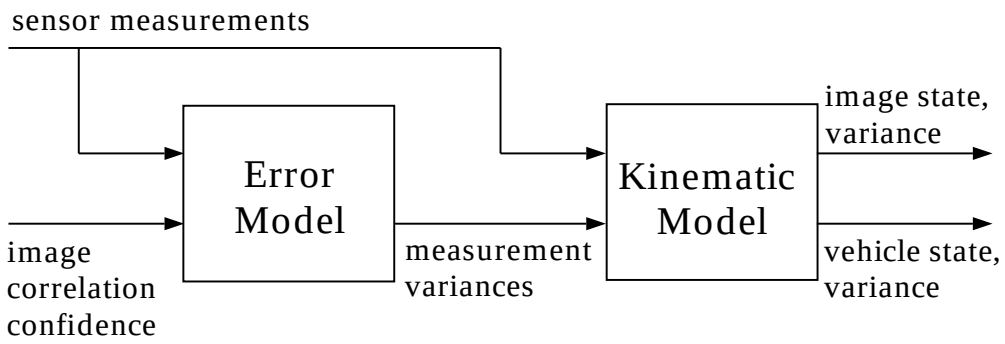

Figure 5.1: **Block Diagram for State Estimator**

- An *altimeter* measures the range from the camera to the image plane.
- An *inclinometer* measures the pitch and roll of the vehicle.
- A *compass* measures the yaw (i.e. heading) of the vehicle.
- *Pan/tilt sensors* measure the orientation of the camera relative to the vehicle.

The sensor measurements are expressed in six different frames, so a complex series of geometric transformations is required to estimate the state outputs of this model relative to a seventh reference frame attached to the ocean floor. Since the sensor measurements are independent, none of the input information is redundant. Therefore, the kinematic relations are sufficient for combining the measurement inputs, and no sensor fusion or error minimization techniques are required. The real difficulty lies in determining the variances of the state estimates. To compute the state variances, the transformation equations are formulated in terms of random variables. The sensor measurements and measurement variances then are propagated through every intermediate frame transformation, until the final mean and variance calculations yield the state estimates and state variances.

The measurement error model (Section 5.2) augments the raw sensor inputs with variance information. In order to compute the variances on the state estimates, the kinematic model assumes that the measurement variances, in addition to the raw sensor data, are

available as inputs. However, this is not automatically provided by most sensors, particularly the image correlator. To address this issue, the measurement noise associated with the image correlator has been characterized empirically. The resulting relation between the measurement confidence and variance is used to determine the variance of the current image correlation measurement dynamically.

Once the current image state estimate and variance have been determined, this information is stored along with the evolving mosaic map. The mosaic model (Section 5.4) represents the mosaic map as a network of nodes connected by links. The image state estimates and variances are stored in the nodes, while the links represent relative displacements between images in the map. Due to the variances on the image state estimates (that result directly from measurement noise), the mosaic model may not be internally self-consistent. In other words, the measured displacement between two mosaic images may vary, depending on which path was taken to perform the measurement. The goal of Chapters 6 and 7 is to detect and minimize these inconsistencies in the mosaic map.

5.2 Measurement Error Model

In attempting to predict the current image and vehicle states accurately to within well-defined error bounds, the raw sensor data alone do not provide sufficient information. The goal of the state estimator is to produce probabilistic estimates of the global image and vehicle states and associated variances; as such, the complete probability distributions on the sensor inputs must be known. For every quantity measured by one of the sensors on-board the underwater vehicle, the following additive error model is assumed:

$$z = x + v \tag{5.1}$$

In this equation, x and v are random variables, representing the quantity to be measured and the measurement error, respectively. z is the actual scalar measurement value recorded

by the sensor. Given knowledge of the sensor measurement z and the error distribution v, the distribution on x can be estimated.

The purpose of the measurement error model is to provide an estimate of x, the true value of the measured quantity, for every sensor relevant to the state estimation process. Equation 5.1 indicates that a knowledge of v is required. For several of the sensors typically found on-board the underwater vehicle, the error distributions can be inferred simply from the specifications supplied by the manufacturer or from observations of experimental data. Given an RMS error or $\pm e$ absolute bound, a particular probability distribution on the error can be assumed, and the relevant parameters can be calculated. (Section 5.2.1 will discuss the assumptions of this model in more detail.)

However, the dominant error is present in the image-correlation local displacement measurements, and no specifications exist for this sensor. Therefore, another method for discovering the error distribution must be developed. In addition to the values of the local displacements in the x and y directions, the image correlator provides a confidence value with every measurement pair. This confidence value is a measure of the degree of correlation between two images, and it is in the range of 50–100% (50% is the confidence value produced when correlating two random images, while 100% represents the correlation between two identical images). While it is not a direct measure of the error bound on the vision measurement, the confidence value could be used to infer valuable properties of the error distribution.

To accomplish this, an empirical relation between vision confidence value and error distribution was discovered, using experimental data from the Space Frame. Specifically, it was conjectured that a one-to-one relation existed between the confidence value and the error variance of the measurement pair. This hypothesis was verified during the experiments, and the collected data and resulting relation are presented in Section 5.2.2.

5.2.1 Assumptions

In deriving models for the error distributions on each of the input sensors, in particular the image correlation measurements, several simplifications were made based on observations of the data. This section lists the approximations that were made and justifies their validity.

Zero-Mean Gaussian Error Distributions

It was stated previously in Equation 5.1 that an additive error model is assumed for every sensor measurement. In addition, a Gaussian, or normal, approximation is used for the specific form of each error distribution. The normal approximation is used often to model physical phenomena because it closely characterizes the nature of random processes, such as measurement noise. It is used here to model the additive noise for every input measurement in the state estimator. For the correlation-based local displacement measurements, the central limit theorem provides additional justification for this choice. Since the local measurements (or derived quantities) ultimately will be summed to produce a global state estimate, the central limit theorem implies that the global state error can be approximated by a normal distribution, regardless of the exact nature of the local measurement error distributions.[1] Thus, provided that empirical values for the mean and variance of the error distribution can be deduced from experimental data, it is valid to use the normal approximation. In Section 5.2.2, the experimental error distribution of the correlation-based local displacement measurements is compared to a theoretical normal distribution to determine the closeness of this approximation.

A random variable with Gaussian distribution has several unique mathematical properties that will aid in the derivation of the kinematic model (Section 5.3):

- A Gaussian random variable x is completely specified by its mean $\bar{x}$ and variance X, and it will be written as $x = \mathcal{N}[\bar{x}, X]$.

[1] This does assume that all local measurement error distributions are identical, which is not necessarily true. While the forms are most likely identical, the variance values for individual measurements will differ slightly.

		m		
		1	2	3
n	1	.683	.955	.997
	2	.394	.865	.989
	3	.200	.739	.971

Table 5.1: **Probabilities for Gaussian Error Bounds**

This table depicts the percent likelihood that a sample from a Gaussian distribution will fall within the given error bound ellipsoid. n represents the number of degrees of freedom in the normal distribution, and m represents the number of multiples of σ used to define the error ellipsoid.

- The standard deviation is defined as $\sigma_x = \sqrt{X}$. The standard deviation (or multiples thereof) provides an error bound within which samples from the distribution will fall, with a specific probabilistic certainty. For an n-DOF normal distribution, the probability of a sample having a value within the m-σ error bound is listed in Table 5.1 [2].

- The linear combination of Gaussian random variables is also a Gaussian variable. This is the only distribution that has this property [2]. If $x = \mathcal{N}[\bar{x}, X]$, $y = \mathcal{N}[\bar{y}, Y]$, and $z = ax + by$ where a and b are scalar values, then $z = \mathcal{N}[a\bar{x} + b\bar{y}, a^2X + b^2Y]$.

- According to the central limit theorem [27], the sum of n independent random variables each with the same distribution is approximately normal, for large n. Thus, if each random variable has an identical (but not necessarily normal) distribution with mean μ and variance V, the sum will be approximately normal with mean $n\mu$ and variance n^2V.

Furthermore, after examining experimental data from all of the sensors, it was concluded that virtually no biases exist on the measurement data. This implies that the error distributions on all sensor measurements have zero means. This assumption also will be verified experimentally for the correlation-based local displacement measurements in Section 5.2.2.

Based on these assumptions and Equation 5.1, the error distribution is $\nu = \mathcal{N}[0, V]$, and the measurement equation can be expressed as follows:

$$x = \mathcal{N}[\bar{x}, X] = \mathcal{N}[z, V] \tag{5.2}$$

where $\bar{x}$ is the maximum likelihood estimate of the random variable x, and X is the variance of the error on this estimate.

Implicit Parameters

The empirical relation between vision confidence value and variance is an average across many variables. In other words, several relevant physical parameters are implicit in the measurement error model. The image correlation confidence value is the end result of a chain of events influenced by several parameters (Figure 5.2). These parameters can be divided into two classes: uncontrolled parameters that are part of the physical environment (Table 5.2), and controlled parameters that are part of the correlation process (Table 5.3).

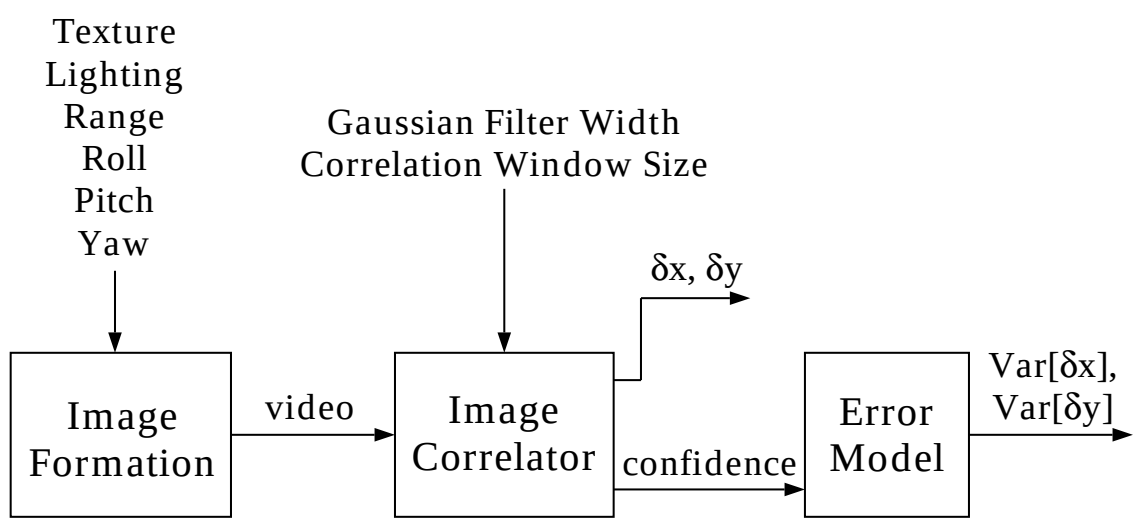

Figure 5.2: **Implicit Parameters in the Measurement Error Model**

Rather than making a futile attempt at including the subtle effects of each of these variables in the error model, the decision was made to sacrifice a small measure of accuracy in order to improve simplicity and robustness. Furthermore, a conscientious effort was made

Texture	This refers to the spatial frequency content of the scene. Since the SLoG filter described in Section 3.4.1 is essentially a band-pass filter, the highest confidence values are achieved when the texture falls within this frequency band.
Lighting	Variations in scene illumination (e.g. spotlight effects, shadows, low lighting levels) can affect the output of the image correlator, although the SLoG filter is generally insensitive to lighting changes.
Range Roll Pitch Yaw	Range is the distance from the on-board camera to the ocean floor. Roll, pitch, and yaw represent the orientation of the camera relative to the ocean floor. The vehicle control system attempts to hold these four degrees of freedom constant. Since the image correlator assumes these parameters are all constant (Section 3.3), any deviations will degrade the confidence value of the local displacement measurement.

Table 5.2: **Uncontrolled Parameters**

Gaussian Filter Width	This controls the amount of smoothing performed on each image during the SLoG filtering. More smoothing tends to degrade the accuracy slightly (and lowers the confidence value), but it greatly improves the robustness of measurements.
Correlation Window Size	This sets the portion of each image that is compared in the correlation stage. Larger windows improve the confidence value, but they require additional computation power.

Table 5.3: **Controlled Parameters**

to exercise the full "range of motion" of each of these variables while collecting the required data.[2] As a result, changes in the confidence value reflect possible changes in any of the implicit parameters, thus creating a true average relation across all variables.

5.2.2 Validation on Space Frame

In order to verify the assumptions and create empirical error models for the sensor measurements, experiments were conducted on the Space Frame to compare the sensor data with truth measurements. The focus of these experiments was the image correlation-based local displacement sensor, since the remaining sensors were assumed to have constant error variances. By collecting large amounts of data and calculating the actual measurement errors, a model for the error distribution on the image correlator measurements was estimated.

The experimental runs consisted of arbitrary, random camera motions over the entire workspace of the Space Frame. Table 5.4 lists the ranges exercised for each of the implicit model parameters. During these runs, the following measurements were collected at approximately 10 Hz: the x, y local displacements in the image plane, vision measurement confidence value, and the x, y, and z global state truth measurement. By reversing the kinematic model transformations derived in Section 5.3, x, y local displacement truth measurements were derived. By subtracting the truth data from the local displacement measurements, the measurement error (in pixels) was calculated.

Since each experimental run lasted several minutes, and the error measurements from several different runs were combined, several thousand data points were used to estimate the probability distribution on the measurement error. Furthermore, since the image correlator is inherently symmetric in the x and y directions of the image plane, the two sets of data points were combined to form a single model applicable to both x and y measurements. For each possible confidence value, a zero-mean Gaussian error distribution was assumed (Section 5.2.1). Thus, two tasks must be accomplished using the set of experimental data:

[2] Lighting conditions were changed, the system geometry was changed during and between runs, etc.

Class	Name	Range
Space Frame Workspace	Fore-Aft Span	.7–2.7 meters
	Port-Starboard Span	.1–1.1 meters
Uncontrolled Parameters	Texture	scene variations between runs
	Lighting	illumination changes between runs
	Range	.1–.85 meters
	Roll, Pitch, Yaw [3]	0 radians
Controlled Parameters	Gaussian Filter Width	10 pixels
	Correlation Window Size	64x64 pixels

Table 5.4: **Parameter Ranges**

investigate the closeness of the zero-mean normal approximation, and calculate the variance values for every possible confidence value.

Figure 5.3 is a scatter plot of the set of error measurements. For each 1% confidence interval, the mean was calculated, as indicated by the '+' signs. Then, the mean across all confidence values was calculated to be -0.86 pixels, as indicated by the solid line. Both the individual means and average mean are on the order of 1 pixel, which is the same order as the resolution of the image correlation measurements. Thus, the zero-mean assumption is a valid one, and it will be assumed for the remainder of this discussion.

The absolute value of the error is plotted in Figure 5.4. The standard deviations for each 1% confidence interval have been calculated and indicated by '+' signs on the scatter plot. The next phase is to devise a model for the variance of the measurement error as a function of the confidence value. The pattern of the standard deviation data points immediately suggests a simple form for the error variance model: a bi-modal approximation consisting of two linear segments, separated at about the 63% confidence mark. For each of these segments, a least-squares estimation was performed to derive a linear model for

[3] The camera/vehicle orientation was held constant during these trials, since the Space Frame does not currently have the capability to measure and control attitude. If these three parameters were to affect the confidence value quite differently than the other parameters, and they varied significantly during testing on-board an underwater vehicle, this would be a likely source of inaccuracy in measuring the error distributions. Fortunately, this did not appear to be the case, as shown in the experiments of Chapter 8.

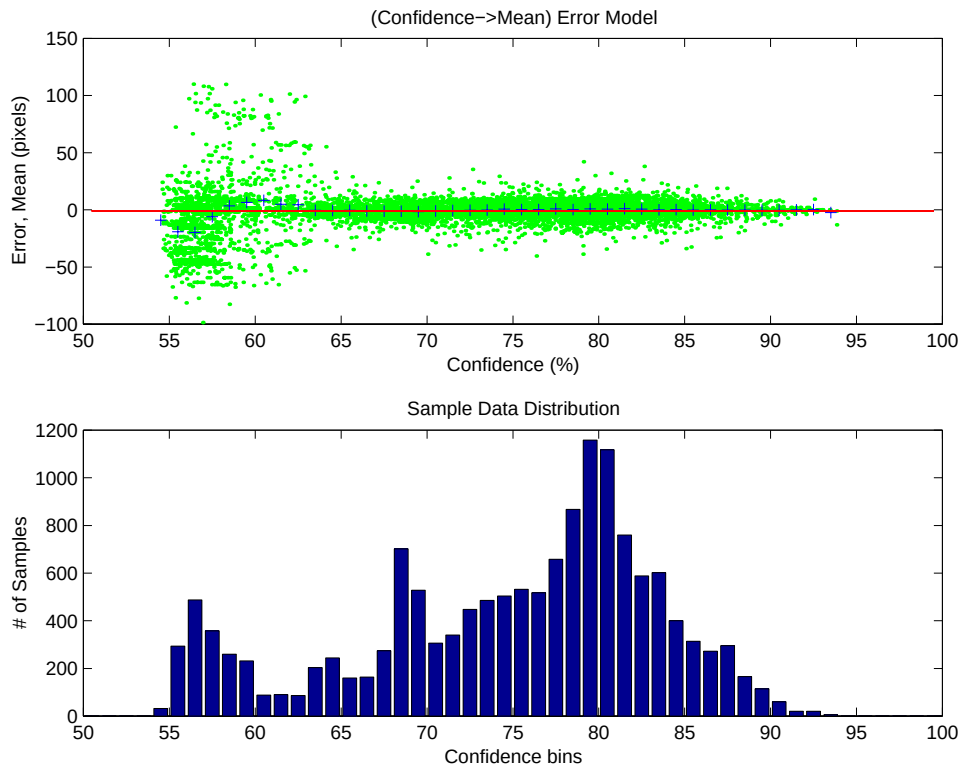

Figure 5.3: **Distribution of Error Measurements: Mean**

The top figure is a scatter plot of the error measurements. The '+' signs indicate the mean values for each 1% confidence interval, and the solid line indicates the mean value of all error measurements. The bottom figure illustrates the number of data samples in each 1% confidence interval.

the standard deviation which best fit the variance data.[4] In other words, a linear model of the form $\sigma = mc + b$ was assumed, where c is the confidence value, and m and b are scalar parameters. Then, a nonlinear parameter estimation was performed to obtain least-squares estimates of m and b for the nonlinear model $V = (mc + b)^2 = m^2c^2 + 2mbc + b^2$. This was designed to obtain a best fit for the variance data, rather than the standard deviation data, while maintaining the linearity of the model for standard deviations. The specific calculated values are given later in this section.

[4] All variance calculations and curve-fitting were done using the square of the error, while Figure 5.4 plots the absolute value of the error and standard deviations, since this is a more physically intuitive representation.

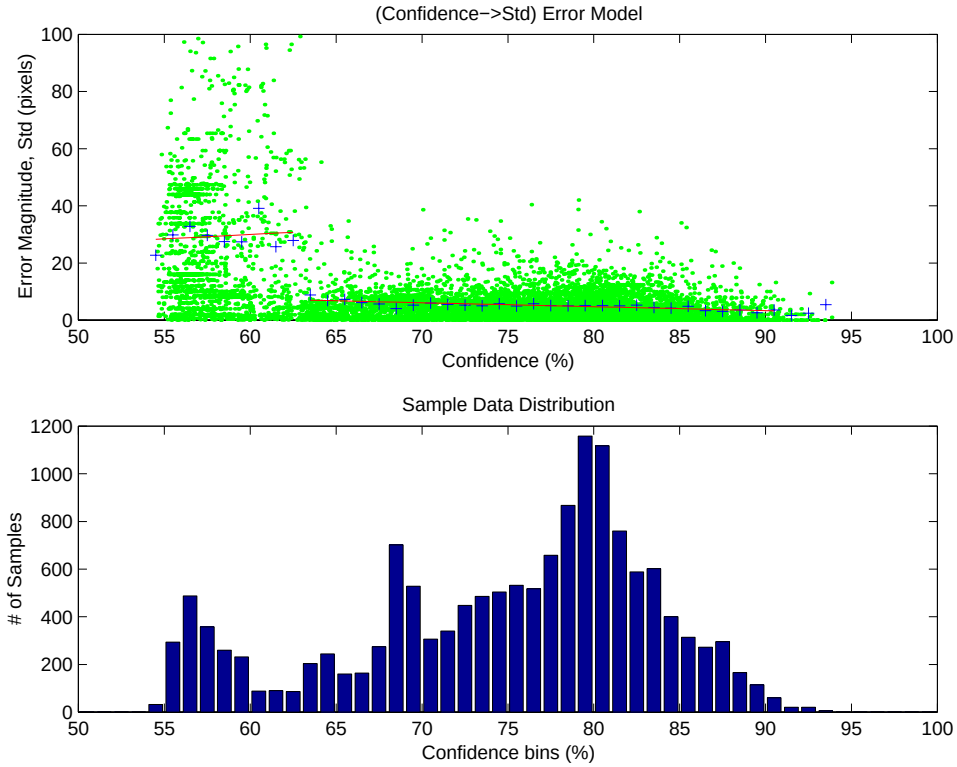

Figure 5.4: **Distribution of Error Measurements: Standard Deviation**

The top figure is a scatter plot of the absolute values of the error measurements. The '+' signs indicate the standard deviations for each 1% confidence interval, and the solid lines indicate the model approximation to the standard deviation as a function of confidence. The bottom figure illustrates the number of data samples in each 1% confidence interval.

To test how well the experimental data are approximated by a normal distribution, the error range (i.e. the y-axis of Figure 5.4) was discretized into 1 pixel intervals, and the number of samples in each interval was counted. The experimental error distribution as a function of confidence is plotted in Figure 5.5, along with a theoretical zero-mean normal distribution with variances identical to the experimental variances at every confidence interval, in Figure 5.6. While the general shape is similar, the experimental distribution has a steeper peak and longer tails than the normal distribution. Based on the central limit theorem, this variation is acceptable, since the sum of several of these distributions will approach a normal distribution.

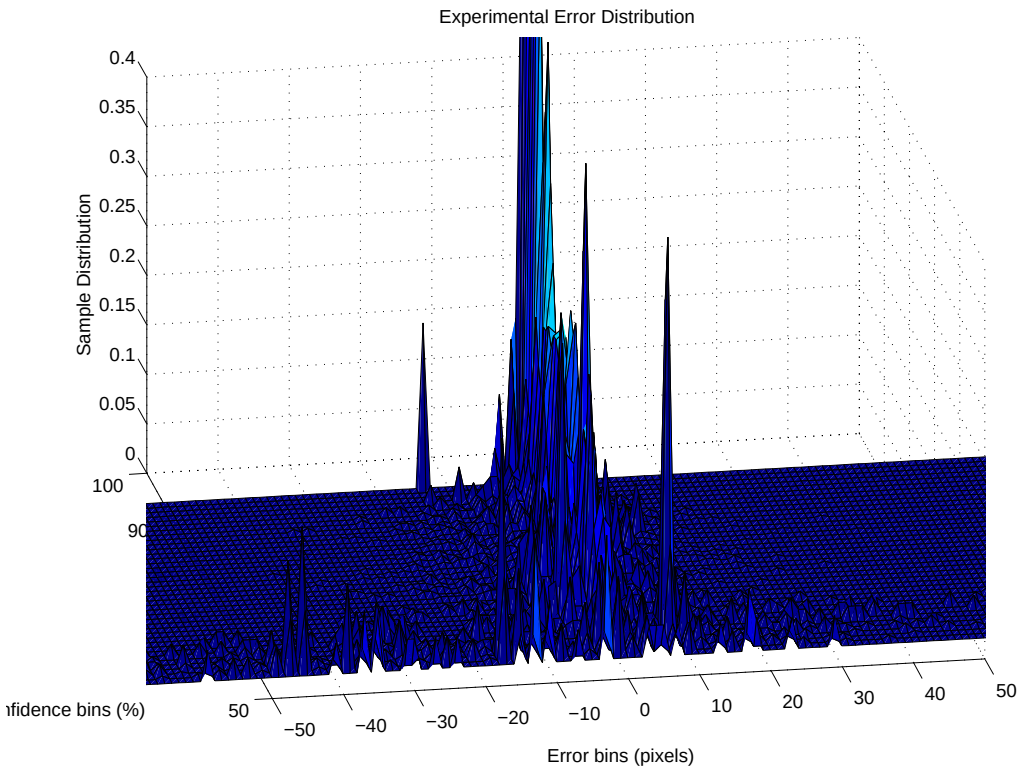

Figure 5.5: **Experimental Error Distributions**

In summary, the measurement errors can be modeled by zero-mean, normally distributed estimates of the errors on each input sensor. The estimates are fully defined by their standard deviations (or alternatively, their variances), and the specific empirical values are listed in Table 5.5. The σ estimate for the image correlator is a function of the measurement confidence value; for all other sensors, the σ estimate is constant.

5.3 Kinematic Model

The purpose of the kinematic model is to transform the local sensor measurements into image and vehicle state estimates in inertial space (Figure 5.1). Specifically, the model is a set of kinematic equations derived from the geometry of the vehicle and its environment. The measurement error model augments the data from the input sensors with Gaussian

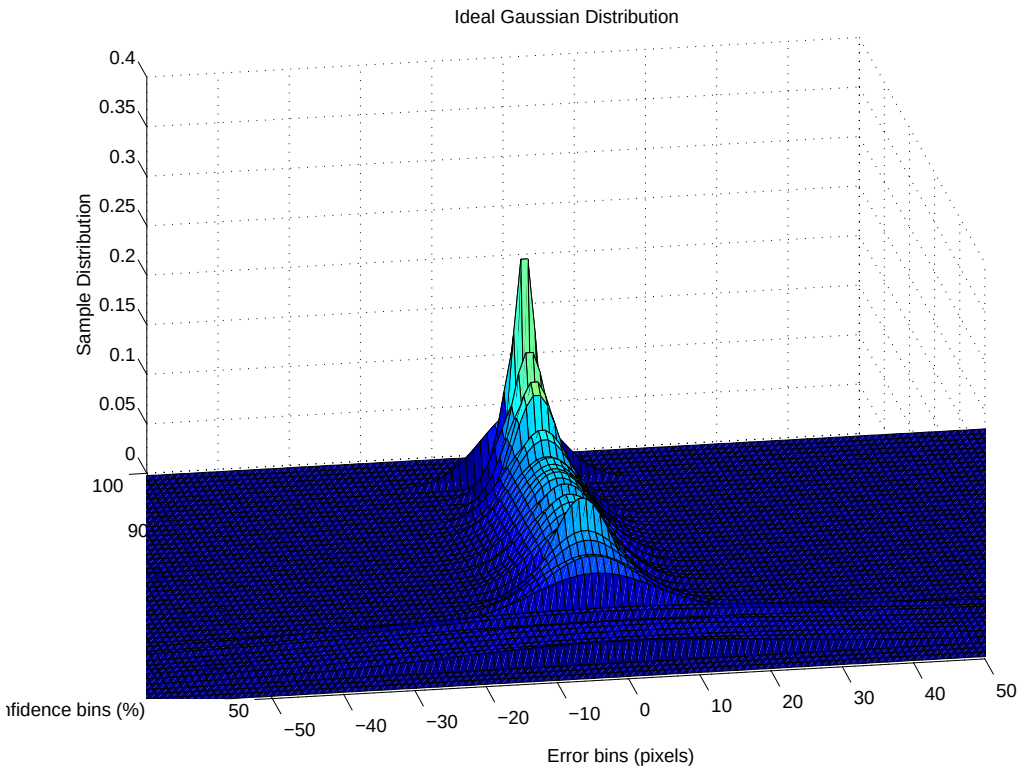

Figure 5.6: **Theoretical Error Distributions**

error bounds, and it inputs these data as random variables into the kinematic model. Since these equations deal with random variables, they transform both means and variances to produce the output distribution.

In order to understand the derivation of the transformation equations, Section 5.3.1 illustrates the vehicle and environment geometries. After Section 5.3.2 lists the relevant assumptions that were made to simplify the problem, Section 5.3.3 derives the transformation equations in full detail. This model is then validated on the Space Frame, by comparing the predicted output to experimental data in Section 5.3.4.

Sensor	Estimated Standard Deviation
Image Correlator	$\begin{cases} .317c + 10.7 \text{ pixels} & c \leq 63\% \\ -.130c + 15.10 \text{ pixels} & c > 63\% \end{cases}$
Altimeter	.1 meters
Inclinometer	$1^\circ \approx .017$ radians
Compass	$1^\circ \approx .017$ radians
Pan/Tilt Sensors	N/A [5]

Table 5.5: **Standard Deviations for Sensor Measurements**

5.3.1 System Geometry

Based on the mechanics of the video mosaicking process, the two fundamental frames used to describe the system geometry are attached to the most recently stored snapshot image and the current image. These two frames are depicted in Figure 5.7 as I and I', respectively. More precisely, the frame I could be written as $I(k)$, since it is the kth snapshot in the image chain that forms the mosaic. However, for the sake of simplicity, this parameter is not explicitly written every time. In essence, frame I represents the relevant section of the mosaic map that is used to localize the vehicle within the map. The origin of each frame coincides with the center of the corresponding image, and the axes are aligned with the camera orientation. The image correlator measures the local x, y displacements of the center of image I' (i.e. the origin of frame I') with respect to frame I.

The frames in Figure 5.7 are closely related to the evolving mosaic. Figure 3.2 illustrates the dynamic mosaic creation process, including the current image that may or may not become a new snapshot in the image chain that forms the mosaic. Frame I is attached to the most recent snapshot, and frame I' is attached to the current image. When a new image is digitized and becomes the current image, two possibilities can occur. If the current

[5] For every testbed used during the experimental phase of this research, the camera orientation was fixed with respect to the vehicle during each experimental run. Thus, the additional three camera DOF were never used, and the pan/tilt sensors (where available) were only used to measure the initial camera geometry.

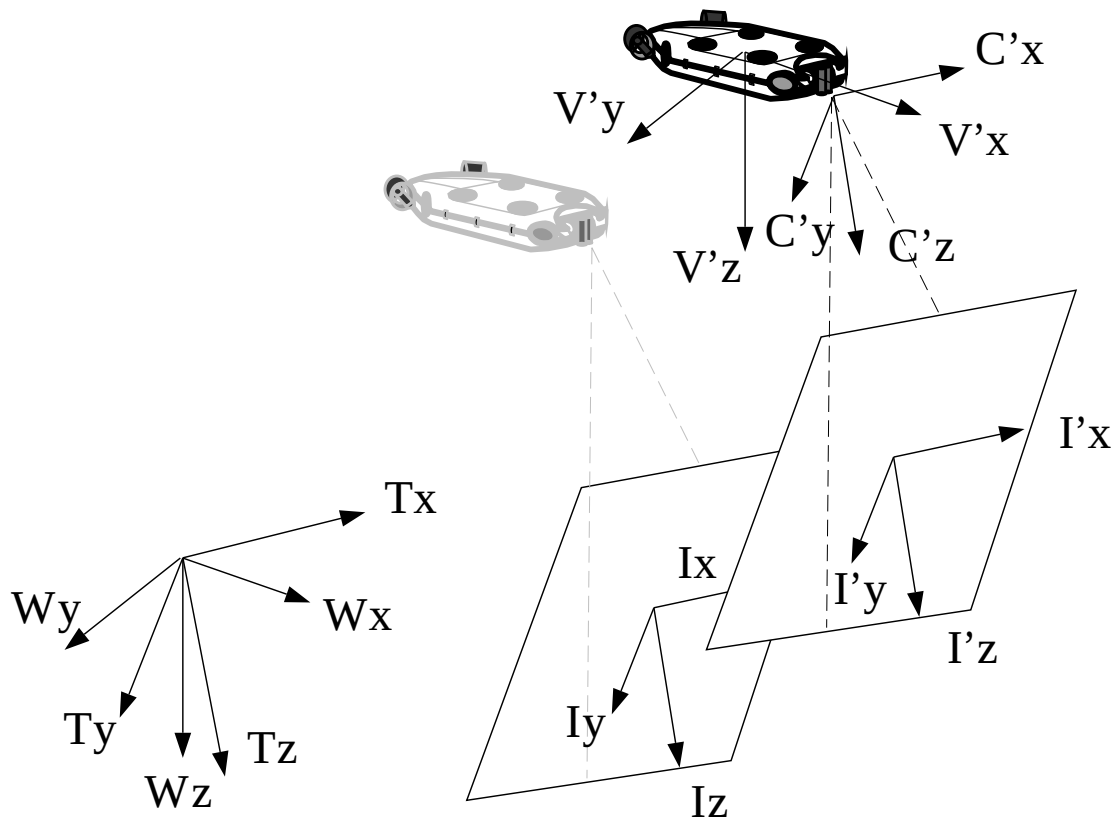

Figure 5.7: **System Geometry**

image did not become a snapshot image in the mosaic, the I' frame attaches to the new current image and the I frame does not change. On the other hand, if the current image does become part of the mosaic, the I' frame becomes the new I frame (since the current image has become the most recent snapshot), and the I' frame moves to the new current image as before.

Two more frames are used to describe the ocean floor environment. Frame T is fixed in inertial space, its origin coincides with the center of the initial image in the mosaic (i.e. the origin of frame $I(0)$), and its axes are aligned with the sloping ocean floor terrain. Frame W is also fixed in inertial space, its origin also coincides with the center of the initial image in the mosaic, but its axes are aligned with gravity.

To describe the vehicle and its components, two frames have been added to Figure 5.7. Frame C' is aligned with the on-board camera, and it represents the camera state when image I' was taken. The altimeter measures the range from the origin of C' (i.e. the center of the camera) to the origin of I' (i.e. the center of the image). Frame V', also taken at the time corresponding to image I', coincides with the vehicle center of mass and is aligned with the vehicle body. The compass and inclinometer measure the orientation of the vehicle frame V' relative to the world frame W, and the pan/tilt sensors measure the orientation of C' relative to V'.

Before proceeding any further, it is necessary to introduce several conventions and mathematical notations that will be adopted for this derivation and the remainder of this dissertation.

- Scalars have already been introduced and will be written as x; vectors will be written in bold as $\mathbf{x}$; matrices will always be assigned capital letters, as in X.

- To denote a position vector $\mathbf{p}$ expressed in a frame A, pointing from the origin of frame B to the origin of frame C, the following notation will be used: ${}^{A}\mathbf{p}_{BC}$. A rotation matrix R that describes frame B relative to frame A will be written as: ${}^{A}_{B}R$. Similarly, a vector q of Euler angles describing the same rotation will be written as: ${}^{A}_{B}\mathbf{q}$.[6] The Euler angle and rotation matrix representations of orientation are used interchangeably in this text, since the relation between the two is well-defined. Specifically, this derivation uses the Z:Y:X body-fixed Euler angle convention [4]. For this case, the vector of Euler angles q is defined as follows:

$$\mathbf{q} = \begin{pmatrix} \psi \\ \theta \\ \phi \end{pmatrix} \tag{5.3}$$

[6] Although a scalar is completely defined by its magnitude, independent of any frame, these notations are often used for scalars as well, particularly when the scalar is a component of a position or Euler angle vector. The frame notations indicate to the reader that the scalar quantity is associated with the particular frame(s) in a physical, if not mathematical, sense.

The corresponding rotation matrix R is:

$$R = \begin{pmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix} \quad (5.4)$$

- In general, a primed notation will simply denote a measurement taken at the current time. An unprimed notation will usually imply that the measurement was taken at the time the last image snapshot was added to the mosaic.

- For probability distributions, the mean of a random variable x will be denoted by $\bar{x}$; the variance will be written as $\mathrm{Var}[x]$. The covariance between two random variables x and y is $\mathrm{Cov}[x, y]$. Similarly, for a random vector $\mathbf{x}$, the mean is $\bar{\mathbf{x}}$ and the covariance matrix is $\mathrm{Cov}[\mathbf{x}]$. Also, it is important to recall the following basic definitions from probability, for both random scalars and random vectors (E stands for the expectation) [27]:

$$\bar{x} = \mathrm{E}[x] \quad (5.5)$$

$$\bar{\mathbf{x}} = \mathrm{E}[\mathbf{x}] \quad (5.6)$$

$$\mathrm{Var}[x] = \mathrm{E}[x^2] - (\mathrm{E}[x])^2 \quad (5.7)$$

$$\mathrm{Cov}[\mathbf{x}] = \mathrm{E}[\mathbf{x}\mathbf{x}^{\mathrm{T}}] - \mathrm{E}[\mathbf{x}]\mathrm{E}[\mathbf{x}^{\mathrm{T}}] \quad (5.8)$$

$$\mathrm{Cov}[x, y] = \mathrm{E}[xy] - \mathrm{E}[x]\mathrm{E}[y] \quad (5.9)$$

$$\mathrm{Cov}[\mathbf{x}, \mathbf{y}] = \mathrm{E}[\mathbf{x}\mathbf{y}^{\mathrm{T}}] - \mathrm{E}[\mathbf{x}]\mathrm{E}[\mathbf{y}^{\mathrm{T}}] \quad (5.10)$$

$$\mathrm{Var}[x + y] = \mathrm{Var}[x] + \mathrm{Var}[y] + 2\mathrm{Cov}[x, y] \quad (5.11)$$

$$\mathrm{Cov}[\mathbf{x} + \mathbf{y}] = \mathrm{Cov}[\mathbf{x}] + \mathrm{Cov}[\mathbf{y}] + \mathrm{Cov}[\mathbf{x}, \mathbf{y}] + \mathrm{Cov}[\mathbf{y}, \mathbf{x}] \quad (5.12)$$

Using this notation, the sensor measurement inputs are listed in Table 5.6. The most recent available data from each sensor are collected during every iteration of the state

Sensor	Measurement	Description	Units
Image Correlator	$\delta u'$	horizontal displacement in image plane	pixels
Image Correlator	$\delta v'$	vertical displacement in image plane	pixels
Altimeter	r'	range from camera to image plane	meters
Compass	ψ'	vehicle yaw (heading)	radians
Inclinometer	θ'	vehicle pitch	radians
Inclinometer	ϕ'	vehicle roll	radians
Pan/Tilt Sensors	${}^{C_0}_{C'}\mathbf{q} = {}^{I_0}_{I'}\mathbf{q}$ [7]	camera orientation	radians

Table 5.6: **Sensor Measurement Inputs**

Each measurement is actually specified by providing its mean and variance.

Name	Description	Units
FOV_x, FOV_y	horizontal and vertical fields of view of image	radians
w, h	width and height of image	pixels
${}^{V'}\mathbf{p}_{V'C'}$	location of camera center relative to vehicle center	meters
${}^{V'}_{C_0}\mathbf{q} = {}^{V'}_{I_0}\mathbf{q}$	initial orientation of camera relative to vehicle	radians
r	reference range from camera to image plane	m
${}^{T}\bar{\mathbf{p}}_{TI}$, ${}^{T}_{I}\bar{\mathbf{q}}$	6-DOF reference image state in terrain frame	m, rad
$\mathrm{Cov}[{}^{T}\mathbf{p}_{TI}]$, $\mathrm{Cov}[{}^{T}_{I}\mathbf{q}]$	covariance of reference image state estimate	m^2, r^2

Table 5.7: **Known Geometric Quantities**

estimator computation loop; the sample rate of this loop is approximately 10 Hz. [8] In addition, Table 5.7 lists other quantities whose values are given, based on the vehicle and scene geometry or past measurements.

5.3.2 Assumptions

Since the kinematic model will ultimately be implemented in real-time, the goal of this derivation is to develop a computationally efficient, first-order approximation to the system

[7] Since the pan/tilt sensors actually measure the camera orientation relative to the vehicle, some pre-processing must be performed to provide the camera orientation relative to its initial orientation.

[8] This sample rate could be increased to 30 Hz simply by increasing the available computational power. Since the image correlator can digitize the camera video at a maximum frame rate of 30 Hz, this sensor is the limiting factor in determining the maximum sampling rate.

kinematics. To accomplish this end, several assumptions have been made to simplify the equations:

1. At a given time step, all sensor measurements are independent. Furthermore, all measurements at time k are independent from those at time $k+1$. Since independence implies that these variables are also uncorrelated, the following two important statements can be made about the independent random variables x and y [27]:

$$\mathrm{E}[xy] = \mathrm{E}[x]\mathrm{E}[y] \tag{5.13}$$

$$\mathrm{Cov}[x, y] = 0 \tag{5.14}$$

2. The scene terrain (e.g. ocean floor, vertical rock face) is assumed to be roughly planar; by definition, it coincides with the X-Y plane of the terrain frame T. The terrain frame may be arbitrarily sloped in relation to the world frame W, and knowledge of this rotation, ${}^{W}_{T}\mathbf{q}$, is assumed. In practice, the axis of the camera is initially aligned by eye to be perpendicular to the ocean floor, and the appropriate measurements are taken to determine the ocean floor orientation. While small height variations of the terrain are allowed, these range deviations must be small compared to the range from camera to terrain plane (i.e. $\left(\frac{\delta h}{r}\right)^2 \ll 1$). Furthermore, the range measurement is taken to be the range from the camera to the ideal plane of the ocean floor, regardless of any surface unevenness.

3. While the vehicle is free to translate in a single plane parallel to the ocean floor, its control system constrains any rotations or range changes. Since the vehicle is under active control, the Euler angles ${}^{W}_{V'}\mathbf{q}$ can be assumed to be small (i.e. $\theta^2 \ll 1$ rad), such that the approximations: $\sin\theta \approx \theta$, $\cos\theta \approx 1$ are valid. Similarly, since the camera axis is nominally perpendicular to the terrain, the small angle approximation can be made for ${}^{T}_{C'}\mathbf{q}$, ${}^{T}_{I}\mathbf{q}$, and ${}^{T}_{I'}\mathbf{q}$. In practice, it is possible to use the exact kinematic state equations, but for the purposes of calculating the variances of these states, the small angle approximation is exceedingly useful. Using this approximation, the rotation

matrix from Equation 5.4 can be approximated as follows:

$$R = \begin{pmatrix} 1 & -\psi & \theta \\ \psi & 1 & -\phi \\ -\theta & \phi & 1 \end{pmatrix} \tag{5.15}$$

Also, active control on the vehicle range ensures that range changes are small compared to the range itself: $\left(\frac{\delta r}{r}\right)^2 \ll 1$.

As stated in the previous assumption, the variance computations will only be accurate to first order, and the equations can be further simplified by taking advantage of other approximations related to the accuracies of the input measurements. This is the topic of the remaining assumptions on this list. For the specific experimental systems used in this work, the sensors are more than capable of satisfying the conditions of these approximations.

4. The local displacements of the current image relative to the most recent stored snapshot are small compared to the range from the camera to the image plane: $\left(\frac{{}^I x_{II'}}{r'}\right)^2 \ll 1$ and $\left(\frac{{}^I y_{II'}}{r'}\right)^2 \ll 1$. Using the condition on terrain variation in Assumption 2 and the small angle approximation from Assumption 3, it can also be stated that the depth variation in the image is small compared to the range from the camera to the ocean floor. These two statements in combination are necessary and sufficient conditions for the assumption of orthographic projection [20]. If ${}^I x_{II'}$ and ${}^I y_{II'}$ were to increase, pitch and roll motions of the camera would cause the view region of the image to become trapezoidal, as opposed to a rectangular view region when the camera is perpendicular to the terrain. This phenomenon is commonly known as the keyhole effect. With an orthographic projection, the keyhole effect becomes a second order effect that can be neglected.

5. The magnitude of the error on the range measurement is much smaller than the magnitude of the range measurement itself: $\mathrm{Var}[r'] \ll (\bar{r}')^2$. This assumption essentially places a restriction on the signal-to-noise ratio of the altimeter.

Name	Description	Units
${}^{T}\bar{\mathbf{p}}_{TI'}$, ${}^{T}_{I'}\bar{\mathbf{q}}$	6-DOF current image state in terrain frame	m, rad
Cov[${}^{T}\mathbf{p}_{TI'}$], Cov[${}^{T}_{I'}\mathbf{q}$]	covariance of current image state estimate	m^2, r^2
${}^{T}\bar{\mathbf{p}}_{TC'}$, ${}^{T}_{C'}\bar{\mathbf{q}}$	6-DOF camera state in terrain frame	m, rad
Cov[${}^{T}\mathbf{p}_{TC'}$], Cov[${}^{T}_{C'}\mathbf{q}$]	covariance of camera state estimate	m^2, r^2
${}^{W}\bar{\mathbf{p}}_{WV'}$, ${}^{W}_{V'}\bar{\mathbf{q}}$	6-DOF vehicle state in world frame	m, rad
Cov[${}^{W}\mathbf{p}_{WV'}$], Cov[${}^{W}_{V'}\mathbf{q}$]	covariance of vehicle state estimate	m^2, r^2

Table 5.8: **State Estimator Outputs**

6. The magnitude of the error on the yaw measurement is small: $\mathrm{Var}[\psi'] \ll 1\ \mathrm{rad}^2$. This assumption essentially places a restriction on the accuracy of the compass.

5.3.3 Derivation of Global State Estimates

The goal of this derivation is to provide a set of equations to transform the raw sensor data into estimates of the image, camera, and vehicle global states. As previously discussed, Table 5.6 lists the sensor inputs, and Table 5.7 lists other relevant geometric quantities to be used in the derivation. The intended outputs of this model are described briefly in Table 5.8. Figure 5.7 will be used as a guide when transforming among the various frames.

Image Displacement in Image Frame

Starting with the local displacements from the image correlator, these measurements must be converted to the image displacement in meters. Based on Assumption 4, the following equation uses an orthographic projection.[9] Since the orthographic projection model only defines the scene-to-image mapping up to a scale factor, the external range measurement and knowledge of the camera FOV's enable determination of this scale factor. Thus, knowledge

[9]Given the assumption of orthographic projection, an affine transformation could be used to construct an image of the same section of the ocean floor from a different camera viewpoint. For instance, this would be useful for realistic display of the mosaic images in a 3-D GUI. The interface used in this research does not perform such transformations, in order to minimize computation and improve the interactivity of the GUI.

of the image width and height in meters enables conversion of the vision measurements from pixels to meters. The transformation from image to scene coordinates is as follows:[10]

$$
\begin{aligned}
{}^{I}\mathbf{p}_{II'} &= \begin{pmatrix} {}^{I}x_{II'} \\ {}^{I}y_{II'} \\ {}^{I}z_{II'} \end{pmatrix} \\
&= \begin{pmatrix} \left[\frac{2}{h}\tan\left(\frac{\mathrm{FOV}_y}{2}\right)\right](r)(\delta v') \\ \left[\frac{2}{w}\tan\left(\frac{\mathrm{FOV}_x}{2}\right)\right](r)(\delta u') \\ {}^{I}z_{II'} \end{pmatrix}
\end{aligned}
\tag{5.16}
$$

The reversal of the u, v coordinates when going from the $\delta u', \delta v'$ sensor measurements to the I frame is due to the fact that the image correlator measures u along the horizontal axis of the image as seen by the camera, while the x-axis is directed forward in the I frame. Similarly, v is vertical in the camera image, while the y-axis points to the right in the I frame. Also, the component ${}^{I}z_{II'}$ has not been computed; it will be determined in a later step of the derivation.

Equation 5.16 provides a relationship among several random variables. However, each random variable input was specified by its mean and variance, and these do not appear in the above equation. To determine the distribution on ${}^{I}\mathbf{p}_{II'}$ as a function of known parameters, the mean and covariance of this equation must be calculated. Furthermore, the expressions have been simplified using Assumption 5:

$$
{}^{I}\bar{\mathbf{p}}_{II'} = \begin{pmatrix} {}^{I}\bar{x}_{II'} \\ {}^{I}\bar{y}_{II'} \\ {}^{I}\bar{z}_{II'} \end{pmatrix}
$$

[10] The assumption of orthographic projection is implicit in the fact that the scene-to-image transform is represented as a coordinate transformation. In other words, the line connecting the 3-D object point on the terrain and its 2-D projection in the image plane is only perpendicular to the image plane (i.e. along the z-axis) under the assumption of orthographic, not perspective, projection.

$$= \begin{pmatrix} \left[\frac{2}{h}\tan\left(\frac{\mathrm{FOV}_y}{2}\right)\right](\bar{r})(\delta\bar{y}') \\ \left[\frac{2}{w}\tan\left(\frac{\mathrm{FOV}_x}{2}\right)\right](\bar{r})(\delta\bar{x}') \\ {}^{I}\bar{z}_{II'} \end{pmatrix} \tag{5.17}$$

$$\mathrm{Cov}[{}^{I}\mathbf{p}_{II'}] = \begin{pmatrix} \mathrm{Var}[{}^{I}x_{II'}] & \mathrm{Cov}[{}^{I}x_{II'}, {}^{I}y_{II'}] & \mathrm{Cov}[{}^{I}x_{II'}, {}^{I}z_{II'}] \\ \mathrm{Cov}[{}^{I}x_{II'}, {}^{I}y_{II'}] & \mathrm{Var}[{}^{I}y_{II'}] & \mathrm{Cov}[{}^{I}y_{II'}, {}^{I}z_{II'}] \\ \mathrm{Cov}[{}^{I}x_{II'}, {}^{I}z_{II'}] & \mathrm{Cov}[{}^{I}y_{II'}, {}^{I}z_{II'}] & \mathrm{Var}[{}^{I}z_{II'}] \end{pmatrix} \tag{5.18}$$

where:

$$\mathrm{Var}[{}^{I}x_{II'}] = \left[\frac{2}{h}\tan\left(\frac{\mathrm{FOV}_y}{2}\right)\right]^2 \left[(\bar{r})^2\mathrm{Var}[\delta v'] + (\delta\bar{y}')^2\mathrm{Var}[r]\right] \tag{5.19}$$

$$\mathrm{Var}[{}^{I}y_{II'}] = \left[\frac{2}{w}\tan\left(\frac{\mathrm{FOV}_x}{2}\right)\right]^2 \left[(\bar{r})^2\mathrm{Var}[\delta u'] + (\delta\bar{x}')^2\mathrm{Var}[r]\right] \tag{5.20}$$

$$\mathrm{Cov}[{}^{I}x_{II'}, {}^{I}y_{II'}] = ({}^{I}\bar{x}_{II'})({}^{I}\bar{y}_{II'})\left(\frac{\mathrm{Var}[r]}{\bar{r}^2}\right) \tag{5.21}$$

Image Orientation in Terrain Frame

Next, it would be desirable to determine the image local displacement vector in terrain coordinates. However, this transformation requires knowledge of the rotation between the terrain frame T and current image frame I'. This rotation is composed of two independent dynamic effects, namely, the vehicle rotation relative to the world frame (${}^{W}_{V'}\mathbf{q}$) and the camera rotation relative to the vehicle (${}^{V'}_{C'}\mathbf{q} = {}^{V'}_{I'}\mathbf{q}$).

The on-board sensors provide the orientation of the vehicle frame V' relative to the world frame W:

$${}^{W}_{V'}\bar{\mathbf{q}} = \begin{pmatrix} \bar{\psi}' \\ \bar{\theta}' \\ \bar{\phi}' \end{pmatrix}, \mathrm{Cov}[{}^{W}_{V'}\mathbf{q}] = \begin{pmatrix} \mathrm{Var}[\psi'] & 0 & 0 \\ 0 & \mathrm{Var}[\theta'] & 0 \\ 0 & 0 & \mathrm{Var}[\phi'] \end{pmatrix} \tag{5.22}$$

According to the algebra of transformations for rotation matrices:

$${}^{T}_{I'}R = ({}^{T}_{W}R)({}^{W}_{V'}R)({}^{V'}_{I'}R) \tag{5.23}$$

Assumption 2 states that the orientation of the terrain frame T relative to the world frame W is defined as the initial orientation of the image frame I' (or camera frame C') relative to the vehicle frame V'. These initial orientations will be denoted by a subscript on the frame designation:

$$ {}^{W}_{T}R = {}^{T}_{W}R^{\mathrm{T}} = {}^{V'}_{I_0}R = {}^{V'}_{C_0}R \tag{5.24} $$

If the camera is mounted on a pan-tilt unit or similar device, ${}^{V'}_{I'}R$ may change over time. The camera pan/tilt sensors provide ${}^{I_0}_{I'}\mathbf{q}$, the current camera orientation relative to its initial orientation, at every time step (Table 5.6).[11] This leads to an expression for ${}^{V'}_{I'}R$:

$$ {}^{V'}_{I'}R = ({}^{V'}_{I_0}R)({}^{I_0}_{I'}R) \tag{5.25} $$

Substituting Equations 5.24 and 5.25 into Equation 5.23:

$$ {}^{T}_{I'}R = ({}^{V'}_{I_0}R^{\mathrm{T}})({}^{W}_{V'}R)({}^{V'}_{I_0}R)({}^{I_0}_{I'}R) \tag{5.26} $$

Using Assumption 3, the rotation matrices ${}^{T}_{I'}R$, ${}^{W}_{V}R$, and ${}^{I_0}_{I'}R$ can be expressed using the small angle approximation. By substituting in for the rotation matrices and performing the computation of Equation 5.26, then solving for the equivalent Euler angle representation, an interesting result is achieved:

$$ {}^{T}_{I'}\mathbf{q} = ({}^{V'}_{I_0}R^{\mathrm{T}})({}^{W}_{V'}\mathbf{q}) + {}^{I_0}_{I'}\mathbf{q} \tag{5.27} $$

This equation provides a formula to transform Euler angles between frames (for the specific geometric configuration given and under the several restrictions mentioned). Solving for the mean and covariance:

$$ {}^{T}_{I'}\bar{\mathbf{q}} = ({}^{V'}_{I_0}R^{\mathrm{T}})({}^{W}_{V'}\bar{\mathbf{q}}) + {}^{I_0}_{I'}\bar{\mathbf{q}} \tag{5.28} $$

$$ \mathrm{Cov}[{}^{T}_{I'}\mathbf{q}] = ({}^{V'}_{I_0}R^{\mathrm{T}})(\mathrm{Cov}[{}^{W}_{V'}\mathbf{q}])({}^{V'}_{I_0}R) + \mathrm{Cov}[{}^{I_0}_{I'}\mathbf{q}] \tag{5.29} $$

[11] During the experimentation phase of this research, the camera was fixed in place once its initial orientation was set. In other words, ${}^{I_0}_{I'}\mathbf{q}$ was equal to the zero vector during experimentation.

Image Displacement in Terrain Frame

The next step in the derivation is to determine the location of the current image center (i.e. origin of I') relative to the reference image center (i.e. origin of I) in terrain frame coordinates. The transformation equation for the quantity in question, ${}^T\mathbf{p}_{II'}$, is as follows:

$$ {}^T\mathbf{p}_{II'} = ({}^T_I R)({}^I\mathbf{p}_{II'}) \tag{5.30} $$

Using Equation 5.16 and the fact that the origin of every image is defined to lie in the plane of the ocean floor, the following substitutions can be made:

$$ \begin{pmatrix} {}^T x_{II'} \\ {}^T y_{II'} \\ 0 \end{pmatrix} = {}^T_I R \begin{pmatrix} {}^I x_{II'} \\ {}^I y_{II'} \\ {}^I z_{II'} \end{pmatrix} \tag{5.31} $$

Substituting in the small angle approximation for the rotation matrix ${}^T_I R$ (Equation 5.15), and solving for ${}^I z_{II'}$:

$$ {}^I z_{II'} = ({}^T_I \theta)({}^I x_{II'}) - ({}^T_I \phi)({}^I y_{II'}) \tag{5.32} $$

Using this value, an expression for ${}^T\mathbf{p}_{II'}$ can be calculated, and it can then be simplified using Assumption 3:

$$ \begin{aligned} {}^T\mathbf{p}_{II'} &= \begin{pmatrix} {}^I x_{II'} - ({}^T_I \psi)({}^I y_{II'}) + ({}^T_I \theta)({}^I z_{II'}) \\ ({}^T_I \psi)({}^I x_{II'}) + {}^I y_{II'} - ({}^T_I \phi)({}^I z_{II'}) \\ 0 \end{pmatrix} \\ &\approx \begin{pmatrix} {}^I x_{II'} - ({}^T_I \psi)({}^I y_{II'}) \\ ({}^T_I \psi)({}^I x_{II'}) + {}^I y_{II'} \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & -{}^T_I \psi & 0 \\ {}^T_I \psi & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} {}^I\mathbf{p}_{II'} \end{aligned} \tag{5.33} $$

To obtain the distribution on ${}^T\mathbf{p}_{II'}$, the mean and variance of Equation 5.33 is calculated and then simplified using Assumption 6:

$$ {}^T\bar{\mathbf{p}}_{II'} = \begin{pmatrix} {}^I\bar{x}_{II'} - ({}^T_I\bar{\psi})({}^I\bar{y}_{II'}) \\ ({}^T_I\bar{\psi})({}^I\bar{x}_{II'}) + {}^I\bar{y}_{II'} \\ 0 \end{pmatrix} \tag{5.34} $$

$$ \mathrm{Cov}[{}^T\mathbf{p}_{II'}] = \begin{pmatrix} \mathrm{Var}[{}^Tx_{II'}] & \mathrm{Cov}[{}^Tx_{II'}, {}^Ty_{II'}] & 0 \\ \mathrm{Cov}[{}^Tx_{II'}, {}^Ty_{II'}] & \mathrm{Var}[{}^Ty_{II'}] & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{5.35} $$

where:

$$ \mathrm{Var}[{}^Tx_{II'}] = \mathrm{Var}[{}^Ix_{II'}] + ({}^I\bar{y}_{II'})^2(\mathrm{Var}[{}^T_I\psi]) - 2({}^T_I\bar{\psi})(\mathrm{Cov}[{}^Ix_{II'}, {}^Iy_{II'}]) \tag{5.36} $$

$$ \mathrm{Var}[{}^Ty_{II'}] = \mathrm{Var}[{}^Iy_{II'}] + ({}^I\bar{x}_{II'})^2(\mathrm{Var}[{}^T_I\psi]) + 2({}^T_I\bar{\psi})(\mathrm{Cov}[{}^Ix_{II'}, {}^Iy_{II'}]) \tag{5.37} $$

$$ \mathrm{Cov}[{}^Tx_{II'}, {}^Ty_{II'}] = ({}^T_I\bar{\psi})(\mathrm{Var}[{}^Ix_{II'}] - \mathrm{Var}[{}^Iy_{II'}]) + \mathrm{Cov}[{}^Ix_{II'}, {}^Iy_{II'}] \tag{5.38} $$

Image State in Terrain Frame

Once the relative image displacement has been calculated in the proper frame, a global estimate of the image state can be determined. Specifically, the image state with respect to the terrain frame will be calculated. This quantity is crucial to the later stages of the estimation process. Essentially, this calculation places each image in the mosaic within a global framework. Re-alignment of the mosaic using new measurement updates is accomplished within this framework.

The calculation of ${}^T\mathbf{p}_{TI'}$, the location of the current image in the terrain frame, can be achieved by summing the relative image displacements along the entire image chain that composes the mosaic. However, noting that the I frame was at some point in the past the I' frame, the position vector ${}^T\mathbf{p}_{TI}$ is available and contains the global image position up until that point in the image chain. Thus, the equation for ${}^T\mathbf{p}_{TI'}$ is relatively simple, based

on standard vector algebra:

$$ {}^{T}\mathbf{p}_{TI'} = {}^{T}\mathbf{p}_{TI} + {}^{T}\mathbf{p}_{II'} \tag{5.39} $$

Since ${}^{T}\mathbf{p}_{TI}$ and ${}^{T}\mathbf{p}_{II'}$ are independent, the mean and variance relations are also straightforward:

$$ {}^{T}\bar{\mathbf{p}}_{TI'} = {}^{T}\bar{\mathbf{p}}_{TI} + {}^{T}\bar{\mathbf{p}}_{II'} \tag{5.40} $$

$$ \mathrm{Cov}[{}^{T}\mathbf{p}_{TI'}] = \mathrm{Cov}[{}^{T}\mathbf{p}_{TI}] + \mathrm{Cov}[{}^{T}\mathbf{p}_{II'}] \tag{5.41} $$

Since state refers to both position and orientation, Equations 5.27– 5.29 complete the state description of the current image frame, I', relative to the terrain frame, T.

Camera State in Terrain Frame

After determining the global displacement of images mapped onto the terrain, the next step is to determine the state of the camera above the terrain. This is most easily accomplished by taking advantage of the fact that the range measurement describes the displacement of the image plane away from the camera, along the optical axis of the camera:

$$ \begin{aligned} {}^{T}\mathbf{p}_{TC'} &= {}^{T}\mathbf{p}_{TI'} + ({}^{T}_{I'}R)({}^{I'}\mathbf{p}_{I'C'}) \\ &= {}^{T}\mathbf{p}_{TI'} + {}^{T}_{I'}R \begin{pmatrix} 0 \\ 0 \\ -r' \end{pmatrix} \end{aligned} \tag{5.42} $$

Using Assumption 3 and multiplying:

$$ {}^{T}\mathbf{p}_{TC'} = {}^{T}\mathbf{p}_{TI'} + \begin{pmatrix} -({}^{T}_{I}\theta')(r') \\ ({}^{T}_{I}\phi')(r') \\ -r' \end{pmatrix} \tag{5.43} $$

The mean and variance equations follow directly from this equation, and they have been simplified using Assumptions 3 and 5:

$$ {}^{T}\bar{\mathbf{p}}_{TC'} = {}^{T}\bar{\mathbf{p}}_{TI'} + \begin{pmatrix} -({}^{T}_{I}\bar{\theta}')(\bar{r}') \\ ({}^{T}_{I}\bar{\phi}')(\bar{r}') \\ -\bar{r}' \end{pmatrix} \tag{5.44} $$

$$ \begin{aligned} \mathrm{Cov}[{}^{T}\mathbf{p}_{TC'}] &= \mathrm{Cov}[{}^{T}\mathbf{p}_{TI'}] + \\ & \begin{pmatrix} (\bar{r}')^2\mathrm{Var}[{}^{T}_{I}\theta'] & -(\bar{r}')^2\mathrm{Cov}[{}^{T}_{I}\theta', {}^{T}_{I}\phi'] & ({}^{T}_{I}\bar{\theta}')\mathrm{Var}[r'] \\ -(\bar{r}')^2\mathrm{Cov}[{}^{T}_{I}\phi', {}^{T}_{I}\theta'] & (\bar{r}')^2\mathrm{Var}[{}^{T}_{I}\phi'] & -({}^{T}_{I}\bar{\phi}')\mathrm{Var}[r'] \\ ({}^{T}_{I}\bar{\theta}')\mathrm{Var}[r'] & -({}^{T}_{I}\bar{\phi}')\mathrm{Var}[r'] & \mathrm{Var}[r'] \end{pmatrix} \end{aligned} \tag{5.45} $$

The various variance and covariance terms in the large matrix in Equation 5.45 are elements of the matrix $\mathrm{Cov}[{}^{T}_{I'}\mathbf{q}]$, as described in Equation 5.29.

For the orientation component of the state description, the orientations of the camera frame and image frame are defined to be identical. Therefore, ${}^{T}_{C'}\mathbf{q} = {}^{T}_{I'}\mathbf{q}$, and Equations 5.27– 5.29 define ${}^{T}_{C'}\mathbf{q}$ as well.

Vehicle State in World Frame

At this point in the derivation, all quantities needed for subsequent estimation stages have been derived. Also, the vehicle has been localized with respect to a global frame; specifically, the point on the vehicle corresponding to the camera center has been localized with respect to the terrain frame. However, for control purposes, the location of the vehicle center of mass with respect to a fixed sensor frame is often required. In this final step of the derivation, the location of the vehicle center of mass will be calculated with respect to the world frame.

Given that the camera position with respect to the terrain frame is already known from Equations 5.43 and 5.27– 5.29, the vehicle position with respect to the terrain frame can

be determined through a straightforward frame transformation:

$$^{T}\mathbf{p}_{TV'} = {}^{T}\mathbf{p}_{TC'} + ({}^{T}_{C'}R)({}^{C'}\mathbf{p}_{C'V'}) \tag{5.46}$$

where

$$^{C'}\mathbf{p}_{C'V'} = -({}^{C'}_{V'}R)({}^{V'}\mathbf{p}_{V'C'}) \tag{5.47}$$

Similarly, the transformation from terrain frame to world frame is as follows:

$$^{W}\mathbf{p}_{WV'} = {}^{W}\mathbf{p}_{WT} + ({}^{W}_{T}R)({}^{T}\mathbf{p}_{TV'}) \tag{5.48}$$

Noting that the origins of the world and terrain frames are coincident ($^{W}\mathbf{p}_{WT} = 0$), substituting Equations 5.24, 5.46, and 5.47 into Equation 5.48 and simplifying:

$$^{W}\mathbf{p}_{WV'} = ({}^{V'}_{C_0}R)({}^{T}\mathbf{p}_{TC'}) - ({}^{W}_{V'}R)({}^{V'}\mathbf{p}_{V'C'}) \tag{5.49}$$

The mean of $^{W}\mathbf{p}_{WV'}$ can now be calculated:

$$^{W}\bar{\mathbf{p}}_{WV'} = ({}^{W}_{T}R)({}^{T}\bar{\mathbf{p}}_{TC'}) - ({}^{W}_{V'}\bar{R})({}^{V'}\mathbf{p}_{V'C'}) \tag{5.50}$$

However, calculating the co-variance is a difficult task, since the random variables are located in the vector $^{T}\mathbf{p}_{TC'}$ and the matrix ${}^{W}_{V'}R$. By taking advantage of Assumption 3 for ${}^{W}_{V'}R$, Equation 5.49 can be re-written as:

$$^{W}\mathbf{p}_{WV'} = ({}^{V'}_{C_0}R)({}^{T}\mathbf{p}_{TC'}) - A({}^{W}_{V'}\mathbf{q}) - {}^{V'}\mathbf{p}_{V'C'} \tag{5.51}$$

where

$$A = \begin{pmatrix} -{}^{V'}y_{V'C'} & {}^{V'}z_{V'C'} & 0 \\ {}^{V'}x_{V'C'} & 0 & -{}^{V'}z_{V'C'} \\ 0 & -{}^{V'}x_{V'C'} & {}^{V'}y_{V'C'} \end{pmatrix} \tag{5.52}$$

Before solving for $\mathrm{Cov}[{}^{W}\mathbf{p}_{WV'}]$, Equation 5.27 can be used to show the following:

$$\mathrm{Cov}[{}^{T}_{C'}\mathbf{q}, {}^{W}_{V'}\mathbf{q}] = ({}^{V'}_{C_0}R^{\mathrm{T}})\mathrm{Cov}[{}^{W}_{V'}\mathbf{q}] \tag{5.53}$$

In conjunction with Equations 5.22, 5.45 and 5.53, the covariance of Equation 5.51 can be determined:

$$\mathrm{Cov}[{}^{W}\mathbf{p}_{WV'}] = ({}^{V'}_{C_0}R)\mathrm{Cov}[{}^{T}\mathbf{p}_{TC'}]({}^{V'}_{C_0}R^{\mathrm{T}}) + A\mathrm{Cov}[{}^{W}_{V'}\mathbf{q}]A^{\mathrm{T}} + \bar{r}'\left(AB + (AB)^{\mathrm{T}}\right) \tag{5.54}$$

where

$$B = \begin{pmatrix} \mathrm{Cov}[{}^{T}_{C'}\theta, \psi'] & -\mathrm{Cov}[{}^{T}_{C'}\phi, \psi'] & 0 \\ \mathrm{Cov}[{}^{T}_{C'}\theta, \theta'] & -\mathrm{Cov}[{}^{T}_{C'}\phi, \theta'] & 0 \\ \mathrm{Cov}[{}^{T}_{C'}\theta, \phi'] & -\mathrm{Cov}[{}^{T}_{C'}\phi, \phi'] & 0 \end{pmatrix} \tag{5.55}$$

and the covariance terms in B are provided by the elements of the matrix in Equation 5.53.

To complete the state description, the mean and co-variance of the vehicle orientation with respect to the world frame are given in Equation 5.22.

5.3.4 Validation on Space Frame

Once the theoretical derivation had been completed, both the measurement error model and the kinematic model were implemented in real-time and tested on the Space Frame. The goal of this testing was to verify the correctness of the derivation and compare the predictions of the state estimator with real experimental data.

During each of several experimental runs, a mosaic was created as the camera followed a rectangular path around the workspace, in a plane parallel to the lab floor. A typical dead-reckoned mosaic from these tests is shown in Figure 5.8. The "vehicle" path started in the lower-left corner of the mosaic, traveled in a clockwise direction, and ended near the lower-left corner. Notice that while the alignment is generally quite good around the loop, the initial and final images are badly mis-aligned in the lower-left corner of the mosaic. This provides clear experimental evidence of the problem with dead reckoning for the purposes

of mapping, state estimation, and navigation. The re-alignment methods developed in this thesis to augment the state estimator are described in Chapters 6 and 7.

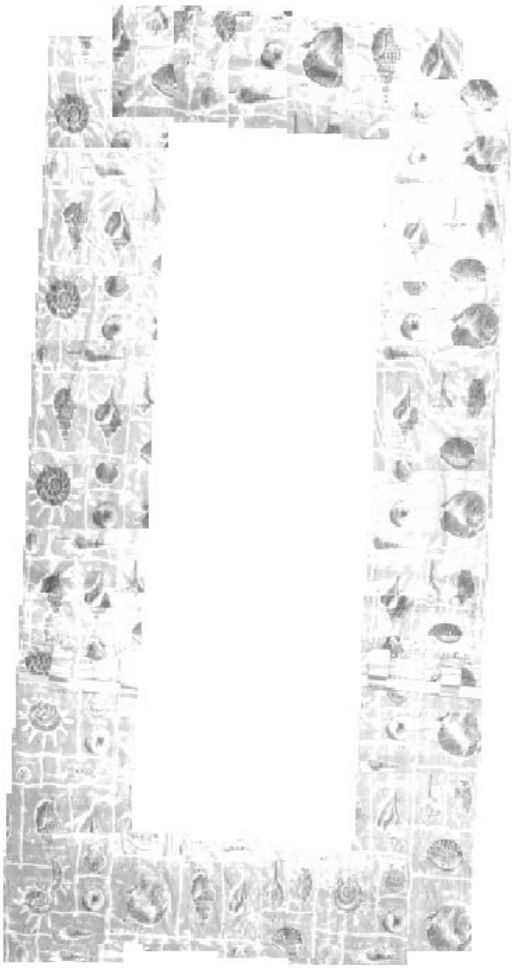

Figure 5.8: **Dead-Reckoned Mosaic**

This mosaic evolved in the clockwise direction, where both the initial and final images are located in the lower-left corner. The mis-alignment of these two images is a result of dead reckoning.

While this mosaic provides excellent qualitative visualization and verification of the mosaicking and state estimation process, the accuracy of the kinematic model is not immediately apparent. To provide a quantitative validation of the state estimator, truth

measurements of the camera position with respect to the lab floor were recorded while creating a mosaic, along with the state estimates. The experimental error was calculated and then compared to the errors predicted by the covariance estimates.

Figure 5.9 compares the actual errors on the x and y position estimates from a typical experimental run with the predicted standard deviation of these errors. Table 5.1 predicts that, for each of the x and y data sets, each data point will fall within the 1-σ error bound with a probability of about 68%. While no statistical prediction of the errors can be inferred from a single data run, it is evident that the growth of these errors over time (or more precisely, over distance traveled) follows the same general trend as the predicted error envelope. Furthermore, the data points fall within the error bound with roughly the predicted certainty.

Since the "vehicle" and camera centers coincide on the Space Frame, the V' frame is identical to the C' frame. Similarly, since the plane of the lab floor is perpendicular to the gravity vector, the global frames W and T coincide. Thus, the experiments represent a partially degenerate case where ${}^{T}\mathbf{p}_{TC'} = {}^{W}\mathbf{p}_{WV'}$. Furthermore, since the Space Frame does not currently have the capability to sense or control the orientation of the camera head, this portion of the theoretical equations could not be fully tested. However, qualitative testing on 6-DOF underwater robots in marine environments (Chapter 8) indicates that the complete equations provide accurate estimates of vehicle state.

5.4 Mosaic Model

In order to utilize real-time image and vehicle state and error estimates, subsequent estimation stages require a mathematical representation of the topology and measurements that comprise a video mosaic. The mosaic representation is created and updated by the state estimator stage whenever another snapshot is added to the mosaic. The later stages of estimation also update the mosaic representation, as discussed in later chapters.

The mosaic model is composed of two different parts, in order to represent the image positions and errors, respectively. These will be explained in the following two sections.

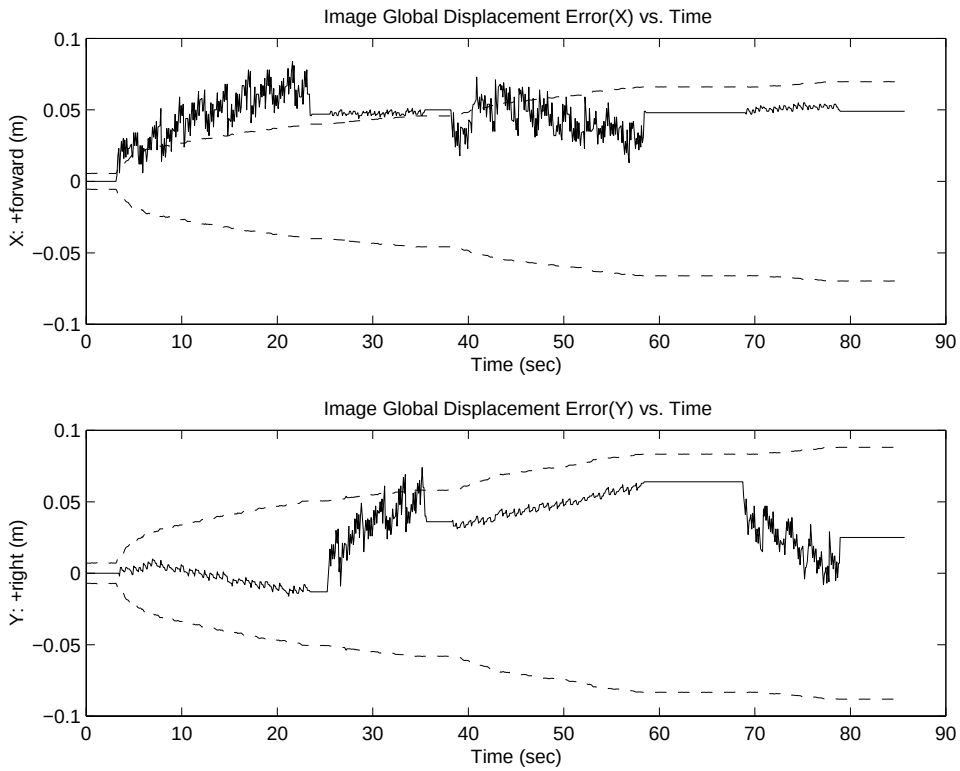

Figure 5.9: **Predicted and Actual Error Data for State Estimator**

The solid lines indicate the actual experimental errors in estimating the image global displacements, as measured by the Space Frame. The dashed lines are error envelopes that correspond to the predicted variances on the global displacement estimates.

5.4.1 Position Network

The representation of image positions within the mosaic takes the form of a node graph, or more precisely, a network with weights on each link. Each node in the network represents the location, ${}^{T}\mathbf{p}_{TI'}$, of a snapshot image in the mosaic. Every link is assigned a weight equal to the local displacement, ${}^{T}\mathbf{p}_{II'}$ between the two connected nodes. An example of a simple position network corresponding to a single-loop mosaic is given in Figure 5.10. The smoothing phase uses the position network to optimize globally the image displacement measurements within the mosaic (Chapter 7).

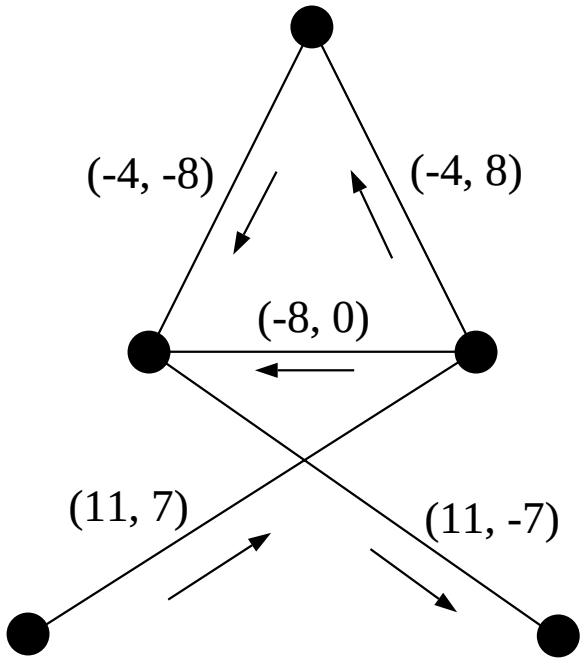

Figure 5.10: **Mosaic Model: Position Network**

This example models a single-loop mosaic. Unlike real mosaic models created with noisy displacement measurements, this model is self-consistent, since the distance measurements between the same two nodes along different paths are identical.

The initial node, n_0, is defined to be the origin of the mosaic.[12] Whenever a new snapshot is added to the mosaic, a new node is added to the network, and a new link connects this node with the node corresponding to the reference image. Up until this point, the newest image in the mosaic is always the reference image, such that the network representation is always a simple serial chain. However, more complex networks will be encountered in Chapter 6 when the next estimation stage takes advantages of crossover points in the vehicle path.

[12]This matches the previous statement that the origin of both the world and terrain frames coincide with the center of the initial image in the mosaic.

5.4.2 Error Network

In the same manner as for the mosaic position model, the alignment errors within a mosaic are stored in a network. In this type of network, each node represents the absolute error covariance of an image location relative to the origin of the mosaic, $\mathrm{Cov}[{}^{T}\mathbf{p}_{TI'}]$. A link is weighted with the covariance on the local displacement measurement, $\mathrm{Cov}[{}^{T}\mathbf{p}_{II'}]$, between the two connected nodes. An error network that could correspond to the position network in Figure 5.10 is given in Figure 5.11. The crossover detection phase utilizes the error network to detect inter-image overlap accurately (Chapter 6).

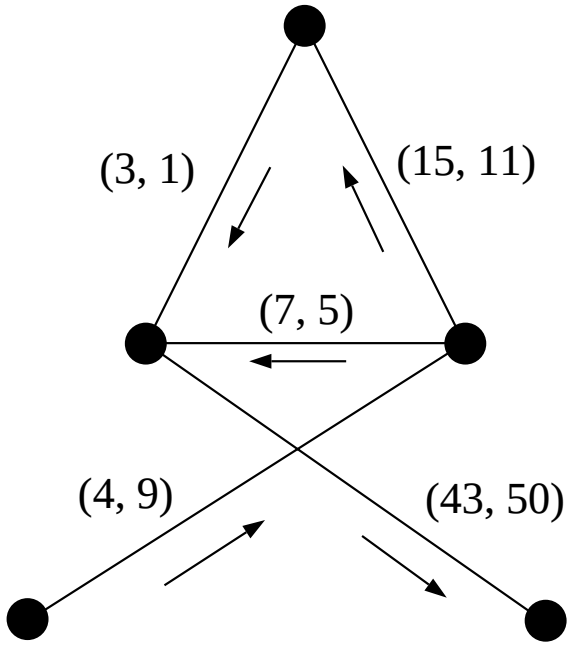

Figure 5.11: **Mosaic Model: Error Network**

This example could correspond to the position network of Figure 5.10. In practice, covariance estimates with these magnitudes would indicate that inconsistencies in the position network are likely.

The error network is created and updated in conjunction with the position network. Whenever a new snapshot image is added to the mosaic, a corresponding node and link is added to each of the two networks.

5.5 Summary

This chapter provided a detailed explanation of the state estimator component within the vision sensor. The goal of this first stage is to estimate the image and vehicle states and associated variances (Figure 2.5), and this was successfully demonstrated on the Space Frame (Figure 5.9). A detailed theoretical derivation was provided for the two sub-components of the state estimator, namely, the measurement error model and the kinematic model.

In addition to the state estimator, the internal representation of mosaics within the vision sensor was described. In particular, network representations for both mosaic position data and error data were described, for use in subsequent estimation stages.

Chapter 6

Crossover Detection and Correlation

6.1 Introduction

The crossover detection and correlation stage is the first step in re-aligning the mosaic map created by the state estimator. Map re-alignment is required to minimize internal inconsistencies in the mosaic as new, and conflicting, information is added to the mosaic model. The mosaic is comprised of a serial chain of overlapping images, and crossover points are simply the locations where the image chain loops back upon itself, such that non-adjacent images overlap. Due to the accumulation of image local alignment errors, the overlapping pair of images at the crossover point often exhibits poor alignment. The overlapping images at the crossover point can be re-aligned, and the change can be propagated throughout the mosaic using the smoothing techniques of Chapter 7.

Crossover points provide a unique opportunity to identify and fix inconsistencies in the mosaic map, so the crossover stage is divided into a detection phase and a correlation phase (Figure 6.1). Although the simplest solution would be to perform both detection and correlation at once by comparing each new image to the entire mosaic map, the required computations are prohibitively expensive for real-time implementation.

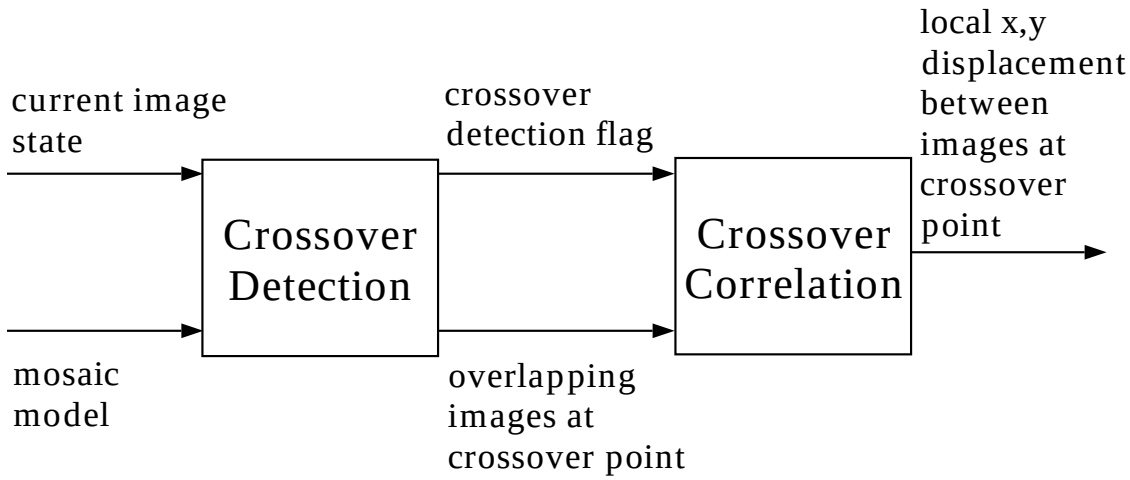

Figure 6.1: **Block Diagram for Crossover Detection and Correlation**

The crossover detection phase (Section 6.2) utilizes image state and variance information from the mosaic model to determine the probability that the new image overlaps each of the other images in the mosaic. Since this is essentially a displacement measurement with associated error bound, it is relatively low-cost. If a suitable pair of images is found (i.e. a crossover point has been detected), the pair is passed to the correlation phase.

The correlation phase (Section 6.3) attempts to fix the inconsistency at the crossover point by correlating the images received from the detection phase. A successful correlation indicates that the images actually do overlap, and the new image is re-aligned accordingly, irrespective of the effect this will have on other parts of the mosaic map. The correlation phase is only executed if the detection phase identifies a possible crossover point and corresponding pair.

6.2 Crossover Detection

In attempting to re-align overlapping images at crossover points in the image chain (Figure 2.3), one possible approach would be to correlate the current image with every other image in the mosaic to discover all potential image alignments. However, the crossover

stage must be performed in real-time, and the image registration process incurs a significant computational cost. Every image correspondence requires an entire cycle to perform the necessary computations, regardless of which two images are being compared. Thus, every attempt to correlate the current image with an image other than the reference image causes the sample rate to slip, destabilizing the mosaicking process.

To solve this problem, the crossover stage separates the task of detecting possible loops and performing the actual image correlation. The purpose of the crossover detection phase is to compare the location of the current image to the locations of all previous images and determine if there is any image overlap. Since this algorithm only compares relative locations of image centers, it is a low-cost computation that can be performed on all images in the evolving mosaic. If a suitable pair of images is found, these are passed to the crossover correlation phase for image registration.

During every loop through the estimation process, the crossover detection algorithm steps through the entire image chain to determine if a crossover correlation is possible. To reduce unnecessary attempts at crossover correlation further, an image selection heuristic has been developed that limits the possible image pairs that may even be considered for re-alignment. Before the relative displacement between a snapshot image and the current image is calculated, two conditions must be satisfied:

Minimum Time Between Detections This condition states that after a crossover correlation has been attempted, no further correlations may be attempted before a specified minimum time has elapsed. This prevents the possibility that a crossover correlation between two non-adjacent images could be performed in between every standard correlation between the current and reference images, thereby halving the effective sampling rate of the vision sensor. This is the first check performed by the detection algorithm, and if the condition is not satisfied, no furthering processing is done.

Minimum Separation Along Image Chain An image in the mosaic is only considered for crossover correlation if there are a specified minimum number of images between it and the current image along the image chain. Thus, as each image is considered, this

check is performed before any relative displacements are calculated. This condition prevents the possibility that the current image and reference image (and possibly more previous images) could be recommended by the crossover detection phase for correlation, even though they were just aligned during the standard correlation.

The details of the detection algorithm are described below. It is introduced first for the case of a vehicle path that only loops back upon itself once. Then, the more complex case of multiple loops in the vehicle path is examined.

6.2.1 Single Loop

To identify a candidate image pair for crossover correlation, the image chain is traversed, starting with the initial mosaic image and ending with the reference image. To test a particular image (Figure 6.2), provided that the image meets the conditions of the snapshot selection heuristic, a check is performed to determine the possibility of overlap with the current image. This check takes the form of several inequality constraints. These constraints will be derived in several steps, with each step taking into consideration an additional factor in the mosaicking and estimation process.

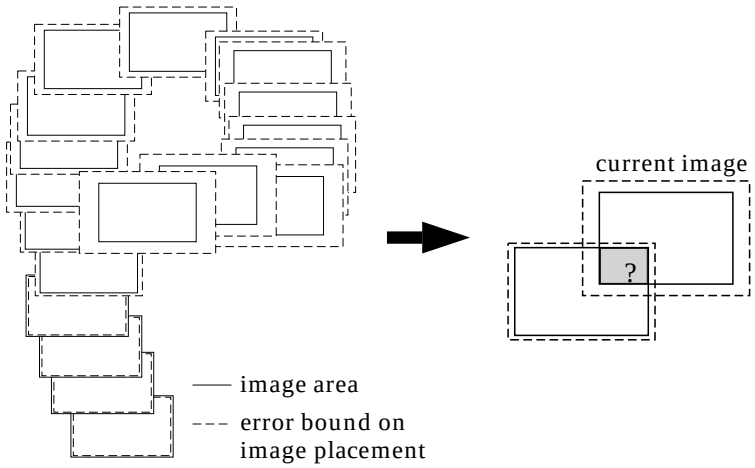

Figure 6.2: **Crossover Detection Process**

To determine if a crossover has occurred, the current image is checked against every other image to determine if there is a possible overlap.

First, the location of the current image center relative to the candidate image center is calculated by differencing the respective global positions. If this relative location falls within the bounds of the candidate image, the current image center overlaps with the candidate image and a crossover correlation should be attempted between these two images. Mathematically, these conditions can be stated as follows:

$$\begin{aligned} \text{If:} \quad & \left| {}^{T}x_{TI}(n) - {}^{T}x_{TI}(k) \right| < h/2 \text{ and} \\ & \left| {}^{T}y_{TI}(n) - {}^{T}y_{TI}(k) \right| < w/2 \\ \text{Then:} \quad & \text{correlate image}(k) \text{ and image}(n) \end{aligned} \tag{6.1}$$

However, this does not take into account that the entire correlation window (centered in the current image) must fall within the bounds of the reference image to ensure the possibility of correlation. If the correlation window is assumed to be square with size c:

$$\begin{aligned} \text{If:} \quad & \left| {}^{T}x_{TI}(n) - {}^{T}x_{TI}(k) \right| < h/2 - c \text{ and} \\ & \left| {}^{T}y_{TI}(n) - {}^{T}y_{TI}(k) \right| < w/2 - c \\ \text{Then:} \quad & \text{correlate image}(k) \text{ and image}(n) \end{aligned} \tag{6.2}$$

Furthermore, since each image center is given by a probabilistic position estimate, the detection algorithm must take this uncertainty in image location into account. In order to provide the most accurate test for overlap, the variances of the *relative* x and y displacements between the two image centers are used. For a single-loop mosaic, this relative variance can be determined by summing the local image displacement variances, ${}^{T}\mathbf{p}_{II'}$, along the image chain, from the candidate image to the current image.[1] Recall that the local image displacement measurements are Gaussian random variables, given the assumptions in Section 5.2.1 and the resultant state estimator computations of Chapter 5. Also,

[1]For the case of a single loop, it would be easier to just subtract the candidate image position variance from the current image position variance. However, the summation approach will be more useful when handling mosaics containing multiple loops.

recall that the normal distribution has the unique property that the sum of Gaussian random variables is also a Gaussian random variable, with variance equal to the sum of the individual variances. Thus, summing the variances is allowable in this situation. For the inequality constraints, these relative variances will be written simply as σ_x and σ_y.

The desired probability that the two images actually overlap can be set by choosing the proper size of the uncertainty ellipsoid. For instance, if the 1-σ error bound is chosen, there is about a 39% chance that the two images actually overlap if they satisfy the following inequality constraints (Table 5.1). The following equations assume that m standard deviations define the uncertainty ellipsoid, and they conservatively estimate the ellipsoid by using a bounding rectangle:

$$\begin{aligned} \text{If:} \quad & \left| {}^{T}x_{TI}(n) - {}^{T}x_{TI}(k) \right| < h/2 - c - m\sigma_x \text{ and} \\ & \left| {}^{T}y_{TI}(n) - {}^{T}y_{TI}(k) \right| < w/2 - c - m\sigma_y \\ \text{Then:} \quad & \text{correlate image}(k) \text{ and image}(n) \end{aligned} \tag{6.3}$$

Thus, if a box centered in the current image and inflated to compensate for the correlation window size and position uncertainty falls within the bounds of the reference image, a crossover correlation is attempted. In addition to assigning an overlap probability to the image pair, the uncertainty ellipsoid provides the search region for the crossover correlation. The center of the search region is located in the reference image at:

$$\begin{aligned} x_{sr} &= {}^{T}x_{TI}(n) - {}^{T}x_{TI}(k) \\ y_{sr} &= {}^{T}y_{TI}(n) - {}^{T}y_{TI}(k) \end{aligned} \tag{6.4}$$

and has the following width and height:

$$\begin{aligned} w_{sr} &= 2m\sigma_x \\ h_{sr} &= 2m\sigma_y \end{aligned} \tag{6.5}$$

In traversing the image chain from start to finish, the first candidate image that satisfies these inequality constraints is passed to the crossover correlation phase, along with the recommended search region. This is because images closer to the origin (i.e. initial image) provide a greater benefit to the crossover scheme. They generally have more accurate position estimates, since the accumulation of errors due to dead reckoning increases with distance traveled.

6.2.2 Multiple Loops

For the more general (and practical) case of vehicle paths containing multiple loops, the same crossover detection algorithm can be utilized, with one significant addition. In the case of a single-loop mosaic, the relative displacement variance between two images was calculated by summing the local displacement variances between them along the image chain. However, while this technique would provide a conservative estimate for multiple loops, a more accurate method would consider the different measurement paths that could be taken between the two images (Figure 6.3).

The ideal method would find the minimum relative variance between the current image and each candidate image. This could be achieved using the mosaic error network of Section 5.4.2, by finding the minimum-length path from the current image to each candidate image, and summing the link variances along this minimum path. There exists a technique known as Dijkstra's algorithm [29] that can accomplish exactly this task.

Figure 6.4 presents a simple 1-D[2] example of a multiple-loop mosaic error network having N nodes, that will be used to demonstrate Dijkstra's algorithm. The goal is to start at the current image, known as the source node s, and find the minimum length to all other nodes. The object of Dijkstra's algorithm is to start at the source node and settle the minimum path to an additional node at every time step. Thus, at every step k, there are k settled nodes (i.e. for which the minimum path from the source node has been determined) and $(n-k)$ unsettled nodes. The objective at each step is to find the path lengths from every

[2]For the 2-D case of x, y position, link A is considered to be less than link B if both the x and y variances of A are less than the corresponding variances of B.

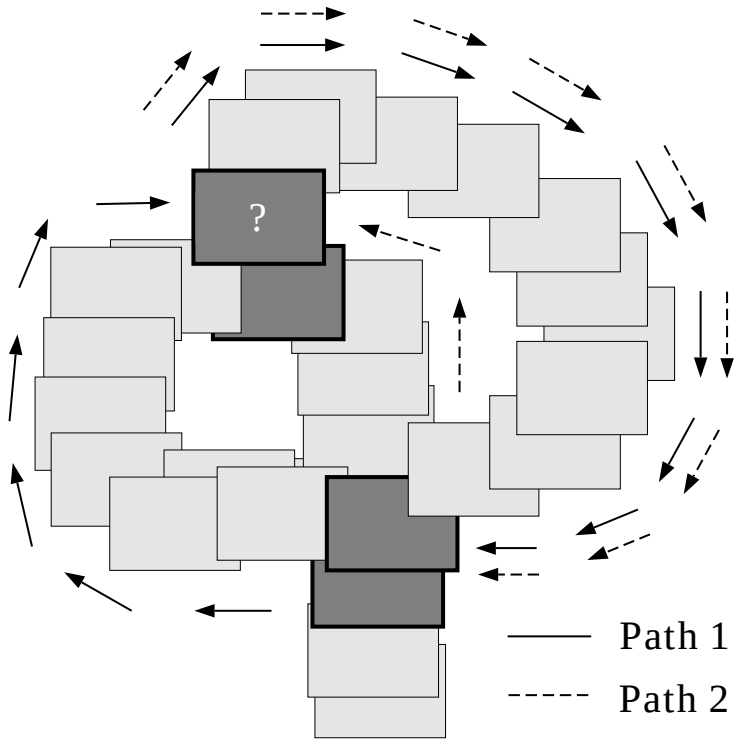

Figure 6.3: **Crossover Detection For Multiple Loops**

The most efficient method for detecting crossovers utilizes the minimum variance on the relative displacement between the two candidate images. To determine the minimum variance, the relative displacement variances along every possible path between the two candidate images must be calculated.

settled node to every unsettled node that is one link away, and then choose the minimum path:

$$\text{Minimize } (d_i + l_{ij}) \text{ at every step } k \tag{6.6}$$

where i represents a settled node, j represents a new node, d_i is the minimum distance from the source node to the settled node i, and l_{ij} is the link length from the node i to the node j. For instance, at step 1, the source node is the only settled node, and the next settled node is chosen to be the one connected to the source node with the smallest link. Thus, in the example of Figure 6.4, node 6 would be settled, with a minimum path of 10.

To reduce redundant computation, the minimum path lengths can be stored in a table and updated at every step in the algorithm, as shown in Table 6.1. Initially, all distances

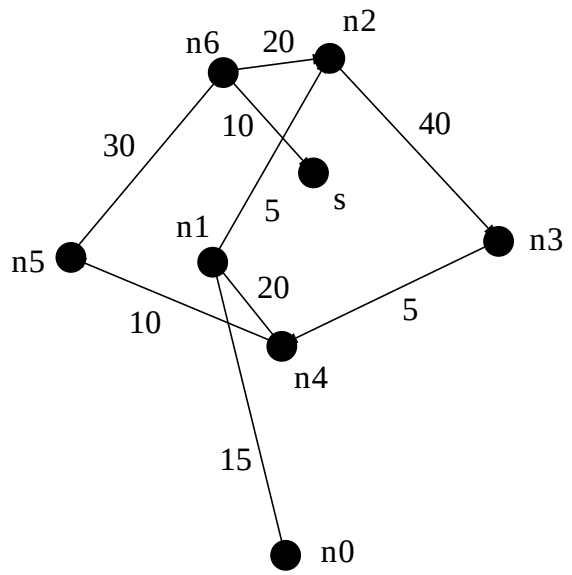

Figure 6.4: **Example of Dijkstra's Algorithm**

This 1-D example provides an error network for a mosaic containing two crossover points. Dijkstra's algorithm enables calculation of the minimum relative variance between the current image (s) and any other image.

are set to infinity. At every step, the distances are calculated to every new node that is directly connected to the *most recently settled node*. Each distance is then compared to the current minimum distance to the relevant node, and if it is smaller, it replaces the current value as the new minimum. Finally, the smallest distance of the remaining nodes is chosen to be the new settled node, as indicated by the boxes in Table 6.1. The dots indicate that once a node is settled, it can be ignored for the remaining steps in the algorithm. Step 4 can be used as an example: after Step 3, node 1 was the most recently settled node. The only two unsettled nodes connected to node 1 are nodes 0 and 4 (from Figure 6.4); the distances to these nodes through node 1 are computed as $35 + 15 = 50$ and $35 + 20 = 55$, respectively. Both of these distances are smaller than the current distances in columns 0 and 4 in the table, so they replace those values. Then, the smallest distance is chosen out of row 4 to be the next settled node: node 5, with a minimum distance of 40. This process continues until all nodes have been settled, and the minimum distances to every node from the source node have been determined. The last row in Table 6.1 indicates these final distances. The order

	Minimum Distances						
Step	d_0	d_1	d_2	d_3	d_4	d_5	d_6
1	∞	∞	∞	∞	∞	∞	$\boxed{10}$
2	∞	∞	$\boxed{30}$	∞	∞	40	$\cdot$
3	∞	$\boxed{35}$	$\cdot$	70	∞	40	$\cdot$
4	50	$\cdot$	$\cdot$	70	55	$\boxed{40}$	$\cdot$
5	$\boxed{50}$	$\cdot$	$\cdot$	70	50	$\cdot$	$\cdot$
6	$\cdot$	$\cdot$	$\cdot$	70	$\boxed{50}$	$\cdot$	$\cdot$
7	$\cdot$	$\cdot$	$\cdot$	$\boxed{55}$	$\cdot$	$\cdot$	$\cdot$
	50	35	30	55	50	40	10

Table 6.1: **Minimum Distances**

Iterations of Dijkstra's algorithm for the example network in Figure 6.4

of computations required to find these minimum distances is $O(N^2)$, where N is equal to the number of nodes in the mosaic (not counting the source node).

By modifying the crossover detection algorithm to use Dijkstra's algorithm to find the minimum relative variances, it is now capable of handling multiple-loop vehicle paths of arbitrary complexity.

6.3 Crossover Correlation

The crossover correlation phase is only executed if the crossover detection algorithm has recommended an image for correlation with the live image. If this is the case, the crossover correlation simply performs the same image registration as the one used to align the current image with the reference image, as explained in Section 3.4. This is accomplished by modifying the image processing pipeline in real-time. If the output of the image correspondence is a valid measurement, it is transformed to global coordinates using the state estimator equations. This new image displacement measurement is then used to update both the current vehicle position estimate and the mosaic model.

6.3.1 Modification of Image Processing Pipeline

In anticipation of performing a crossover correlation, two changes must be made to the image processing pipeline discussed in Section 3.4.2. Recall that the pipeline is depicted graphically in Figure 3.1. First, a snapshot is taken and added to the mosaic. This assures that, if the crossover correlation is successful, the resulting measurement will represent the local displacement between two images that actually exist in the mosaic.[3]

Next, the candidate image from the detection phase is retrieved from the image buffer and becomes the new reference image. The image registration is then performed, which outputs the relative displacement between the two images and the measurement confidence value. Upon completion of the image registration, the latest snapshot is retrieved from the image buffer, a new live image is digitized, and the standard mosaicking process continues.

If the measurement confidence value of the crossover correlation is below a specified threshold, the data are considered invalid and the attempt at crossover correlation is completely ignored. If the confidence is above the threshold, the valid data are used to perform two different updates, as explained in the following sections.

6.3.2 Global Position Update

Upon successfully completing a crossover correlation, the new data are used to update the live image position estimate, which propagates through the state estimator equations and updates the current vehicle position estimate. Prior to performing the crossover correlation, the live image position was estimated by adding the most recent displacement data from image correspondence to the estimate of reference image position. After the crossover correlation, a new estimate of live image position can be calculated by adding the crossover displacement data to the estimated position of the crossover image.

Each of these two live image position estimates has an associated error variance. After successful crossover correlation, these two variances are compared, and the estimate with the

[3]Recall that image correspondence is performed at rates up to 30 Hz on the incoming live images, while images are only added to the mosaic when deemed necessary by the mosaicking algorithm(s) (Section 3.4.3).

lower variance is chosen to be the new estimate of live image position. Even if the successful crossover correlation does not provide a superior estimate, it is not ignored entirely. While no difference is evident after completion of the crossover detection and correlation stage, the crossover estimate is used during the smoothing stage to optimize the image alignment within the mosaic, which ultimately affects future vehicle position estimates.

6.3.3 Mosaic Model Update

A successful crossover correlation is also used to update the internal mosaic data representation, namely the mosaic position and error networks. First of all, a new node is automatically added by the standard mosaicking process, since a new snapshot was taken before attempting the crossover correlation. Second, if the correlation data are valid, a link is added between the candidate crossover node and the new snapshot node. Depending on whether the position or error network is being updated, the new link is weighted with the crossover displacement or variance, respectively.

6.4 Validation on Space Frame

To verify the effectiveness of crossover detection and correlation for the real-time improvement of vehicle state estimates, the algorithms were implemented and tested on the Space Frame in the same manner as for the state estimator. In order to show the improvements obtained, a mosaic containing a single loop and the associated experimental data are presented in this section. In addition, the crossover detection and correlation algorithm was tested fully on mosaics with multiple loops. These tests verified that the crossover stage of the vision sensor is fully capable of handling arbitrarily complex vehicle paths. Since the multiple-loop results provide no additional quantitative validation and the effects of a single loop is easier to visualize, only the data from the single-loop vehicle path are presented.

Figure 6.5 depicts a single-loop mosaic after successful crossover correlation. This mosaic was created while the camera followed the same rectangular path as before, in a plane parallel to the lab floor. The camera started in the lower left corner, traveled clockwise,

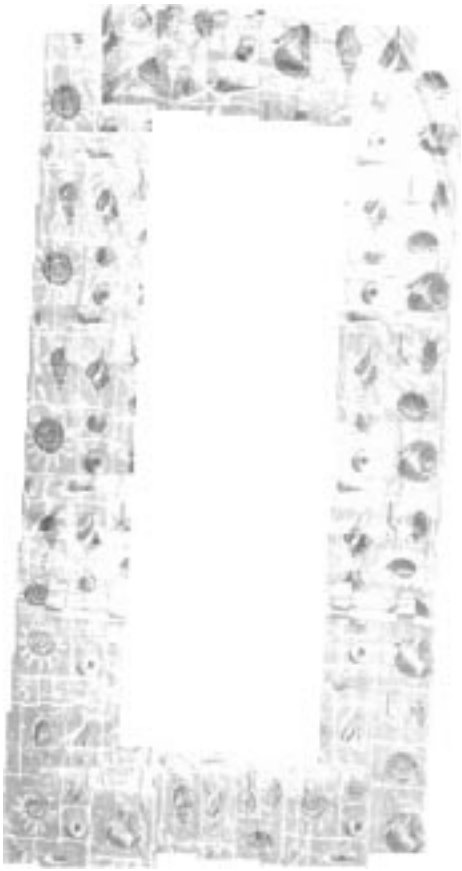

Figure 6.5: **Crossover-Aligned Mosaic**

The initial and final images in the lower-left corner of this mosaic have been re-aligned by the crossover detection and correlation stage. However, this re-alignment has caused mis-alignment between the final image and the previous image in the image chain.

and ended in the same corner. In contrast to Figure 5.8, the initial and final images in the lower left corner align quite well. This re-alignment is due to a successful crossover correlation between these two images. However, while this improves the current and future position estimates, compare the final image with the previous image in the chain. In essence, the misalignment has been shifted from images 0 and n to images n and $(n-1)$. The purpose of the smoothing stage (Chapter 7) is to fix this by re-aligning the entire mosaic.

Quantitatively, it can be seen in Figure 6.6 that the crossover algorithm has achieved its intended task. As the camera is completing its rectangular trajectory after about 95

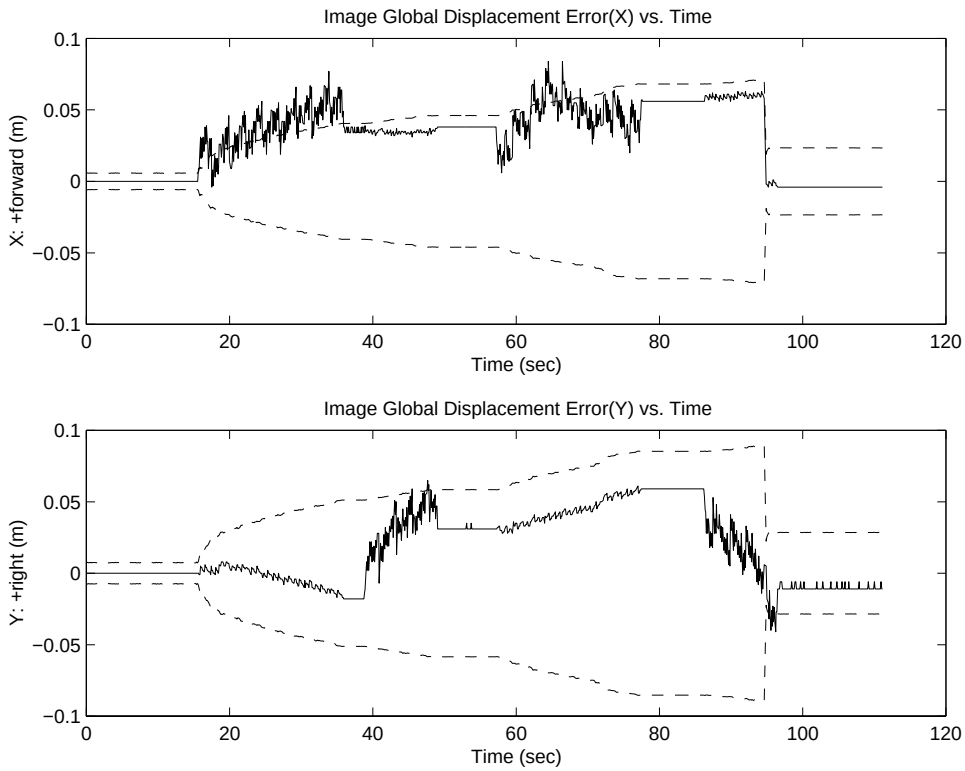

Figure 6.6: **Predicted and Actual Error Data after Crossover**

The solid lines indicate the actual experimental errors in estimating the image global displacements, as measured by the Space Frame. The dashed lines are error envelopes that correspond to the predicted variances on the global displacement estimates.

seconds, the crossover algorithm detects a possible overlap between the initial and final images, attempts a correlation, and produces a valid displacement measurement between these two images. The dashed line on the plot indicates the predicted reduction in global measurement variance, and the solid line presents the experimental error measurements. The experimental data follow the predicted trend and stay roughly within the 1-σ error bound, producing a marked improvement in future position estimates.

Comparing Figure 6.6 to Figure 5.9 further demonstrates the advantage of the crossover method over simple dead reckoning techniques. By taking advantage of the loop in the mosaic, especially one that crosses so near to the origin (i.e. the initial image), the crossover algorithm results in a pronounced reduction in the estimation error. If this path were to

continue, the errors would continue to grow in dead-reckoned fashion, until another loop was encountered. At that point, another successful crossover correlation would reset the dead-reckoned error to a much lower value, depending on the point of crossover and the variance of the crossover measurement.

6.5 Summary

This chapter presented a method to improve the current global vehicle position estimate and mosaic alignment when loops in the vehicle path are encountered. This crossover method is the second estimation stage of the vision sensor, and its goal is to reduce the estimation errors that occur when dead reckoning, as depicted in Figure 2.5. This method was demonstrated successfully on the Space Frame, and the results in Figure 6.6 verified the intended improvement over state estimation through dead reckoning.

The crossover method was accomplished in two phases: crossover detection and crossover correlation. This chapter described the reasons for approaching the problem in this manner, and it provided the technical details of the algorithms underlying the detection and correlation phases.

Chapter 7

Smoothing

7.1 Introduction

The role of the smoothing stage is to re-align the mosaic map using data from the mosaic model, including the crossover data. Optimal estimation techniques are utilized to minimize the variances on the image global positions, thereby improving the internal consistency of the mosaic map. Specifically, *optimal* refers to the maximum likelihood (or minimum variance) estimate of image positions. The core concept of the smoothing process is to take advantage of crossover points in the image chain, and it is illustrated graphically in Figure 7.1. To accomplish this, the smoother utilizes measurements from the previous two stages and outputs an optimally re-aligned mosaic (Figure 7.2).

In devising an algorithm to perform this optimal estimation for map re-alignment, there are several issues that must be considered. Since one of the goals of this stage is to perform

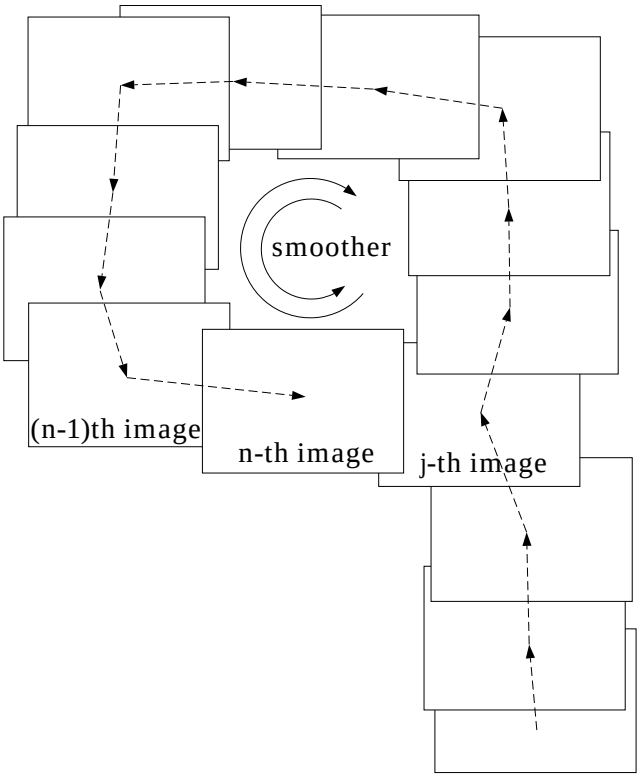

Figure 7.1: **Error Reduction in Image Chain**

By re-aligning the overlapping images when the image chain loops back upon itself and propagating the re-alignment around the loop, the errors in absolute image alignment are reduced. This improves the internal consistency of the mosaic map.

the optimization online,[1] the computation time of the chosen algorithm is important. Because of their simplicity in both concept and implementation, batch algorithms are considered in Section 7.2. To improve efficiency, several sequential algorithms have been developed based on their batch counterparts (Section 7.3).

The presence or absence of a dynamic vehicle model significantly affects the choice of algorithm. While no dynamic models exist for any of the experimental vehicles used in this research, the advantages of a model and methods to incorporate it into the re-alignment

[1]By definition, real-time performance is impossible for this stage. Since optimal estimation utilizes all available measurements to re-align the mosaic map, the position estimate of a given image depends on both past and future measurements. Thus, the goal is to finish the computation as soon as possible after the relevant data have been collected, so that the results may be used to improve performance during the same mapping/navigation run.

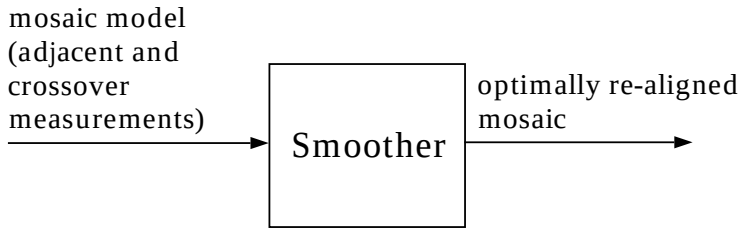

Figure 7.2: **Global State Estimation: Smoothing**

process are considered. Both batch and sequential algorithms have been developed to take advantage of the existence of a dynamic model.

A final consideration is the complexity of the image chain comprising the mosaic map, namely, whether it contains a single loop or multiple loops. Optimization methods are presented only for the more complex case of multiple loops in the image chain. While a derivation for the single-loop case often provides physical intuition into the algorithmic computations, multiple-loop derivations are much more practical for real-world experimentation and contain the single-loop case as a natural subset. Furthermore, the algorithms for single-loop image chains do not exhibit any significant simplifications, except for the degenerate case where the crossover occurs at the origin.

7.2 Batch Methods

This section provides the technical details on two batch methods for computing optimal estimates of image placement within the mosaic. The first method provides a solution for the case where no dynamic model is available for the vehicle, while the second method utilizes the presence of a dynamic model to full advantage. In addition, the order of computations required is calculated for each algorithm, for later comparison with similar sequential methods.

7.2.1 No Dynamic Model

For the case where no dynamic model of the robot is present, it is fairly straightforward to determine a batch algorithm for optimal mosaic re-alignment. As an aid to understanding this algorithm, consider again the mosaic position network of Section 5.4.1. Any mosaic can be viewed as a collection of link measurements connecting the image nodes. The goal of this algorithm is to determine the node locations using the link measurements.

This task is identical to the problem of finding an optimal estimate of the vector x given a vector of measurements z related to x as follows:

$$\mathbf{z} = C\mathbf{x} + \mathbf{v} \tag{7.1}$$

where $\mathbf{v}$ is a zero-mean Gaussian random vector representing the measurement error:

$$\mathbf{v} = \mathcal{N}[\mathbf{0}, V] \tag{7.2}$$

and a prior estimate of $\mathbf{x}$ is given:

$$\mathbf{x} = \mathcal{N}[\bar{\mathbf{x}}, \bar{P}] \tag{7.3}$$

For the case of the mosaic network, $\mathbf{z}$ is a vector of the link measurements, both for adjacent nodal displacements and for crossover nodal displacements:[2]

$$\mathbf{z} = \begin{pmatrix} \vdots \\ \mathbf{z_m} \\ \vdots \end{pmatrix} \text{ for } m = 0, \ldots, (M-1) \tag{7.4}$$

where M is the total number of links and $\mathbf{z_m}$ represents the 2-D local displacement measurement from an image node i_m to an image node j_m. Thus, for every link measurement, there is a mapping $[m \rightarrow (i_m, j_m)]$ based on the network geometry. Since these measurements are

[2]For the sake of simplicity, the former frame notation has been dropped for the discussion of optimization techniques in this chapter. Where there is little danger of confusion, simple variables are used to represents states, measurements, etc.

independent from each other, the covariance matrix is diagonal:

$$V = \begin{pmatrix} \ddots & & 0 \\ & V_m & \\ 0 & & \ddots \end{pmatrix} \text{ for } m = 0, \ldots, (M-1) \tag{7.5}$$

Similarly, $\mathbf{x}$ is a vector of the image node locations:

$$\mathbf{x} = \begin{pmatrix} \vdots \\ \mathbf{x_n} \\ \vdots \end{pmatrix} \text{ for } n = 1, \ldots, (N-1) \tag{7.6}$$

where N is the total number of nodes, and $\mathbf{x_n}$ represents the 2-D position of node n. Since the initial node is defined to be the origin, $\mathbf{x_0} \equiv 0$, it is not included in the position vector. The matrix C reflects the relationship between $\mathbf{z}$ and $\mathbf{x}$. For every measurement $\mathbf{z_m}$, the corresponding row in the C matrix has an identity sub-matrix(I) in the j_m^{th} column and a negative identity sub-matrix($-I$) in the i_m^{th} column. Since $\mathbf{x_0} \equiv 0$, if $i_m = 0$ or $j_m = 0$, then only the I or $-I$ entry, respectively, is present.

The maximum likelihood estimate of $\mathbf{x}$ can be derived using Bayes' rule to find $p(\mathbf{x}|\mathbf{z})$, the probability of $\mathbf{x}$ given the measurement vector $\mathbf{z}$ [2, Ch. 3, p. 2]. In determining this normal probability distribution, it is found that:

$$p(\mathbf{x}|\mathbf{z}) \sim e^{-J} \tag{7.7}$$

where:

$$J = \frac{1}{2}(\mathbf{z} - C\mathbf{x})^{\mathrm{T}} V^{-1} (\mathbf{z} - C\mathbf{x}) + \frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^{\mathrm{T}} \bar{P}^{-1} (\mathbf{x} - \bar{\mathbf{x}}) \tag{7.8}$$

Thus, J becomes the performance metric for optimality, so the goal is to minimize J with respect to x. This is accomplished by the following measurement update equations, where $\hat{\mathbf{x}}$ is the maximum likelihood estimate of $\mathbf{x}$ that minimizes J, $\hat{P}$ is the corresponding minimum

variance, and no independent prior estimate of $\mathbf{x}$ is given:

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + K(\mathbf{z} - C\bar{\mathbf{x}}) = K\mathbf{z} \tag{7.9}$$

$$K = \hat{P}C^{\mathrm{T}}V^{-1} \tag{7.10}$$

$$\hat{P}^{-1} = \bar{P}^{-1} + C^{\mathrm{T}}V^{-1}C = C^{\mathrm{T}}V^{-1}C \tag{7.11}$$

Equations 7.9–7.11 provide an optimal estimate for the location of every image in the mosaic. The order of operations required to complete this algorithm is $O(N^3)$, where N equals the number of nodes in the network. Given the fact that new nodes are only added whenever a new image snapshot is taken for the mosaic (e.g. every couple seconds), this algorithm can achieve online performance for relatively large mosaics. During experimentation, mosaics approaching 200 images in size with on the order of 10 crossover correlations were smoothed online using this algorithm in a few seconds.

7.2.2 Dynamic Model

If a dynamic model of the robot is available, it can be used to advantage and incorporated into an optimization algorithm. For this section, the presence of a linear dynamic model of the following form will be assumed:

$$\mathbf{x}(k+1) = \Phi(k)\mathbf{x}(k) +, (k)\mathbf{u}(k) \text{ for } k = 0, \ldots, (K-1) \tag{7.12}$$

$$\mathbf{u}(k) = \mathcal{N}[\bar{\mathbf{u}}(k), U(k)] \tag{7.13}$$

where K is the number of time steps. The state vector $\mathbf{x}$ has been re-defined for this algorithm; it no longer refers to the position and orientation of an image or vehicle. Instead, $\mathbf{x}$ may contain any variables needed by the dynamic vehicle model. For instance, if the vehicle is described by a second order dynamic model, the state vector typically would contain the position and velocity components of the vehicle.

To include the dynamic model, a new metric must be chosen that penalizes both measurement errors and process noise (i.e. errors in the dynamic model). Also, it is now

advantageous to utilize the local displacement measurements at every time step, rather than only at times when a new snapshot is taken. Since the crossover measurements are still taken asynchronously with respect to the discrete dynamic model, the following performance index will be used:

$$\begin{aligned} J &= \frac{1}{2}[\mathbf{x}_0 - \mathbf{x}(0)]^{\mathrm{T}} S_0 [\mathbf{x}_0 - \mathbf{x}(0)] \\ &+ \frac{1}{2}\sum_{c=0}^{C-1}[\mathbf{d}_c - (\mathbf{x}(t_c) - \mathbf{x}(h_c))]^{\mathrm{T}} T_c [\mathbf{d}_c - (\mathbf{x}(t_c) - \mathbf{x}(h_c))] \\ &+ \frac{1}{2}\sum_{k=0}^{K}\mathbf{v}^{\mathrm{T}}(k)Q(k)\mathbf{v}(k) + \frac{1}{2}\sum_{k=0}^{K-1}\mathbf{u}^{\mathrm{T}}(k)R(k)\mathbf{u}(k) \end{aligned} \tag{7.14}$$

subject to the constraints of Equation 7.12–7.13 and:

$$\mathbf{y}(k) = \mathbf{x}(k) - \mathbf{x}(r_k) + \mathbf{v}(k) \text{ for } k = 0, \ldots, K \tag{7.15}$$

$$\mathbf{v}(k) = \mathcal{N}[0, V(k)] \tag{7.16}$$

In the above equations, C is the number of crossover measurements, where each $\mathbf{d}_c$ is the crossover measurement vector with the head starting at the center of the snapshot image taken at time h_c, and a tail ending at the center of the snapshot image taken at time t_c. Thus, for every crossover measurement, there is a mapping $[c \rightarrow (h_c, t_c)]$. The W_c matrices are the covariances of the crossover measurements $\mathbf{d}_c$. The vector $\mathbf{y}(k)$ represents the adjacent displacement measurement taken at time step k, where r_k is the time when the reference image for that particular measurement was taken. These vision-based measurements are taken at the sensor sample rate (10–30 Hz). The P_0 matrix is the covariance on the initial condition estimate $\mathbf{x}_0$. The information matrices Q, R, S, and T are defined as follows: $Q(k) = V^{-1}(k)$, $R(k) = U^{-1}(k)$, $S_0 = {P_0}^{-1}$, and $T_c = {W_c}^{-1}$.

The goal of this batch algorithm is to minimize the performance index J with respect to $\mathbf{x}(0)$ and $\mathbf{u}(k)$ for $k = 0 \ldots (K-1)$.[3] This can be achieved in a single computation,

[3] $\mathbf{x}(k)$ for $k = 0 \ldots (K-1)$ can be computed using $\mathbf{x}(0)$, $\mathbf{u}(k)$, and the dynamic equation of motion, Equation 7.12.

Weights	Terms
S_0	$\mathbf{x}_0 - \mathbf{x}(0)$
T_c	$\mathbf{d}_c - [\Phi(t_c, h_c+1) - I]\Phi(h_c, 0)\mathbf{x}(0)$ $- [\Phi(t_c, h_c+1) - I] \sum_{j=0}^{h_c-1} \Phi(h_c, j+1),\ (j)\mathbf{u}(j)$ $- \sum_{j=h_c}^{t_c-1} \Phi(t_c, j+1),\ (j)\mathbf{u}(j)$ for $c = 0, \ldots, (C-1)$
$Q(k)$	$\mathbf{y}(k) - [\Phi(k, r_k+1) - I]\Phi(r_k, 0)\mathbf{x}(0)$ $- [\Phi(k, r_k+1) - I] \sum_{j=0}^{r_k-1} \Phi(r_k, j+1),\ (j)\mathbf{u}(j)$ $- \sum_{j=r_k}^{k-1} \Phi(k, j+1),\ (j)\mathbf{u}(j)$ for $k = 0, \ldots, K$
$R(k)$	$\mathbf{u}(k)$ for $k = 0, \ldots, (K-1)$

Table 7.1: **Terms of Performance Index**

with the proper reformulation of the problem [2]. First, every term of Equation 7.14 can be written in terms of the dependent random variables $\mathbf{x}(0)$ and $\mathbf{u}(k)$ and the measurements $\mathbf{x}_0$, $\mathbf{d}_c$, and $\mathbf{y}(k)$. Specifically, Equation 7.12 can be propagated forward to specify $\mathbf{x}(k)$ in terms of $\mathbf{x}(0)$ and $\mathbf{u}(k)$, and the constraint equations 7.12 and 7.15 can be substituted into Equation 7.14. For this purpose, the transition matrix from the ith step to the jth step, $\Phi(j, i)$, is defined as follows:

$$\Phi(j, i) = \Phi(j)\Phi(j-1)\cdots\Phi(i) \tag{7.17}$$

such that $\Phi(j+1, j) = \Phi(j)$ and $\Phi(j, j) = 0$. The results of reformulating each of the four terms in Equation 7.14 are shown in Table 7.1.

Next, Equation 7.14 can be written in the following form:

$$J = [\mathbf{z}_{bat} - C_{bat}x_{bat}]^{\mathrm{T}} S_{bat} [\mathbf{z}_{bat} - C_{bat}x_{bat}] \tag{7.18}$$

$$\mathbf{z}_{bat} = \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{d}_c \\ \mathbf{y}(k) \\ 0 \end{pmatrix} \text{ for } c = 0, \ldots, (C-1) \text{ and } k = 0, \ldots, K \tag{7.19}$$

$$\mathbf{x}_{bat} = \begin{pmatrix} \mathbf{x}(0) \\ \mathbf{u}(k) \end{pmatrix} \text{ for } k = 0, \ldots, (K-1) \tag{7.20}$$

$$S_{bat} = \begin{pmatrix} S_0 & & & 0 \\ & T_c & & \\ & & Q(j) & \\ 0 & & & R(k) \end{pmatrix}$$
$$\text{for } c = 0, \ldots, C,\ j = 0, \ldots, K, \text{ and } k = 0, \ldots, (K-1) \tag{7.21}$$

and C_{bat} is constructed from the terms in Table 7.1.

By comparing Equation 7.18 with Equation 7.8, it is seen that the new batch formulation is identical to a single static estimation problem, with no prior estimate given. From Equations 7.9–7.11, the optimal estimate of $\mathbf{x}_{bat}$ and the corresponding covariance matrix X_{bat} are:

$$\mathbf{x}_{bat} = X_{bat} {C_{bat}}^{\mathrm{T}} S_{bat} \mathbf{z}_{bat} \tag{7.22}$$

$$X_{bat} = \left[{C_{bat}}^{\mathrm{T}} S_{bat} C_{bat} \right]^{-1} \tag{7.23}$$

This algorithm will provide an optimal estimate for every point along the vehicle path, taking into account both the dynamic model and all measurements (including crossover measurements). However, the unnecessary calculation of intermediate vehicle path points between image locations results in a significant computational cost. The order of computations for this algorithm is $O((C+K)^3)$. Since the number of time steps is almost always orders of magnitude larger than the number of crossover measurements, this is approximately $O(K^3)$. This result can be compared to the batch algorithm that does not use a dynamic model, if one makes the conservative assumption that a new snapshot image is taken approximately once per second for a sample rate of 10 Hz. In addition, if the vehicle state consists of three position components and three velocity components for a typical second order model, this is three times larger than the 2-D position vector of Section 7.2.1. Therefore, this algorithm's computational cost is equivalent to $O((30N)^3) = 27,000 \cdot O(N^3)$.

In other words, given identical vehicle paths and processing power, the algorithm including the dynamic model takes between 4 and 5 orders of magnitude longer to complete. While this algorithm would be excellent for post-processing, its computational cost would make it extremely difficult to achieve online performance.

7.3 Sequential Methods

In an attempt to improve the performance of the batch optimal estimation algorithms, three sequential algorithms are presented in this section. Two of the sequential algorithms are intended for the case where a dynamic vehicle model is absent, and the remaining sequential algorithm includes the presence of a dynamic vehicle model. The following sections describe the algorithms in detail and calculate the improved order of computations. In addition, the limitations of these sequential algorithms are discussed.

7.3.1 No Dynamic Model

Two different sequential algorithms to solve the static estimation problem are presented. The first one takes a unique approach by viewing the estimation process as a weighted average of a series of paths to each individual node. The second one uses the corresponding batch algorithm as a starting point and modifies it so that each successive optimization utilizes the results of the previous optimization to improve the efficiency of the algorithm.

Path-Based Method

For the case of the batch algorithm of Section 7.2.1, all nodes were optimized simultaneously in a single static estimation. In an effort to improve the order of computations, a sequential algorithm can be developed that optimizes the location of each node in the mosaic network individually. The key to this approach is to recognize that the optimal estimate of a nodal position relies on the consideration of all possible paths (that do not contain internal loops) from the origin node to the goal node. Assuming that all possible non-looping paths from the origin are known for a given node n, a static estimation can be setup that conforms to

Equations 7.1–7.3 from the batch case. The difference lies in the state and measurement vector definitions. For the sequential case, $\mathbf{x}$ represents the 2-D position of a single node n. $\mathbf{z}$ is a vector of measurements of $\mathbf{x}$ made along every possible path from node 0 to node n. Since no prior estimate of $\mathbf{x}$ is available, the solution to this static estimation problem is entirely similar to the batch case, and it is given in Equations 7.9–7.11. Whenever a new measurement is taken (i.e. a new link is added to the network), this static estimation procedure is performed for every node that has acquired a new path through the new link in the network. For instance, if another sequential image is added the chain, only this new node location needs to be estimated. For a crossover measurement, many nodes will have to be re-optimized. A method to keep track of which nodes need optimization is described later in this section.

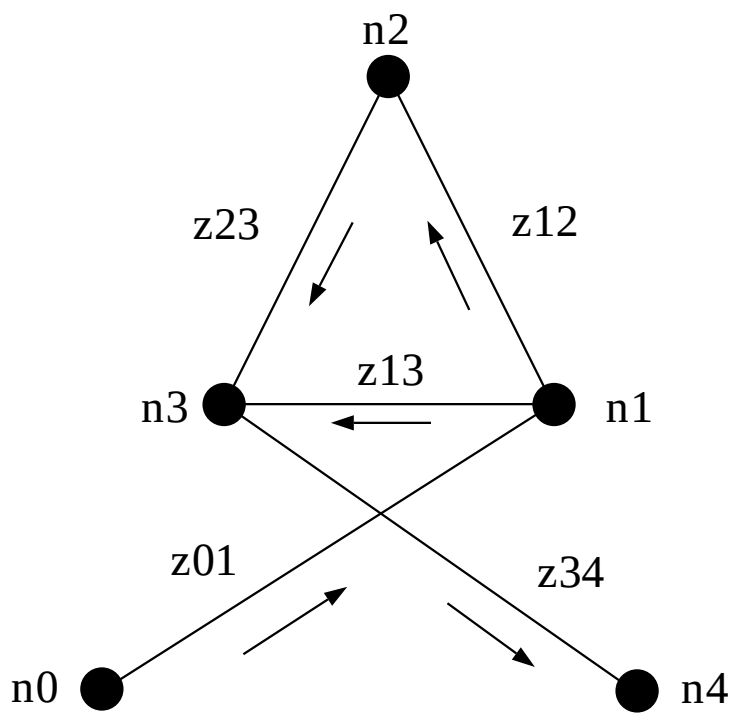

Figure 7.3: **Sequential Position Estimation Algorithm: No Dynamic Model**

In this example, the nodes nk of the position network represent the image positions, and each link zij is the displacement measurement between nodes i and j.

Figure 7.3 presents a simple example of a mosaic position network to illustrate the procedure. For $n = 2$, $\mathbf{x}$ would equal the x, y position of node 2, and $\mathbf{z}$ would be as follows:

$$\mathbf{z} = \begin{pmatrix} \mathbf{z}_{01} + \mathbf{z}_{12} \\ \mathbf{z}_{01} + \mathbf{z}_{13} + \mathbf{z}_{32} \end{pmatrix} \tag{7.24}$$

where $\mathbf{z}_{ij}$ is the displacement measurement from image i to image j and ${}^{j}\mathbf{z}_{i} = -{}^{i}\mathbf{z}_{j}$.

To determine the variance V, recall that all link measurements $\mathbf{z}_{ij}$ are independent. Thus, the diagonal terms are simply the sum of the covariances of each link in the given path, and the off-diagonal terms are equal to the sum of the covariances of shared links between the two paths:

$$V = \begin{pmatrix} V_{01} + V_{12} & V_{01} \\ V_{01} & V_{01} + V_{13} + V_{32} \end{pmatrix} \tag{7.25}$$

This method of determining V can be applied for $\mathbf{z}$ vectors of any size, provided that the chain of links composing each path is known.

Since every $\mathbf{z}_{ij}$ measurement is an estimate of $\mathbf{x}$, the C matrix is a column vector of identity sub-matrices:

$$C = \begin{pmatrix} I \\ I \end{pmatrix} \tag{7.26}$$

Similarly, for the general case, C is a column of p identity sub-matrices, where p equals the number of paths to the specified node.

The optimal estimate of $\mathbf{x}$ for node 2 can now be found by substituting $\mathbf{z}$, V, and C into Equations 7.9–7.11. The resulting maximum likelihood estimate and corresponding covariance is:

$$\mathbf{x}_2 = \mathbf{z}_{01} + \left(\frac{V_{13} + V_{32}}{V_{12} + V_{13} + V_{32}}\right)(\mathbf{z}_{12}) + \left(\frac{V_{12}}{V_{12} + V_{13} + V_{32}}\right)(\mathbf{z}_{13} + \mathbf{z}_{32}) \tag{7.27}$$

$$\hat{P}_2 = V_{01} + \frac{(V_{12})(V_{13} + V_{32})}{V_{12} + V_{13} + V_{32}} \tag{7.28}$$

Essentially, the optimal estimate is the sum of the shared link ($\mathbf{z}_{01}$) and a covariance-weighted average of the two different paths from node 1 to node 2, a result that matches intuition. The identical approach can be utilized to optimize the locations of all other nodes in Figure 7.3.

While this algorithm is conceptually simple, the inherent difficulty is keeping track of all possible paths from the origin to every other node in the mosaic. To perform this task, another algorithm can be constructed to produce a path tree from the origin node. This path-tree algorithm consists of a series of rules that are checked whenever a new link is added to the network. The set of rules is:

- If the new measurement is a sequential link with reference node $(n-1)$ and new node n:
    1. Add a leaf corresponding to node n to every instance of node $(n-1)$ in the path tree.
    2. Estimate the location of node n by adding the new link measurement to the latest optimal estimate of the node $(n-1)$ location.
- If the new measurement is a crossover link with head node h and tail node n (i.e. the most current node):
    1. At every instance of node h in the path tree:
        (a) Add node instances to the path tree, by traversing the network from node h through node n to all possible branches.
        (b) Record which nodes have had new instances added to the path tree.
        (c) To prevent paths containing internal loops, stop traversing a particular network branch when the next node to add is already in the branch of the path tree.
    2. At every instance of node n in the path tree:
        (a) Add node instances to the path tree, by traversing the network from node n through node h to all possible branches.

(b) Record which nodes have had new instances added to the path tree.

(c) To prevent paths containing internal loops, stop traversing a particular network branch when the next node to add is already in the branch of the path tree.

3. Perform a static estimation for every node that has been recorded in the list of modified nodes.

Figure 7.4 provides a clear example of the evolution of a path tree using this rule set, for a vehicle path that will ultimately contain two crossover points.

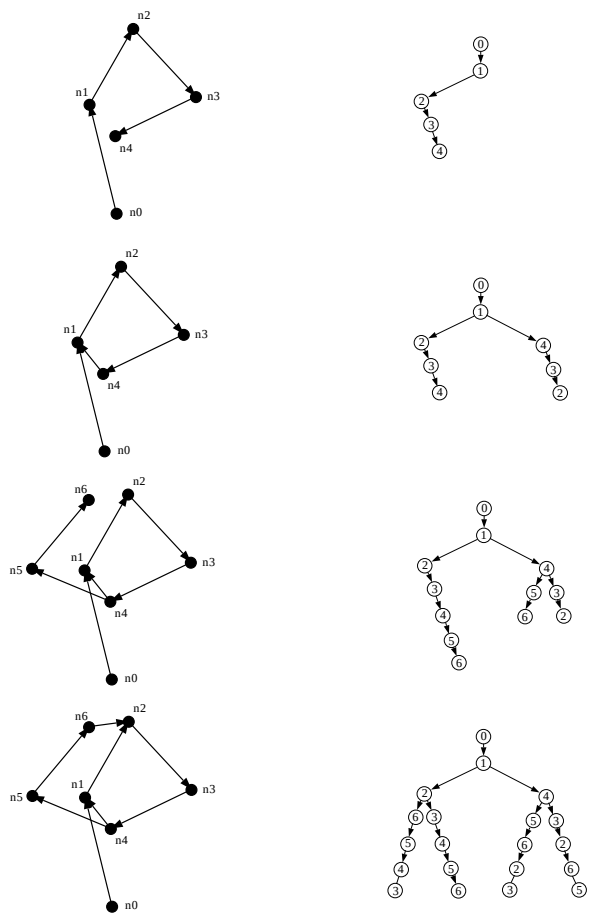

Figure 7.4: **Path-Tree Algorithm for Multiple-Loop Mosaic**

The left side of this figure depicts stages of the evolving mosaic model, while the right side shows the corresponding path trees.

At first glance, this algorithm is a vast improvement over the corresponding batch algorithm. Since static estimation is performed for each node individually, the computational cost simply depends on the number of measurements, i.e. the number of different paths to the node in question. Thus, the order of computations is $O(P^3 N^*)$, where N^* is the number of nodes with new paths since the last optimization, and P is the average number of paths to each of the N^* nodes. Thus, since $N^* \leq N$, this algorithm is at most linear in the total number of nodes, compared to a cubic relation for the batch algorithm.

However, there is a hidden problem with this method, due to the fact that the order of computations is cubic in the average number of paths to each node. The problem is that the path tree tends to growth exponentially in relation to the number of nodes. The mosaic network is far from fully connected; it is much closer to the other end of the spectrum, namely, a serial chain of nodes. This fact helps to minimize the exponential path growth. Unfortunately, path explosion becomes most problematic when more successful crossover measurements occur. In other words, the path problem becomes worse when the potential for improvement is highest, since a greater number of crossover measurements increases the accuracy of the state estimates.

This is a fundamental limitation of this algorithm, yet it also suggests the possibility of highly efficient, sub-optimal solutions to the estimation problem. In particular, various path pruning methods could be devised to limit the exponential growth of the path tree, significantly improving its scalability. One of the most straightforward path pruning methods simply would be to use the mosaic error network and Dijkstra's algorithm from Section 6.2.2 to find the minimum variance path from the origin node to each of the other nodes. This reduces the smoothing phase to a summation of local displacements along the minimum variance path. In fact, the $O(N^2)$ computations required by Dijkstra's algorithm would dominate the performance. While these types of methods would not take into account all measurements in optimizing a particular node location, the tradeoff in performance vs. accuracy may be highly desirable for many applications.

Link-Based Method

This method adds a straightforward modification to the batch algorithm of Section 7.2.1 to reduce the required computation greatly. The modification involves the use of state augmentation and a prior estimate at each stage of the computation to store the results of the previous optimization.

Equations 7.1–7.3 still govern the static estimation process. To start the process, the initial node is defined to be the origin, and there is no prior estimate of the state. Every time an adjacent link measurement is received, the state $\mathbf{x}$ is augmented with the position of the new node, and the estimates $\hat{\mathbf{x}}$ and $\hat{P}$ are updated by summing the adjacent link measurement, $\mathbf{z}_m$ and the position of the most recent reference image, $\mathbf{x}_N$. So, given a prior estimate of $\bar{\mathbf{x}}$ and $\bar{P}$, where:

$$\bar{\mathbf{x}} = \begin{pmatrix} \vdots \\ \bar{\mathbf{x}}_n \\ \vdots \end{pmatrix} \text{ for } n = 1, \ldots, (N-1) \tag{7.29}$$

$$\bar{P} = \begin{pmatrix} \ddots & & & \vdots \\ & \bar{P}_n & & \bar{P}_{(col\ \ N-1)} \\ & & \ddots & \vdots \\ \cdots & \bar{P}_{(row\ \ N-1)} & \cdots & \bar{P}_{N-1} \end{pmatrix} \tag{7.30}$$

the current estimate after taking into account the new adjacent measurement and node is:

$$\hat{\mathbf{x}} = \begin{pmatrix} \bar{\mathbf{x}} \\ \bar{\mathbf{x}}_{N-1} + \mathbf{z}_m \end{pmatrix} \tag{7.31}$$

$$\hat{P} = \begin{pmatrix} \bar{P} & \bar{P}_{(col\ \ N-1)} \\ \bar{P}_{(row\ \ N-1)} & \bar{P}_{N-1} + V_m \end{pmatrix} \tag{7.32}$$

Every new adjacent measurement augments the state vector $\mathbf{x}$, such that it always consists of the 2-D node positions for every node currently in the mosaic network.

Whenever a new crossover measurement occurs, a static estimation update can be performed, where $\mathbf{z}$ and V equal the crossover measurement and its variance, respectively. If the crossover measures the displacement from an image i to image j, then C is a 1 by N block matrix, with an identity sub-matrix (I) in the jth column, and a negative identity sub-matrix ($-I$) in the ith column. Furthermore, the latest estimate $\hat{\mathbf{x}}$, $\hat{P}$ becomes the prior estimate $\bar{\mathbf{x}}$, $\bar{P}$. Using all of these above values, and re-writing Equations 7.9–7.11 in an alternate form, the maximum likelihood estimate, $\hat{\mathbf{x}}$ and $\hat{P}$ can be found quite efficiently:

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + K(\mathbf{z} - C\bar{\mathbf{x}}) \tag{7.33}$$

$$K = \bar{P}C^{\mathrm{T}}[V + C\bar{P}C^{\mathrm{T}}]^{-1} \tag{7.34}$$

$$\hat{P} = \bar{P} - KC\bar{P} \tag{7.35}$$

Because the alternate formulation of these equations only requires the inversion of a small matrix, and by taking advantage of the zeros in the C matrix, this sequential method can be completed in $O(N)$ computations, a remarkable improvement over the $O(N^3)$ required for the equivalent batch algorithm. Furthermore, this method has none of the path-growth limitations of the previous sequential method, so it is ideal for online optimal estimation in the smoother stage of the vision sensor.

7.3.2 Dynamic Model

In Section 7.2.2, a batch algorithm utilizing a vehicle dynamic model was developed to estimate every point along the vehicle path history. This results in wasted computational effort, since knowledge of the vehicle path between images in the mosaic is unnecessary; only estimates of the image locations are required to re-align the mosaic and improve the performance of future crossover attempts. Thus, a sequential algorithm utilizing a dynamic model can take advantage of this fact and only smooth the image locations, rather than the entire vehicle path history, as long as all available measurements have been taken into account.

Thus, the approach of this sequential algorithm is to develop a Kalman filter capable of dynamically estimating both the current vehicle state and past image node states. Again, the definition of state is different when dealing with dynamic models: the state vector may include position, velocity, or other information relevant to the dynamic model.

As in the batch algorithm case, it is assumed that a vehicle dynamic model of the form specified in Equations 7.12 and 7.13 is given, where the state $\mathbf{x}(k)$ is defined as needed by the dynamic model. The only additional constraint on the dynamic model is that $\mathbf{x}(k)$ must include the vehicle position, $\mathbf{p}(k)$. The standard Kalman filter implementation also requires real-time measurement updates $\mathbf{y}(k)$ of the form:

$$\mathbf{y}(k) = C(k)\mathbf{x}(k) + \mathbf{v}(k) \text{ for } k = 0, \ldots, K \tag{7.36}$$

$$\mathbf{v}(k) = \mathcal{N}[0, V(k)] \tag{7.37}$$

The first two estimation stages of the vision sensor provide local displacement measurements at every time step, whether they are measurements along the image chain or crossover measurements. However, these local displacements are always referenced to a node location, either the reference image (i.e. most recent) node for adjacent measurements, or an arbitrary image node for crossover measurements. As a result, they cannot be specified in the form of Equation 7.36, since $\mathbf{x}(k)$ contains only current state information.

To solve this problem, it is possible to augment the state vector $\mathbf{x}(k)$ with additional components. Thus, whenever a new snapshot image is taken, the state is augmented with the snapshot image location. When augmenting the state vector, it is also necessary to augment the current estimate covariance $\hat{P}(k)$, to reflect the new state. By noting that the new node position and the current position are identical at the instant the snapshot is taken, $\hat{P}k$ can be augmented by adding new rows and columns corresponding to the new state component and copying the relevant covariances from existing entries in the covariance matrix.

This state augmentation enables specification of the sensor measurements in the form of Equation 7.36, and it integrates the static estimation of image node positions with the

dynamic estimation of current vehicle state. For instance, after the initial reference image snapshot is taken, the augmented system of equations looks like:

$$\mathbf{x}_{aug}(k+1) = \Phi_{aug}(k)\mathbf{x}_{aug}(k) + ,_{aug}(k)\mathbf{u}(k) \tag{7.38}$$

$$\mathbf{y}(k) = C_{aug}(k)\mathbf{x}_{aug}(k) + \mathbf{v}(k) \tag{7.39}$$

$$\mathbf{x}_{aug}(k) = \begin{pmatrix} \mathbf{x}(k) \\ \mathbf{p}_0 \end{pmatrix} \tag{7.40}$$

$$\Phi_{aug}(k) = \begin{pmatrix} \Phi(k) & 0 \\ 0 & I \end{pmatrix} \tag{7.41}$$

$$,_{aug}(k) = \begin{pmatrix} ,(k) \\ 0 \end{pmatrix} \tag{7.42}$$

$$C_{aug} = \begin{pmatrix} \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} & -I \end{pmatrix} \tag{7.43}$$

$$\tag{7.44}$$

where $\mathbf{p}_n$ equals the position of node n. After n snapshot images have been taken, the augmented variables are:

$$\mathbf{x}_{aug}(k) = \begin{pmatrix} \mathbf{x}(k) \\ \mathbf{p}_n \\ \vdots \\ \mathbf{p}_0 \end{pmatrix} \tag{7.45}$$

$$\Phi_{aug}(k) = \begin{pmatrix} \Phi(k) & 0 \\ 0 & I \end{pmatrix} \tag{7.46}$$

$$,_{aug}(k) = \begin{pmatrix} ,(k) \\ 0 \end{pmatrix} \tag{7.47}$$

$$C_{aug} = \begin{pmatrix} \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} & -I & 0 \end{pmatrix} \tag{7.48}$$

(7.49)

In the above sets of equations, C_{aug} represents the matrix corresponding to adjacent measurements, where $\mathbf{p}_n$ is always the reference image. For the case of crossover measurements, two measurements are performed in the same time step: an adjacent measurement and a crossover measurement, both utilizing the current image. So, when a successful crossover measurement occurs, the two simultaneous measurements can be indicated as follows:

$$\mathbf{y}(k) = \begin{pmatrix} \mathbf{y}_{adj} \\ \mathbf{y}_{cross} \end{pmatrix} \tag{7.50}$$

$$C_{aug} = \begin{pmatrix} \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} & -I & 0 & \cdots & 0 \\ \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} & 0 & \cdots & -I & \cdots \end{pmatrix} \tag{7.51}$$

The Kalman filter solves Equations 7.12, 7.13, 7.36, and 7.37 recursively, by utilizing successive time and measurement updates [2, Ch. 3, pp. 9–10]. In the recursive equations to follow, $\bar{\mathbf{x}}$ and $\bar{P}$ are prediction estimates (i.e. before the measurement update), and $\hat{\mathbf{x}}$ and $\hat{P}$ are current estimates (i.e. after the measurement update). The time updates are as follows:

$$\bar{\mathbf{x}}_{aug}(k+1) = \Phi_{aug}(k)\hat{\mathbf{x}}_{aug}(k) + ,_{aug}(k)\mathbf{u}(k) \tag{7.52}$$

$$\bar{P}_{aug}(k+1) = \Phi_{aug}(k)\hat{P}_{aug}(k)\Phi^{\mathrm{T}}_{aug}(k) + ,_{aug}(k)U(k),^{\mathrm{T}}_{aug}(k) \tag{7.53}$$

The measurement updates are the same as for the case of static estimation (Equations 7.9–7.11, but can be written in a different form:

$$\hat{\mathbf{x}}_{aug}(k) = \bar{\mathbf{x}}_{aug}(k) + K[\mathbf{y}(k) - C_{aug}(k)\bar{\mathbf{x}}_{aug}(k)] \tag{7.54}$$

$$K = \bar{P}_{aug}(k)C^{\mathrm{T}}_{aug}(k)[V(k) + C_{aug}(k)\bar{P}_{aug}(k)C^{\mathrm{T}}_{aug}(k)]^{-1} \tag{7.55}$$

$$\hat{P}_{aug}(k) = KC_{aug}(k)\bar{P}_{aug}(k) \tag{7.56}$$

Using this modified Kalman filter method, the maximum likelihood image node positions can be read directly from the state estimates $\hat{\mathbf{x}}_{aug}(k)$. Furthermore, since the method is sequential in nature, the computation required at each time step is much reduced. By calculating the update relations in the form of Equations 7.52–7.56, and by taking advantage of the zeros in the C_{aug} matrix, the required computations at each time step are $O(N)$, where N is the number of nodes. By reducing the computation from a cubic to a linear dependence on the number of images in the mosaic, this sequential algorithm has proven itself to be ideal for the case when a dynamic model of the robot is available.

7.4 Validation on Space Frame

Before proceeding to the experimental validation of the optimal estimation stage, it was necessary to decide which of the methods is most suitable for implementation. Simple dynamic models do exist for the specific experimental platforms utilized in this research, namely the Space Frame, OTTER, and Ventana. However, the control and disturbance inputs to these models are not well-known; as a result, the dynamic models would provide little information to improve image and vehicle state estimation and mosaic re-alignment. Furthermore, one of the goals of the experimental phase of this work is to demonstrate that the vision measurements in conjunction with the estimation techniques discussed in this dissertation are sufficient for robot navigation, without resorting to a dynamic model for corroboration. For these reasons, the choice of smoother algorithms was limited to one of the two that does not rely on a dynamic model.

The batch algorithm was chosen over the sequential algorithm for the sake of simplicity. The batch algorithm is easier to implement in code; it can be performed in a single matrix computation, and there is no need for complicated algorithms to keep track of the paths to each node.

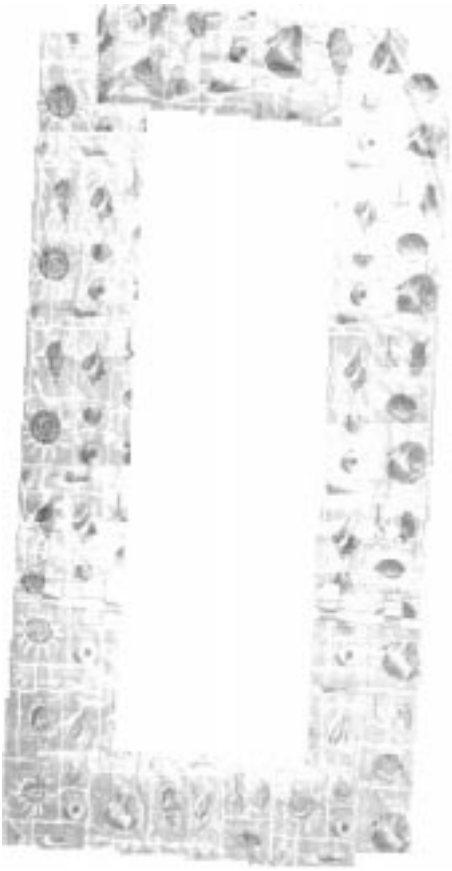

Figure 7.5: **Smoothed Mosaic**

The initial and final images in the lower-left corner of this mosaic have been re-aligned, and the smoother has minimized the alignment errors throughout the mosaic.

The batch smoothing algorithm for a system with no dynamic model was implemented and tested on the Space Frame. This stage completed the development of the entire vision sensor, so it was possible to test the full capability of the sensor for producing optimally aligned mosaics. With the smoother online, mosaics were created as the vehicle followed a single loop (Figure 7.5) and for arbitrary complex vehicle paths consisting of multiple loops (Figure 7.6). For the multiple-loop mosaic, it can be seen that the alignment among all images in the mosaic is quite accurate, despite the many crossover points in the image where non-adjacent images in the image chain overlap. This demonstrates the capability to

Figure 7.6: **Smoothed Mosaic with Multiple Loops**

create multiple-loop mosaics, but for the sake of clarity, only the data from the single-loop mosaic will be presented to quantitatively verify the smoother stage of the vision sensor.

The predicted variances and measurement errors for a single-loop smoothed mosaic are displayed in Figure 7.7. Because the data in between snapshot images are not used for smoothing and thus are not stored, the smoother algorithm only estimates the node locations. These node locations are drawn on the plot as diamonds and circles, before and after optimal estimation, respectively. For the single-loop case, the post-smoothing estimate of image position is simply a weighted average of the two estimates obtained by summing the local displacements around the loop in the forward and backward directions,

from the initial node to the node of interest. Therefore, position estimates for nodes near the crossover point (i.e. near the start and end times in the plot) are dominated by the short path around the loop. Farther around the loop, the path variances are larger and roughly equal, so the two estimates are weighted more equally. These trends are verified by Figure 7.7. Furthermore, even after smoothing, the estimation errors in node positions generally fall within the predicted 1-σ (68%) error envelope, providing further evidence of the validity of the entire state estimation process.

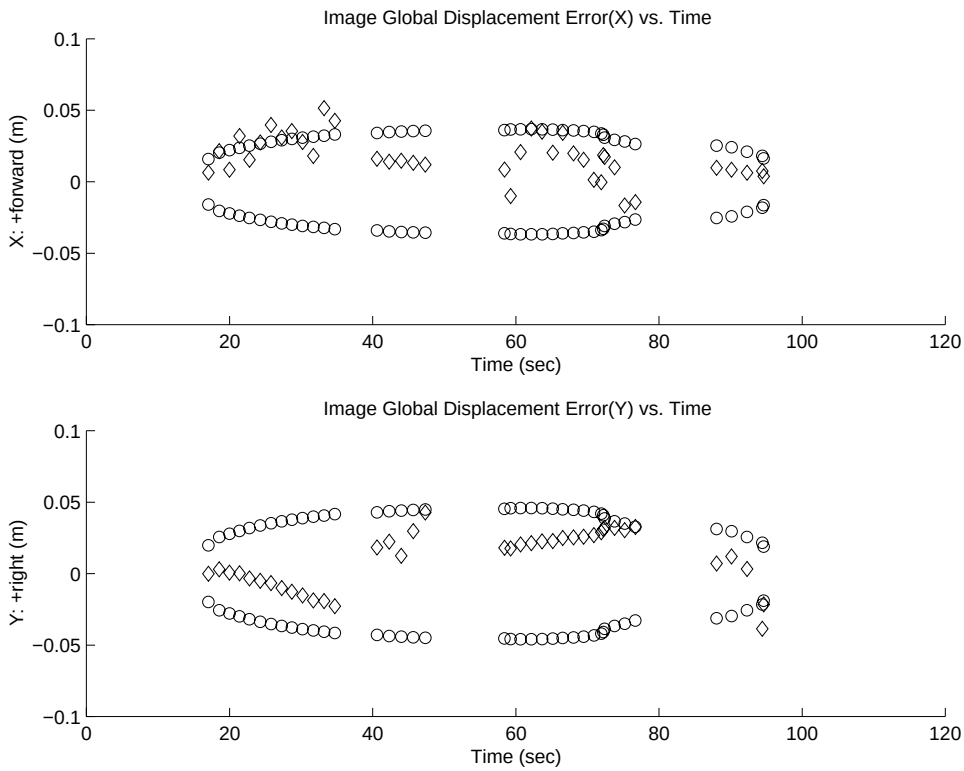

Figure 7.7: **Predicted and Actual Error Data after Smoothing**

The circles represent the predicted error envelopes at the image snapshot times after smoothing (since the 10–30 Hz data between snapshots is not saved for smoothing). The diamonds are the actual errors in estimating the image global displacements after smoothing.

The increase in global position accuracy for all of the images in the mosaic can be seen by comparing the smoother data with the data recorded immediately after the crossover

correlation. Figures 7.8 and 7.9 show the predicted and actual reduction in error, respectively, for the typical run highlighted in this discussion. While the optimal estimation stage provides only marginal improvement in position estimates immediately after the crossover time (about 95 seconds), it provides significant improvement in estimating the past node locations. This is crucial for two reasons: it improves the alignment and visual quality of the existing mosaic; and it enhances the impact of future crossover points, by providing more accurate node locations with which to align.

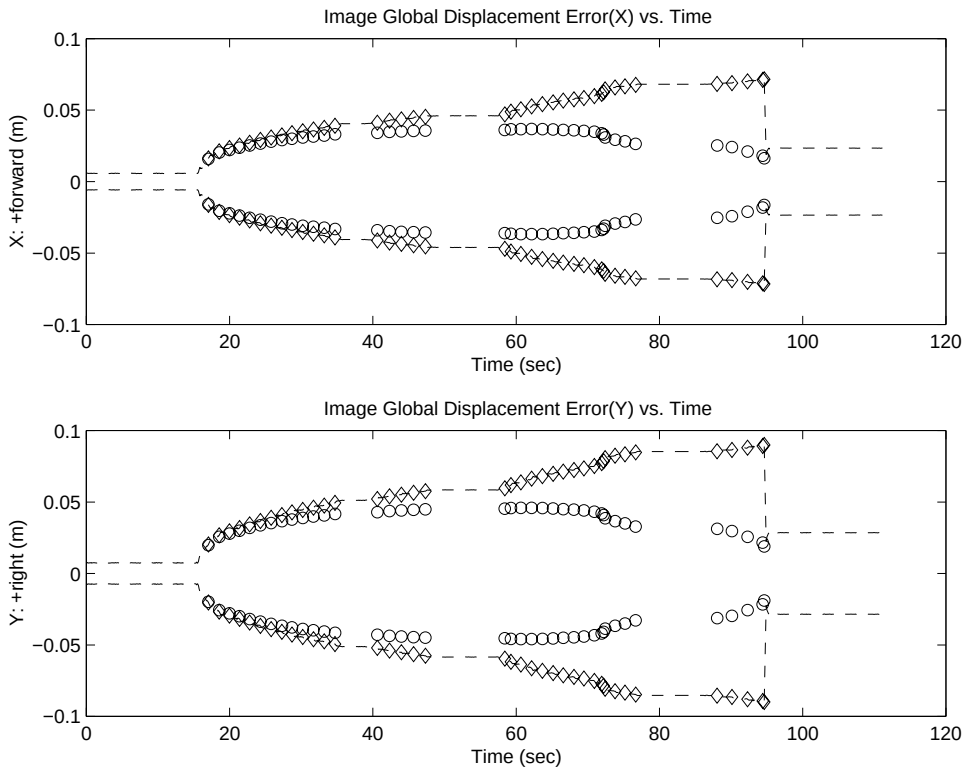

Figure 7.8: **Comparison of Predicted Error Bounds**

The dashed lines are error envelopes that correspond to the predicted variances on the image global displacement estimates before smoothing. The diamonds indicate the same data at the image snapshot times (for direct comparison to the data after smoothing). The circles depict the error envelopes at the image snapshot times after smoothing.

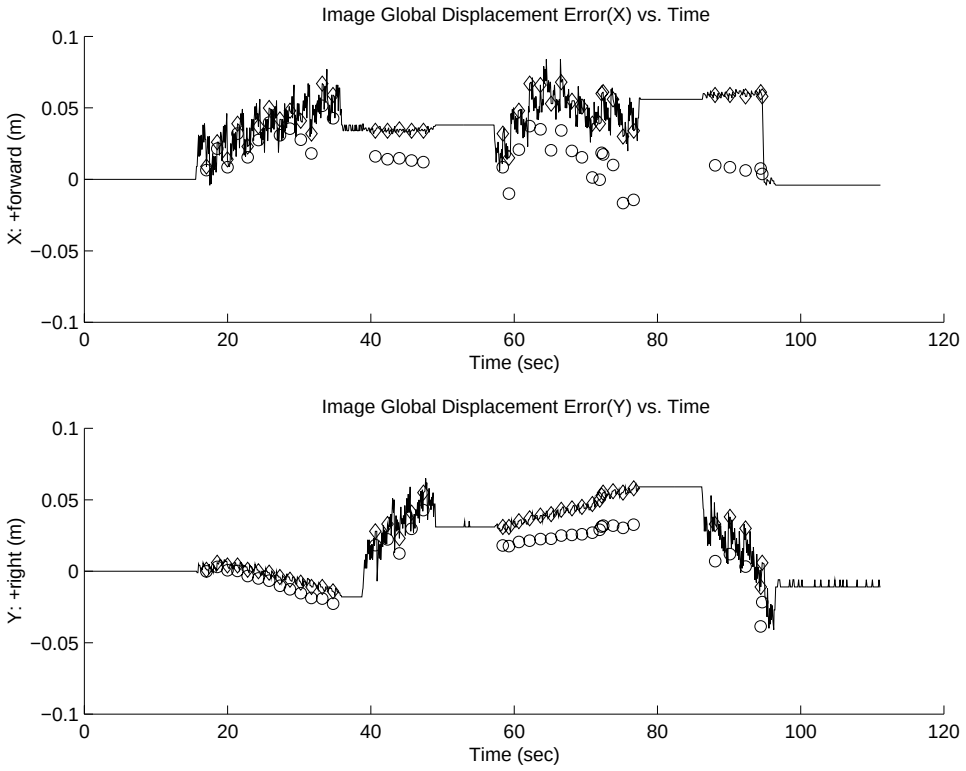

Figure 7.9: **Comparison of Actual Error Data**

The solid lines indicate the actual experimental errors in estimating the image global displacements before smoothing. The diamonds indicate the same data at the image snapshot times (for direct comparison to the data after smoothing). The circles indicate the actual experimental errors at the snapshot times after smoothing.

7.5 Summary

To complete the explanation of the vision sensor estimation process (Figure 2.5), this chapter described several possible methods for the optimal estimation of mosaic node positions after a successful crossover correlation has occurred. These methods can be differentiated by such factors as: batch vs. sequential algorithm, presence vs. absence of a dynamic model, and single vs. multiple loops.

Based on the constraints of the experimental systems, a suitable method was chosen and implemented. Using this smoother stage in conjunction with the previous two stages, the proper operation of the complete vision sensor was validated on the Space Frame. The tests

also verified the improved mosaic re-alignment by the smoother stage, and these results were compared with the output of the previous two stages in Figures 7.8–7.9.

Chapter 8

Field Tests

This chapter presents the results from field tests of the complete vision-based, bounded-error mapping, state estimation, and navigation system. Experiments were conducted on underwater robots in both the test tank and open ocean environments.

8.1 Introduction

In Chapters 5–7, each estimation stage of the vision sensor was tested individually on the Space Frame in ARL. The unique advantage of using the Space Frame is that it provides truth measurements for quantitative evaluation of particular methods and comparison with theoretical performance estimates, whereas field systems rarely have this capability. However, as discussed briefly in Section 5.3.4, the Space Frame has the drawback that it is not a perfect imitation of a typical real-world environment:

- the camera images have a different frequency content than those of an underwater environment,
- the non-visual sensors exhibit much lower noise and higher accuracy characteristics than typical sensors on-board an underwater robot,
- the orientation of the camera and vehicle are fixed in space and align with the axes of the global frame,

- the camera and "vehicle" centers are constrained to coincide,
- and the scene surface is constrained to be perpendicular to gravity.

All of these variations are permissible within the assumptions of the state estimation algorithms, but they are not fully exploited by experiments run on the Space Frame.

In order to verify the proper operation of the vision sensor under these additional environmental variations, field tests were performed on vehicle platforms in two different underwater environments. These experiments provided the opportunity to combine the vision sensor, control system, and graphical interface components into a complete navigation system and to test this system in a real-world environment.

The first phase of testing was performed in a large test tank at MBARI using the OTTER AUV. These experiments were intended to simply validate the ability to accomplish the navigation task on a real vehicle using the vision sensor developed in this thesis. Section 8.2 describes the tests that were conducted for this purpose.

The second phase of testing took place in the Monterey Bay, in the deep waters of the Monterey Canyon. MBARI provided both the support ship Pt. Lobos and the Ventana ROV for this effort. The goal of these tests was to enable a more thorough investigation of the estimation results from real sensor data and of the control performance using input data from the vision sensor. Section 8.3 provides a detailed description of the results from ocean experimentation.

8.2 OTTER

The demonstration of bounded-error, vision-based navigation on OTTER is the first of its kind in the field of underwater robotics. In comparison to the Space Frame, the OTTER AUV represents a generalization of the problem of autonomous underwater navigation. Real sensors (e.g. pressure depth sensor, SHARPS, compass, inclinometers, IMU) and their associated noise characteristics provide the input data for the vision sensor. The vehicle is free to deviate in all 3-DOF from its nominal orientation, within the control system

constraints.[1] The cameras are mounted on the front of the vehicle, such that the camera and vehicle centers are a significant distance away from each other. Thus, qualitative validation of the navigation system on OTTER provides significant evidence that the system has successfully achieved its design goals.

Implementation of the entire mapping and navigation system was straightforward, given that OTTER was designed to be an experimental testbed for new technologies, and several Ph.D.-level experiments had been performed using OTTER in the past. The navigation software had already been implemented on a dual-Pentium PC for the Space Frame experiments, so this same machine and software provided the vision sensor and GUI functionality. The OtterLink thread was written to communicate via AVPNet to a network node running on a Sun UNIX workstation, which then transformed the data into NDDS packets for direct communication with the on-board computer. The smoother computations were performed remotely in MATLAB on another SUN UNIX workstation at Stanford ARL. The ComputeServerLink thread enabled communication via AVPNet between the vision sensor code and the optimal estimation code. Due to the small amount of data transferred between the PC and the compute server, no problems were experienced as a result of low-bandwidth or high-latency issues associated with the Internet. The control system had already been implemented on-board OTTER, so the PC simply sent sensor data and reference commands in real-time to the robot control system.[2]

To provide texture resembling the ocean floor, several shower curtains with patterns of large seashells were placed on the bottom of the test tank. This provided a total usable area of approximately 12 square meters. The vehicle was placed into position above the shower curtains using the SHARPS network and switched into vision-based navigation mode. From this point on, the cameras provided x, y translational data, the pressure depth sensor provided range estimates, and all vehicle movements were specified through the vision-based GUI.

[1] The control system has sufficient authority to maintain the vehicle attitude close enough to its nominal pose to permit the use of the small angle approximation.

[2] Complete technical details on the OTTER control system, and the associated implementation issues, are discussed fully in Howard Wang's dissertation [34].

The experiments that were conducted all focused around use of the GUI point-and-click navigation capability to drive OTTER around the test tank. While demonstrating this functionality, the corresponding mosaics were recorded for analysis. To test the complete vision sensing system, OTTER was driven in a single loop with all three estimation stages enabled. During this test run, the smoothed mosaic of Figure 8.1 was created. As a final verification of the system functionality, OTTER was navigated along an arbitrary path with the GUI, while the vision sensor detected several crossover points and smoothed the evolving mosaic several times. The resultant mosaic is depicted in Figure 8.2.

Figure 8.1: **Smoothed Mosaic Created With OTTER**

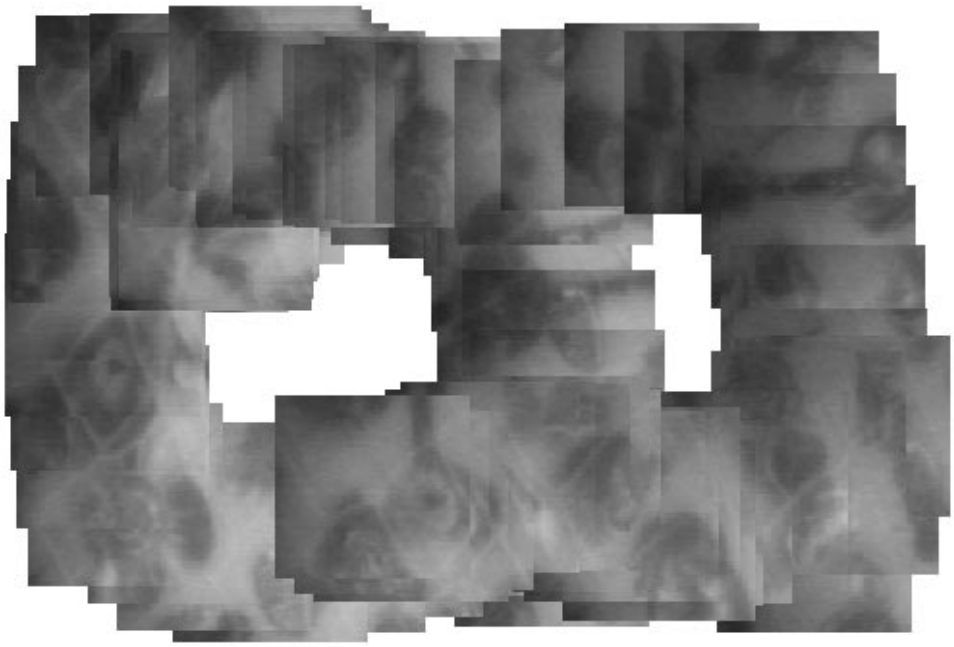

Figure 8.2: **Smoothed Mosaic Created During Navigation on OTTER**

8.3 Ventana

As the culmination of this research, MBARI provided the exciting opportunity to work with ROV pilots and marine scientists aboard the Pt. Lobos research vessel. The goal of this effort was to transfer the navigation technology developed at ARL to an operational vehicle at MBARI, namely, the Ventana ROV. To this end, the first-ever demonstration of bounded-error, vision-based, task-level navigation of an ROV in an unexplored, unstructured area of the near-bottom ocean environment was achieved.

The challenge of implementing the navigation system on Ventana represents a further generalization of the problem of autonomous underwater navigation. In addition to the issues the OTTER experiments addressed, a couple other factors must be taken into account. Real underwater imagery in the deep ocean often contains significant noise, due to effects such as backscatter, marine snow and lighting variations (Sections 3.4.1 and 3.5.1), or areas of insufficient texture for correlation. These image effects degrade the vision-based

displacement measurements, causing a major increase in invalid measurements. Furthermore, the terrain surface (e.g. ocean floor, vertical rock face) can no longer assumed to be perpendicular to the direction of gravity. While the state estimator assumptions constrain the surface to be roughly planar, it may be arbitrarily oriented with respect to both gravity and the vehicle. Thus, Ventana is the only vehicle platform utilized in this research to exercise the full freedom of the vision sensor assumptions.

Since Ventana is an ROV, the goal of transferring technology developed at ARL is to provide pilot aids by automating routine and tedious tasks, freeing the pilot to concentrate on higher-level, mission-related issues. For the case of the vision sensor, the original intent was to automate the station keeping task, since this is a very common yet boring task for the pilots during marine science missions. After successfully demonstrating this technology for a horizontal ocean floor [11], it was decided to extend this work and enable mosaicking and navigation for arbitrarily oriented ocean floor terrain.

Given that Ventana and its support ship Pt. Lobos are operational systems that perform daily marine science missions off the coast of Monterey, it was necessary to interconnect the navigation software with existing subsystems while minimizing disturbances to the vehicle/ship operations. This took some additional effort, but fortunately, several methods and protocols already existed for the integration of new components. The only required change to the vehicle hardware was the addition of a sonar altimeter, mounted along the line-of-sight of the main camera. Since the main camera is mounted on a 2-DOF tilt unit, the altimeter provided a range measurement that was always orthogonal to the vision-based displacement measurements. As with the Space Frame and OTTER, the vision sensor and GUI functionality was performed by a dual-pentium PC running the Sensor 0.7 application. However, since the standard mode of operation for Ventana is under remote pilot control, the only control modes available were auto-heading and auto-depth modes. Thus, an automatic control system was implemented to control the three translational degrees of freedom.[3] The

[3]The auto-depth mode was never used, since the depth measurement is not always orthogonal to the vision-based measurements.

remaining two degrees of freedom, pitch and roll, are passively stable for Ventana. To minimize the impact to the system, the control commands masqueraded as joystick commands that were summed with the pilot's controls before being sent to the thrusters. The PC was connected to the ship-side computer via a serial line, so the VentanaSerialLink thread was written to perform the serial communications task. To perform the smoothing computations, the MATLAB compute server was run remotely on a Sun UNIX workstation at Stanford ARL. The Pt. Lobos has a microwave network link to land to provide Internet access to all computers aboard the ship, so the ComputeServerLinkThread was again able to provide communications support.

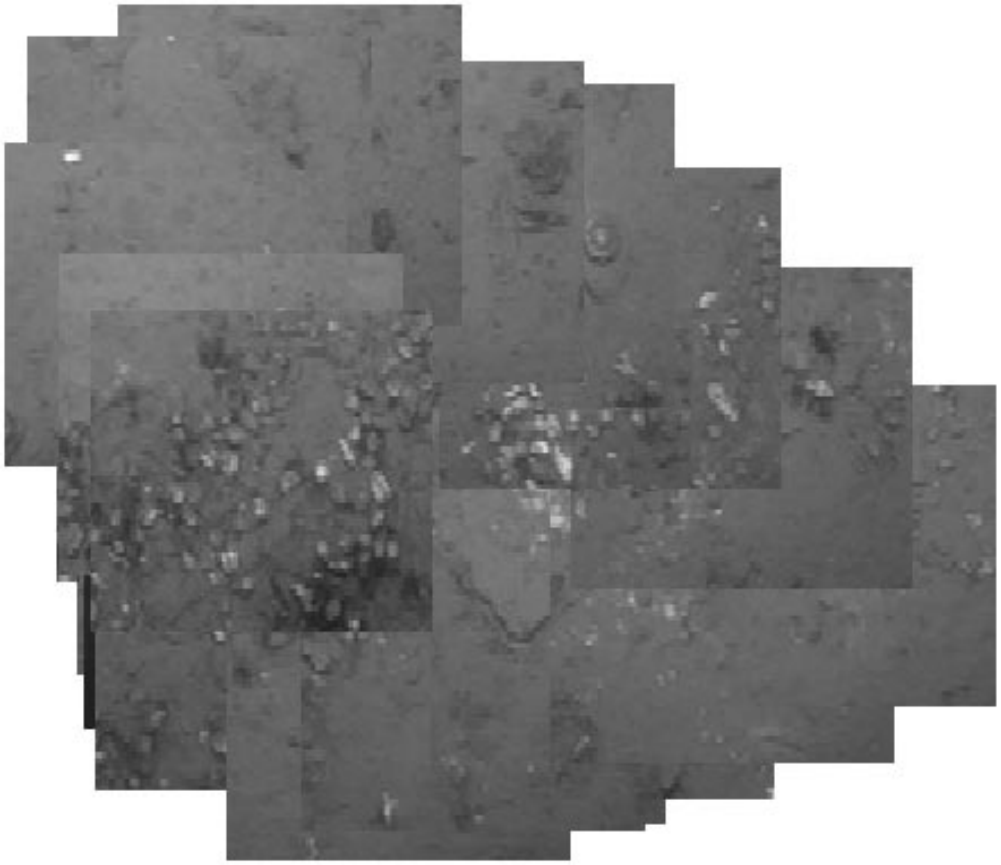

Figure 8.3: **Dead-Reckoned Mosaic Created With Ventana**

To begin each experimental run, the pilot would position Ventana at a suitable range above the ocean floor, align the main camera to be perpendicular to the ocean floor, and enable the auto-heading control mode. At this point, the vehicle and camera were assumed to be at their nominal orientations, and the pilot transferred control to the automatic navigation system by pressing a button on one of the touch-screen interfaces to the robot. The navigation system would then hold station at the nominal position, until a new goal location was specified via the mosaic-based GUI.

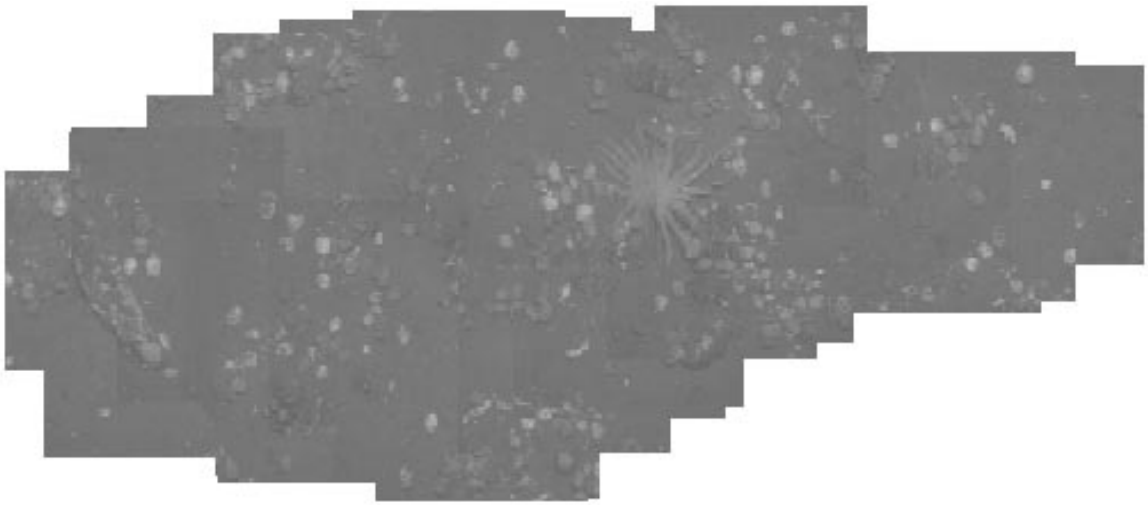

Figure 8.4: **Smoothed Mosaic Created During Navigation on Ventana**

The first level of experimentation was a proof-of-concept demonstration of the mosaicking and navigation task in the deep ocean environment. In particular, tests were conducted to verify the expected performance of the state estimation process. To test the baseline system, a mosaic was created using only the state estimator to perform dead reckoning estimation. The resultant mosaic is depicted in Figure 8.3. All stages of the vision sensor were then brought online to complete the state estimation strategy. To validate the ability to recognize multiple loops during unconstrained robot navigation and minimize the image registration errors, the optimally aligned mosaic in Figure 8.4 was created from a typical navigation run.

The next level of testing explored the robustness of the state estimation outputs to noisy input data, including spurious measurements from the image correlator and altimeter. Figures 8.5–8.9 represent a sequence of data taken from the same experimental run. In this test, the vehicle was directed to follow an arbitrary path containing multiple loops, using the mosaic-based navigation interface.

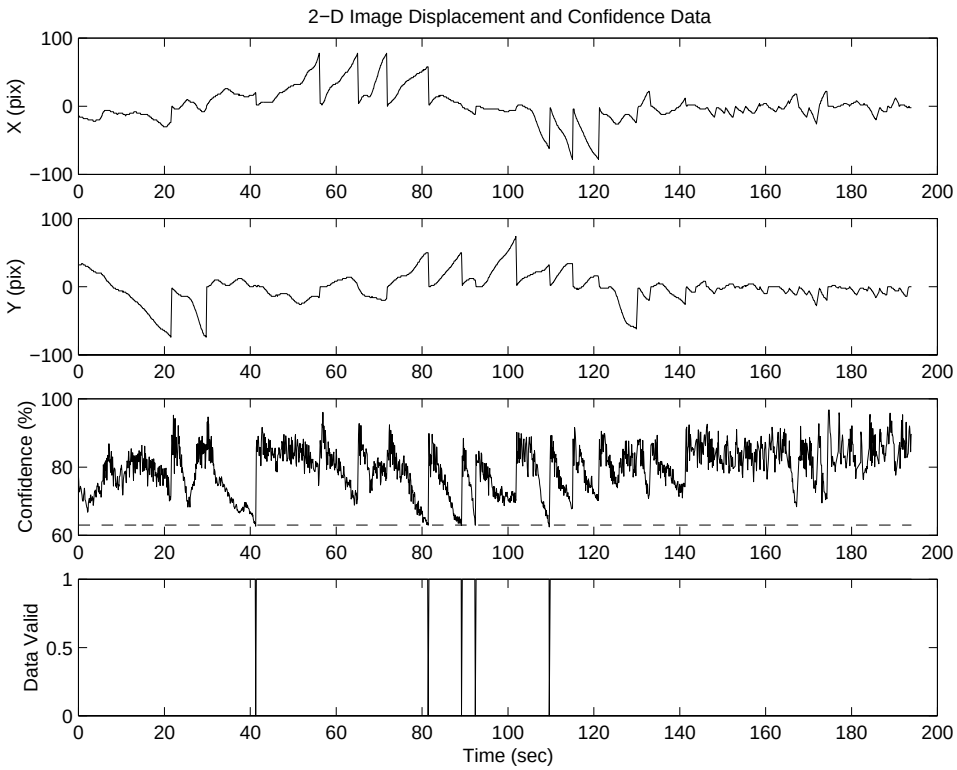

Figure 8.5: **Raw x, y Data and Confidence from Image Correlator**

The dashed line on the confidence plot indicates the confidence threshold (63%). This threshold determines the validity of the data. The data valid flag has a value of 1 if the data is valid and 0 if the data is invalid.

Figures 8.5 and 8.6 plot the raw data coming directly from the sensors. Notice that several dropouts occur in the image correlator x, y outputs. Depending on the scene texture, these often occur much more frequently than in the case of this particular experimental run. To recover from dropouts, measurement filters have been added between the sensors and the state estimation. For the image correlator, measurements that have an associated

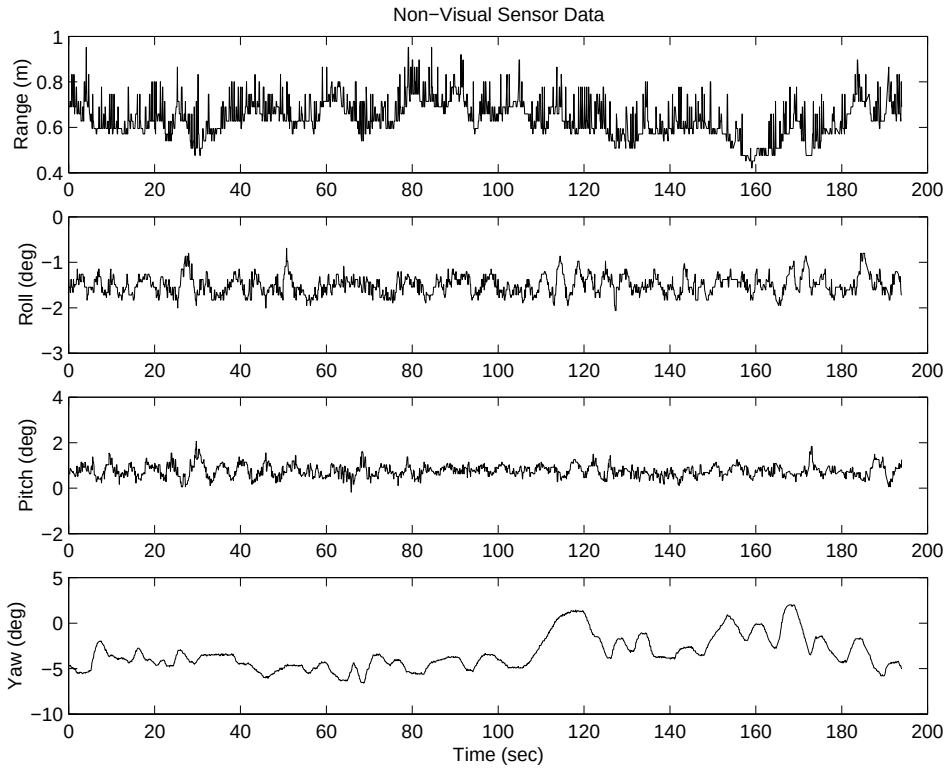

Figure 8.6: **Raw Range and Attitude Data**

confidence below a specified threshold are considered invalid (Figure 8.5). These are filtered out by assuming the vehicle has not moved and retaining the most recent valid data. If a sufficient number of consecutive dropouts occur, a new reference image is snapped and the image registration process continues. The altimeter also experiences occasional dropouts, although none are shown in Figure 8.6. Whenever a dropout occurs, the range measurement spikes to its maximum value. The filter to remove these dropouts checks for a value outside a specified bound; if invalid data are detected, the most recent valid data are used in their place. For the remaining attitude degrees of freedom, Figure 8.6 demonstrates that the compass and inclinometers are more robust and do not produce any spurious measurements that must be filtered out.

Once the input data have been filtered properly, they are passed to the state estimation portion of the vision sensor. Figure 8.7 plots the image position estimate, where the z component should be identically zero since all image centers are assumed to lie on the plane of the ocean floor.

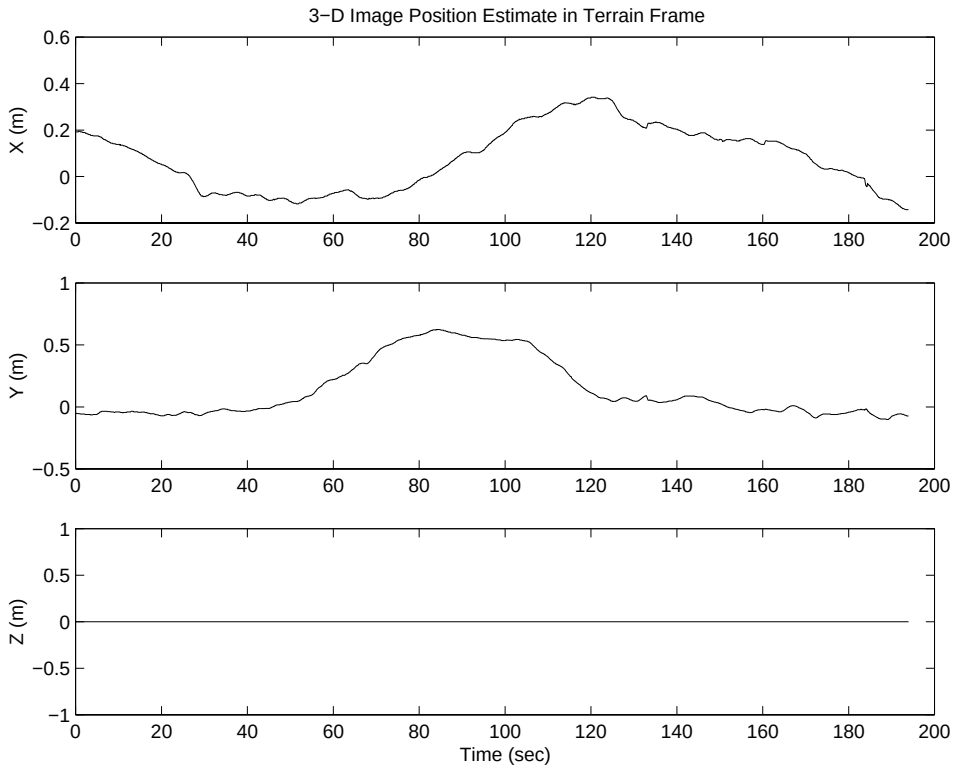

Figure 8.7: **Image Position Estimate from Vision Sensor**

The standard deviations of the x and y image state components are depicted in Figure 8.8. The effect of multiple crossover points can be seen in this plot: the solid line indicates the output of the crossover stage, while the asterisks indicate the node locations after the smoother stage has performed a mosaic re-alignment. While the actual sensing errors cannot be measured with Ventana, these predicted error bounds indicate a significant reduction in errors around the loop.

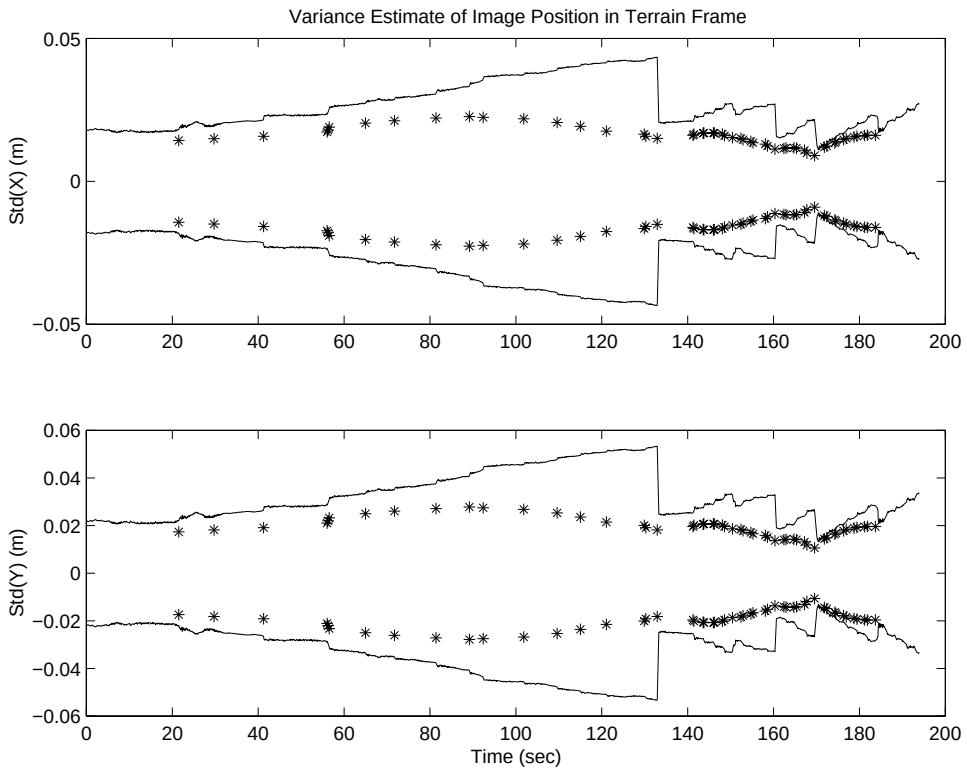

Figure 8.8: **Image Position Variances from Vision Sensor**

The solid lines indicate the predicted variances in estimating image position after crossover detection and correlation, but before smoothing. The '' signs indicate the predicted variances at the image snapshot times after smoothing.*

Figure 8.9 presents the 3-DOF vehicle position estimate for this same time period. In comparison to Figure 8.7, additional noise can be seen on the x and y state estimates; this is due to the imperfect range measurement that is used to determine the vehicle state from knowledge of the image state. While there is noise present on these signals, pre-filters have removed all spurious data from these real-time estimates of image and vehicle state.

The final level of experiments quantified the vehicle control performance. Using the vehicle state estimates of Figures 8.9 as the sensor inputs, three different controllers were implemented: Proportional-Derivative (PD), PID, and Sliding Mode. The PD controller proved to be the most accurate and robust method. Figure 8.10 shows the control error between the desired and actual position for all three translational degrees of freedom.

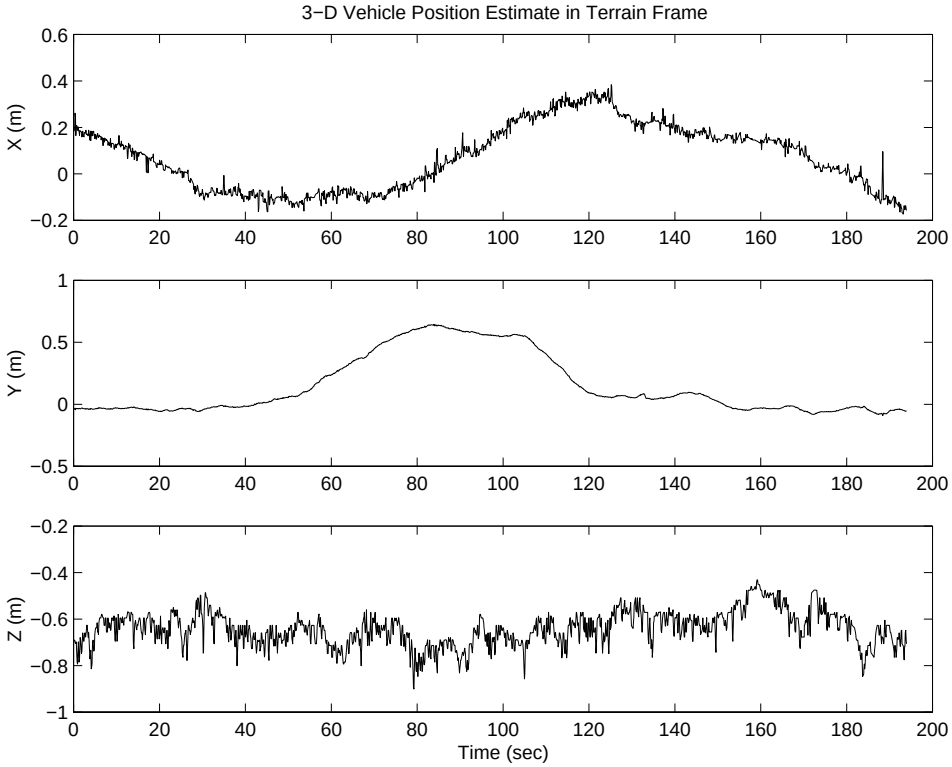

Figure 8.9: **Vehicle Position Estimate from Vision Sensor**

Furthermore, the plot of control authority in Figure 8.11 indicates that the PD controller produces control values that are both reasonable in magnitude and frequency for output to the vehicle thrusters. The PID controller displayed similar response characteristics (Figure 8.12) as the PD controller, and the destabilizing effect of the integrator was minimal, although definitely present. Since no attempts were made at aggressive integral control, the control authority for this controller was reasonable as well (Figure 8.13). It was assumed initially that the sliding mode controller would provide superior control performance; it would maintain reasonable control authority near the desired vehicle path, while providing bang-bang control outside the acceptable control error envelope. Instead, the step increases in control at the edges of the envelope caused sharp increases in vehicle velocity, that in turn degraded the image correlator measurements. This produced significant oscillations in

the output, leading to poor tracking performance (Figure 8.14) and unacceptable variations in control magnitude and frequency (Figure 8.15).

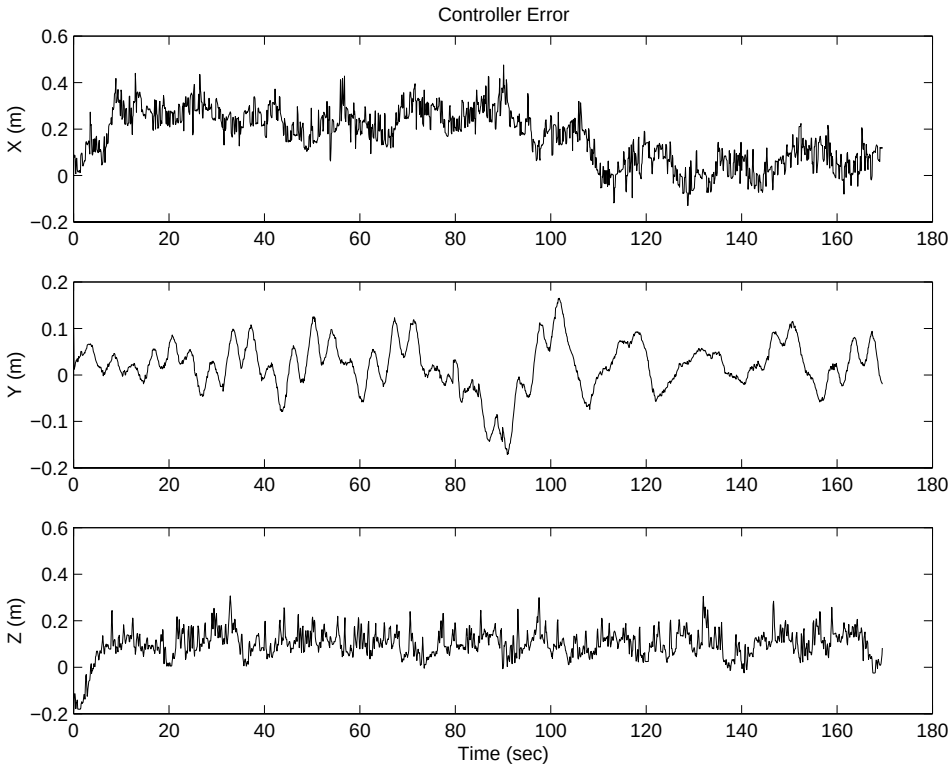

Figure 8.10: **Control Error in Position for PD Controller**

The most important and exciting result of the Ventana experiments on-board the Pt. Lobos was the reaction of the ROV pilots to the new technology. They were very enthusiastic about its potential to improve the daily operations of Ventana, and they demonstrated incredible patience during the debugging and testing phases of this research. Throughout the entire process, they were quite helpful in providing feedback on the practical usefulness of various aspects of the navigation system, and they helped determine which features to add in later revisions of the software. These recommendations were incorporated directly into the design process, leading to a software application that will hopefully be of use to Ventana ROV pilot operations in the near future.

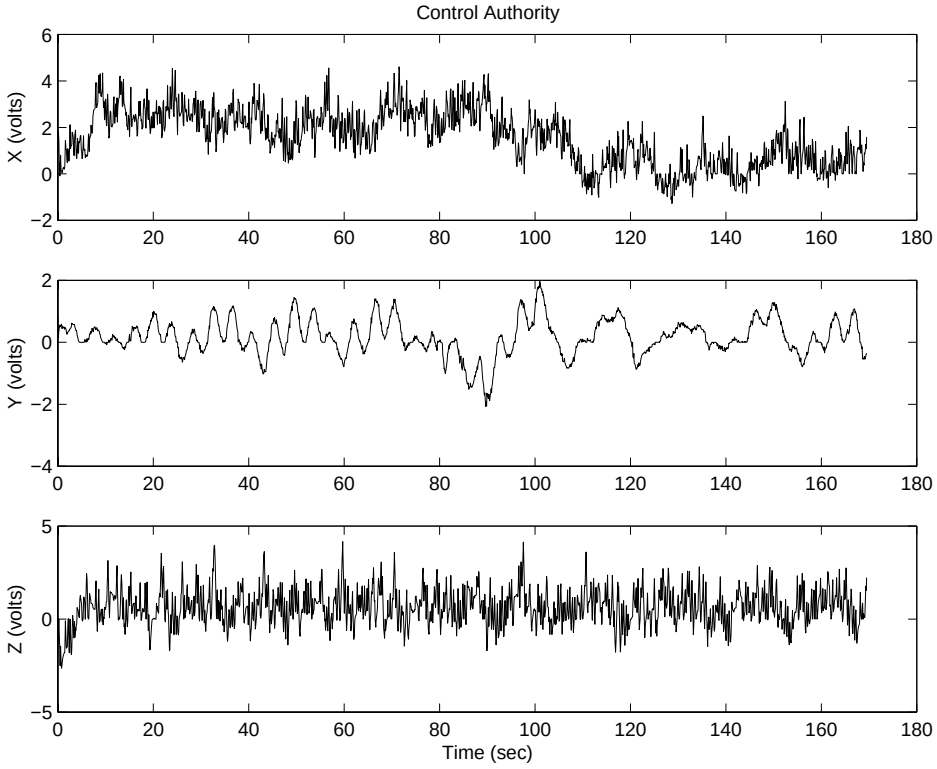

Figure 8.11: **Position Control Authority for PD Controller**

8.4 Summary

The field test results presented in this chapter prove the usefulness and verify the expected performance of the real-time, vision-based navigation capability for underwater robots. The complete system implementation was tested on two separate vehicle platforms in different environments: the OTTER AUV in the test tank, and the Ventana ROV in the open ocean.

The tests on OTTER provided a first-ever demonstration of bounded-error, vision-based navigation on an unmanned underwater vehicle. Specifically, they demonstrate the vision sensor's ability to produce real-time state estimates suitable for vehicle control, reduce mapping error by re-aligning the mosaic images, and provide an image-based interface to the user for goal specification.

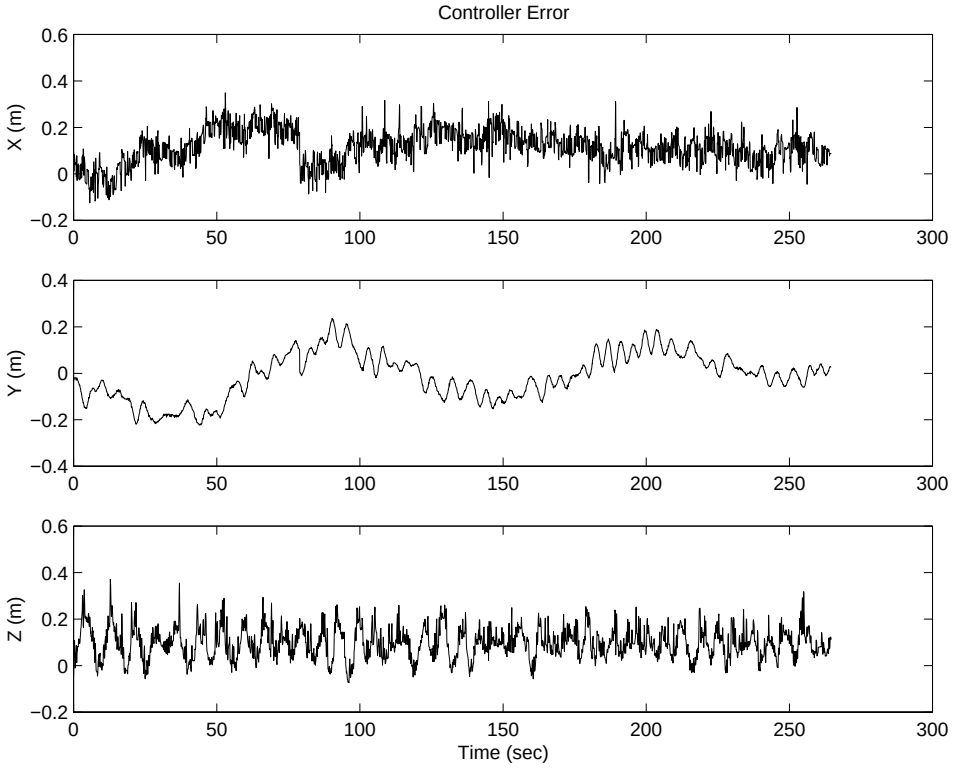

Figure 8.12: **Control Error in Position for PID Controller**

The successful demonstration of the navigation system on Ventana in a real ocean environment represents another first-ever breakthrough in navigation technologies for ocean exploration. This set of experiments exploited the full freedom of the vision sensor assumptions to validate the successful performance of every aspect of the vision sensor, control system, and user interface. In particular, this testing achieved quantitative results on the reliability of the state estimation despite spurious data inputs, and on the vehicle control performance using the vision sensor estimates and GUI reference commands as control system inputs.

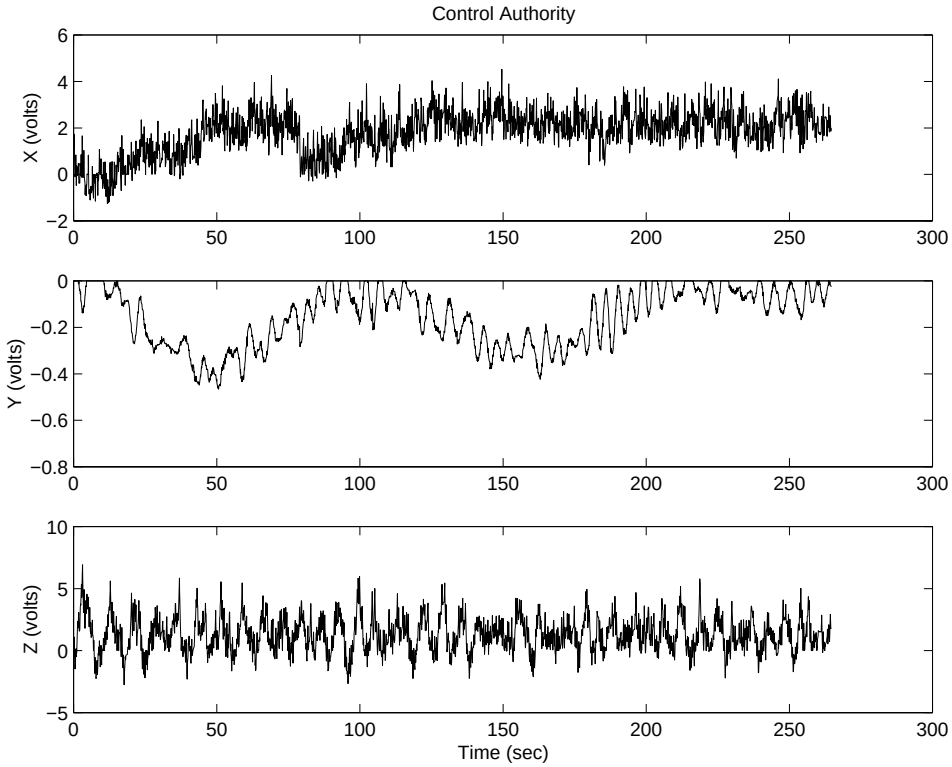

Figure 8.13: **Position Control Authority for PID Controller**

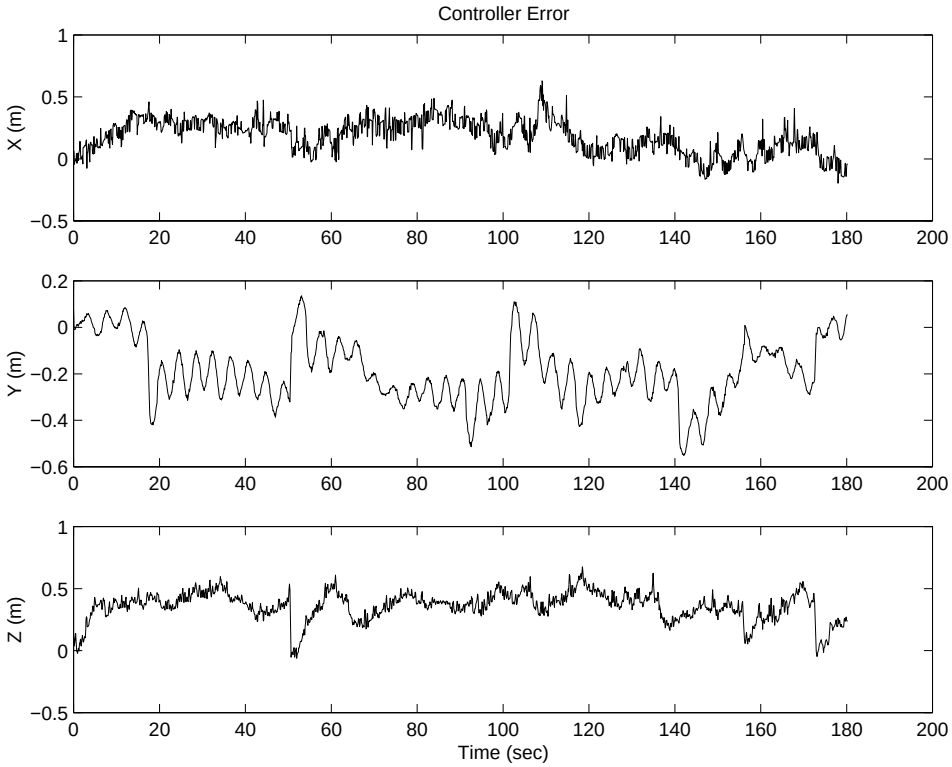

Figure 8.14: **Control Error in Position for Sliding Mode Controller**

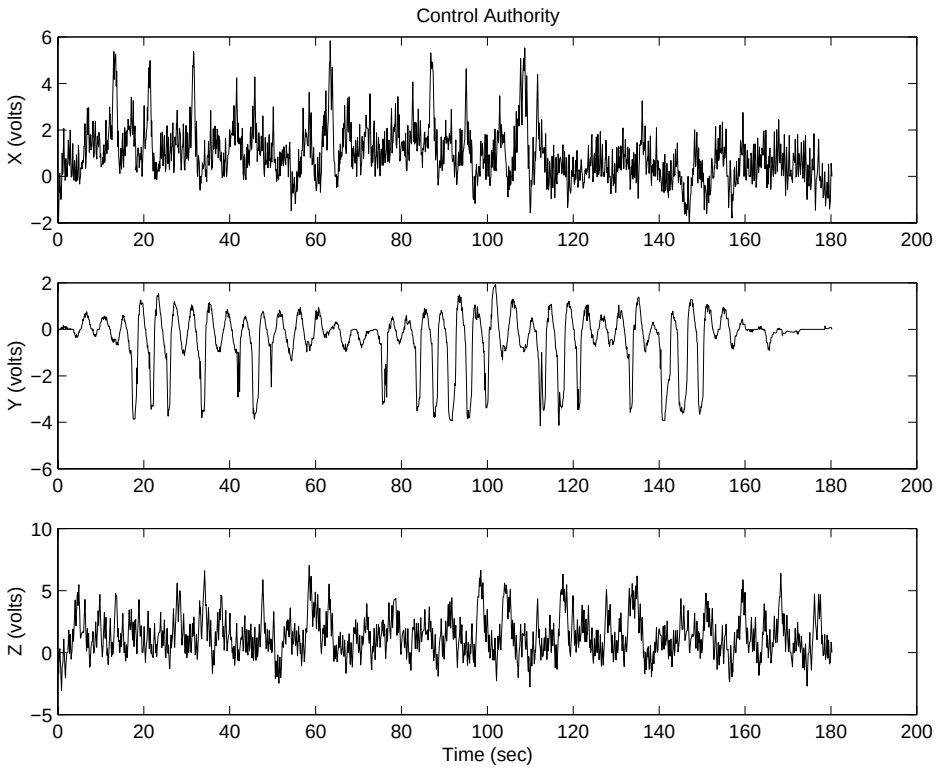

Figure 8.15: **Position Control Authority for Sliding Mode Controller**

Chapter 9

Conclusions

This chapter summarizes the achievements of this work, and it presents several possible ideas for extending the capabilities of the current vision sensing and navigation system in future research efforts.

9.1 Summary of Results and Contributions

The research described in this dissertation has achieved its intended purpose by creating an entirely new visual sensing technology for the remote exploration of unknown, unstructured environments. This novel sensor for mobile robotic platforms, in conjunction with other on-board sensors, simultaneously estimates the global position and attitude of the robot and maps the terrain in real-time. The vision sensing system provides the opportunity to enhance the utility of mobile robots for scientific missions. By integrating the vision sensor with other robot subsystems, new capabilities can be created to achieve higher levels of intelligence and autonomy.

This claim has been validated in this work through the creation and demonstration of one such capability: bounded-error, visual map-based navigation. Specifically, the application of interest is near-bottom ocean exploration by unmanned underwater vehicles. This dissertation has presented the results of several successful demonstrations of autonomous

navigation, performed on the OTTER AUV in the test tank and the Ventana ROV in the deep ocean waters of the Monterey Canyon.

The following two sections review the results and contributions of the two major accomplishments of this research: the development of the vision sensing system, and the demonstration of UUV navigation.

9.1.1 Vision Sensing System

The development of the vision sensing system is the fundamental advancement enabled by this research effort. This system is uniquely beneficial in comparison to other vehicle state sensors, in that it produces two useful sets of information:

Global State Estimates Using the image correlator and a complement of on-board sensors to supply input measurements, real-time estimates of vehicle position and attitude are output at rates up to 30 Hz. These vehicle state estimates are calculated with respect to the mosaic map coordinates. To maintain map self-consistency, the map coordinates are registered with respect to a common global frame of reference. The estimation method is superior to existing dead reckoning techniques in that the magnitudes of both the navigation errors and the absolute position estimation errors are bounded over time and distance, despite the absence of any direct measurements of absolute x, y position.

Video Mosaics Every view of the terrain captured by the vehicle camera is aggregated into a single high-resolution, composite-image map to provide a macroscopic view of the environment. As an offline product, this mosaic is a natural and intuitive representation of the scene data collected by the camera. Moreover, the primary advantage of this mapping system is that the mosaics are available online and dynamically evolve as new sensor data are received: images from unexplored territory increase the coverage area of the mosaic, and overlapping images improve the self-consistency, accuracy, and visual quality of the map through mosaic re-alignment techniques.

The theoretical advances that made it possible to provide this information in real-time represent the core achievements of this research. In other words, the primary technical contribution of this work is the invention of a novel method and structure for bounded-error navigation, vehicle global position estimation, and mosaic creation. As part of this overall contribution, several key developments were achieved during the implementation of the vision sensing stages:

- A kinematic model and an empirical error model were derived to estimate errors in the video mosaicking process. This characterization of the input errors to the vision sensor and the transformation geometry of the scene enables a determination of the errors on the state outputs, and it provides a quantitative basis for optimization techniques to re-align the mosaic.

- An optimal crossover detection and correlation algorithm was developed. This algorithm is essentially an automated procedure that tests for loops in the vehicle path. Since the measurements between overlapping image pairs at the crossover points provide redundant information on the placement of images within the mosaic, these are the critical data in the re-alignment procedure.

- The theory of optimal estimation was adapted to correct image alignment errors on-line. This improves both the visual quality and global position accuracy of the mosaic map. Thus, future estimates of the vehicle global state improve after smoothing the mosaic data.

The results of these theoretical developments were validated on the Space Frame. Truth measurements from the Space Frame were used to compare the predicted and actual errors from typical mosaicking runs, as reported in Chapters 5–7. The experiments verified the reduction in image alignment errors and vehicle state estimation errors as a result of the novel algorithms developed in this research. In particular, the testing demonstrated that errors present in the new vision sensing system were bounded with respect to vehicle path length.

The most important conclusion to be drawn from this work is the importance of utilizing all available information to increase the reliability and accuracy of mosaic alignment and vehicle state estimation. The results of this work demonstrate the marked improvement of bounded-error estimation over previous dead reckoning techniques. Through both the theoretical contributions and experimental results of the vision sensor development, all of the relevant research objectives in Section 1.3 have been achieved.

9.1.2 UUV Navigation

While the vision sensor represents a significant achievement in itself, its impact on marine science, and more generally other mobile robotic applications, is determined by the capabilities it enables. To demonstrate the importance of this novel sensor, the visual navigation task was chosen for implementation on underwater vehicles in the test tank and deep ocean environments. Moreover, this task is useful in ocean operations, beyond its role as a proof-of-concept demonstration for the vision sensor.

The experimental demonstration of bounded-error, mosaic-based navigation is a major contribution of this research, and it is the first time this task has ever been accomplished. The initial phase of testing was performed on the OTTER AUV, in the MBARI test tank facility. Under these controlled conditions, the vision sensor, mosaic-based user interface, and vehicle control system were integrated into a unified navigation architecture, and all subsystems were verified to be operating properly. The final testing phase utilized the Ventana ROV to demonstrate the visual navigation task on an operational system in the deep ocean waters of the Monterey Canyon. During this demonstration, the effect of the optimal estimation techniques on image position accuracy was evidenced by the visual improvement in mosaic quality. Most importantly, the ROV pilots deemed the testing a major success, and have expressed serious interest in incorporating the navigation system on a permanent basis for use in daily ROV operations. Chapter 8 provided a detailed review of the results achieved from both sets of experiments.

The success of these visual navigation demonstrations verify that both relevant research objectives in Section 1.3 have been achieved, and it has led to several interesting conclusions.

The OTTER and Ventana tests have proven the feasibility of vision-based navigation, and they have established this method as a viable option for directing the motions of unmanned underwater vehicles. More generally, this research has shown that the advantages of vision-based sensing are particularly beneficial for the task of navigation: an on-board camera is the only requirement for the remote exploration of unknown, unstructured environments, and the intuitive nature of visual data is ideal for human-robot interaction.

9.2 Suggestions for Future Work

The research described in this dissertation has enabled the development of a first-generation vision sensor and bounded-error navigation system. As is the case for any invention, additional developments can create incremental improvements that would benefit future generations of the product. The following sections describe possible extensions to the current contributions of this work that build upon the successful outcome of this research.

9.2.1 Robust Computer Vision Methods

Due to the continual increase in processor speeds for the same cost, more computational power is available today than was utilized during the course of this research. As a result, these new computational capabilities enable more sophisticated methods to be implemented as part of the vision sensing system, while still satisfying the constraint of real-time performance. The following two extensions to the computer vision techniques currently utilized in the vision sensor have the potential to increase the robustness of the image correlator measurements substantially, thereby reducing the associated measurement errors.

Geometric Image Information Extraction Methods

New computer vision methods for geometric image information extraction, such as those described in Section 3.5, could be implemented. These low-level image processing algorithms would replace the current solution described in Section 3.4. While data-intensive

image correspondence methods and more complex transformation models would require significantly more calculations, they have the potential to provide several benefits. Increases in the amount of data analyzed and the model detail could lead to both improvements in measurement robustness and increases in measurement confidence and accuracy. Furthermore, instead of relying only on other on-board sensors for the measurement of range, yaw, pitch, and roll, some or all of these additional degrees of freedom could be provided through image registration. These new vision measurements could either replace the other sensor data or be fused with the redundant measurements to form more robust estimates.

Parameter Adaptation

For all of the image registration methods considered in this dissertation, their performance is highly dependent on the scene content. If an input image could be analyzed to determine its spectrum of spatial frequencies, this would provide valuable information concerning the probability of success of image correspondence, whether it is based on edges, features, intensities, or texture. More specifically, this frequency analysis could be utilized to adapt the free parameters of the correspondence method to optimize performance. For instance, the texture-based correspondence method requires tuning of several parameters, such as the correlation window size, correlation window location, and Gaussian filter width. These parameters could be tuned adaptively, based on online image analysis, to maintain optimal robustness and accuracy as the scene content changes over time.

9.2.2 Extensions to the Vision Sensing System

In transforming the vision sensing system from a successful prototype to a tool for operational use, several modifications may be suggested to enhance its usefulness for scientific missions. The following sections describe a couple ideas for extending the capabilities of the current system.

Methods for Reduction in Computation

During the course of this research, the optimal estimation method that was implemented for the experiments exceeded the online performance goals, and several more methods were derived to achieve significant gains in computational efficiency. However, for large enough mosaics and/or sufficiently complex dynamic vehicle models, it is possible that the methods outlined in this thesis will eventually fail to meet online performance specifications. For these extreme cases, alternate methods could be developed that provided sub-optimal mosaic re-alignment, but at a greatly reduced computational cost. For instance, the path-based method of Section 7.3.1 relies on the construction of a path tree from the origin to all nodes, in order to compute a weighted average of all paths to the node. Various tree pruning methods could be devised to remove high-variance paths from the tree, thereby reducing the required calculations while minimally impacting the node position estimates.

Fusion of Multiple Mosaics

In its current configuration, the vision sensor constructs a mosaic of the scene when the mosaicking mode of the sensor is enabled, then terminates the mosaic when the sensor switches to its idle mode. The mosaic can be stored for later use, but it is no longer dynamic, and it can no longer be used as a real-time reference map for active vehicle control and navigation. An extremely useful feature that could be developed is the ability to combine multiple overlapping mosaics taken at different times, to form a single connected mosaic of the scene. This may require an exhaustive offline search and/or manual intervention to align the mosaics. If this new mosaic map could then be re-loaded for real-time correlation with live images from the vehicle, this would enable the vision sensor both to re-align and expand any previously created mosaic, whenever the underwater robot returned to the same scene.

9.2.3 Novel Navigation Techniques

The final set of possibilities for future work entails further development of the demonstration of bounded-error, vision-based navigation. The two suggestions to follow take opposite approaches in modifying the current navigation scheme: the first strives to increase the allowable vehicle motions that the pilot or scientist is able to control, while the second proposes to automate the process to maximize the mosaic accuracy.

4-DOF Navigation

In the current design of the navigation task, the human user is able to direct the underwater robot within a plane parallel to the terrain, and the remaining four degrees of freedom are either passively stable or are handled by the automatic control system. These two translational degrees of freedom represent significantly less freedom of motion than the user is normally given for normal remote operations. Thus, it would be greatly beneficial if the pilot or scientist were given the freedom to manage the vehicle heading and altitude as well. In order to enable 4-DOF navigation, the low-level computer vision methods would need to replaced. As previously discussed, more sophisticated image registration methods would enable correlation of image pairs that are both scaled and rotated with respect to each other. By relaxing these control constraints on the vehicle, it would enable the navigation software to be used as an ROV pilot aid that would not limit the vehicle capabilities available to the pilot.

Optimal Coverage Patterns

For some navigation applications, the goal may be to create a map of the ocean floor that is as accurate as possible. In this case, the vehicle path specification could be completely automated to ensure complete coverage of the area. Optimal coverage patterns could be derived that achieve the most accurate mosaics, given the constraints of the mosaic re-alignment process, the vehicle, and the environment.

The development of a real-time, bounded-error, vision-based sensor for navigation of AUV's has demonstrated the potential for visual sensing in the underwater environment. As the navigation task is brought to maturity and new capabilities are created using the vision sensor, the contributions of this dissertation to marine science and mobile robotics will come to full fruition.

Appendix A

Additional Video Mosaics

A.1 Introduction

This appendix provides a collection of video mosaics that have been created during the course of this research. Most of the mosaics were created in real-time, using various versions of the Sensor mosaicking and navigation software application. Some of the earlier smoothed mosaics were first created by the Sensor application, then re-aligned offline in MATLAB or C++. All of the mosaics consist of real images collected during the experimental phase of this work, and they have been organized based on the environments in which the mosaicking experiments were conducted.

A.2 Mosaics

The mosaics have been categorized according to the five different environments in which they were constructed: the laboratory (using a Sony videocamera and tripod), outdoors (using the same Sony videocamera and tripod), the Space Frame, the OTTER AUV, and the Ventana ROV.

A.2.1 Lab Mosaics

Figure A.1: **Single-Column Mosaic of ARL (Room 017)**

Figure A.2: **Multiple-Column Mosaic of ARL (Room 017)**

Figure A.3: **Multiple-Column Mosaic of ARL (Room 017)**

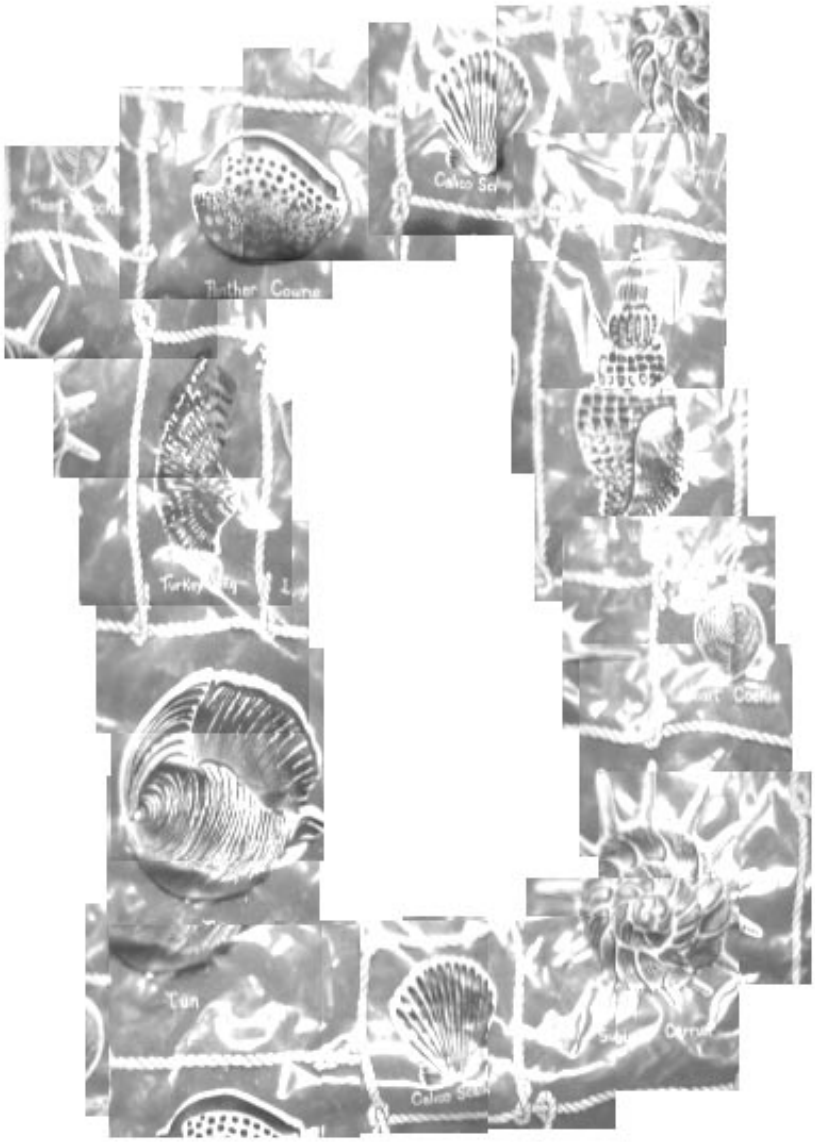

Figure A.4: **Single-Loop Mosaic (Before Smoothing) in MBARI lab**

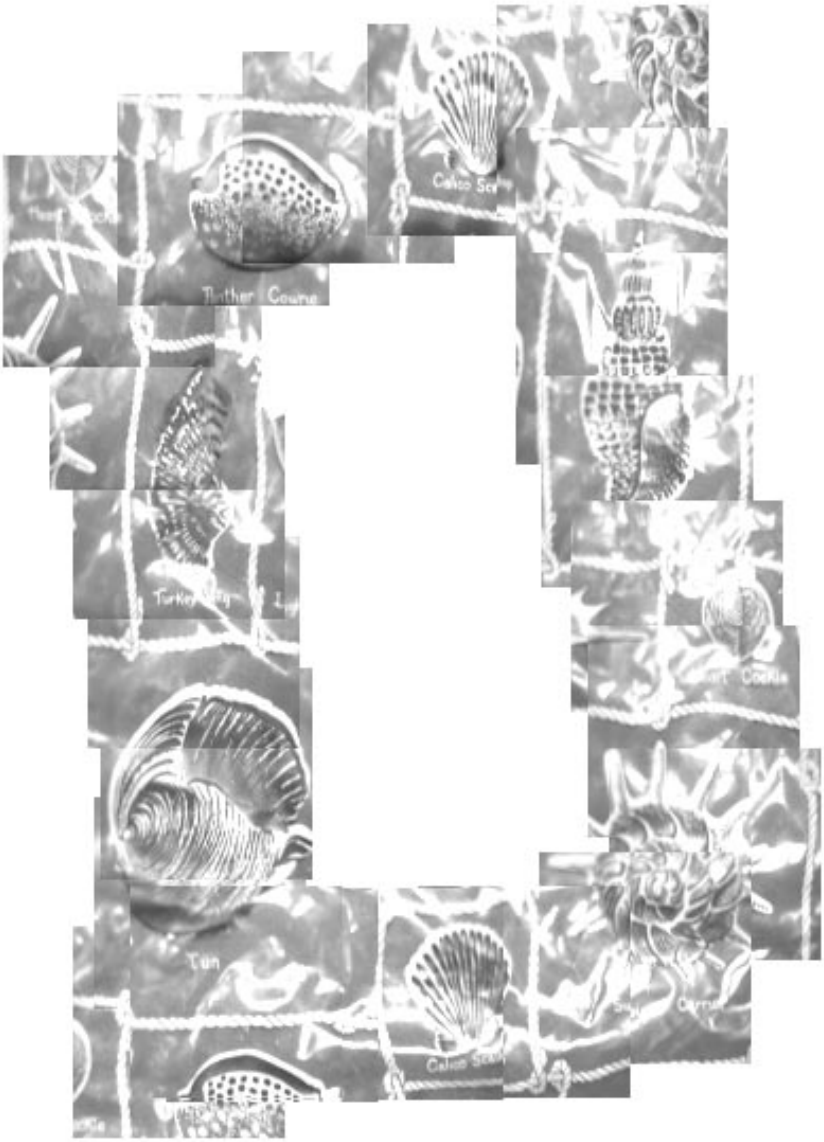

Figure A.5: **Single-Loop Mosaic (After Smoothing) in MBARI lab**

A.2.2 Outdoor Mosaics

Figure A.6: **Mosaic of Stanford Campus (Rains) Before Smoothing**

Figure A.7: **Mosaic of Stanford Campus (Rains) After Smoothing**

A.2.3 Space Frame Mosaics

Figure A.8: **Mosaic Created on the Space Frame**

A.2.4 OTTER Mosaics

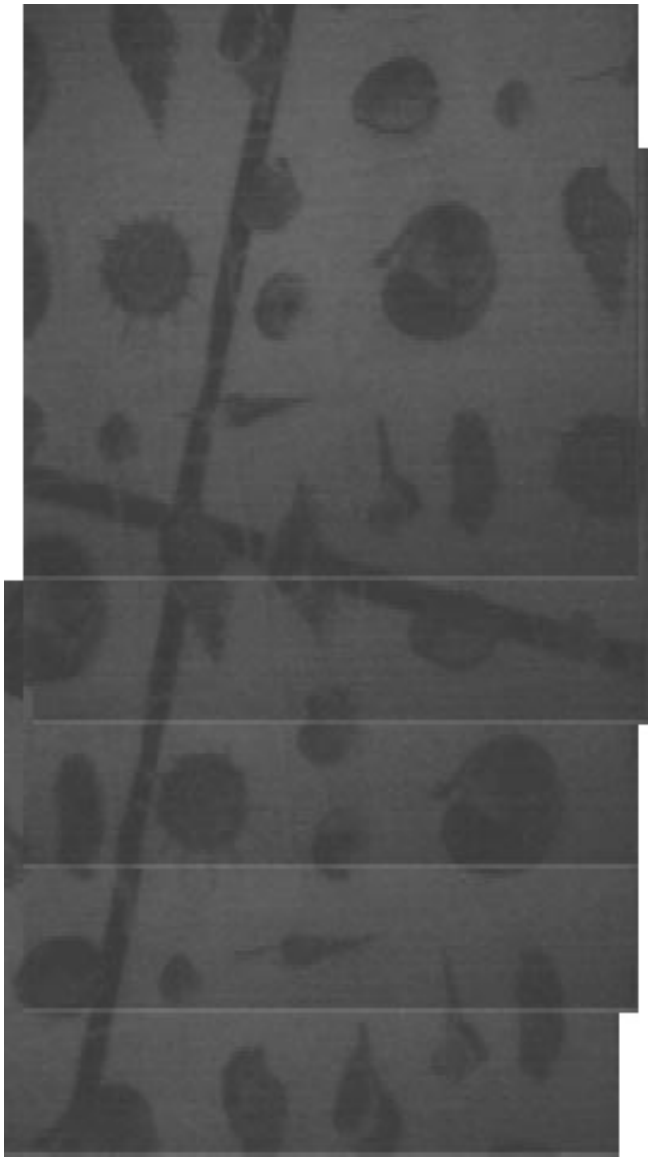

Figure A.9: **Mosaic Created During Vision-Based Control of OTTER**

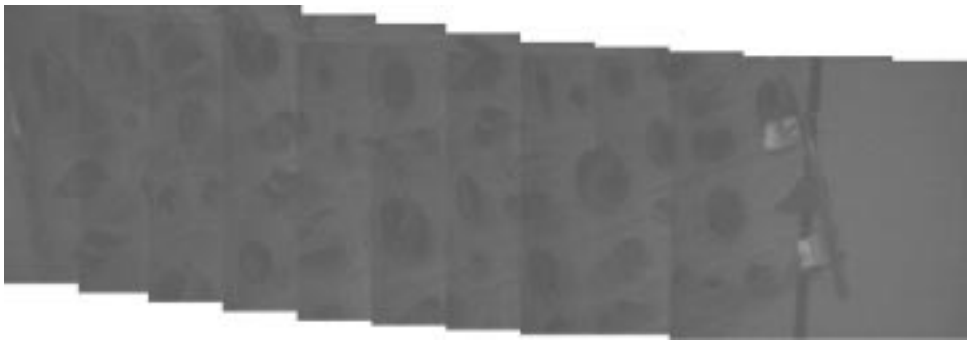

Figure A.10: **Mosaic Created During Vision-Based Control of OTTER**

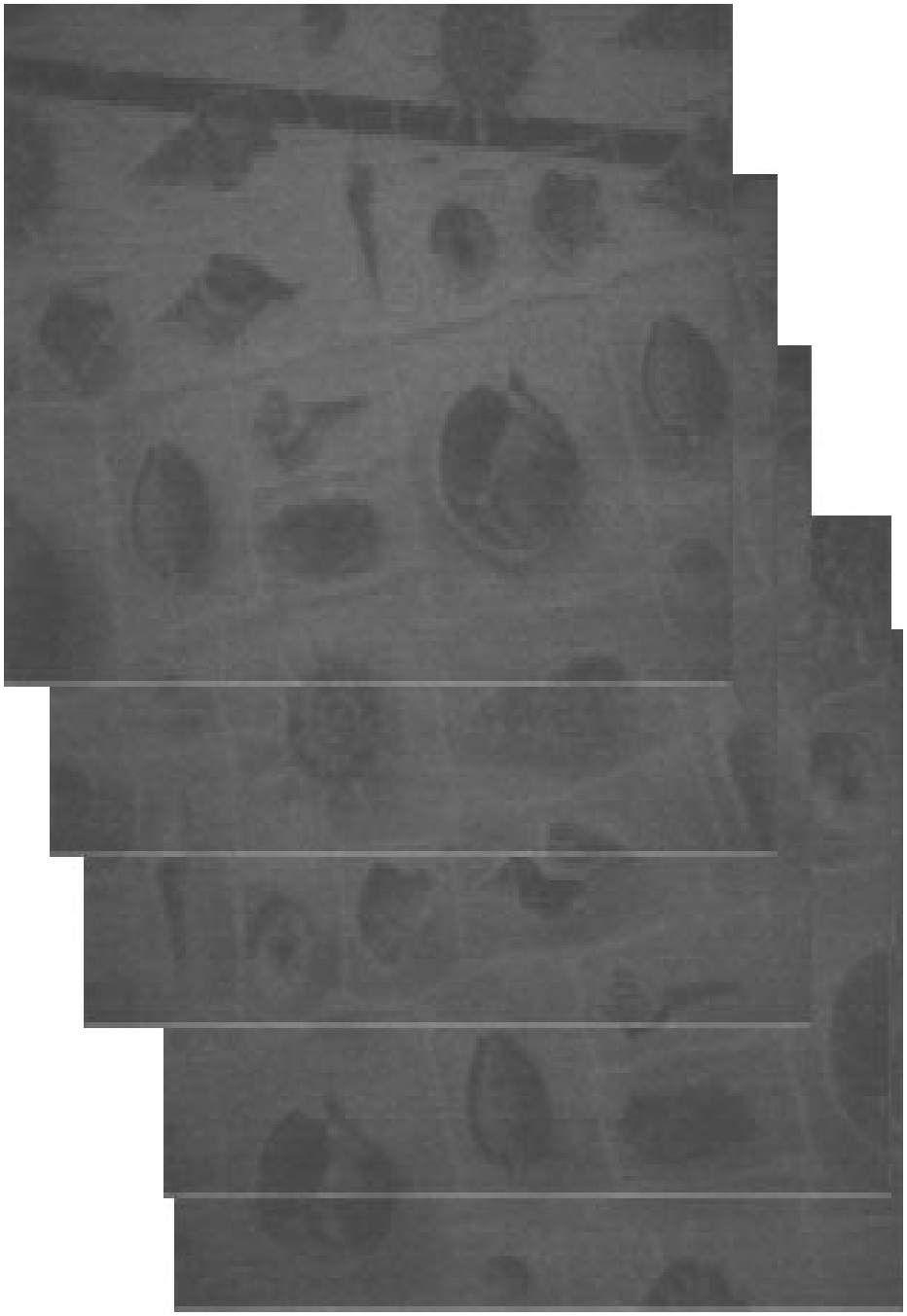

Figure A.11: **OTTER Mosaic Created During Autonomous Mosaicking Mission**

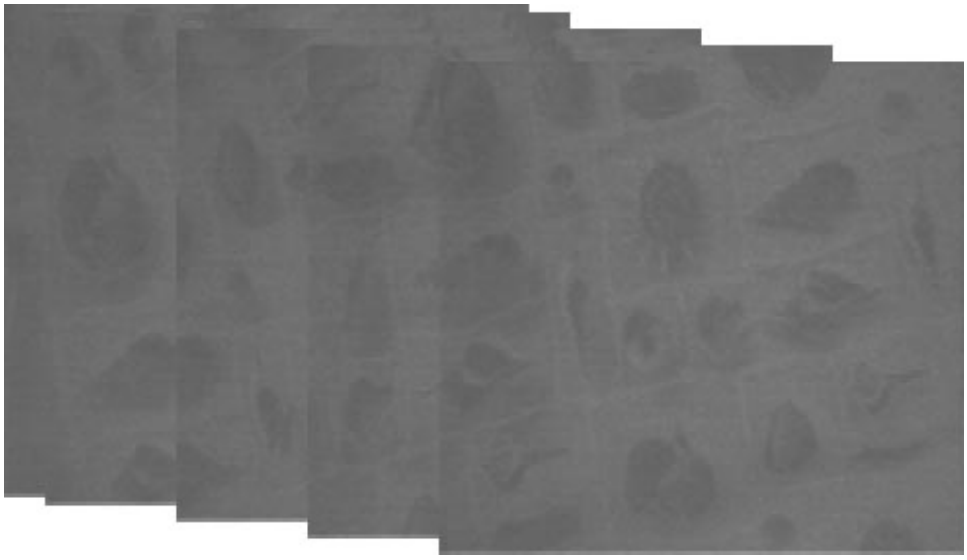

Figure A.12: **OTTER Mosaic Created During Autonomous Mosaicking Mission**

Figure A.13: **OTTER Mosaic Created During SHARPS Navigation**

Figure A.14: **OTTER Mosaic Created During SHARPS Navigation**

Figure A.15: **OTTER Mosaic Created During Dead-Reckoned Navigation**

A.2.5 Ventana Mosaics

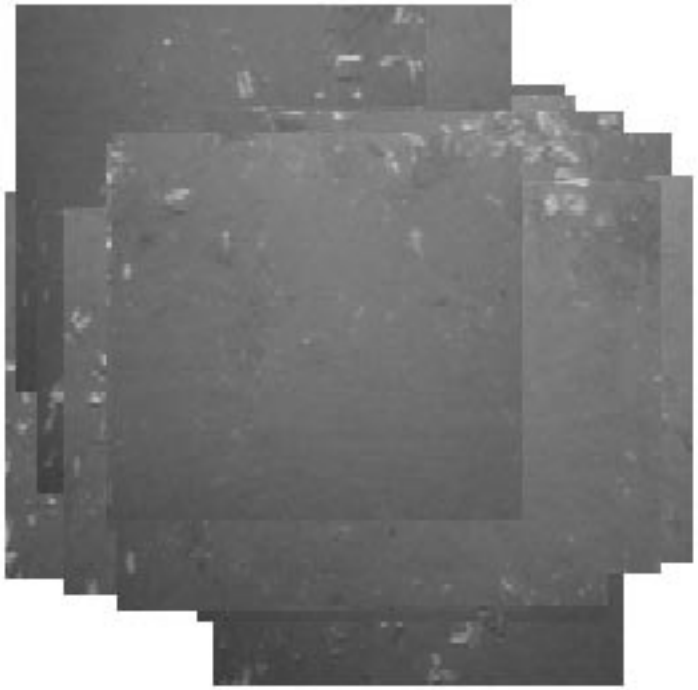

Figure A.16: **Mosaic of Brachipods from Ventana**

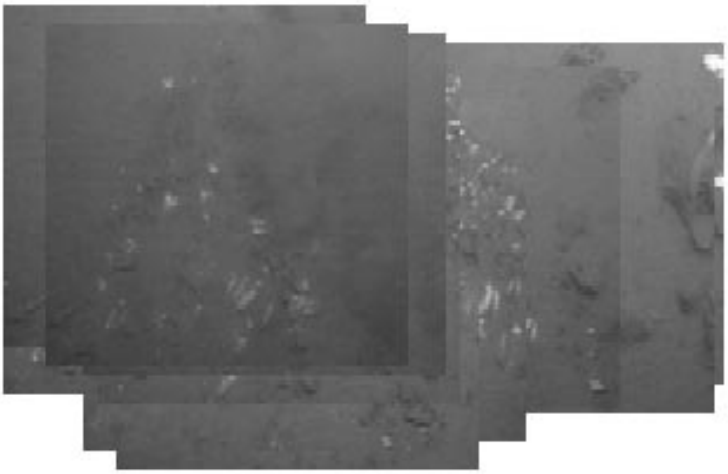

Figure A.17: **Mosaic of Brachipods from Ventana**

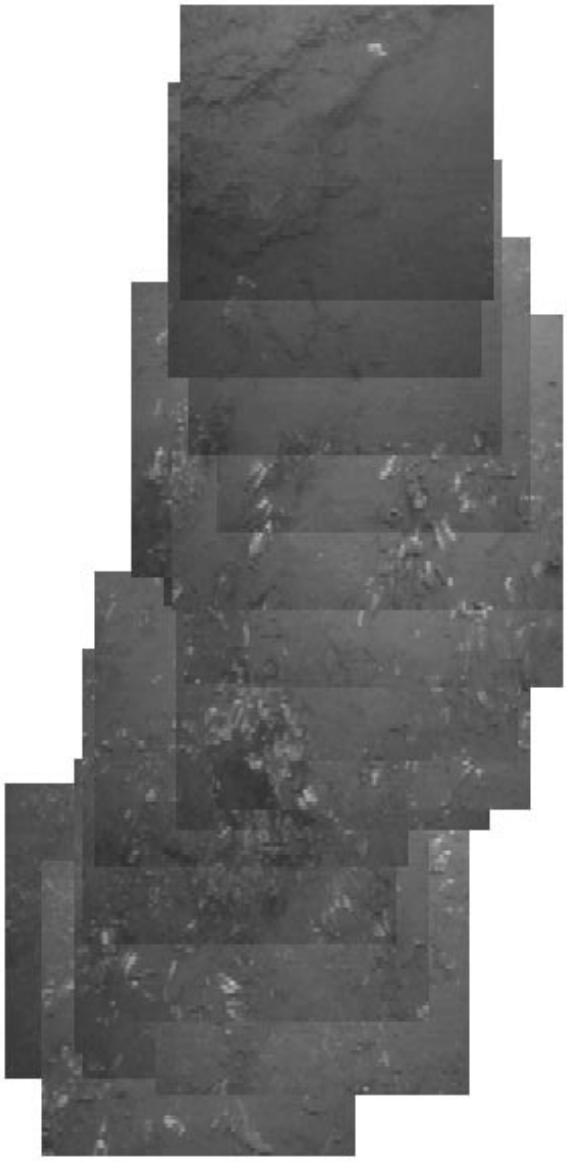

Figure A.18: **Mosaic of Brachipods from Ventana**

Figure A.19: **Mosaic of Brachipods with Crossover**

Bibliography

[1] W.B. Bell. Image correlation under full-perspective distortion. In *Visual Information Processing V*, pages 38–49, Orlando, FL, April 1996. SPIE.

[2] A. Bryson. Dynamic Optimization with Uncertainty. Paper in progress, March 1993.

[3] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8:679–698, November 1986.

[4] J. Craig. *Introduction to Robotics.* Addison-Wesley Publishing Company, 2nd edition, 1989.

[5] D. Greenwood. *Principles of Dynamics.* Prentice-Hall, Inc., 2nd edition, 1988.

[6] S. Hert, S. Tiwari, and V. Lumelsky. A Terrain-Covering Algorithm for an AUV. *Autonomous Robots*, 3(2-3):91–119, June-July 1996.

[7] Andreas Huster, Stephen D. Fleischer, and Stephen M. Rock. Demonstration of a vision-based dead-reckoning system for navigation of an underwater vehicle. In *Proceedings of the OCEANS 98 Conference*, Nice, France, September 1998. IEEE. submitted.

[8] J. Illman, H. Milburn, and R. Macdonald. Integrated system for navigation and positioning of an rov for scientific exploration. In *Proc. IEEE Oceans*, pages 499–503, 1993.

[9] M. Irani and S. Peleg. Improving resolution by image registration. *CVGIP: Graphical Models and Image Processing*, 53:3:231–239, 1991.

[10] L. Jin. Real-Time Vision-Based Stationkeeping System For Underwater Robotics Applications. In *Proceedings of the OCEANS 96 Conference*. MTS/IEEE, 1996.

[11] K. N. Leabourne, S. M. Rock, S. D. Fleischer, and R. L. Burton. Station Keeping of an ROV Using Vision Technology. In *Proceedings of the OCEANS 97 Conference*, Halifax, Nova Scotia, October 1997. MTS/IEEE.

[12] R. L. Marks, M. J. Lee, and S. M. Rock. Visual sensing for control of an underwater robotic vehicle. In *Proceedings of IARP Second Workshop on Mobile Robots for Subsea Environments*, Monterey, May 1994. IARP.

[13] R. L. Marks, S. M. Rock, and M. J. Lee. Automatic object tracking for an unmanned underwater vehicle using real-time image filtering and correlation. In *Proceedings of IEEE Systems, Man, and Cybernetics*, France, October 1993. IEEE.

[14] R. L. Marks, S. M. Rock, and M. J. Lee. Real-time video mosaicking of the ocean floor. In *Proceedings of IEEE Symposium on Autonomous Underwater Vehicle Technology*, Cambridge, MA, July 1994. IEEE.

[15] R. L. Marks, H. H. Wang, M. J. Lee, and S. M. Rock. Automatic visual station keeping of an underwater robot. In *Proceedings of IEEE Oceans 94 Osates*, Brest, France, September 1994. IEEE.

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

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

[18] D. Marr and E. Hildreth. Theory of edge detection. *Proc. of the Royal Society of London*, pages 187–217, 1980.

[19] Timothy W. McLain. *Modelling of Underwater Manipulator Hydrodynamics with Application to the Coordinated Control of an Arm/Vehicle System.* PhD thesis, Stanford University, August 1995. Also published as SUDAAR 670.

[20] V. Nalwa. *A Guided Tour of Computer Vision.* Addison-Wesley Publishing Company, 1993.

[21] S. Negahdaripour. Passive Optical Sensing For Near-Bottom Stationkeeping. In *Proceedings of the OCEANS 90 Conference.* OES/IEEE, 1990.

[22] S. Negahdaripour. Undersea Optical Stationkeeping. Improved Methods. In *Journal of Robotic Systems*, June 1991.

[23] H.K. Nishihara. Prism: a practical realtime imaging stereo matcher. In *SPIE*, pages 134–142, 1983.

[24] H.K. Nishihara. Practical realtime imaging stereo matcher. *Optical Engineering*, 23(5):536–545, 1984. Also in *Readings in Computer Vision: issues, problems, principles, and paradigms.*

[25] F. Parthiot and J. Denis. A better way to navigate on deep sea floors. In *Proc. IEEE Oceans*, pages 494–498, 1993.

[26] S. Peleg and J. Herman. Panoramic mosaics by manifold projection. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 338–343, San Juan, PR, June 1997. IEEE.

[27] J. Pitman. Probability. Book in progress, January 1993.

[28] Heung-Yeung Shum and R. Szeliski. Construction and refinement of panoramic mosaics with global and local alignment. In *Proceedings of IEEE 6th International Conference on Computer Vision*, pages 953–956, Bombay, India, January 1998. IEEE.

[29] G. Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, 1986.

[30] Zhaohui Sun and A.M. Tekalp. Reconstruction of 3-d affine and euclidean mesh models from video. In *Visual Communications and Image Processing '98*, pages 820–830, San Jose, CA, January 1998. SPIE.

[31] R. Szeliski and Sing Bing Kang. Direct methods for visual scene reconstruction. In *Proceedings IEEE Workshop on Representation of Visual Scenes*, pages 26–33, Cambridge, MA, June 1995. IEEE.

[32] S. Tiwari. Mosaicking of the Ocean Floor in the Presence of Three-Dimensional Occlusions in Visual and Side-Scan Sonar Images. In *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology*. OES/IEEE, June 1996.

[33] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9:2:137–154, November 1992.

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

[35] S. Wasielewski. Dynamic Vision for ROV Stabilization. In *Proceedings of the OCEANS 96 Conference*. MTS/IEEE, 1996.

[36] X. Xu and S. Negahdaripour. Vision-based Motion Sensing for Underwater Navigation and Mosaicing of Ocean Floor Images. In *Proceedings of the OCEANS 97 Conference*. MTS/IEEE, 1997.

[37] J. Yuh. Control and Optical Sensing in Underwater Robotic Vehicles (URVs). In *Proceedings of the OCEANS 90 Conference*. OES/IEEE, 1990.