

## ABSTRACT

Title of Thesis: OPTIMAL VISION-BASED POSITION  
ESTIMATION OF AN UNDERWATER  
SPACE SIMULATION ROBOT

Jeffrey Russell Smithanik  
Master of Science, 2004

Thesis Directed By: Assistant Professor Ella M. Atkins  
Associate Professor Robert M. Sanner  
Department of Aerospace Engineering

This thesis describes the development of the Vision Positioning System (VPS), a real-time 3-D inertial translational state estimation system for free-flying neutral buoyancy space simulation robots. Key contributions include a technique for the accurate calibration of long-baseline underwater vision systems, and a three degree of freedom Extended Kalman Filter (EKF) that merges camera measurements with robot telemetry to create an optimal estimate of 3-D translational position and velocity. Results from static and dynamic underwater positioning tests are presented that characterize the system accuracy. Static tests indicate VPS is capable of locating the robot with 3 to 4 cm accuracy, while dynamic test results show similar accuracy given ideal lighting conditions and flight in a region of complete camera coverage. The results indicate that with further development to correct for lighting and better reject erroneous camera measurements, VPS has the potential for accuracy comparable to that achieved by GPS navigation systems.

OPTIMAL VISION-BASED POSITION ESTIMATION OF AN UNDERWATER  
SPACE SIMULATION ROBOT

By

Jeffrey Russell Smithanik

Thesis submitted to the Faculty of the Graduate School of the  
University of Maryland, College Park, in partial fulfillment  
of the requirements for the degree of  
Master of Science  
2004

Advisory Committee:

Assistant Professor Ella M. Atkins, Co-Chair

Associate Professor Robert M. Sanner, Co-Chair

Associate Professor David L. Akin

© Copyright by  
The Space Systems Laboratory  
The University of Maryland, College Park  
2004

## Dedication

*I dedicate this thesis to my precious wife Nicole*

*Thank you, with all my heart*

## Acknowledgements

Many thanks are deserved by many people who all contributed in very significant ways, both directly and indirectly, to the work described in this thesis. First, my gratitude must go to Drs. Ella Atkins and Rob Sanner, who supported me patiently and persistently, and provided independence and oversight in a ratio that was just right. I especially want to thank Ella for her guidance, sacrifice and friendship - and for not giving up on me when she found to her horror in my first week at the SSL that I did not know what `grep` meant! I want to thank Dr. David Akin for his kind mentorship and for the privilege of working in the fantastic environment that is the SSL. I also am thankful to Rhiannon Peasco and her undergrads, from whom I inherited VPS, and upon whose excellent work this thesis is built and without which none of this would have been possible. A huge thank-you goes to the star undergrads Cat McGhan (the mother of `VPS_client`), Mike Naylor and Tim Wasserman, who contributed to VPS enormously with their bodies and minds, and at times suffered various sorrows without complaint as reward for being on my team. My friends, thank-you. I gratefully thank all of those (you are too numerous for all to be named) who supported me on my many VPS and SSV dives, especially Elisa, Daniel, Emily, Dave H., Jeff B. and Meghan. Thanks for all the happy and fun hours in the water! I also am indebted especially to Nick and Mike for their enormous last minute efforts at creating the simulator and gathering and plotting the results at the end of Chapter 7. I owe you a favor, and I expect you to collect some day. Finally, thank-you to all the SSLers that I have worked with over these past years for the friendship, fantastic

conversations, support, and for sharing all that you have taught me. The SSL is a special place, one of a kind in the world not only for its facilities, but for its people as well. It was an honor and a dream come true to work in such an environment. Thank-you all.

# Table of Contents

Dedication	ii
Acknowledgements	iii
Table of Contents	v
List of Tables	ix
List of Figures	xi
Nomenclature	xvii
Acronyms	xvii
Symbols	xviii
Chapter 1	1
Introduction	1
1.1    Description of Problem	3
1.2    Solution	6
1.3    Outline	7
Chapter 2	9
Vision Positioning System Background	9
2.1    System Hardware	9
2.1.1    Description of Cameras and Camera Coverage	10
2.1.2    Description of SCAMP SSV	16
2.1.3    Description of VPS Computer Hardware	20
2.2    System Software Overview	22
2.2.1    VPS_client	22

2.2.2	Control Station Software (Raptor)	24
2.2.3	SCAMP SSV Flight Code	24
Chapter 3		25
Camera Model and Calibration Background		25
3.1	Camera Calibration and Camera Models	25
3.2	Mathematical Development of the VPS Camera Model	28
3.2.1	Step One: $[X_G Y_G Z_G]^T$ to $[X_C Y_C Z_C]^T$ Conversion	29
3.2.2	Step Two: $[X_C Y_C Z_C]^T$ to $[X_U Y_U A]$ Conversion	31
3.2.3	Step Three: $[X_U Y_U A]$ to $[X_D Y_D A]$ Conversion	33
3.2.4	Step Four: $[X_D Y_D A]$ to $[X_{FD} Y_{FD} A]$ Conversion	36
3.3	General Approach To Camera Calibration	40
Chapter 4		45
Calibration of VPS		45
4.1	Motivation for a New Calibration Technique	45
4.2	Overview of New Calibration Technique	47
4.3	VPS Intrinsic Calibration	48
4.3.1	VPS Intrinsic Calibration: Results And Analysis	53
4.4	VPS Extrinsic Calibration	57
4.4.1	VPS Extrinsic Calibration: Results and Analysis	78
Chapter 5		81
VPS Software Components		81
5.1	Image Acquisition and Processing Program: VPS_client	81
5.1.1	VPS_client Initialization	83



5.1.2	Image Acquisition and Processing	84
5.2	Control Station Software (Raptor)	95
5.3	SCAMP SSV Flight Software	98
Chapter 6		100
Optimal State Estimation and the Kalman Filter		100
6.1	Optimal Estimation Theory	100
6.1.1	Introduction to State Estimation	100
6.1.2	The Kalman Filter	108
6.1.3	The Extended Kalman Filter	115
6.2	Implementation of VPS Extended Kalman Filter	116
6.3	Noise Characterization	126
6.3.1	Process Noise Characterization	126
6.3.2	Measurement Noise Characterization	129
6.3.3	VPS EKF Implementation on the Control Station Computer	132
Chapter 7		135
Positioning Results		135
7.1	Static Positioning Results	136
7.1.1	Static Position of Hanging Ball, Double EKF	136
7.1.2	Static Position of SCAMP SSV, Single EKF	139
7.1.3	Static Position of SCAMP SSV, Double EKF	143
7.2	Dynamic State Evolution With No Measurement Data	144
7.3	Dynamic SCAMP SSV State Estimation, Single EKF	151
7.4	Dynamic SCAMP SSV State Estimation, Double EKF	168

7.4.1	Non-Ideal Test Conditions	168
7.4.2	Near-Ideal Test Conditions	178
Chapter 8		190
Conclusion and Future Work		190
8.1	Conclusion: VPS Calibration	191
8.2	Conclusion: VPS Extended Kalman Filter	192
8.3	Future Work	194
8.3.1	Hardware and Operational Upgrades	195
8.3.2	Further Testing and Parameter Updates	197
8.3.3	VPS_client and Raptor/EKF Enhancements	199
8.3.4	Future Research Applications for VPS	205
Appendix		207
Bibliography		211

## List of Tables

Table 2.1 Legend for tank coverage maps	13
Table 3.1 List of intrinsic VPS calibration parameters	27
Table 3.2 List of extrinsic VPS calibration parameters	28
Table 3.3 Data transition states, from 3D global coordinates to 2D computer image coordinates	29
Table 4.1 VPS intrinsic calibration results	55
Table 4.2 Error bounds on the VPS intrinsic calibration parameters	55
Table 4.3 Target 3D initial position estimates	68
Table 4.4 Target 3D position estimates, after optimization	69
Table 4.5 Optimized, final 3D target positions	72
Table 4.6 VPS extrinsic calibration results	79
Table 4.7 Error metrics for VPS extrinsic calibration	80
Table 5.1 vps_data structure sent to raptor	93
Table 5.2 global_data structure sent to VPS_client	94
Table 7.1 Double EKF Architecture	137
Table 7.2 Double EKF static black ball position estimates, three positions	138
Table 7.3 Double EKF static black ball position standard deviations, three positions	138
Table 7.4 Static SCAMP SSV position data, using all 6 cameras	139

Table 7.5 Static SCAMP SSV standard deviation data, using all 6 cameras	140
Table 7.6 Position-to-position distance values: actual and as calculated by VPS	140
Table 7.7 Static SCAMP SSV position data, double EKF	143
Table 7.8 Static SCAMP SSV standard deviation data, double EKF	143
Table 7.9 Approximate force directions in SCAMP SSV body and global frames	145
Table 7.10 Deviation between predicted and measured camera data	185
Table 7.11 Deviations between 2-camera EKF and 6-camera EKF simulations	189

## List of Figures

Figure 1.1 SCAMP SSV in free flight	5
Figure 2.1 Bottom ring VPS camera	12
Figure 2.2 Top ring VPS camera	12
Figure 2.3 Camera coverage, depth = 3'	13
Figure 2.4 Camera coverage, depth = 7'	14
Figure 2.5 Camera coverage, depth = 12.5'	14
Figure 2.6 Camera coverage, depth = 18'	15
Figure 2.7 Camera coverage, depth = 22'	15
Figure 2.8 SCAMP SSV above the NBRF	17
Figure 2.9 SCAMP SSV with side access covers removed	18
Figure 2.10 SCAMP SSV control station, ReCS	19
Figure 2.11 SCAMP SSV control station GUI	20
Figure 2.12 VPS layout diagram	23
Figure 3.1 Conversion from global coordinates $[X_G Y_G Z_G]^T$ to camera coordinates $[X_C Y_C Z_C]^T$	30
Figure 3.2 Conversion from camera coordinates $[X_C Y_C Z_C]^T$ to ideal image coordinates $[X_U Y_U A]$	32
Figure 3.3 Conversion from ideal image coordinates $[X_U Y_U A]$ to distorted image coordinates $[X_D Y_D A]$	33
Figure 3.4 Conversion from distorted image coordinates $[X_D Y_D A]$ to distorted computer image coordinates $[X_{FD} Y_{FD} A]$	37

Figure 3.5 Generalized diagram of calibration techniques	41
Figure 4.1 Sample checkerboard images for VPS intrinsic calibration	50
Figure 4.2 VPS calibration frame installed in the NBRF	59
Figure 4.3 VPS Calibration frame ball locations and reference frame	61
Figure 4.4 VPS calibration frame hanging above NBRF	62
Figure 4.5 Sliding measuring block on VPS calibration frame caliper	63
Figure 4.6 Illustration of VPS calibration frame reference targets	66
Figure 4.7 Image of calibration frame from camera 2	73
Figure 4.8 Zoomed calibration frame target	74
Figure 5.1 Information flow diagram for VPS_client	82
Figure 5.2 VPS_client algorithm	83
Figure 5.3 VPS image in its initial state	86
Figure 5.4 VPS background image	88
Figure 5.5 VPS image after background subtraction	88
Figure 5.6 VPS image after thresholding	89
Figure 5.7 VPS image after noise removal	91
Figure 5.8 Information flow diagram for raptor	95
Figure 5.9 Information flow diagram for SCAMP SSV	99
Figure 6.1 Simple open-loop block diagram	101
Figure 6.2 Simple closed-loop block diagram	101
Figure 6.3 Discontinuous closed-loop block diagram	102
Figure 6.4 Closed-loop block diagram with observer	103
Figure 6.5 Luenberger Observer in a closed-loop system	105

Figure 6.6 State estimator in a realistic closed-loop diagram	108
Figure 6.7 SCAMP SSV rigidly attached to fixture for sensor noise characterization tests	129
Figure 6.8 Algorithm of the VPS Extended Kalman Filter	132
Figure 7.1 Black ball, simulating SCAMP SSV, hanging in the NBRF	137
Figure 7.2 SCAMP SSV in static position 2, viewed from camera 1	141
Figure 7.3 Image from Figure 7.2, after vision processing	141
Figure 7.4 Global force inputs applied to SCAMP SSV	146
Figure 7.5 Estimated global X-axis position versus time, propagation test	147
Figure 7.6 Estimated global X-axis velocity versus time, propagation test	147
Figure 7.7 Estimated global Y-axis position versus time, propagation test	148
Figure 7.8 Estimated global Y-axis velocity versus time, propagation test	148
Figure 7.9 Estimated global Z-axis position versus time, propagation test	149
Figure 7.10 Estimated global Z-axis velocity versus time, propagation test	149
Figure 7.11 Trajectory of SCAMP SSV in 3D	152
Figure 7.12 SCAMP SSV global X-axis position vs. time	152
Figure 7.13 SCAMP SSV global X-axis velocity vs. time	153
Figure 7.14 SCAMP SSV global X-axis position vs. time, magnified	153
Figure 7.15 SCAMP SSV global X-axis position vs. time, with $\pm 3\sigma$ error bounds	154
Figure 7.16 SCAMP SSV global X-axis velocity vs. time, with $\pm 3\sigma$ error bounds	155

Figure 7.17 SCAMP SSV global X-axis position vs. time, with $\pm 3\sigma$ error bounds, magnified for T=105 s to T=112 s	156
Figure 7.18 Camera 1 pixel measurements	157
Figure 7.19 Camera 2 pixel measurements	157
Figure 7.20 Camera 3 pixel measurements	158
Figure 7.21 Camera 4 pixel measurements	158
Figure 7.22 Camera 5 pixel measurements	159
Figure 7.23 Camera 6 pixel measurements	159
Figure 7.24 Background image for camera 1, coordinates [260 320] indicated	164
Figure 7.25 SCAMP SSV with glare on top panels, raw image	165
Figure 7.26 SCAMP SSV with glare on top panels, thresholded image	166
Figure 7.27 Example of glare from direct sunlight	167
Figure 7.28 Trajectory of SCAMP SSV in 3D, measured by EKF1	168
Figure 7.29 Trajectory of SCAMP SSV in 3D, measured by EKF2	169
Figure 7.30 SCAMP SSV global X-axis position vs. time, EKF1 and EKF2	169
Figure 7.31 SCAMP SSV global Y-axis position vs. time, EKF1 and EKF2	170
Figure 7.32 SCAMP SSV global Z-axis position vs. time, EKF1 and EKF2	170
Figure 7.33 SCAMP SSV global X-axis position vs. time, EKF1, with $\pm 3\sigma$ error bounds	172
Figure 7.34 SCAMP SSV global X-axis velocity vs. time, EKF1, with $\pm 3\sigma$ error bounds	172



Figure 7.35 SCAMP SSV global Y-axis position vs. time, EKF1, with $\pm 3\sigma$ error bounds	173
Figure 7.36 SCAMP SSV global Y-axis velocity vs. time, EKF1, with $\pm 3\sigma$ error bounds	173
Figure 7.37 SCAMP SSV global Z-axis position vs. time, EKF1, with $\pm 3\sigma$ error bounds	174
Figure 7.38 SCAMP SSV global Z-axis velocity vs. time, EKF1, with $\pm 3\sigma$ error bounds	174
Figure 7.39 SCAMP SSV global X-axis position vs. time, EKF2, with $\pm 3\sigma$ error bounds	176
Figure 7.40 SCAMP SSV global Y-axis position vs. time, EKF2, with $\pm 3\sigma$ error bounds	176
Figure 7.41 SCAMP SSV global Z-axis position vs. time, EKF2, with $\pm 3\sigma$ error bounds	177
Figure 7.42 SCAMP SSV global X axis position (top) and velocity (bottom) vs. time	178
Figure 7.43 Camera 1 measured and predicted camera data	180
Figure 7.44 Camera 2 measured and predicted camera data	180
Figure 7.45 Camera 3 measured and predicted camera data	181
Figure 7.46 Camera 4 measured and predicted camera data	181
Figure 7.47 Camera 5 measured and predicted camera data	182
Figure 7.48 Camera 6 measured and predicted camera data	182

Figure 7.49 SCAMP SSV global X axis position vs. time, as computed by four EKF/camera configurations	186
Figure 7.50 SCAMP SSV global Y axis position vs. time, as computed by four EKF/camera configurations	187
Figure 7.51 SCAMP SSV global Z axis position vs. time, as computed by four EKF/camera configurations	187

# Nomenclature

## ***Acronyms***

3D, 2D	Three-dimensional, Two-dimensional
A/D	Analog to Digital
CCD	Charge Coupled Device
DC	Direct Current
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
EVA	Extra-Vehicular Activity
FOV	Field of View
GPS	Global Positioning System
GUI	Graphical user Interface
IMU	Inertial Measurement Unit
NBRF	Neutral Buoyancy Research Facility
N-E-D	North-East-Down
ReCS	REconfigurable Control Station
SCAMP SSV	Secondary Camera And Maneuvering Platform – Space Simulation Vehicle
SSL	Space Systems Laboratory

## Symbols

$\hat{\phantom{x}}$	A hat above variable indicates either the estimate of the variable, or the unit vector of an axis, as appropriate
$[\dots]^T$	Transpose of a vector or matrix
$A$	3x4 intermediate matrix, used in intrinsic calibration algorithm
$A$	Object Image Area ( $\text{mm}^2$ , $\text{pixels}^2$ )
$A$	System dynamics matrix for an LTI system
$B$	Matrix of input dynamics for an LTI system
$C$	Matrix of measurement model for an LTI system
$C_{DT}$	Drag coefficient
$C_k()$	Partial derivatives of $c_k()$
$c_k()$	Estimate measurement
$C_x, C_y$	Coordinates of the principal point on the digital image plane
$D_x'$	X axis mm-to-pixel conversion factor
$D_y$	Y axis mm-to-pixel conversion factor
$e$	Error vector
$E$	Sum of the error terms $e_{jk}$
$e_{jk}$	Scalar error term arising from measurement $L_{j,k}$
$ E_{POS} $	Magnitude of the position error in calibration target 3D location estimate
$\varepsilon_v, E_v$	Estimated velocity error terms, scalar and matrix forms
$f$	Camera focal length

$F_{DRAG, i}$	Drag force along axis $i$ , $i=X, Y, Z$
$\sum F_{xG}$	Sum of the forces in the global X direction
$G$	Process noise scaling matrix
$H_{CCD}$	Height of CCD chip
$J$	Error function (used several places)
$J(L)$	Estimator objective function
$K_t, K_r$	Tangential and radial distortion coefficients
$K, K_A, K_B$	First-order radial lens distortion parameter
$L, L_k$	Observer/estimator gain matrix
$L_{j,k}$	Measurement between ball $j$ and ball $k$
$Ls_{j,k}$	Synthetic measurement between ball $j$ and ball $k$
$m$	Mass
$m$	Number of measurements
$\mathbf{m}$	2D image coordinate, $[X_{FD} \ Y_{FD}]$
$\mathbf{M}$	3D global location, $[X_G \ Y_G \ Z_G]^T$
$n$	Number of targets
$N_{CX}$	Number of pixels in the X direction (frame grabber)
$N_{FY}$	Number of sels in the Y direction (CCD)
$N_{FX}$	Number of sels in the X direction (CCD)
$P, P_k, P_{k-1}$	Estimate error covariance matrix
${}^G\mathbf{P}$	Object position in global frame, $[X_G \ Y_G \ Z_G]^T$
${}^C\mathbf{P}$	Object position in camera frame, $[X_C \ Y_C \ Z_C]^T$

${}^C P_{G,org}$	Translation vector from the origin of the camera frame to the origin of the global frame, in camera frame coordinates, or $[T_X T_Y T_Z]^T$
$Q, Q_k, Q_{k-1}$	Process noise covariance matrix
$r$	Image radius of example spherical object
$r$	Image distance from optical axis, distorted and undistorted coordinates
$r_{11}, r_{12}, \dots, r_{33}$	Elements of ${}^C R$
$R$	Physical radius of example spherical object
$R, R_k$	Measurement noise covariance matrix
${}^C R$	3x3 rotation matrix that rotates the global frame into the camera frame
$R_X$	Rotation about the fixed camera X axis, $X_C$
$R_Y$	Rotation about the fixed camera Y axis, $Y_C$
$R_Z$	Rotation about the fixed camera Z axis, $Z_C$
$s_X$	Ratio of pixel spacing in X- and Y- directions
$T$	The extrinsic calibration translation vector, ${}^C P_{G,org}$ , or $[T_X T_Y T_Z]^T$
$T_X$	$X_C$ component of translation vector from origin of camera coordinate frame to origin of global coordinate frame
$T_Y$	$Y_C$ component of translation vector from origin of camera coordinate frame to origin of global coordinate frame

$T_Z$	$Z_C$ component of translation vector from origin of camera coordinate frame to origin of global coordinate frame
$\mathbf{u}$	Control input vector
$\mathbf{v}$	Measurement noise vector
$v_i$	Velocity (inertial) along axis $i$ , $i=X, Y, Z$
$\mathbf{w}$	Process noise vector
$W_{CCD}$	Width of CCD chip
$\mathbf{x}, \dot{\mathbf{x}}$	General state vector and derivative
$\tilde{\mathbf{x}}$	Estimate error vector
$X_A, Y_A, Z_A$	Example coordinate axes
$[X_C \ Y_C \ Z_C]^T$	X, Y, and Z coordinates, camera frame
$[X_D \ Y_D]$	Real distorted 2D image coordinates in X and Y directions (mm)
$[X_{FD} \ Y_{FD}]$	Digital distorted 2D image coordinates in X and Y directions (pixels)
$[X_{FU} \ Y_{FU}]$	Digital undistorted 2D image coordinates in X and Y directions (pixels)
$[X_G \ Y_G \ Z_G]^T$	X, Y and Z coordinates, global (inertial) frame
$[\dot{X}_G \ \dot{Y}_G \ \dot{Z}_G]^T$	X, Y, and Z velocities, global (inertial) frame
$[X_j \ Y_j \ Z_j]^T$	Global coordinates of ball $j$ , $j=1,2,\dots,20$
$[X_U \ Y_U]$	Real undistorted 2D image coordinates in X and Y directions (mm)

$\partial\mathcal{Y}_t, \partial\mathcal{Y}_r, \text{etc.}$	Distortion on an axis due to radial or tangential distortion
$\mathbf{z}$	Measurement vector
$\Phi_{k-1}$	Discrete state transition matrix
$\Lambda_{k-1}$	Discrete input propagation matrix
$\sigma$	Standard deviation
$\sigma_F$	Standard deviation of unmodeled forces
$\lambda$	Parameter defining how big a neighborhood around a black pixel is
$\rho$	Number of black pixels in the neighborhood of another black pixel



# Chapter 1

## Introduction

Operations in low Earth orbit introduce many challenges, whether they involve human astronauts, remotely operated equipment, or autonomous or semi-autonomous systems. Inertial navigation is a key enabling technology common to all space operations. Because of extremely high costs associated with launching equipment into space, systems must be thoroughly tested to ensure proper function, but such testing is often difficult, due to the challenge presented by recreating the space environment on Earth, especially conditions of weightlessness and free motion capability in three dimensions. The Space Systems Laboratory (SSL) at The University of Maryland uses underwater neutral buoyancy simulation of space activities as a way of researching techniques and equipment for use in orbital operations. Other techniques to simulate weightlessness exist, such as air bearing tables [1], aircraft following special parabolic trajectories (e.g. NASA's KC-135) [2], drop towers [3], overhead weight-bearing cables [4], and six-degree-of-freedom (6DOF) gantry robot manipulators [5]. None of these techniques, including neutral buoyancy, meet all of the desirable criteria for weightlessness simulation, but neutral buoyancy testing is the only way to conduct long-duration, full-scale, free flight, three-dimensional simulations of weightlessness on Earth.

Since many space systems have 6DOF navigation requirements – three DOF for system inertial attitude, three DOF for system inertial or relative position – it is important for neutral buoyancy analogs of space systems to possess similar navigational capabilities. Implementation of rotational navigation systems on neutral buoyancy robots has been achieved [6][7] through the use of an on-board inertial measurement system (IMU) analogous to those found on aircraft and undersea vehicles, but a robust translational navigation system, has proven more elusive, with limited successes achieved with variants of an acoustic positioning system [7][8][9]. The typical technology choices for inertial, translational navigation– GPS [10] for outdoor applications, Pseudolite [11] systems for indoor applications – are not suitable due to signal attenuation in water.

This thesis describes efforts made to extend the testing capability of Earth-based space simulation by providing an accurate inertial navigation capability for a neutral buoyancy environment. This document describes the development of a vision-based inertial positioning system that provides, in real-time, a full translational state estimate (inertial position and velocity) to SCAMP SSV (Supplemental Camera and Maneuverability Platform - Space Simulation Vehicle), a free-flying neutral buoyancy robot used to simulate autonomous and teleoperated spacecraft. The key developments presented herein include an accurate vision system calibration technique, and an Extended Kalman Filter for the three vehicle translational DOF that merges processed camera measurements with dynamic information from robot telemetry to create an

optimal state estimate. Results from underwater testing are also presented that characterize the accuracy of the system.

## ***1.1 Description of Problem***

The goal of the work described in this thesis is to field a functional and accurate vision-based inertial navigation system for a neutral buoyancy environment. For this work, VPS hardware was installed and tested in the University of Maryland's Neutral Buoyancy Research Facility (NBRF), and the tracked "target" requiring inertial navigation was a free-flying robotic vehicle known as SCAMP SSV (Supplemental Camera and Maneuvering Platform). SCAMP SSV [6], pictured below, was designed as a high-fidelity neutral buoyancy simulation of a rigid, free flying, 6DOF teleoperated robotic camera platform capable of performing critical inspection tasks in space.

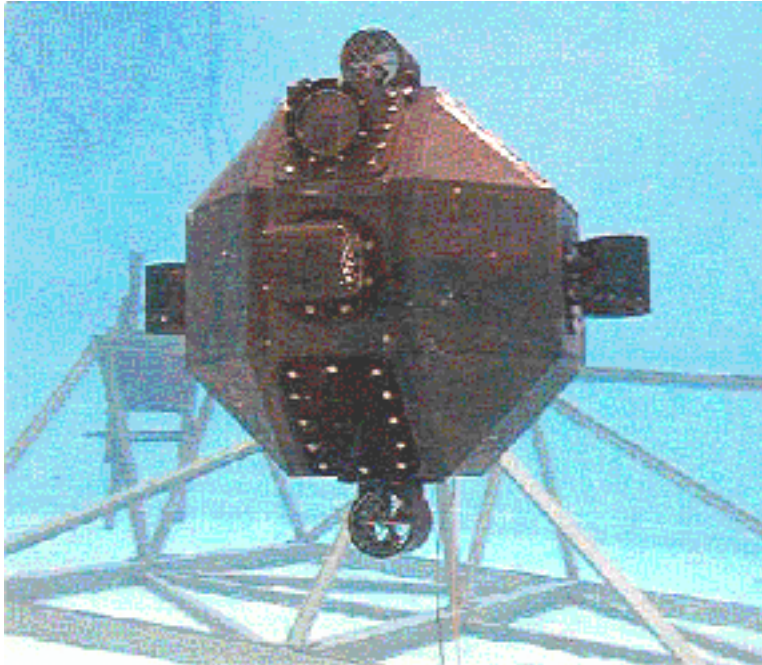


Figure 1.1 SCAMP SSV in free flight in the NBRF

Because SCAMP SSV has both an onboard computer and an onboard IMU that together calculate rotational state feedback (inertial orientation and body angular rates), standard closed loop control techniques are used to cause SCAMP SSV to autonomously hold a desired attitude, or to follow a desired attitude trajectory over time.

To enable closed-loop translational control of SCAMP SSV, a step toward semi-autonomous or autonomous operation, accurate inertial position and velocity feedback is required. Some work has been done on a vision-based *relative* navigation system for SCAMP SSV using a commercially available color tracking system to enable identification of a specially colored target [12]. This approach, however, requires a “leader” that must be kept in view at all times. In previous

work, an acoustic positioning system was deployed that provided an inertial navigation capability for other robots in the neutral buoyancy environment [7][8][9]. However, this system was challenged with multi-path reflections in the enclosed tank environment and by the requirement to place a substantial amount of support hardware on each tracked vehicle.

Previous research [13] established a baseline vision-based inertial navigation system for the NBRF known as The Vision Positioning System (VPS). The goal of this previous effort was to accurately (single-digit cm-scale) calculate the inertial 3-dimensional (3D) translational motion of SCAMP SSV, with direct camera-based position estimates and velocities inferred by differentiation. VPS was originally comprised of eight CCD TV cameras, rigidly mounted underwater to the walls of the NBRF. A single computer on the surface, using a high-speed frame grabber, sampled the video signals and processed the resulting images to calculate vehicle position. In previous work, 3D positions were computed from individual images, with centroid providing local (camera) X and Y coordinate estimates, and local range (Z coordinate) based on tracked object image area. Each local 3D estimate was then transformed into a global frame. Camera measurements were then combined with a least-squares algorithm.

In order to be deployed as an accurate inertial navigation system, the baseline VPS previously developed required two fundamental enhancements, the contributions of this thesis: accurate characterization of internal and external

camera calibration parameters, and a real-time image processing and Extended Kalman Filter algorithm capable of combining camera measurements with dynamic information provided by the robot into position and velocity estimates.

Previously, of the five internal camera calibration parameters typically considered when using consumer-grade CCD TV cameras for machine vision, only one, focal length, had been measured. All six external calibration parameters had been considered, but the technique used to find them was not sufficiently accurate or robust to disturbances (e.g. water currents). This work describes a robust method to compute the internal and external parameters, as well as test results that characterize calibration accuracy and measurement repeatability.

An Extended Kalman Filter was implemented to merge camera measurements and vehicle dynamics into real-time state estimates. This required derivation of filter equations and integration of all data from frame grabbers, SCAMP SSV, and the control station) and will enable full 6DOF closed-loop vehicle control in the future.

## **1.2 Solution**

With the overall goal of fielding VPS as a functional inertial navigation system for a neutral buoyancy simulation environment, work towards that goal was divided into three main categories.

- 1) Development of a new VPS calibration technique: A new, two-step calibration technique was developed, using algorithms and software that are standard in the computer-vision community. Five internal and six external camera parameters were characterized, and the accuracy of these parameters was also estimated.
- 2) Implementation of a real-time optimal state estimator (EKF): An Extended Kalman Filter was implemented in the SCAMP SSV control station software that combines the VPS position measurements with current vehicle dynamic data to form an optimal, real-time estimate of the position and velocity of SCAMP SSV.
- 3) Integration of the complete system: Communication systems between the VPS component programs were implemented to enable real-time data flow, and structures were put in place to allow subsequent implementation of 6DOF closed-loop control on SCAMP SSV.

### **1.3 Outline**

This thesis describes the development and deployment of VPS as a functional, moderately accurate inertial navigation system for SCAMP SSV, focusing on the categories listed above.

Chapter 2 provides a brief introduction to the hardware and software components that comprise VPS, followed by Chapter 3 that provides background about camera models and the related problem of camera calibration. Chapter 4

contains a detailed description of the hardware and image processing algorithms required for VPS calibration, while Chapter 5 describes the software architecture and algorithms for continuously sampling and processing image and sensor data required to create the optimal translational state estimates. Chapter 6 introduces the Extended Kalman Filter algorithm and provides details about its application to VPS. Chapter 7 presents inertial state estimation results for SCAMP SSV, from both piloted flight and stationary (static) test series. Chapter 8 concludes the thesis and identifies directions for future development of VPS.



## Chapter 2

### Vision Positioning System Background

This chapter describes the major system components of VPS, including hardware, software, and computer elements.

#### **2.1 System Hardware**

The Neutral Buoyancy Research Facility at The University of Maryland is a cylindrical, fiberglass tank of water, 7.62 m (25') deep and 15.24 m (50') in diameter. The water is filtered for exceptional visual clarity and maintained at approximately 32 deg. C (90 deg. F). There are 12 brackets, or "hard points", built into the fiberglass of the tank wall. These provide rigid mounting points for equipment that is to be placed in the water. There is a ring of four hard points, spaced at approximately 90° intervals, near the surface of the water, a similar ring at mid-depth, and a final ring near the bottom of the tank. The top and bottom hard points are approximately aligned with each other, while the middle ring hard points are rotated 45 degrees with respect to the others. It is to the top and middle rings that the VPS camera boxes are attached because of the potential for cameras on the bottom hard points having their views blocked by hardware resting on the bottom of the tank. Currents in the tank introduced by natural convection and the water-heating system are minimal but not negligible, and can introduce significant disturbance forces on unsupported, floating objects.

### 2.1.1 Description of Cameras and Camera Coverage

Two types of CCD TV cameras are used as sensors in VPS: the Elmo TSE401 black and white security camera, and the Elmo TNC4614DN color security camera. The CCD chip in both models is 4.88 mm x 3.66 mm, with 768 sels (or “sensing elements”) in the horizontal direction, and 494 sels in the vertical direction. When in use, each camera is sealed in a waterproof box designed and manufactured by SSL personnel. Pictured below in Figures 2.1 and 2.2 mounted in the NBRF, these camera boxes consist of a white PVC tube, and two Delrin™ end plugs that seal on the inner diameter of the PVC tube with radial o-rings. One end plug has a clear, flat Plexiglas™ window that seals with an axial o-ring, providing a minimally-distorted camera viewing port<sup>1</sup>. The other end plug has an opening for passing the camera’s video (standard co-axial) and power cables out of the camera box. The opening is sealed with epoxy. All parts of the camera box that come in contact with the water are epoxy, a type of plastic, or stainless steel (the fasteners). The camera boxes on the middle ring are mounted to point radially inward, while the boxes on the top ring are fitted with a single pivot axis, allowing the cameras to be tilted down. This increases the amount of the tank that is visible to the top ring cameras, which otherwise would have a large portion of their fields-of-view (FOV) occupied by the surface of the water.

---

<sup>1</sup> Previous designs saw anodized aluminum contacting the water. Anodic corrosion on the aluminum, due to its contact with the stainless steel, made this design not suitable for long-term submersion in the NBRF.

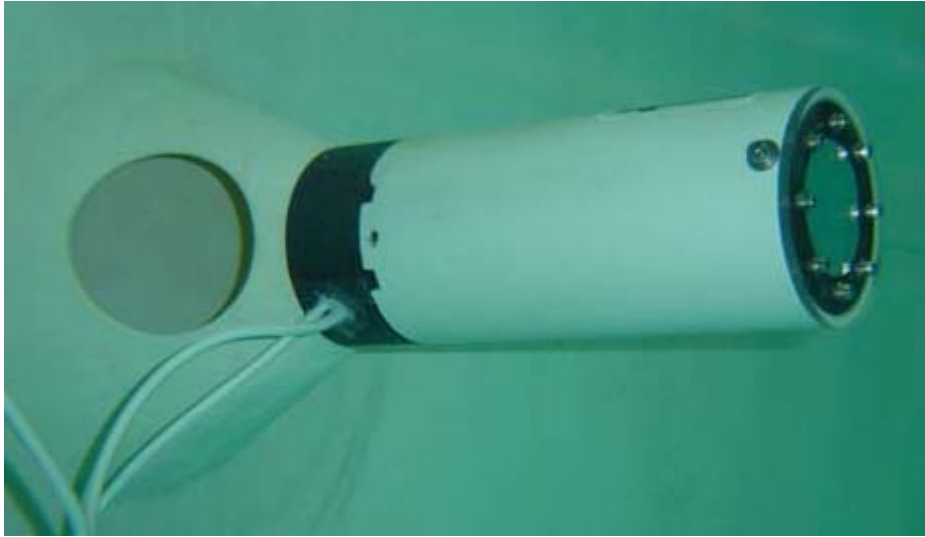


Figure 2.1 Bottom ring VPS camera



Figure 2.2 Top ring VPS camera

Cables carrying both power and the video signal run from the camera boxes to the surface of the water, and then along handrails to terminal blocks. DC power is provided to the terminal blocks from a wall electrical outlet via a medical-grade

isolation transformer. The video signals are carried from the terminal blocks by another set of co-axial cables to the computers that perform the vision processing operations.

Camera coverage in the NBRF is not consistent throughout the tank, but varies with position. A significant portion of the center of the tank is visible by all eight cameras, while some areas (such as the center of the tank near the top and bottom) are not visible by any cameras. The cameras were each aimed and focused by hand, and no measures were taken to ensure symmetrical coverage since there were no requirements for such actions. Therefore, the cameras have varying focal lengths and aiming points, causing asymmetrical tank coverage. For example, this can be seen in Figure 2.4. Camera 2, mounted on the middle ring and aimed radially from West to East, is focused more widely than the camera opposite it, camera 4. This means that camera 2 sees more of the tank than camera 4, but camera 4 will provide slightly better resolution due to its more narrow focus. Figures 2.3 through 2.7 show the camera coverage at the depths 3' (0.91m), 7' (2.13m), 12.5' (3.81m), 18' (5.49m) and 22' (6.71m). Table 2.1 contains the legend for the tank coverage maps.

Table 2.1 Legend for tank coverage maps

Symbol	Camera Coverage (# cams.)
+	0
•	1
○	2
△	3-4
×	5-6
*	7-8

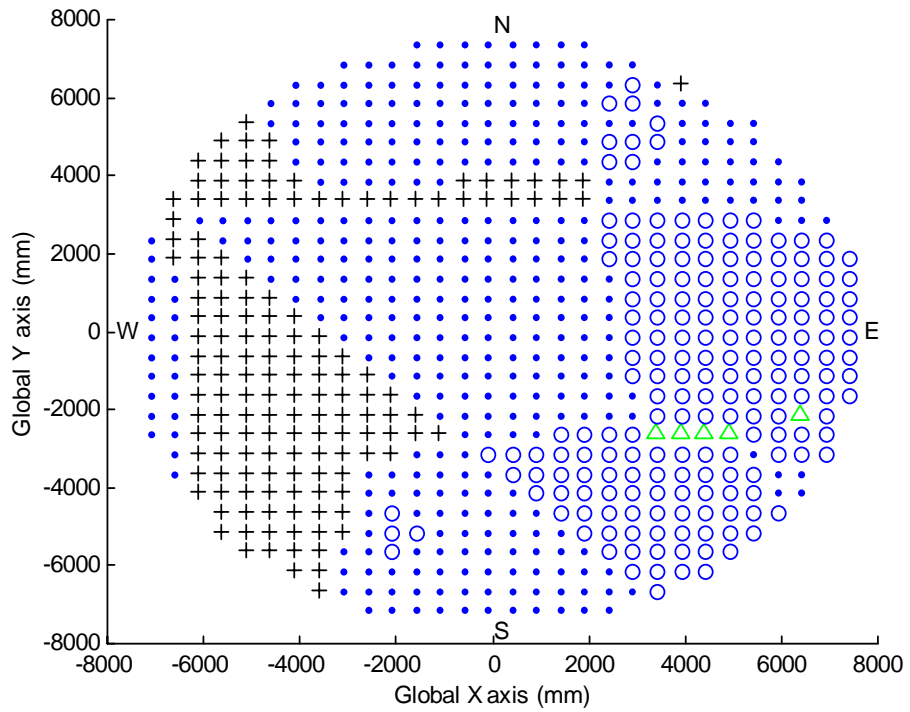


Figure 2.3 Camera coverage, depth = 3'

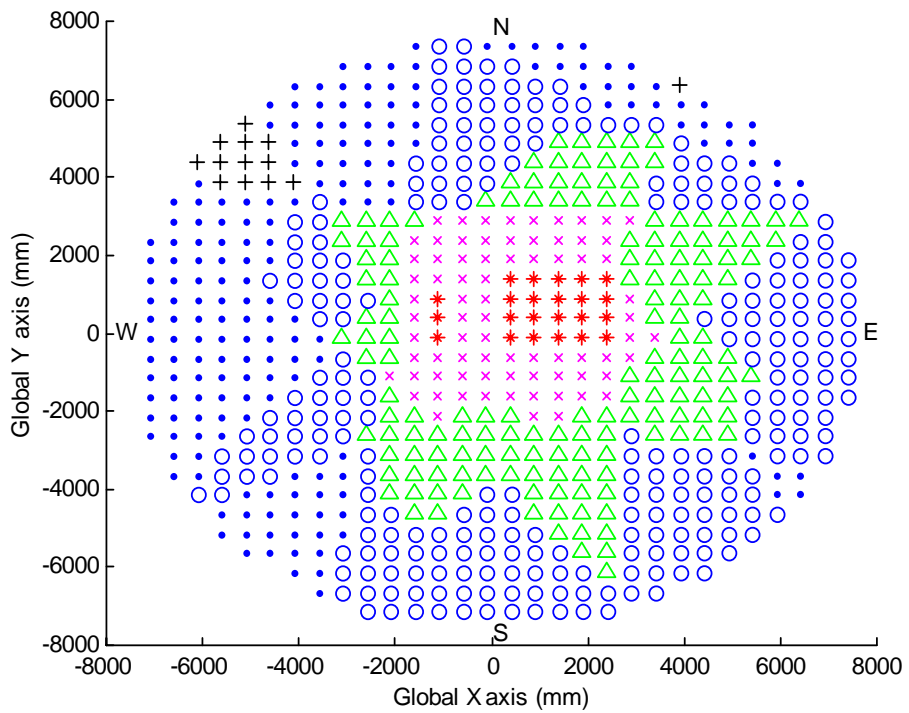


Figure 2.4 Camera coverage, depth = 7'

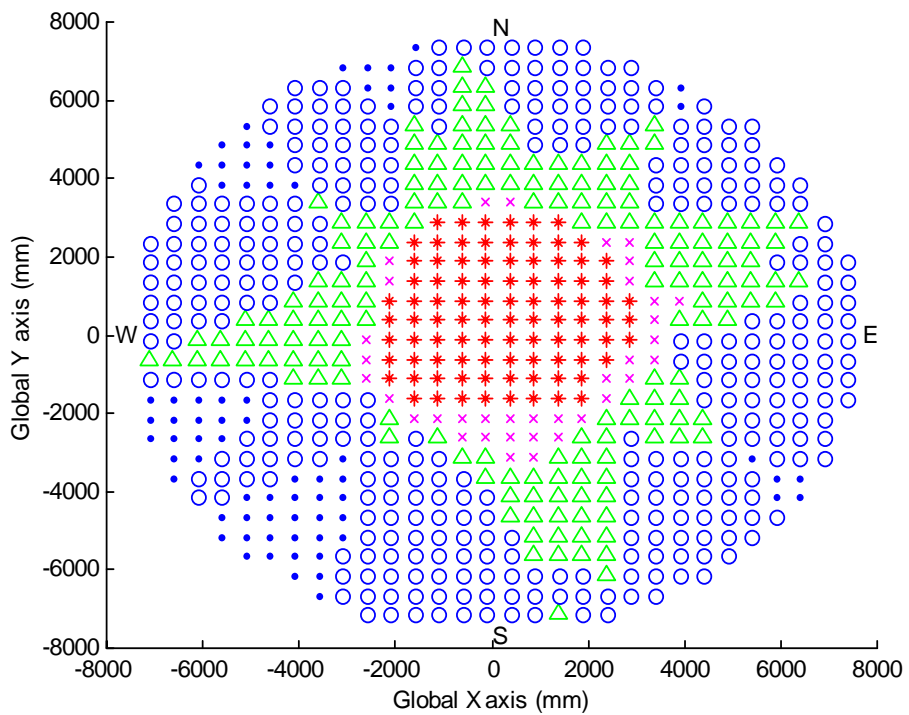
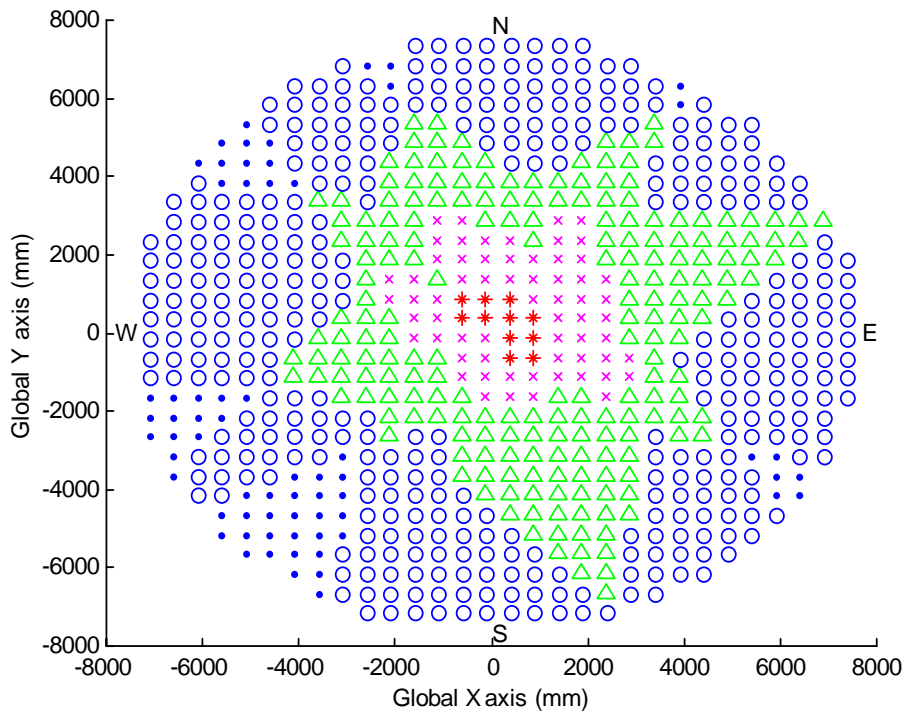


Figure 2.5 Camera coverage, depth = 12.5'



2.6 Camera coverage, depth = 18'

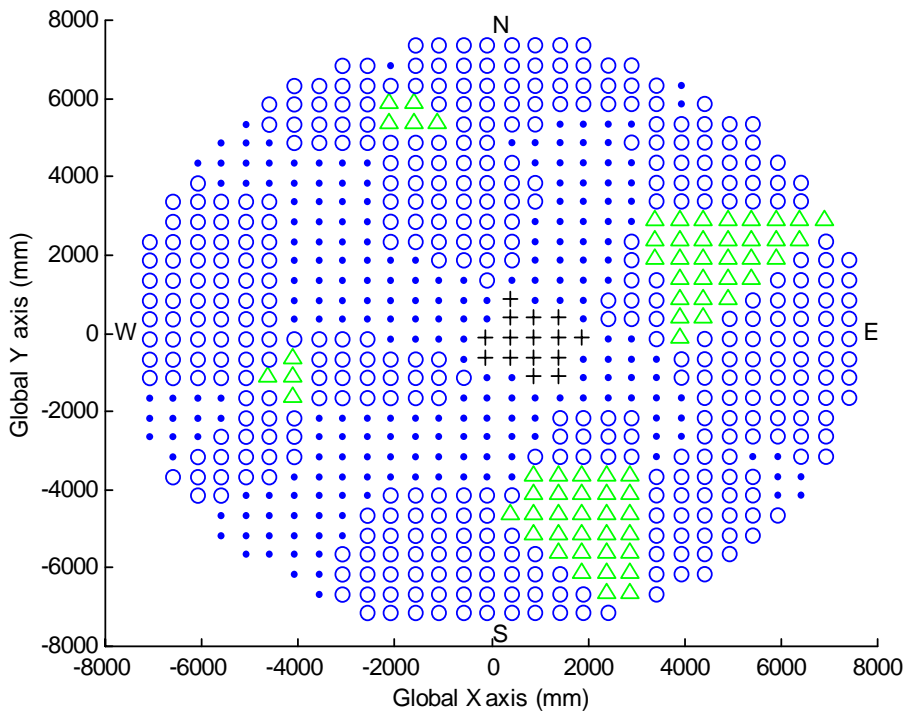


Figure 2.7 Camera coverage, depth = 22'

### 2.1.2 Description of SCAMP SSV

With appropriate object recognition software, VPS could theoretically track any object in the tank. However, the goal for this work was to specifically provide inertial navigation for SCAMP SSV, the Supplemental Camera and Maneuvering Platform, Space Simulation Vehicle. SCAMP SSV is one of two SCAMP-class robots currently operational at the SSL. The SCAMP-class robots were built to provide roving camera views for larger simulations, simulate teleoperated and autonomous inspection missions as an underwater analogue to NASA JSC's AERCam [14], and to serve as a test bed for experiments in spacecraft guidance, navigation and control. They are also capable, with some simple modifications such as the addition of a tool carrier, of acting as limited astronaut assistants in EVA simulation. SCAMP SSV is shown below in Figure 2.8, and with its two side access covers removed in Figure 2.9.

SCAMP SSV has a hull made of welded, black-anodized aluminum which, when assembled, forms a nearly spherical, 26-sided closed polyhedron. A housing for a video camera, which provides the onboard view necessary for a pilot to fly the robot, protrudes from the top-front section of the vehicle. Six bi-directional thrusters, aligned in pairs along the body axes of the vehicle, are mounted to the hull. The end caps are rapidly assembled onto the hull by pressurizing pneumaseals, and equipment access panels are attached with seal screws. When properly balanced, SCAMP SSV is neutrally buoyant in water in both



translation (it neither sinks nor floats) and in rotation (it has no preferred orientation), providing the free 6DOF motion that simulates a spacecraft on orbit.

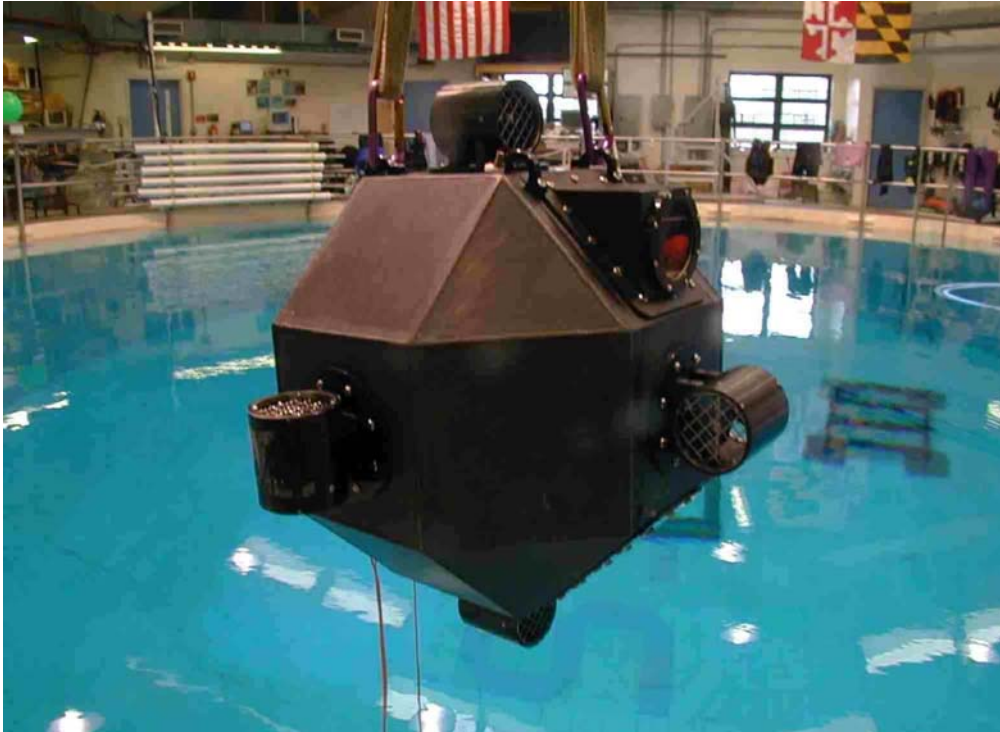


Figure 2.8 SCAMP SSV above the NBRF

The central electronics compartment contains the computer stack and the IMU. The computer is a 100 MHz Compact PCI Power PC running the VxWorks real-time operating system, version 1.0.1. The IMU includes a triaxial accelerometer to sense the gravity vector, a triaxial magnetometer to sense the magnetic north vector, and three rate gyros, aligned along the body axes of the robot, to sense angular velocity. These sensors, with an A/D board on the computer stack, allow SCAMP SSV to compute in real-time an accurate estimate of its inertial orientation. Control algorithms implemented in the onboard flight code allow

SCAMP SSV to autonomously hold or track desired attitudes or attitude trajectories. The computer stack also contains LM629-based motor controller boards for each of the six thrusters.

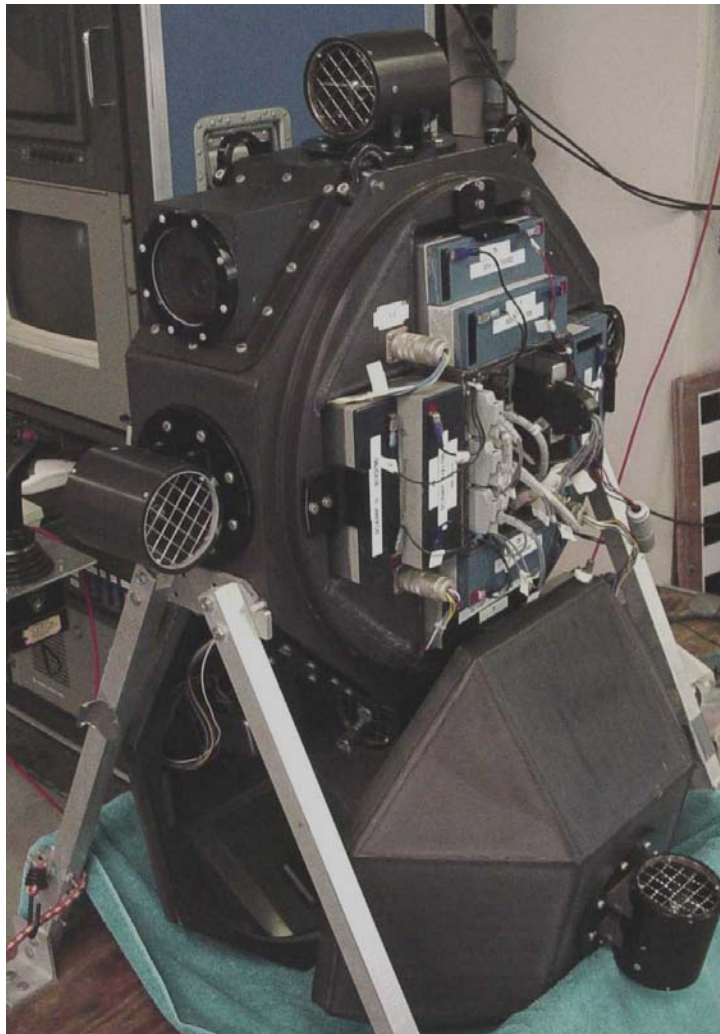


Figure 2.9 SCAMP SSV with side access covers removed

Sixteen 6V batteries on board the robot provide power to the electronics and thrusters. The batteries are arranged into a 30V control power pack, and three

12V packs, each powering a pair of thrusters. A final 12V battery pack powers the onboard camera.

Communication to and from a surface control station computer occurs over fiber-optic cables, with a single fiber video uplink, and a dual fiber pair for Ethernet-based data communications. A pilot controls SCAMP SSV from the control station pictured below in Figure 2.10, using the video view from the on board camera, a GUI showing telemetry data, and two analog hand controllers to issue motion commands. This control station is known as ReCS, the Reconfigurable Control Station.



Figure 2.10 SCAMP SSV control station, ReCS

The hand controllers are standard for space shuttle operations, with the one on the left controlling translation, while the one on the right controls rotation. The GUI, shown in Figure 2.11, displays real-time telemetry coming from the robot, including attitude, angular velocity, and, when VPS is fully functional, inertial position and velocity.

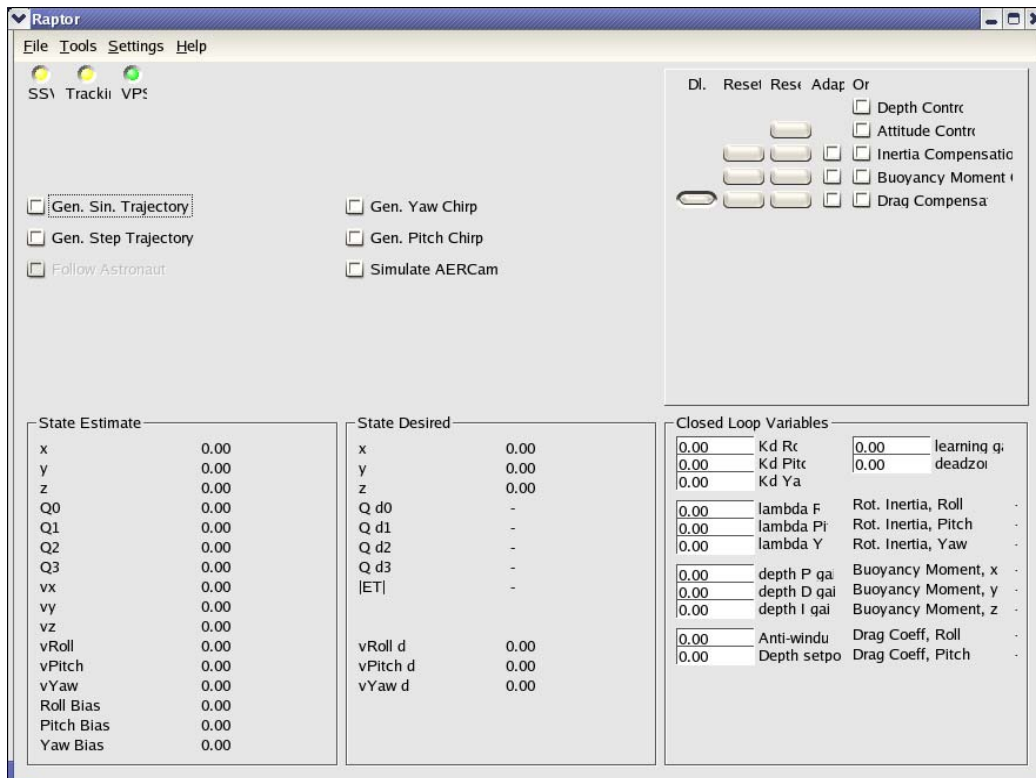


Figure 2.11 SCAMP SSV control station GUI

### 2.1.3 Description of VPS Computer Hardware

Four desktop computers house the frame grabber boards that accept the video signals directly from the video cameras, and run the image processing software that creates the measurements VPS uses to compute the position of SCAMP

SSV. These computers are referred to as the vision computers, Vision 1 through Vision 4. Currently, three of these computers are identical: each is a 1.7GHz PC with 512 MB of RAM, runs on the Windows 2000 operating system, and has two FlashBus™ MV Lite single-channel frame grabber boards. Each board has one camera attached directly to it, meaning each vision computer controls two individual cameras. Vision 1 controls cameras 1 and 2, Vision 2 controls cameras 3 and 4, and so on.

The fourth vision computer was originally used for proof-of-concept testing, and has a single, nine-channel Data Translations™ DT3133 frame grabber board, that is alone capable of controlling up to nine cameras. This board currently has two cameras attached to it, cameras 7 and 8. Due to differences between the FlashBus™ and Data Translations™ frame grabber boards, Vision 4 and its connected cameras cannot currently operate as part of VPS.

ReCS is built around a 1.0 GHz PC with 256 MB of RAM, running Red Hat Linux, version 9, and the KDE desktop application. The analog hand controllers, mentioned above, interface with the control station computer via a National Instruments™ PCI-6023E A/D board. The control station software that handles communication with VPS and SCAMP SSV, creates and updates the GUI, and records telemetry for post-processing is called raptor.

## **2.2 System Software Overview**

This section provides an introduction to the major software components used to operate VPS. Figure 2.12 is a system-level diagram showing the various VPS computers, the programs that run on them, and how they are connected to each other. More details on VPS\_client, raptor, and the SCAMP SSV flight code, which are introduced below, can be found in Chapter 5.

All messages passed between VPS\_client, raptor, and SCAMP SSV are sent through Ethernet-based UDP (datagram) sockets. The execution rates of each of the software components vary, due to the limitations of hardware (the frame grabbers) in the case of VPS\_client, and the overhead of refreshing the GUI in the case of raptor.

Note that several software elements were created and adapted for use in VPS calibration. Because this software is not used at run-time, it is discussed in Chapter 4, which is wholly dedicated to the new VPS calibration technique, and not discussed here.

### **2.2.1 VPS\_client**

All of the VPS computer vision code exists in a program called VPS\_client. This program currently runs on Vision 1, 2 and 3, and with minor modifications required by the different frame grabber board, will be able to also run on Vision 4.

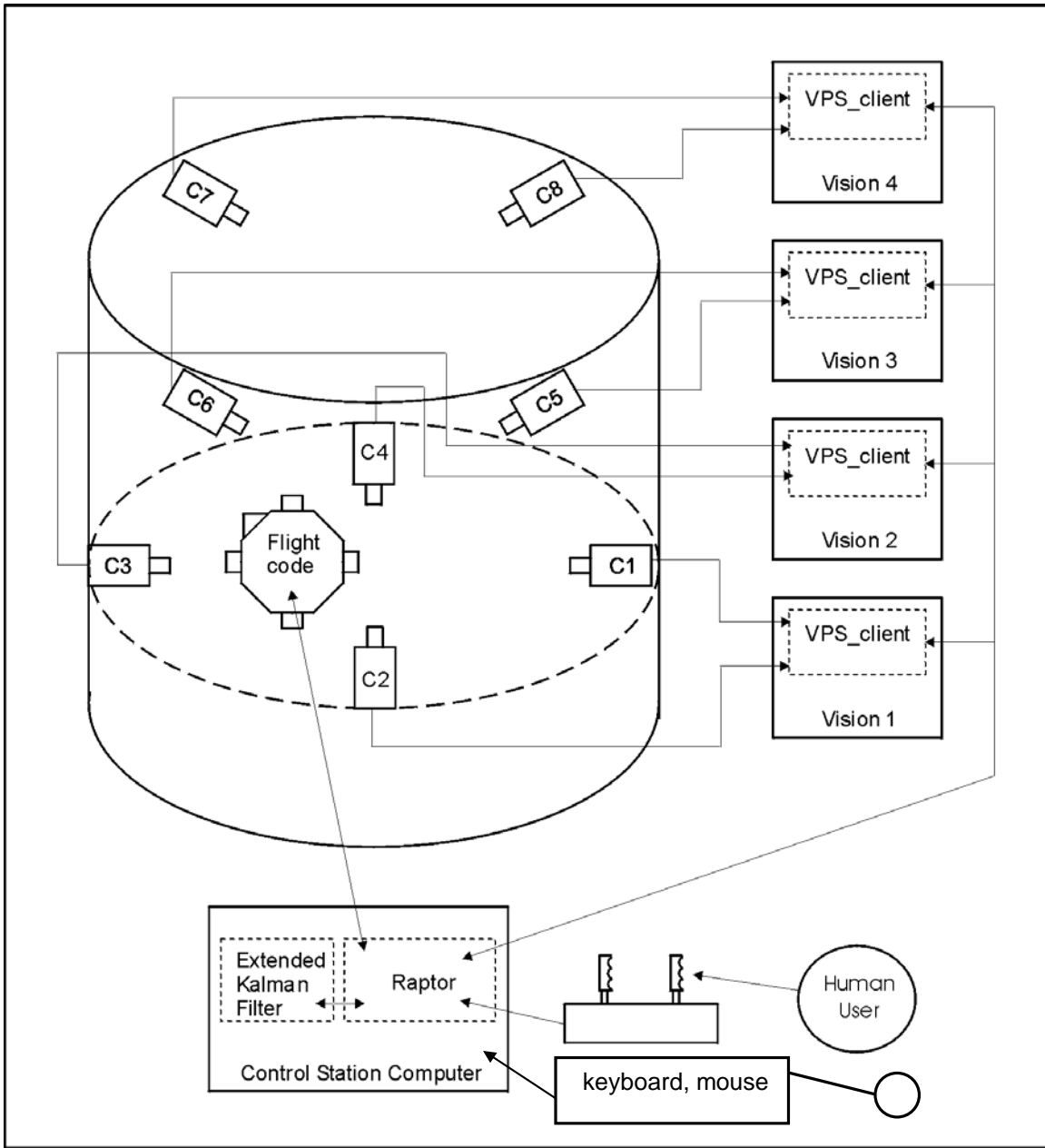


Figure 2.12 VPS layout diagram

After initialization, VPS\_client begins a loop to acquire an image, process the image to find the area and centroid of SCAMP SSV, and send this data to the control station, that in turn computes an inertial state estimate,

$[X_G \ \dot{X}_G \ Y_G \ \dot{Y}_G \ Z_G \ \dot{Z}_G]^T$ . Each VPS\_client process receives an updated

copy of the global position estimate from the control station to determine if the vehicle is in the FOV of a camera before an image is taken. If the vehicle is not visible to a camera, VPS\_client will not acquire images from that camera until the vehicle is visible.

### 2.2.2 Control Station Software (Raptor)

Raptor, the control station program, is an object-oriented C++ program. The GUI was created using the Qt open-source GUI development package. Socket communication and GUI update operations in raptor occur as fast as the computer is capable of executing them, with data from raptor sent to its SCAMP SSV and VPS\_client processes only when a message from that client arrives. Execution of the VPS EKF state estimation algorithm occurs at regular intervals (10 Hz) and is controlled by a software timer process independent of the other raptor operations.

### 2.2.3 SCAMP SSV Flight Code

The flight software operating on SCAMP SSV has three separate threads running independently: a 50 Hz communications thread, a 25 Hz state estimation thread, and a 25 Hz controller thread. This software receives and interprets pilot commands, calculates the current robot attitude state based on sensor measurements, and calculates and commands the control forces applied by the six thrusters.



## Chapter 3

### Camera Model and Calibration Background

Machine vision is often used, as with VPS, to infer 3D object information from 2D image data. For VPS, the inertial positions of an object, SCAMP SSV, relative to some selected world coordinate frame, are to be inferred from images. This requires accurate knowledge of the correspondence between the computer image coordinate frame and the selected world coordinate frame. A camera model is a set of equations that relates the two coordinate frames, by representing the creation of a computer image from incoming light. Camera calibration parameters are the constants that appear in those equations, and camera calibration refers to the act of determining the numerical values of those constants. A camera model, with the values of its associated calibration parameters properly determined, allows one to make the desired 2D to 3D inference. This chapter presents background on the topics of camera models, calibration parameters, and how to determine the values of those parameters, laying the groundwork for a description of the calibration of VPS, described in chapter 4.

#### ***3.1 Camera Calibration and Camera Models***

In the description of any machine vision investigation, it is important to specify the camera model that was used. The camera model consists of the

mathematical equations and parameters that are used to model what physically takes place in the camera – the process of converting incoming light into a computer image stored in digital memory. Just as there are different ways of modeling the dynamics of physical systems (Newtonian formulations, Lagrangian formulations, etc.), there are also a variety of camera models used by the computer vision community.

Given the equations for a certain camera model, the constants in those equations are referred to as the calibration parameters. The camera model and calibration parameters are related to each other in that either one, once specified, defines the form and nature of the other.

Each calibration parameter in a camera model typically defines a specific optical, geometrical, or physical value relating the computer image frame and a selected global coordinate frame. In some camera models, calibration parameters represent artificial, mathematical combinations of the “real” parameters. In either case, camera calibration involves measuring and or calculating the numerical values of the constants for each camera in a vision system. Accurate values for some parameters are available from camera specifications, but the majority require experimental determination.

The VPS camera model was derived from Tsai [15], a popular and accurate model used by the vision community. Calibration parameters can be divided into

two groups. The first group is the internal, or intrinsic calibration parameters. These are constants in the equations that model the interaction of light with optical and electronic components inside a camera. The VPS cameras required the five intrinsic parameters defined in Table 3.1.

Table 3.1 List of intrinsic VPS calibration parameters

<b><i>Symbol</i></b>	<b><i>Parameter</i></b>
$f$	Camera focal length (mm)
$C_x, C_y$	Coordinates of the principal point on the digital image plane (pixels)
$s_x$	Ratio of pixel spacing in X- and Y- directions (no units)
$K$	First-order radial lens distortion parameter (no units)

The extrinsic calibration parameters, or external parameters, define the physical placement, in both translation and rotation, of a camera with respect to the selected world coordinate frame. The extrinsic calibration parameters used in the characterization of VPS are listed and defined below in Table 3.2. For the Table 3.2 definitions, each camera has an orthogonal, right-handed coordinate system, the camera frame  $[X_c Y_c Z_c]^T$ , attached to it. The parameters below represent rotations and translations that are required to transform a camera from an initial state aligned and coincident with the selected global reference frame, to the final “calibrated” state.

Table 3.2 List of extrinsic VPS calibration parameters

<b>Symbol</b>	<b>Definition</b>
$R_X$	Rotation about the fixed camera $X$ axis, $X_C$ (degrees)
$R_Y$	Rotation about the fixed camera $Y$ axis, $Y_C$ (degrees)
$R_Z$	Rotation about the fixed camera $Z$ axis, $Z_C$ (degrees)
$T_X$	$X_C$ component of translation vector from origin of camera coordinate frame to origin of world coordinate frame (mm)
$T_Y$	$Y_C$ component of translation vector from origin of camera coordinate frame to origin of world coordinate frame (mm)
$T_Z$	$Z_C$ component of translation vector from origin of camera coordinate frame to origin of world coordinate frame (mm)

As can be seen in the above tables, eleven camera calibration parameters characterized each of the VPS cameras. Each of these will be discussed in detail below.

### **3.2 Mathematical Development of the VPS Camera Model**

Consider a 3D world position  $[X_G \ Y_G \ Z_G]^T$ . This section traces its conversion through three intermediate states to the 2D image state stored in computer memory and introduces the equations that govern that conversion. Together, the four conversion steps require all eleven calibration parameters.

Note that in practice, these equations are actually applied in the opposite order (and thus in inverted form) because the goal is generally to convert captured 2D image data into 3D world position data. These equations are developed from 3D world to 2D image coordinates because it is more straightforward to follow, and is the standard direction for the presentation of such equations in the literature.

The states achieved in transition from global (world) to computer image coordinates are given in Table 3.3. Each coordinate transformation is described below.

Table 3.3 Data transition states, from 3D global coordinates to 2D computer image coordinates

<b>Data State</b>	<b>Definition</b>
$[X_G Y_G Z_G]^T$	3D coordinates in the global reference frame (mm)
$[X_C Y_C Z_C]^T$	3D coordinates in the local camera reference frame (mm)
$[X_U Y_U A]$	Real undistorted 2D image coordinates in X and Y directions (mm), and object image area (mm <sup>2</sup> )
$[X_D Y_D A]$	Real distorted 2D image coordinates in X and Y directions (mm), and object image area (mm <sup>2</sup> )
$[X_{FD} Y_{FD} A]$	Digital distorted 2D image coordinates in X and Y directions (pixels), and object image area (pixels)

### 3.2.1 Step One: $[X_G Y_G Z_G]^T$ to $[X_C Y_C Z_C]^T$ Conversion

Figure 3.1 depicts a spherical object of radius  $R$  and two coordinate frames - a global frame and a camera frame. The position of the object in the global frame is  ${}^G\mathbf{P} = [X_G Y_G Z_G]^T$ , while its position in the camera frame is  ${}^C\mathbf{P} = [X_C Y_C Z_C]^T$ . The relationship between  ${}^G\mathbf{P}$  and  ${}^C\mathbf{P}$  is given by the rigid body transformation equation:

$${}^C\mathbf{P} = {}^C R \cdot {}^G\mathbf{P} + {}^C\mathbf{P}_{G, org} \quad (3.1)$$

where

${}^cR = 3 \times 3$  rotation matrix that rotates the global frame into the camera frame, the elements of which are  $r_{11}, r_{12}, r_{13}, r_{21}, \dots, r_{33}$

and

${}^cP_{G,org} = [T_X \ T_Y \ T_Z]^T$ , translation vector from the origin of the camera frame to the origin of the global frame, in camera frame coordinates, as shown in Figure 3.1

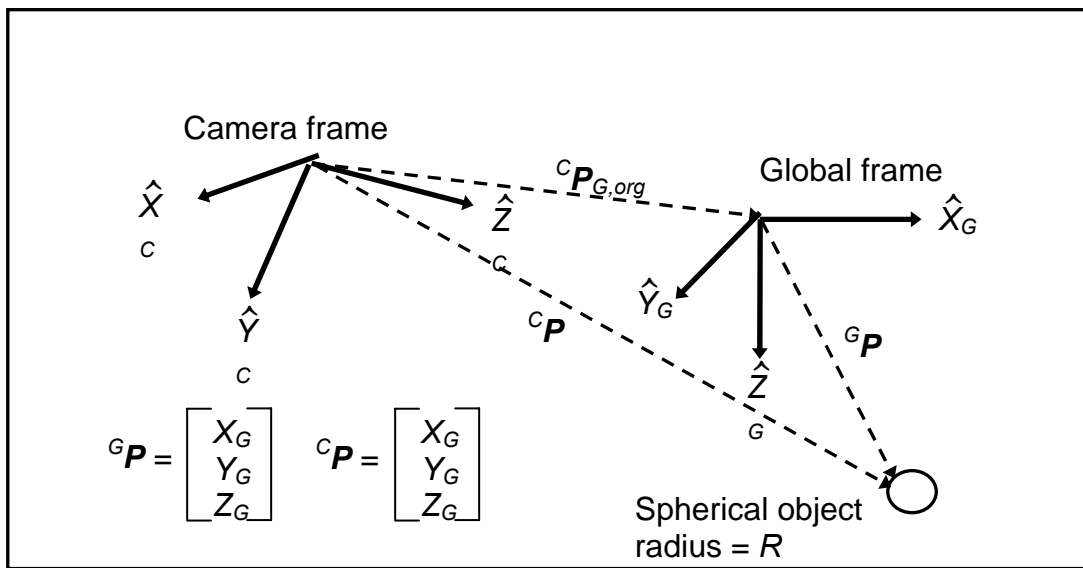


Figure 3.1 Conversion from global coordinates  $[X_G \ Y_G \ Z_G]^T$  to camera coordinates  $[X_c \ Y_c \ Z_c]^T$

A rotation matrix must be orthonormal, setting constraints on column unit-vector magnitude, and inter-column orthogonality. With these six constraints in mind, only three independent calibration parameters are required to fully specify the nine-member matrix  ${}^cR$ . There are a variety of ways to reduce a rotation matrix to an equivalent set of three parameters. The technique used in the calibration of VPS is known as X-Y-Z fixed angles [16]. In this formulation, three rotation

angles,  $R_X$ ,  $R_Y$  and  $R_Z$ , which can be alternately thought of as roll, pitch and yaw angles, are used to specify a rotation matrix.

To illustrate the use of X-Y-Z fixed angles, consider two frames,  $A$  and  $B$ , initially aligned. To orient frame  $B$  in the orientation specified by a set of X-Y-Z fixed angles, frame  $B$  would first be rotated by  $R_X$  about the  $X_A$  axis, then by  $R_Y$  about the  $Y_A$  axis, and finally by  $R_Z$  about the  $Z_A$  axis. Notice that the reference axes, about which the rotations are applied, retain a constant orientation.

The three X-Y-Z fixed angles, combined with the three components of the vector  ${}^C P_{G,org}$ , comprise the six extrinsic calibration parameters that define the physical orientation and position of a camera with respect to a selected global reference frame.

### 3.2.2 Step Two: $[X_C Y_C Z_C]^T$ to $[X_U Y_U A]$ Conversion

The initial creation of an image is described mathematically with perspective projection presuming pinhole camera geometry (Figure 3.2).

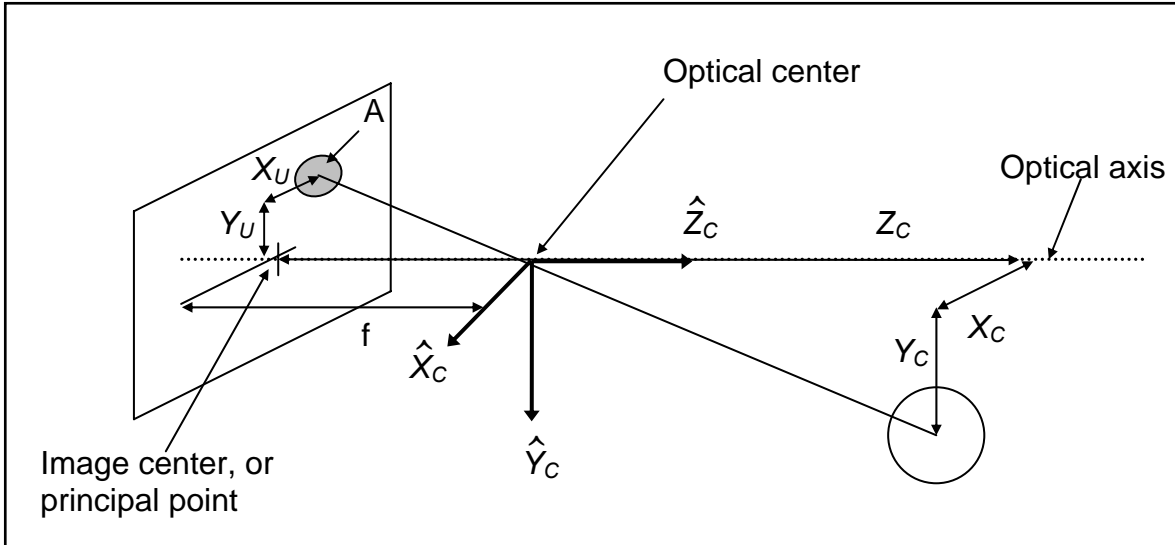


Figure 3.2 Conversion from camera coordinates  $[X_c \ Y_c \ Z_c]^T$  to ideal image coordinates  $[X_u \ Y_u \ A]$

Assume that a spherical object, of radius  $R$ , is being observed. Relative to the camera, the object is at the position  $[X_c \ Y_c \ Z_c]^T$ . The third component of the position vector in camera coordinates is known as the range of the object, or the distance from the camera to the object along the camera's optical axis. Under the pinhole perspective projection assumption, the image coordinates of the object, in units of length (e.g. mm), are given by the equations

$$X_u = \frac{f \cdot X_c}{Z_c} \quad (3.2)$$

and

$$Y_u = \frac{f \cdot Y_c}{Z_c} \quad (3.3)$$

The image area of the spherical object will be



$$A = \pi \cdot r^2 \quad (3.4)$$

where

$$r = \frac{f \cdot R}{Z_c} \quad (3.5)$$

To convert from camera frame coordinates to undistorted image coordinates, the only camera calibration parameter required is the focal length,  $f$ .

### 3.2.3 Step Three: $[X_U \ Y_U \ A]$ to $[X_D \ Y_D \ A]$ Conversion

Camera systems that are composed of optical elements with spherical surfaces suffer from unavoidable geometric distortions. The image coordinates of an observed object will be displaced farther from (pin-cushion distortion) or closer to (barrel distortion) the optical axis compared to the coordinates predicted by the pinhole projection model. This displacement grows in magnitude as the image coordinates get farther from the optical axis because the angle between the incoming ray of light and optical axis is greater at these coordinates.

Distortion occurs both radially and tangentially from the optical axis. Radial distortion is illustrated below in Figure 3.3. The displacement due to radial distortion is typically modeled by the equations

$$\partial X_r = X_D \cdot (K_{r1} \cdot r^2 + K_{r2} \cdot r^4 + \dots) \quad (3.6)$$

$$\partial Y_r = Y_D \cdot (K_{r1} \cdot r^2 + K_{r2} \cdot r^4 + \dots) \quad (3.7)$$

where

$$r^2 = X_D^2 + Y_D^2 \quad (3.8)$$

Similarly, the displacement due to tangential distortion can be modeled by the equations

$$\partial X_t = Y_D \cdot (K_{t1} \cdot r^2 + K_{t2} \cdot r^4 + \dots) \quad (3.9)$$

$$\partial Y_t = X_D \cdot (K_{t1} \cdot r^2 + K_{t2} \cdot r^4 + \dots) \quad (3.10)$$

Note that for equations 3.6 to 3.12, all values of  $X$ ,  $Y$ , and  $r$  are in physical units of length (e.g. mm). Due to the physics of optical elements, radial and tangential distortion is proportional to the even powers of  $r$ , and thus only the even powers of  $r$  appear in the distortion models. The values  $K_{r_i}$  and  $K_{t_i}$  are the distortion coefficients that must be characterized in order for a camera system to be calibrated for distortion. In many vision applications using off-the-shelf CCD TV cameras and lenses, tangential distortion is insignificant and ignored, and only the first power series of the radial distortion model is retained. This standard was applied to VPS, yielding equations relating the undistorted ( $U$ ) and distorted ( $D$ ) image coordinates as

$$X_U = X_D \cdot (1 + K \cdot r^2) \quad (3.11)$$

$$Y_U = Y_D \cdot (1 + K \cdot r^2) \quad (3.12)$$

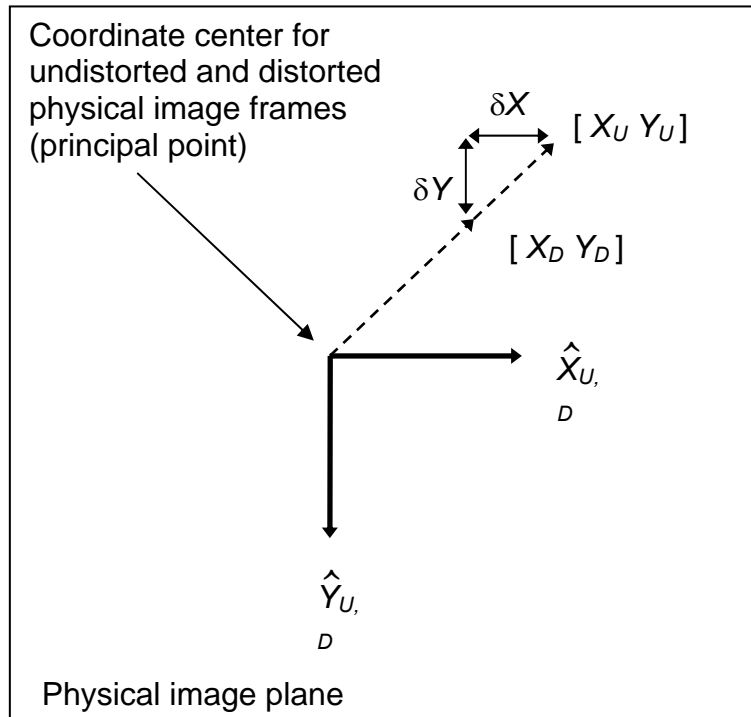


Figure 3.3 Conversion from ideal image coordinates  $[X_D Y_D A]$  to distorted image coordinates  $[X_U Y_U A]$

The conversion between undistorted image coordinates, and the real, distorted image coordinates thus only requires one camera parameter, the first-term radial distortion coefficient  $K$ . Note in (3.11) and (3.12), conversion from  $[X_U Y_U]$  to  $[X_D Y_D]$  involves solving a system of 2 nonlinear equations, while conversion from  $[X_D Y_D]$  to  $[X_U Y_U]$  involves only simple addition and multiplication. This is fortunate, since the latter conversion is required for VPS. Note also that the object area  $A$ , as it is used in VPS, is not significantly affected by distortion.

### 3.2.4 Step Four: $[X_D Y_D A]$ to $[X_{FD} Y_{FD} A]$ Conversion

The distorted image coordinates are in units of length, typically mm. Before an image can be processed in software, it must be sampled and digitized. Thus, the distorted image coordinates of an object must be transformed into digital image coordinates, or frame coordinates (as in a captured frame, not a coordinate frame).

In modern CCD video cameras, light collected by the optical system falls onto a CCD (Charge-Coupled Device) chip, and is converted into an analog or digital video signal. A CCD chip, typically a few mm in length and width, is made up of a two dimensional array of very small sensor elements, or sels. A single frame, or snapshot at some instant in time, can be captured using a frame grabber. A frame grabber is a device that can sample the analog video signal originating from the CCD chip and record that sample as a digital image. A digital image is simply an array of intensities saved in a computer memory medium. The individual, discrete intensities in the digital image are called pixels. Thus, coordinates in a digital image are measured as the number of pixels away from the origin of the image in the X and Y directions. The origin of an image is by convention set to be the upper left-hand corner, with positive X pointing to the right, and positive Y pointing down. A digital image array is two-dimensional if the image is black-and-white, with each pixel containing a gray-scale intensity value. A color image is represented by a set of three two-dimensional arrays of

intensities, one for each of the three primary colors: red, green and blue. The conversion from distorted image coordinates to frame coordinates involves several parameters. Figure 3.4 illustrates this conversion.

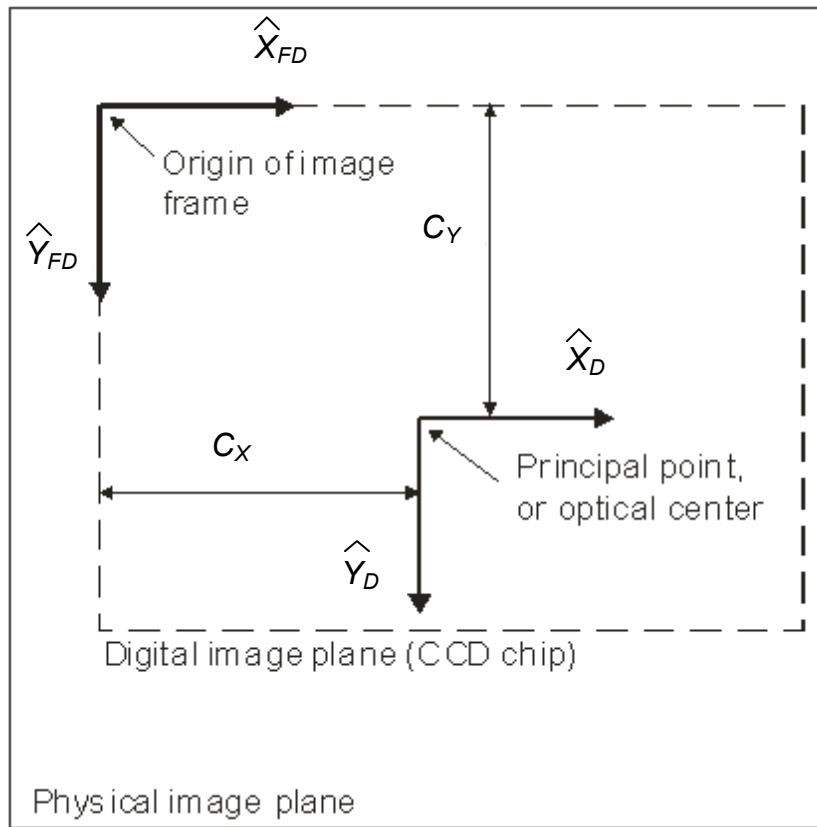


Figure 3.4 Conversion from distorted image coordinates  $[X_D Y_D A]$  to distorted computer image coordinates  $[X_{FD} Y_{FD} A]$

Ideally, the center of the digital image plane, or CCD chip, would be perfectly aligned with the optical axis, or center of the physical image plane. This is never the case in reality though, and the coordinates of the optical axis in the digital image frame must be found experimentally. These coordinates, measured in pixels, are often referred to as the piercing point  $[C_x, C_y]$ . Physically, the optical

axis intercepts the CCD chip, not the frame grabber, so intuitively,  $C_X$  and  $C_Y$  should be measured in sels, not pixels. They are in fact measured in pixels, however, because the important value in vision algorithms is the computer image equivalent location of the piercing point, not the physical piercing point.

Since distorted image coordinates are (typically) measured in mm, and image coordinates are in pixels, unit conversion must also occur. For the Y direction, this term ( $D_Y$ ) depends on the number of sels in the CCD chip in the Y direction, and their physical spacing.

$$D_Y = \frac{H_{CCD}}{N_{FY}} \quad (3.13)$$

where

$H_{CCD}$  = height (Y direction dimension) of CCD chip, mm

$N_{FY}$  = number of sels on the CCD chip in the Y direction

The distorted computer image Y coordinate in pixels is then given by:

$$Y_{FD} = \frac{Y_D}{D_Y} + C_Y \quad (3.14)$$

In the X direction, the mm-to-pixels conversion term is slightly different because a frame grabber samples a video signal in the X direction. The output from the CCD chip is transmitted by the camera one row followed by another, and

sampled in a similar manner by the frame grabber. Therefore, at the frame grabber, the sample spacing in the Y direction is determined by the Y direction spacing of the sels on the CCD chip. In the X direction, however, the frame grabber sampling determines the spacing, so the conversion factor depends on the number of X direction pixels that are sampled. The conversion factor ( $D_{X'}$ ) is defined by the equation

$$D_{X'} = \frac{W_{CCD}}{N_{FX}} \cdot \frac{N_{FX}}{N_{CX}} \quad (3.15)$$

where

$W_{CCD}$  = width of the CCD chip, mm

$N_{FX}$  = number of sels on the CCD chip in the X direction

$N_{CX}$  = number of pixels sampled by the frame grabber in the X direction

The equation for the X direction coordinate in the distorted digital image is then defined by the equation

$$X_{FD} = \frac{X_D}{D_{X'}} + C_X \quad (3.16)$$

One final parameter is required to accurately convert from real image data to digital image data, and it is related to the horizontal scanning mentioned above. The analog signal coming from a CCD video camera is initially a discrete, or staircase, signal. Each “step” on the staircase signal corresponds to a different

sel on the CCD chip. The long series of steps is formed from the outputs from one row of sels on the CCD chip, then the output from the next row, and so on. The signal is then low-pass filtered to blur the boundaries between sels, and form a smooth signal. When a frame grabber samples such a signal, the image is being scanned in the horizontal direction. A problem arises when this sampling occurs, because the X direction spacing of the frame grabber samples is never exactly equal to the X direction spacing of the CCD sels. Thus, a scaling parameter,  $s_x$ , must be introduced to account for this horizontal sampling error. This is only a problem in the X direction, because the Y direction sel spacing controls the frame grabber sampling in the Y direction.

When the ratio of pixel spacing is taken into account, equation (3.16) becomes

$$X_{FD} = s_x \cdot \frac{X_D}{D_{X'}} + C_x \quad (3.17)$$

The three parameters,  $C_x$ ,  $C_y$ , and  $s_x$ , required to model the transformation of real, distorted image coordinates into distorted digital image coordinates, are the last of the eleven calibration parameters used for VPS.

### **3.3 General Approach to Camera Calibration**

Many camera calibration techniques have been developed over the years, by both the machine vision and photogrammetry communities [17]. Each differs in details, but nearly all of the calibration techniques follow the same general path



to finding the numerical calibration parameter values. The calibration processes are overviewed in Figure 3.5. Calibration algorithms typically accept two sets of input data, known 3D coordinates and corresponding measured 2D image points. This data is then passed through a set of equations to compute the calibration parameters.

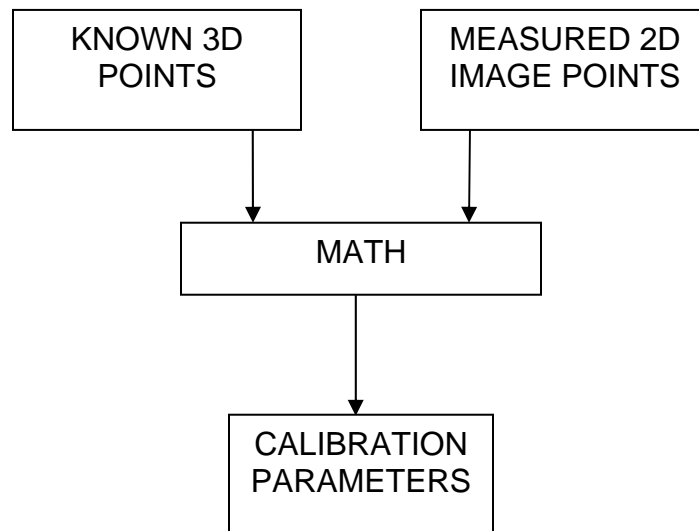


Figure 3.5 Generalized diagram of calibration techniques

The first input to most calibration algorithms is a series of calibration points, or locations, in 3D space. These points are typically targets or features on a calibration fixture. Some calibration algorithms are designed to use non-coplanar 3D calibration points – points dispersed throughout a volume. To simplify the mathematics involved, other algorithms use coplanar calibrations points. This simplification comes at a price however, as coplanar calibration algorithms have

reduced accuracy and capability when compared to non-coplanar techniques [15][17][18][19].

3D calibration targets must have two key characteristics in order to serve as input to a calibration algorithm. First, when the calibration fixture is viewed in a computer image, the 2D image coordinates corresponding to each 3D world location must be easily and accurately identifiable. To facilitate this, targets are usually defined as the centers of spheres in non-coplanar calibration fixtures. For coplanar calibration fixtures, they are usually defined as the intersection of lines or the vertices of squares. The classic coplanar calibration pattern is a checkerboard.

The second characteristic is that the 3D locations  $[X_G \ Y_G \ Z_G]^T$  of the calibration points or targets must be known relative to one another. It is convenient, but not necessary, to allow one of the calibration targets to define the origin for the calibration fixture and of the global coordinate system –once the calibration fixture is removed, the former location of the target remains the global origin. The known 3D coordinates of the calibration points form the first input to a camera calibration algorithm.

The second input is the set of 2D image coordinates corresponding to each (or to as many as possible) of the 3D calibration targets. To obtain these 2D coordinates, an image of the calibration fixture is taken with the camera to be

calibrated. The image can then be analyzed, either automatically or manually, to identify the 2D image locations of the 3D targets. Care must be taken that specific 2D coordinates are paired with the correct 3D coordinates. Otherwise, the calibration algorithm will render invalid output.

Given a series of 3D world coordinates, and a series of corresponding 2D coordinates from a captured computer image as inputs, it is possible to find the calibration parameters of the camera that was used to generate the image. This can be done in one of two ways. In the first, each point in the 2D and 3D coordinate tables is used to define an equation. The system of equations developed from all of the points, once solved, provide the calibration parameter values. The second way is to use each point to create a term in an error function, and then use an optimization algorithm to find the combination of calibration parameters that minimizes the error function.

While similar in approach, calibration techniques can differ greatly from each other mathematically. Optimizing calibration algorithms differ from each other in optimization techniques, the structure of the objective function, the manner in which the initial conditions are generated, etc. Algorithms that do not employ nonlinear optimization solve linear equations, often in a least-squares sense, generated from the world and image coordinates. As with the nonlinear optimizing calibration algorithms, a variety of linear techniques are also available.

See references [15] and [17] for more details on the advantages and disadvantages of the many calibration techniques that exist.

Note that because lens distortion is an inherently non-linear process, it must be ignored by the purely linear techniques. Therefore, precision vision applications must rely on nonlinear calibration methods, such as the method developed by Roger Tsai [15]. This calibration technique is discussed in more detail in Chapter 4 and was used to calibrate VPS because of its characterization of a full set of camera calibration parameters, distortion coefficients included, and its nonlinear refinement of the parameter estimates.

## Chapter 4

### Calibration of VPS

The calibration of the VPS cameras presented several challenges seldom encountered in other machine vision tasks. Typically, when calibrated volumes are small, a few cubic meters at most, small, highly accurate calibration fixtures can be used. Large volumes mean both long distances, where errors are magnified, and large calibration fixtures, which are difficult to manufacture and characterize accurately. Because the cameras are all pointing inwards instead of in one general direction, simple, coplanar calibration patterns could not be used for extrinsic calibration. The underwater environment also created additional procedural overhead to all calibration activities.

This chapter describes how these challenges were addressed. The first section discusses previous VPS calibration work that motivates the new technique. Section 4.2 describes the VPS calibration process, while Sections 4.3 and 4.4 presents the intrinsic and extrinsic calibrations, respectively.

#### ***4.1 Motivation for a New Calibration Technique***

In previous VPS calibration work [13], a partial set of intrinsic and a full set of extrinsic parameters were obtained with a single ball target, moved to multiple tank locations. This was achieved through a technique developed from the

ground up at great effort. The initial calibration successfully allowed VPS to produce static position estimates that agreed, from camera to camera, to approximately 15 cm. It had several limitations, however. Distortion, piercing point and the X and Y axis scaling factor  $s_x$  were not modeled. This is understandable, because these parameters are often only barely significant. Also, it was impossible, despite great effort, to immobilize the calibration target due to water currents, which made it also impossible to locate it with a high degree of accuracy. Finally, the range of the calibration target was calculated based on its image area, and this value was subsequently used in calculating calibration parameters. Object image area is an extremely difficult parameter to measure accurately due to its sensitivity to illumination conditions, and is typically wise to avoid if possible.

Based on the lessons learned from the initial calibration effort, a new VPS calibration method was developed. To facilitate implementation and capitalize on previous work by the vision community, a strategy of using calibration techniques and software that have been validated in academia and industry was adopted. This strategy provided several advantages. First, techniques existed that allowed the full complement of intrinsic parameters to be estimated simultaneously. This was important because while distortion is small, it can be significant in portions of the FOV of the VPS cameras. Also,  $f$  should be estimated simultaneously with the distortion coefficient, because distortion can change the apparent size of an object, and thus  $f$ . A characterization of the

accuracy of the calibration could also be generated. The intrinsic parameters, especially  $f$ , act as inputs to the extrinsic calibration process. Errors in the intrinsic parameters are magnified in the extrinsic parameter estimation. It was hoped that the accuracy of VPS calibration could be improved to allow vehicle position estimation in the accuracy range of a few cm. Some techniques developed in the computer vision community also avoided the use of area, which is desirable. Finally, it was hoped that this strategy would greatly reduce the time required to create a new VPS calibration procedure set since it employed existing software and techniques.

## **4.2 Overview of New Calibration Technique**

The new VPS calibration technique is a two-step process that calibrates intrinsic and extrinsic parameters separately. Both steps are performed with the cameras mounted in their operational positions, which ensures calibration parameters include the effects of the camera housing assembly and the underwater NBRF environment.

The first step involves taking multiple images of a checkerboard pattern and processing them automatically to compute the intrinsic parameters  $f$ ,  $C_x$ ,  $C_y$ , and  $s_x$ . The second step involves a large calibration fixture with 20 fixed spherical calibration targets attached, each with a known relative position. Images of the calibration fixture are taken and processed manually to find the 2D image locations of each target. The 2D image target coordinates and their

corresponding 3D physical target coordinates are inputs to an algorithm. The algorithm computes the extrinsic calibration of the camera that took the image. This extrinsic calibration is relative to a global reference frame attached to the calibration fixture. As part of the extrinsic calibration, the distortion coefficient is also calculated.

The algorithm used in the extrinsic calibration can, in fact, be used to calculate all of the calibration parameters. The intrinsic parameters are calibrated separately, however, because they can be found with far greater accuracy in this way. The extrinsic calibration algorithm begins by computing all eleven parameters. It then discards its values for  $f$ ,  $C_x$ ,  $C_y$ , and  $s_x$ , and replaces them with the accurate values from the earlier intrinsic calibration. It then uses these intrinsic parameters, and its original estimate of  $K$ , to recalculate the extrinsic parameters to complete the calibration process.

A description of both algorithms and results from the intrinsic-extrinsic calibration sequence are found below. Because these methods were taken from the literature, the discussion focuses on the implementation details and accuracy of the results. Step-by-step instructions on calibrating VPS can be found in [20].

### **4.3 VPS Intrinsic Calibration**

The VPS cameras were intrinsically calibrated using the MATLAB™ calibration toolbox [21], a MATLAB™ implementation the algorithm reported in [19].



As is standard in the vision community, the 3D calibration points in this algorithm are the distinct corners of squares on a black and white checkerboard. For VPS calibration, a large checkerboard was made that was suitable for use under water. The squares are 57.85 mm on each side and the usable checkerboard is 9 squares by 9 squares. The calibration process begins by taking a series of images of the checkerboard with the camera that is to be calibrated. The images must capture a wide variety of checkerboard orientations, and should be nearly filled by the checkerboard. This can be accomplished if a diver slowly changes the orientation of the checkerboard while suspending it in front of a camera that is taking images. Each orientation of the checkerboard provides a series of 3D calibration points with known relative locations.

After the user loads the checkerboard images into the MATLAB™ calibration program, the program asks the user for the number and size of the squares, and prompts the user to identify on each of the images the outer corners of the usable checkerboard. The term “usable checkerboard” means the part of the checkerboard that does not include any boundary squares. These boundary squares may not be full squares. The outer corner of the usable checkerboard will be one boundary in from the absolute edge of the checkerboard. The first corner identified by the user in each image is considered to be the global origin. The rest of the points (or corners) will have 3D coordinates (relative to the origin) determined by the number of squares between them and the origin. Both the

global X and Y coordinates of each point will be a multiple of 57.85 mm (the dimension of the squares), and the global Z coordinate will be zero, because the points are all coplanar. Figure 4.1 shows two typical checkerboard images used for the intrinsic calibration.

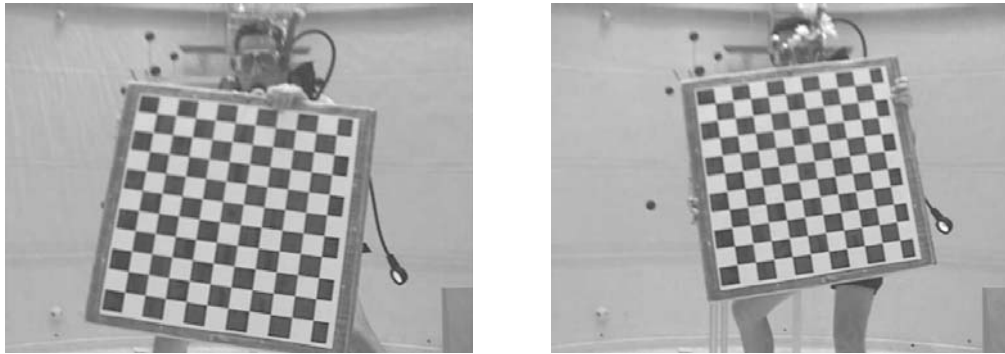


Figure 4.1 Sample checkerboard images for VPS intrinsic calibration

Using initial guesses based on the square size and the user-identified outer corners of the checkerboard, the program then searches for and finds the corners of all of the squares using edge-detection. The locations of the corners in the image become the 2D coordinates corresponding to the 3D planar locations.

The algorithm that solves for the intrinsic parameters has three steps. A closed form, analytical solution provides initial estimates of the extrinsic parameters<sup>2</sup>, and the intrinsic parameters minus the radial distortion coefficients. It is followed

---

<sup>2</sup> In this algorithm, a set of extrinsic parameters, spatially relating the camera to the checkerboard, will be generated for each image, whereas all of the images are used to compute the intrinsic parameters. The extrinsic parameters calculated in this step are discarded.

by a linear optimization to estimate the radial and tangential distortion coefficients, and finally a full nonlinear optimization to refine all of the parameter estimates.

If lens distortion is ignored, the relationship between the 2D image coordinates  $[X_{FD} \ Y_{FD}]^T$  and the 3D coordinates  $[X_G \ Y_G \ Z_G]^T$  is linear. Setting  $\mathbf{m} = [X_{FD} \ Y_{FD}]^T$ , and  $\mathbf{M} = [X_G \ Y_G \ Z_G]^T$ , that relationship is represented by the following system

$$s \cdot \begin{bmatrix} \mathbf{m} \\ 1 \end{bmatrix} = A \cdot \begin{bmatrix} {}^G R \\ {}^C P_{G,org} \end{bmatrix} \begin{bmatrix} \mathbf{M} \\ 1 \end{bmatrix} \quad (4.1)$$

Here,  $s$  is a scaling factor,  ${}^G R$  and  ${}^C P_{G,org}$  are the extrinsic calibration parameters (the rotation matrix and translation vector), and  $A$  is a 3x4 matrix whose elements are algebraic combinations of the intrinsic calibration parameters, excluding any distortion parameters. By the means of several linear algebra manipulations, three  $(\mathbf{m}, \mathbf{M})$  sets will provide a closed-form solution to  ${}^G R$ ,  ${}^C P_{G,org}$ , and the intrinsic parameters. More data sets are used as input to a linear (least-squares) optimization of the parameters. This first, linear solution step is nearly identical to the first step of the extrinsic calibration algorithm, described in the Section 4.4.

Assuming that distortion is small, these estimated parameters should be quite close to the truth. The next step is to use these estimates to find estimates of the

distortion parameters.  $[X_{FD} \ Y_{FD}]^T$  and  $[X_{FU} \ Y_{FU}]^T$  are the 2D distorted and undistorted, respectively, digital (measured in pixels) image coordinates, while  $[X_D \ Y_D]^T$  and  $[X_U \ Y_U]^T$  are their 2D real, or physical (measured in mm), counterparts. The estimated parameters are used with a set of 3D coordinates  $\mathbf{M}$  to calculate both the ideal real image coordinates,  $[X_U \ Y_U]^T$ , and the ideal digital image coordinates,  $[X_{FU} \ Y_{FU}]^T$ . For two-term radial distortion, these can be combined with equations (3.6), (3.7), (3.14), and (3.17) to form the system

$$\begin{bmatrix} (X_{FD} - C_X) \cdot (X_U^2 + Y_U^2) & (X_{FD} - C_X) \cdot (X_U^2 + Y_U^2)^2 \\ (Y_{FD} - C_Y) \cdot (X_U^2 + Y_U^2) & (Y_{FD} - C_Y) \cdot (X_U^2 + Y_U^2)^2 \end{bmatrix} \begin{bmatrix} K_{r1} \\ K_{r2} \end{bmatrix} = \begin{bmatrix} X_{FU} - X_{FD} \\ Y_{FU} - Y_{FD} \end{bmatrix} \quad (4.2)$$

where  $[X_{FD} \ Y_{FD}]^T$  are the 2D coordinates  $\mathbf{m}$  corresponding to  $\mathbf{M}$ . With  $n$  corresponding  $(\mathbf{m}, \mathbf{M})$  sets, the obvious  $2n$ -dimension linear system is then constructed to solve for  $K_{r1}$  and  $K_{r2}$  in a least-squares sense. It is important to point out that these distortion coefficients are not actually used in the VPS model, but are necessary to refine the other intrinsic parameters. The value of  $K$  actually used in VPS is calculated during extrinsic calibration, and is discussed below.

Finally, nonlinear optimization, to refine all of the parameters simultaneously, is performed by minimizing the cost functional

$$\sum_{i=1}^n \left\| \mathbf{m}_i - \mathbf{m}(A, K_{r1}, K_{r2}, C, R, P_{G,org}, \mathbf{M}_i) \right\|^2 \quad (4.3)$$

using the same  $n$  sets of  $(\mathbf{m}, \mathbf{M})$ .

#### 4.3.1 VPS Intrinsic Calibration: Results and Analysis

Table 4.1 below summarizes the intrinsic calibration results for the VPS cameras.

These results all fall within the expected ranges of the respective parameters.

The lenses of the VPS cameras accommodate focal length  $f$  adjustment from five mm to 40 mm, and each camera was focused to give it nearly as wide of a field of view as possible (the shortest possible  $f$ ). Thus,  $f$  values near 5 mm were expected.  $C_X$  and  $C_Y$  should generally not be far from their ideal values, 320 pixels and 240 pixels, respectively, which is the case. The  $C_Y$  values for cameras 3 and 4 are a bit higher than expected, but still acceptable. The  $s_X$  and  $K$  values are all near to the ideal values of one and zero, as expected.

It is important to note that the camera model used in the intrinsic parameter estimation software uses representations of focal length and distortion different from those used in VPS and the extrinsic calibration software [22]. This was not a problem with regards to focal length  $f$ , as a simple calculation converts from one representation to the other. The distortion representations, however, could not be equated. The MATLAB™ calibration toolbox represented distortion as in [18].

$$X_D = X_U \cdot (1 + K_A \cdot r^2) \quad (4.4)$$

$$Y_D = Y_U \cdot (1 + K_A \cdot r^2) \quad (4.5)$$

where

$$r = X_U^2 + Y_U^2 \quad (4.6)$$

The representation of distortion in VPS was identical to that in the extrinsic calibration software, described in [15].

$$X_U = X_D \cdot (1 + K_B \cdot r^2) \quad (4.7)$$

$$Y_U = Y_D \cdot (1 + K_B \cdot r^2) \quad (4.8)$$

where

$$r = X_D^2 + Y_D^2 \quad (4.9)$$

Since  $K_A$  cannot readily be converted into  $K_B$ , the distortion coefficient calculated using the checkerboard images could not be used in the determination of the extrinsic parameters, or in the VPS vision algorithms. For this reason, only the other four intrinsic parameters calculated in this step were used, while the distortion coefficients were computed concurrently with the extrinsic parameters, by the technique discussed in the Section 4.4.

Table 4.1 VPS intrinsic calibration results

	<b><math>f</math> (mm)</b>	<b><math>C_x</math> (pix)</b>	<b><math>C_y</math> (pix)</b>	<b><math>s_x</math></b>	<b><math>K</math></b>
<b>Camera 1</b>	6.3301	279.19	225.98	1.0251	0.007183
<b>Camera 2</b>	7.1428	324.20	217.64	1.0268	0.005184
<b>Camera 3</b>	7.9902	303.38	303.14	1.0274	0.002120
<b>Camera 4</b>	6.7967	314.00	325.01	1.0265	0.008442
<b>Camera 5</b>	5.7769	320.03	230.10	1.0266	0.009502
<b>Camera 6</b>	6.5814	300.19	216.75	1.0257	0.005405
<b>Camera 7</b>	7.3099	298.95	253.82	1.0255	0.005726
<b>Camera 8</b>	6.8419	330.62	242.48	1.0264	0.005833

The MATLAB™ calibration toolbox program supplies, along with calibration parameter values, estimates of the accuracy of those values. Below is a table containing the upper bounds for the errors in the intrinsic calibration parameters for each camera, as computed by the program.

Table 4.2 Error bounds (+/-) on the VPS intrinsic calibration parameters

	<b><math>f</math> (mm)</b>	<b><math>C_x</math> (pix)</b>	<b><math>C_y</math> (pix)</b>	<b><math>s_x</math></b>	<b><math>K</math></b>
<b>Camera 1</b>	0.0061	1.30	1.30	1.54E-05	n/a
<b>Camera 2</b>	0.0065	1.52	1.42	1.40E-05	n/a
<b>Camera 3</b>	0.0092	1.95	1.85	1.83E-05	n/a
<b>Camera 4</b>	0.0051	1.18	1.20	1.18E-05	n/a
<b>Camera 5</b>	0.0066	1.43	1.53	1.83E-05	n/a
<b>Camera 6</b>	0.0058	1.60	1.51	1.39E-05	n/a
<b>Camera 7</b>	0.0072	1.71	1.72	1.59E-05	n/a
<b>Camera 8</b>	0.0045	1.22	1.12	1.04E-05	n/a

When each of the checkerboard image sets was initially analyzed with the MATLAB™ calibration toolbox, the calculated errors were much higher. The

errors were reduced by three different techniques. First, the total number of images used to calculate the parameter estimates was increased, and the parameters were recalculated. This drove down the uncertainty in the estimates. Second, using error analysis tools built in to the program, specific images that were causing large amounts of error were eliminated, and the parameters were again recalculated. Finally, 2D image coordinates of checkerboard vertices that were causing large errors were identified, again using built-in error analysis tools. The program was instructed to recalculate the coordinates of these vertices by re-running the edge-detection process. With the refined 2D coordinates, a final set of calibration parameters was computed. Accuracy goals of 0.01 mm for focal length, and two pixels for  $C_x$  and  $C_y$  were used as guidelines to know when to stop refining the intrinsic calibration. This appears to be roughly the limit of capability for the VPS equipment and configuration, but is seen as sufficient to give our cm-accuracy final positioning goal. The accuracy of  $s_x$  in the vision equations is not as important as that of  $f$ ,  $C_x$ , or  $C_y$ , and was accepted once those other parameters were determined to be sufficiently accurate. The values of  $K$  in Table 4.1 were calculated by the algorithm discussed in Section 4.4. The implementation of this algorithm does not calculate accuracy estimates for  $K$ , but does provide estimates for overall calibration quality, as will be discussed. Thus, accuracy information for  $K$  to put in Table 4.2 is unavailable.



#### **4.4 VPS Extrinsic Calibration**

The extrinsic calibration of the VPS cameras follows a method reported in [15], known as Tsai's calibration method. A C language implementation of this method [22] was modified for use with VPS. Tsai's method computes the intrinsic and extrinsic calibration parameters simultaneously. The major modifications made to the software were to allow the use of the four intrinsic parameters found using the MATLAB<sup>TM</sup> calibration toolbox ( $f$ ,  $C_x$ ,  $C_y$ ,  $s_x$ ) to compute the extrinsic parameters.

The 3D calibration points for the extrinsic calibration were more difficult to create than for the intrinsic calibration. A calibration fixture was required that would define the 3D calibration reference points. The fixture had the following requirements:

1. It must have a minimum of 14 highly visible 3D targets: The algorithm required 14 data points in order to do a full optimization for all extrinsic parameters.
2. A minimum of 14 targets must be visible from all eight VPS cameras: All eight cameras must take their calibration images of a motionless fixture at the same time, for the cameras to all reference the same global coordinate frame. This suggested a small target volume, visible from all cameras.
3. The targets must be accurately "locatable" in images: Viewing a computer image of the fixture, a user must be able to easily and accurately locate

the image coordinates of the targets (minimum 14), from any viewing angle. This suggested a large target volume where targets would not occlude each other in any VPS camera view.

4. It must be easy to obtain accurate measurements of the target 3D positions: The  $[X_G \ Y_G \ Z_G]^T$  positions of the targets, with one target acting as the origin, or the  $[0 \ 0 \ 0]$  point, are required. These positions need to be an order of magnitude more accurate than the accuracy of the desired vision task [15]. Since the positioning accuracy goal of VPS is single-digit cm-scale accuracy, the target positions were required to within better than single-digit mm accuracy.
5. The targets must be dimensionally stable: The targets need to be rigid so that between calibrations their positions will not change significantly.
6. The targets must fill as large of a volume as possible: The accuracy of the calibration over the entire field of view of the camera improves if the fixture can occupy a large fraction of that field.
7. The fixture must be stored safely and easily in the NBRF: Disassembly would be undesirable because of dimensional stability and ease of use issues.
8. The fixture coordinate system must be aligned with the N-E-D inertial reference frame: For the positional and rotational reference frames to align, the fixture's X axis must be able to be aligned with magnetic North, and its Y axis with the East vector.

9. The fixture must be inexpensive: Only a few hundred dollars were available for the construction of the fixture.

Figure 4.2 shows the fixture that was constructed to meet the above requirements. The calibration frame is a cube, 127 cm (50") in dimension, made of square commercial structural aluminum extrusions.

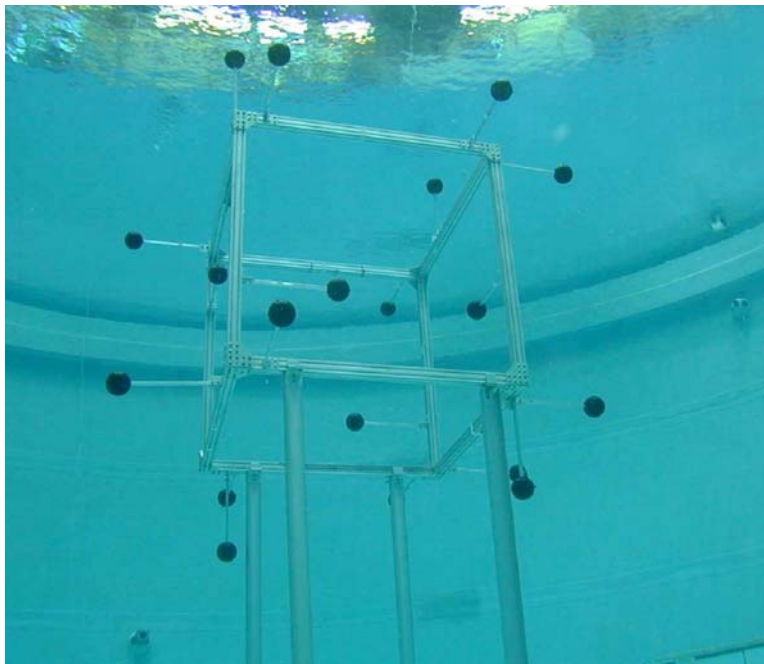


Figure 4.2 VPS calibration frame installed in the NBRF

Attached to the cube are 20 aluminum posts, approximately 36 cm (14") in length, with hollow plastic spheres attached to the end of each to act as the calibration targets. This number of targets allows a camera to still see more than the minimum of 14 targets even when a few are blocked from view by the

structure. The spheres were painted black to enhance their visibility and had holes drilled in them to allow water to enter and exit the fixture.

When in the NBRF, the calibration frame sits on top of four, approximately 3 m long stilts. Before a calibration, the stilts are rigidly attached to a heavy truss that remains permanently in the tank, and the frame is lowered onto the stilts using underwater air bags. Ball 19 is the origin of the global reference frame, and the vector from ball 19 to ball 18 defines the global X axis. Between these two balls is a fixture onto which a waterproof compass can be attached. This allows the frame to be slightly rotated to align its X axis with magnetic North. The Y axis is perpendicular to the X axis, and, assuming the frame is level, points East. Figure 4.3 illustrates the ball locations, where the compass attaches, and the calibration fixture's reference frame. When not in use, the calibration fixture hangs above the neutral buoyancy tank, as shown in Figure 4.4, from a dedicated crank-operated crane.

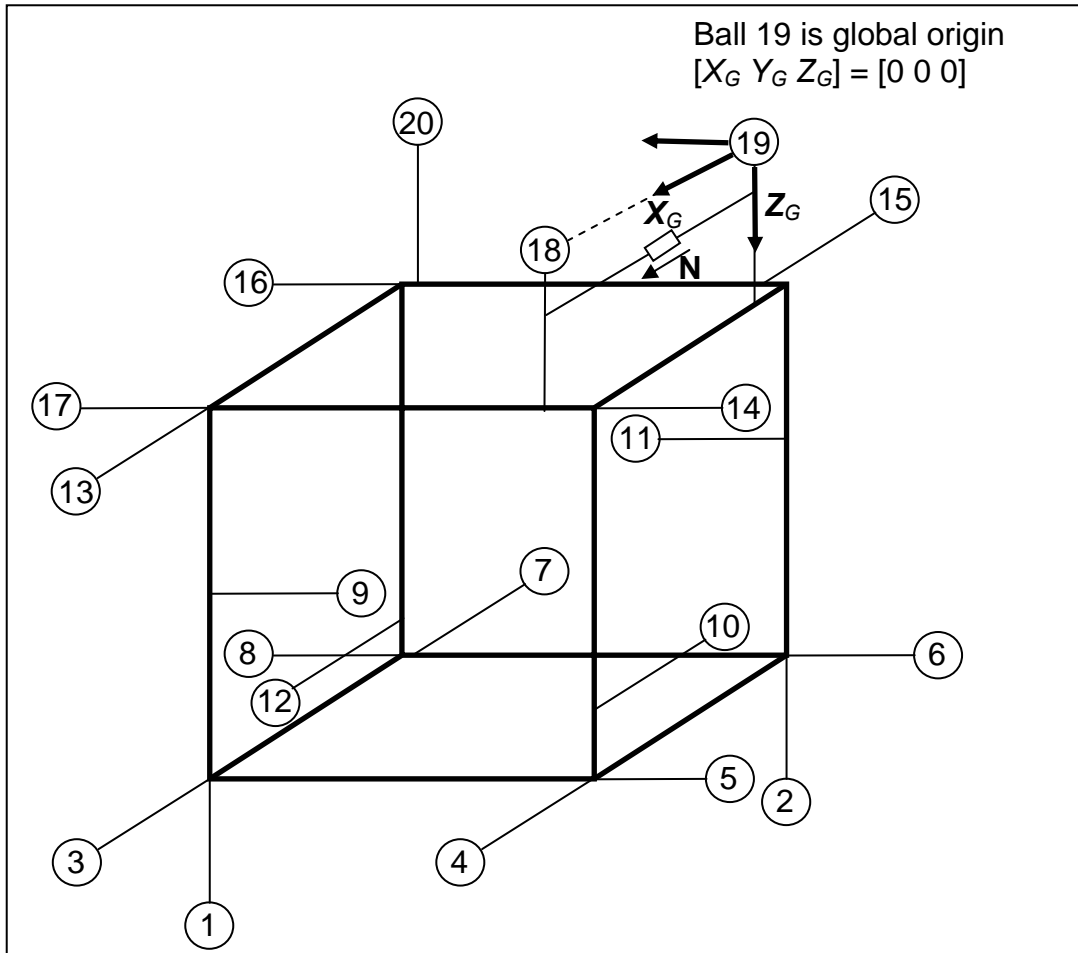


Figure 4.3 VPS calibration frame ball locations and reference frame



Figure 4.4 VPS calibration frame hanging above NBRF

In order to use the calibration fixture, accurate positions of each of the targets was required. Measuring the target positions in all three axes at once would be difficult and require complicated equipment. Measuring the positions one axis at a time would require three perpendicular global reference planes, and would also require complex measurement devices.

Therefore, the ball-to-ball distances were measured. These then acted as inputs into a minimization problem that solved for the 3D positions. The ball-to-ball distances were measured using a large set of calipers built especially for this purpose. They consist of a long aluminum square tube with a metal tape measure rigidly attached to it. Two sliding measuring blocks can be moved along

the length of the tube, and fixed at any desired position using cap screws. Attached to the measuring blocks are templates with semi-circular cutouts that match the diameter of the spherical targets. Beneath the semi-circular cutout, aligned with its center, is an edge for reading the millimeter mark on the tape measure. With the semi-circular templates positioned over two targets, the distance between the targets in mm can be found by subtracting the lower measurement from the higher one. Based on operational experience, the measurements taken with the caliper are considered accurate to a tolerance of plus or minus two mm. A picture of one of the sliding measuring blocks is shown below.



Figure 4.5 Sliding measuring block on VPS calibration frame caliper

With  $n$  targets, the potential number of target-to-target measurements,  $m$ , is defined by

$$m = \frac{(n^2 - n)}{2} \quad (4.10)$$

Thus, for 20 targets, there are 190 potential measurements. Because the caliper was occasionally obstructed by the structure of the frame, only approximately 180 measurements could actually be taken. Table A1 in the Appendix shows the measurements recorded for the frame. Any field that contains a zero indicates either that the measurement was not obtainable, or that its value is elsewhere in the table.

With 20 targets, there were 60 unknowns that had to be solved– the X, Y, and Z coordinates for each target. This number could have been reduced to 57, because one target was set to be the origin, but it was left at 60 to simplify the numerical analysis. The 180 measurements each created one term in a scalar objective function. If  $L_{j,k}$  was the measurement from ball j to ball k, and  $[X_j \ Y_j \ Z_j]^T$  and  $[X_k \ Y_k \ Z_k]^T$  were the true coordinates of balls j and k, then the term in the objective function for that measurement would be

$$e_{jk} = \left( \left\{ (X_j - X_k)^2 + (Y_j - Y_k)^2 + (Z_j - Z_k)^2 \right\} - L_{j,k}^2 \right)^2 \quad (4.11)$$

The scalar objective function would then be the sum of all of the terms



$$E = \sum_{j=1}^{20} \sum_{k=1}^{20} e_{jk} \quad (4.12)$$

Since the distance from any ball to itself is zero,  $e_{jk} = 0$  if  $j=k$ .

A minimization with a 180-term scalar objective function in 60 unknowns will have a multitude of local minima, making the solution quite sensitive to initial conditions. Figure 4.6 illustrates how a sufficiently accurate initial guess of the target coordinates was calculated, and how the global coordinate system was attached to the calibration frame.

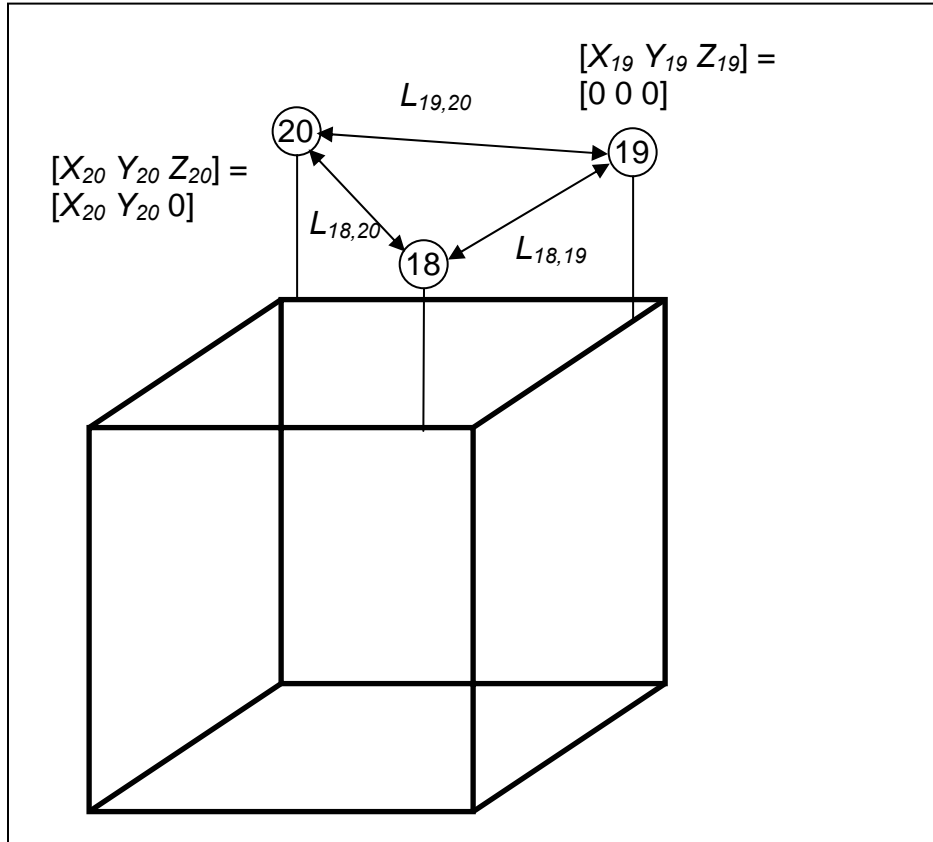


Figure 4.6 Illustration of VPS calibration frame reference targets

Accurate initial guesses of the target locations were computed deterministically. The plane defined by the top three targets, balls 18 thru 20, was defined as the X-Y plane. Balls 18 thru 20, by this definition, had a Z coordinate of 0. Ball 19 was assigned to be the origin and ball 18 was assumed to lie on (and thus define) the X axis. Ball 20 was assumed to be on the X-Y plane, at the location  $[X_{20} \ Y_{20} \ 0]^T$ . With the above assumptions, the 3D positions of targets 18 thru 20 can be solved deterministically using only the three inter-target distances,  $L_{18,19}$ ,  $L_{18,20}$ , and  $L_{19,20}$ , where  $L_{i,j}$  is the measured distance between ball  $i$  and ball  $j$ . Ball 19 defines the global coordinate system origin, while ball 18 defines the X axis at  $[L_{18,19} \ 0 \ 0]^T$ . The coordinates of ball 20 can then be solved:

$$L_{18,19}^2 = X_{20}^2 + Y_{20}^2 \quad (4.13)$$

$$L_{19,20}^2 = Y_{20}^2 + (X_{19} - X_{20})^2 \quad (4.14)$$

where  $X_{20}$  and  $Y_{20}$  are the only unknowns. This is illustrated in Figure 4.6.

The remaining 17 targets all lie on one side of the X-Y plane, thus all have positive Z coordinates. As a result, the 3D position of each subsequent target can be computed using only the three measurements between it and targets 18 thru 20. The system of equations for any ball  $k$  is

$$L_{k,18}^2 = X_k^2 + Y_k^2 + Z_k^2 \quad (4.15)$$

$$L_{k,19}^2 = (X_k - X_{19})^2 + Y_k^2 + Z_k^2 \quad (4.16)$$

$$L_{k,20}^2 = (X_k - X_{20})^2 + (Y_k - Y_{20})^2 + Z_k^2 \quad (4.17)$$

In this system of equations, only  $X_k$ ,  $Y_k$ , and  $Z_k$  are unknown. This system was solved algebraically for each of the remaining 17 targets. Using the resulting set of 3D positions as initial conditions, the minimization algorithm is then executed to adjust the estimated positions and find the combination that will best match the

physical measurements. Minimization was performed in MATLAB™ using the built-in FMINNS function, which is an implementation of the Nelder-Mead simplex minimization algorithm.

Table 4.3 shows the initial position estimates of the targets calculated deterministically using 54 measurements (3 measurements for targets 18 thru 20, and  $3 \times 17 = 51$  measurements for the remaining targets). Table 4.4 shows the position estimates after optimization with respect to all of the measurements.

Table 4.3 Target 3D initial position estimates

Ball #	X (mm)	Y (mm)	Z (mm)
1	1398.85	740.24	2089.87
2	8.43	9.42	2088.96
3	1789.22	570.14	1681.24
4	1520.81	-330.44	1675.8
5	850.79	-685.65	1670.35
6	-70.29	-411.05	1674.35
7	-205.3	1180.2	1686.93
8	509.49	1602.76	1651.37
9	1314.55	451.91	1076.24
10	654.09	-185.65	1371.74
11	31.18	471.09	794.28
12	677.42	1098.78	1283.04
13	1815.6	631.97	420.98
14	868.11	-687.5	428.81
15	-479.06	277.06	427.01
16	492.8	1603.85	425.64
17	1445.31	1319.23	441.31
18	1189	0	0
19	0	0	0
20	216.18	1152.91	0

Once an optimized set of 3D positions is obtained, a set of “synthetic” target  $j$  to target  $k$  distances,  $Ls_{j,k}$  can be calculated from

$$Ls_{j,k} = \sqrt{(X_j - X_k)^2 + (Y_j - Y_k)^2 + (Z_j - Z_k)^2} \quad (4.18)$$

Table 4.4 Target 3D position estimates, after optimization

Ball #	X (mm)	Y (mm)	Z (mm)
1	1397.93	738.90	2090.29
2	8.23	9.66	2088.10
3	1790.44	571.66	1680.08
4	1517.64	-333.17	1677.56
5	849.49	-688.31	1669.34
6	-64.94	-406.14	1679.47
7	-204.65	1180.65	1687.10
8	507.64	1600.93	1652.87
9	1313.35	447.97	1076.97
10	654.88	-187.56	1371.11
11	30.22	470.44	794.00
12	677.19	1097.55	1283.97
13	1815.94	633.41	420.27
14	868.62	-689.73	423.67
15	-478.86	277.79	424.83
16	494.40	1605.58	422.74
17	1446.52	1319.50	438.75
18	1189.35	0.00	0.00
19	0.00	0.00	0.00
20	216.81	1152.53	0.00

These synthetic measurements are then compared to the physical measurements. Since some measurements will naturally be better than others, measurements that exhibit large differences between the physical and synthetic values, and contribute disproportionately to the error term, are then discarded

and the minimization is run again to improve the optimization. Table A2 in the Appendix shows a set of synthetic ball-to-ball measurements, calculated after optimization. These measurements can be compared to the raw measurements in Table A1.

Raw measurements were removed from the optimization until the difference between each raw and synthetic measurement was less than 1.0 mm.

Approximately 30 of the 180 measurements were discarded in this manner. All of the discarded measurements differed from their synthetic counterparts by 1 to 3 mm, except for one measurement that differed by 5 mm. The discarded measurements, highlighted in Table A2, are dispersed among the targets roughly evenly. Note that because a new table of synthetic measurements was computed after the removal of each measurement, some of the highlighted measurements are identical to their corresponding real measurements to the precision shown. In the calculation of subsequent synthetic measurements, a difference of 1 mm or greater was observed, and these measurements were discarded.

If the minimum of three measurements is used to compute the position of target  $k$ , and they are all in error by only 1.0 mm, then  $|\mathbf{E}_{POS}|$ , the magnitude of the maximum error possible in the position vector of a ball  $k$ , is given by

$$|\mathbf{E}_{POS}|_{MAX,k} = \sqrt{3 \cdot 1.0^2} = 1.73 \text{ mm} \quad (4.19)$$

From (4.19), the true center of each target will be assumed to reside within a sphere of radius  $\sqrt{3}$  mm, centered on the optimized target position. Reference [15] indicates that in general, knowledge of the 3D coordinates of calibration fixture targets should be an order of magnitude more accurate than the desired accuracy of the vision task. This means that based on these target locations, and assuming correspondingly good 2D image coordinate data is used, an extrinsic calibration can be computed that will be of a sufficient quality to allow the desired cm-level accuracy for VPS. This does not guarantee the desired system accuracy, because that depends on the vision processing algorithms, the state estimator, etc. It only means that errors due solely to the calibration are unlikely to make the desired accuracy unattainable.

The final optimized 3D target positions are shown in Table 4.5.

Table 4.5 Optimized, final 3D target coordinates

Ball #	X (mm)	Y (mm)	Z (mm)
1	1398.20	738.28	2090.70
2	8.11	9.58	2088.52
3	1790.59	571.78	1679.89
4	1517.14	-333.85	1678.18
5	849.19	-688.38	1669.80
6	-65.42	-405.73	1681.57
7	-204.53	1180.84	1687.28
8	507.80	1600.59	1653.46
9	1312.64	447.13	1077.74
10	654.61	-187.98	1371.50
11	30.08	470.35	794.38
12	677.38	1097.37	1284.06
13	1815.66	632.62	420.51
14	868.23	-690.53	424.99
15	-479.07	277.74	425.40
16	494.45	1604.96	422.44
17	1446.82	1318.94	438.98
18	1188.89	0.00	0.00
19	0.00	0.00	0.00
20	217.33	1151.23	0.00

A MATLAB™ program `compute3Dballcoords.m` computed the 3D target positions from the raw measurements. The program outputs the resulting 3D target positions to a file named `frm_3dpos_date.dat`, where, by convention, *date* is the day the raw frame measurements were taken.

To obtain the 2D image coordinates of the calibration targets, a MATLAB™ image-processing program named `set2Dballcoords.m` was written that allows a user to open a bitmap image of the calibration frame taken with the camera that is to be calibrated, such as the image shown in Figure 4.5. The program allows the user to zoom in on the targets, one at a time, and center a circular tool over



the target. This is illustrated in Figure 4.7. The user then clicks the mouse to record the image coordinates of the target. The user can skip targets that are too obstructed for their center to be accurately identified. The program writes the 2D image coordinates to a .dat file for later use. For an image file named by convention runXX-camX.bmp, the output file will be named runXX-camX\_2Dp.dat.

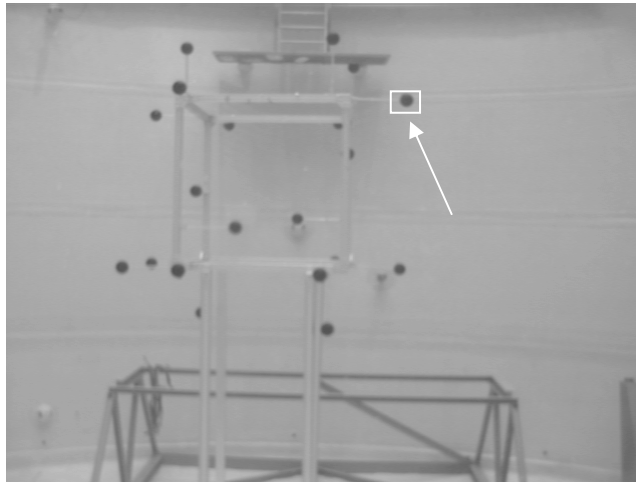


Figure 4.7 Image of calibration frame from camera 2

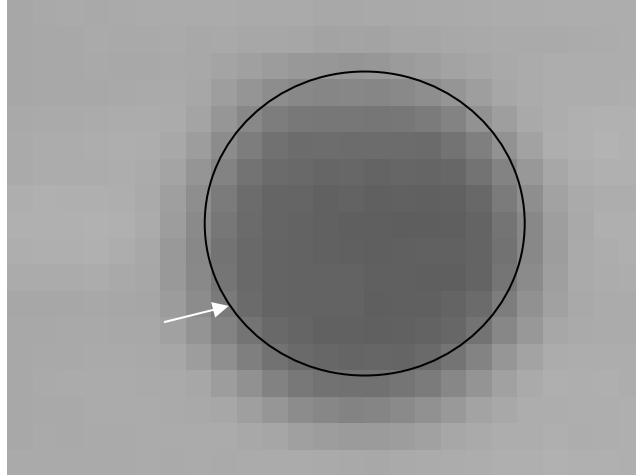


Figure 4.8 Zoomed calibration frame target

A program could be written that, through the use of vision algorithms, could automatically find the centers of each ball, thus eliminating the human error involved with doing so manually, and also saving the user the time required to carefully center the targeting tool on the balls. This option was not pursued for several reasons. A program capable of doing this task as well as a human would have had to have been reasonably complex in order to handle the many partially obstructed balls encountered in the images. It would likely have used edge detection to find the edge of the balls, then analyzed the edge to select part or parts on which to base the creation of an arc, and then found the center of that arc. This analysis would have had to have taken into account that one or more sections of the same ball might be visible, that balls could appear merged (one in front of the other), and that due to illumination differences between the tops and bottoms of the balls the tops would appear slightly flattened.

All of these, if unaccounted for, could have caused errors that were a large fraction of the ball radius. A human can account for these things very easily, and can reliably record the center of partially obstructed balls to within a tenth or twentieth of a ball radius if care is taken. In the end a human would need to be called upon to make the decision of whether or not to discard a ball from the list, and would be needed to number each ball in the images, so time savings over the current method would not have been spectacular. The large amount of time (likely as much as several weeks) to write such a program was seen as needless since the current 2D ball-finding technique was implemented in a few days, and need not be repeated unless a camera is disturbed.

Another MATLAB<sup>TM</sup> program, called `make_extcal_rd.m`, or “make extrinsic calibration raw data”, combines the output from the optimization program (the 3D target locations from `frm_3dpos_date.dat`) and the image-processing program (the 2D manually identified target image coordinates) into one file. This program automatically removes the 3D coordinates for targets that were skipped when finding the 2D image coordinates.

The program `cal_fuse.c`, which is a modified version of software found in [22], accepts as input the intrinsic calibration results (based on the checkerboard images), and the 2D and 3D calibration frame data for the extrinsic calibration. It then calculates, based on the calibration frame data alone, a full set of parameters. Because of the relatively small number of targets on the calibration

frame, and the long distances involved, this initial calibration is not highly accurate. It then discards the just-computed values of  $C_X$ ,  $C_Y$ ,  $f$ , and  $s_X$ , replaces them with those from the separate intrinsic calibration, and recomputes the extrinsic parameters, treating the new intrinsic parameters as fixed. By doing this, the minimization algorithm in `cal_fuse.c` (described below) needs only to optimize for the six extrinsic parameters instead of for all eleven, allowing better convergence. Also, since the intrinsic parameters calculated from the checkerboard images are based on more data points than those calculated initially by `cal_fuse.c`, they are of a higher accuracy. These can thus be used to calculate extrinsic parameters of higher accuracy than if only the calibration frame data was used.

The calibration algorithm implemented in `cal_fuse.c` is described in detail in [15]. First,  $C_X$  and  $C_Y$  are assumed to be their ideal values, 320 and 240, respectively, and distortion is ignored. With these assumptions, a linear equation can be derived that relates a global 3D coordinate,  $[X_G \ Y_G \ Z_G]^T$ , to the X and Y image coordinates,  $[X_{FD} \ Y_{FD}]$ . The parameters in the equation are  $s_X r_{11}$ ,  $s_X r_{12}$ ,  $s_X r_{13}$ ,  $r_{21}$ ,  $r_{22}$ ,  $r_{23}$ ,  $s_X T_X$ , and  $T_Y$ , where each  $r_{ij}$  is the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column of the rotation matrix  ${}^C_G R$ . With eight or more control data points ( $[X_G \ Y_G \ Z_G]^T$  and  $[X_{FD} \ Y_{FD}]$  sets), a system of these equations is solved to render estimates for these eight parameters. Since each equation is homogeneous, one of the parameters is set to a constant value to avoid the trivial solution. This means that the solved parameter values are linearly scaled versions of the true values.

The next step is to force orthonormality of the first two rows of  ${}^c_oR$  by scaling the row vectors to unit magnitude. The third row of the rotation matrix is then recovered by taking the cross product of the first two rows.

The next step is to calculate estimates for the third component of the translation vector,  $T_z$ , and the focal length,  $f$ . Since we have estimates for the full rotation matrix, a linear equation can be derived with  $f$  and  $T_z$  as the only two unknowns. Again, with two or more control point data sets, estimates for  $f$  and  $T_z$  can be solved.

Using the estimates computed or assumed above, the next step is a nonlinear optimization to refine the parameters. The error function to be minimized is:

$$J = \sum_{i=1}^n (X_i - X_i')^2 + \sum_{i=1}^n (Y_i - Y_i')^2 \quad (4.20)$$

where

$n$  = number of calibration targets used

$(X_i, Y_i)$  = observed image coordinates of target  $i$

$(X_i', Y_i')$  = image coordinates target  $i$ , predicted based on current  
parameter estimates

This minimization is achieved using the Levenberg-Marquardt minimization technique. In order for this method to be used, a non-redundant representation

of rotation is required. A rotation matrix has redundant information, since it uses nine members to represent 3 degrees of freedom. Therefore, the rotation matrix at this step is converted into Euler angles, a representation of relative orientation similar to X-Y-Z fixed angles introduced in Chapter 3.

#### 4.4.1 VPS Extrinsic Calibration: Results and Analysis

Table 4.6 below contains the results of the VPS extrinsic calibration. The rotation angles,  $R_x$ ,  $R_y$ , and  $R_z$ , are X-Y-Z fixed angles, with the camera frame as the fixed coordinate system. The translation components,  $T_x$ ,  $T_y$ , and  $T_z$ , are in camera frame coordinates. Table 4.7 contains the error metrics calculated for the VPS extrinsic calibration, along with a record of the number of calibration targets that were used for the calibration of each camera.

Table 4.6 VPS extrinsic calibration results

	$R_x$	$R_y$	$R_z$	$T_x$ (mm)	$T_y$ (mm)	$T_z$ (mm)
<b>Camera 1</b>	94.72°	-80.21°	175.40°	-614.66	-1208.47	6921.82
<b>Camera 2</b>	91.12°	7.65°	179.89°	283.17	-1245.12	6792.34
<b>Camera 3</b>	-83.61°	83.70°	5.86°	565.41	-1500.57	7413.26
<b>Camera 4</b>	-90.82°	-9.15°	0.08°	-489.02	-1229.02	7555.31
<b>Camera 5</b>	-54.55°	-61.39°	-34.30°	-258.17	-223.93	7391.42
<b>Camera 6</b>	66.50°	-22.27°	-169.92°	341.59	-831.95	6914.25
<b>Camera 7</b>	41.41°	56.24°	132.38°	588.11	-1468.69	7206.47
<b>Camera 8</b>	-68.66°	20.05°	6.65°	-222.95	-949.48	7689.74

Unlike the program used to compute the intrinsic parameters, the C implementation of Tsai's algorithm ([15], [22]) does not provide tolerances for individual calibration parameters. Instead, the accuracies of all of the parameter estimates for an individual camera are combined into three error metrics. The first, the Normalized Pixel Error (in pixels) is an amalgamation of different pixel-space errors that accumulated during the parameter estimation calculations, and is difficult to interpret physically. It serves as a way of getting a feel for how one camera's calibration compares to that of another.

More meaningful error metrics are the Average Object Space Error and the Maximum Object Space Error, both in mm. The first is the average error between the actual 3D global locations of the calibration targets, and the predicted 3D global locations of the targets. The predictions of the target locations are made based on the 2D image coordinates and the final calibration parameters. The Maximum Object Space Error is simply the largest discrepancy,

among all of the calibration targets, between actual and predicted 3D locations. It is important to note that these error metrics take into account the inaccuracies of both extrinsic and intrinsic calibration parameters.

Table 4.7 Error metrics for VPS extrinsic calibration

	<b>Normalized Pixel Error (pix)</b>	<b>Avg. Object Space Error (mm)</b>	<b>Maximum Object Space Error (mm)</b>	<b>No. Targets Visible</b>
<b>Camera 1</b>	0.60	2.12	4.78	17
<b>Camera 2</b>	0.64	1.91	5.97	18
<b>Camera 3</b>	1.18	3.03	8.59	18
<b>Camera 4</b>	0.76	2.32	4.49	18
<b>Camera 5</b>	0.73	3.10	6.61	19
<b>Camera 6</b>	1.07	3.92	11.21	20
<b>Camera 7</b>	0.76	2.31	7.43	20
<b>Camera 8</b>	0.70	2.29	6.40	20

The error metrics can be conservatively interpreted. For a given camera, and for a vision task being performed at a range similar to that of the calibration frame (in the middle of the tank), the error introduced by calibration inaccuracies alone should not exceed the maximum object space error listed for that specific camera. The error is also unlikely to be smaller than the average object space error in the table (2 – 4 mm).



## Chapter 5

### VPS Software Components

This chapter contains a detailed description of the software applications associated with VPS. Section 5.1 provides details about VPS\_client, the program responsible for acquiring images in real-time from the VPS cameras and processing them to create usable measurements. Section 5.2 describes the raptor program. This is the software that runs on the SCAMP SSV control station, and is also the program in which the optimal state estimator, the Extended Kalman Filter, is implemented. Section 5.3 contains a description of the software that runs on SCAMP SSV during its operation.

#### ***5.1 Image Acquisition and Processing Program: VPS\_client***

Figure 2.7 illustrated that VPS\_client executes on the four vision computers, each grabbing frames from two VPS cameras. Figure 5.1 is a more detailed graphical representation of how VPS\_client interacts with the other components of VPS. The algorithm executed by VPS\_client is shown in Figure 5.2.

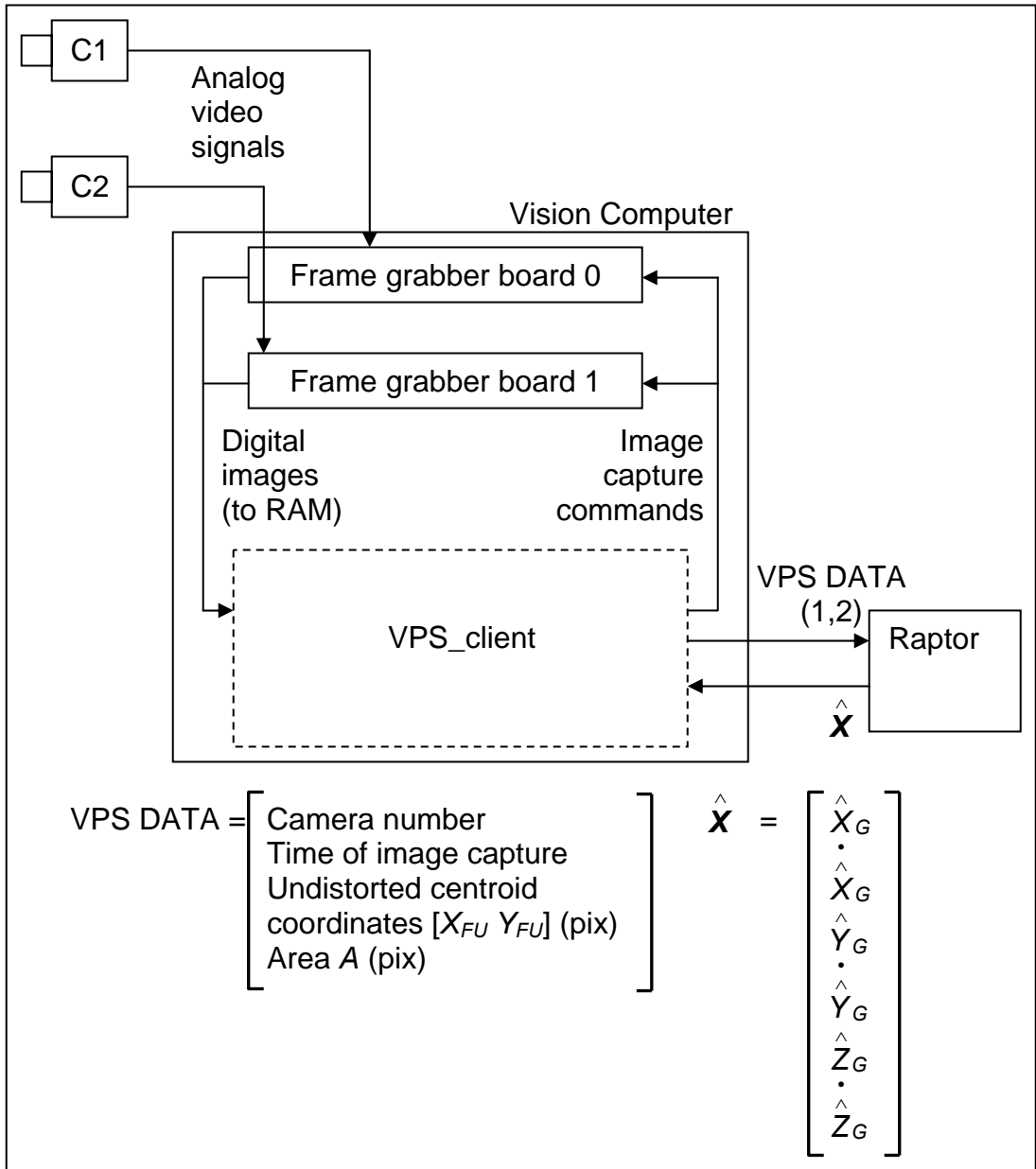


Figure 5.1 Information flow diagram for VPS\_client

1. Initialize camera numbers and frame grabber boards.
2. Load background image pixel arrays  $bg[i]$  (cameras  $i=1, 2$ )
3. Load calibration data.
4. Set camera number  $i=1$
5. If (SSV is in FOV for camera  $i$ ) then
  - a) Capture and process image  $pix[i]$  from camera  $i$
  - b) Send measurement data to raptor; read state estimate update from raptor
6. Else if (SSV is out of FOV for camera 1 and camera 2) then
  - a) Pause 1 second; request new state estimate from raptor
  - b) If raptor has not responded in 10 requests for state estimates
    - i) Capture, process image from camera  $i$
    - ii) Send measurement data to raptor; read state estimate update from raptor
7. Switch to other frame grabber/camera (If  $i=1$  then  $i=2$ ; if  $i=2$  then  $i=1$ )
8. Goto Step 5.

Figure 5.2 VPS\_client algorithm

### 5.1.1 VPS\_client Initialization

At run time, VPS\_client first looks for a file called camnums.dat. This file must reside on each vision computer, in the directory that houses VPS\_client. The camnums.dat file for Vision 1 contains the following data:

```
2
1
2
-1
```

The first integer notifies VPS\_client how many camera/frame grabber sets it will need to control – in this case, two. The second and third integers identify by number which cameras are connected to the boards. In this case, we see that board 0 has camera 1 attached, and board 1 has camera 2 attached. The final number identifies that the end of the list has been reached. The number of cameras present and an end-of-list identifier are redundant. They both exist

because of legacy code that allowed one 8-channel frame grabber to control all eight VPS cameras. After accessing `camnums.dat`, other global variables required for the operation of `VPS_client` are also initialized.

Next, the frame grabber boards are initialized. Memory is allocated where their captured images ( $pix[ij]$ , cameras  $i=1, 2$ ) can be stored, their specific parameters required for operation with VPS are set, and the powered-on status of the cameras is verified. The background images ( $bg[ij]$ , cameras  $i=1, 2$ ) and calibration data, based on the camera numbers found in `camnums.dat`, are then loaded into the appropriate locations in memory.

Next, `VPS_client` prompts the user for input indicating how it is to function. The user can choose from normal operation, or a series of special functions. The special functions allow the user to capture and store background images, pass live video through to the computer monitor, test network communications, and perform other system tests and trouble shooting operations. This step is not shown in Figure 5.2, as it does not occur under normal VPS operations, but only at the beginning of tests or during system development.

### 5.1.2 Image Acquisition and Processing

During normal operation, the image acquisition and processing functions and the raptor communications functions execute in a loop. After each iteration of the loop, a local copy of the global state estimate of the tracked object is checked. If

its position is in the field of view (FOV) of the camera, then the frame grabber is commanded to capture an image with that camera, and the time is recorded locally in VPS\_client. Then the image is processed, the measurement sent to raptor, and a new global state estimate is received. Note that raptor only sends a global state estimate after it has received information from VPS\_client. This process is repeated over both frame grabbers at approximately 8 Hz.

If the tracked object is out of the FOV of one camera that camera is skipped. A new iteration of the while loop is then begun, this time with the next camera. If the tracked object is not in the FOV of either of the two VPS\_client cameras, no images are grabbed, and a copy of the latest global state estimate is requested from raptor at 1-second intervals. VPS\_client makes this request by sending dummy data to raptor – data with a value of -1 in each of the camera data positions. The EKF knows to reject this data, but having received something, still returns the latest global estimate. Once the global estimate indicates that the tracked object has returned to the FOV of one or both of the cameras, normal execution of the loop will resume.

If the tracked object were outside the FOV of all of the VPS cameras, none of the cameras would be commanded to take images with which to update the global estimate. In this case, it is possible for the global state estimate to get “stuck”, especially with poor or absent EKF dynamic propagation, and to remain constant even if the tracked object returns to the FOV of one or more cameras. To

prevent this, if VPS\_client has had to request data from raptor 10 times consecutively, the vision functions (acquisition and processing) and the normal communications are executed. If the tracked object remains outside all camera FOVs, then the vision processing algorithms will not generate useful data, and will send, as before, a dummy set of data with pixel values of -1.

The image acquisition and processing functions are encapsulated in a function named DoVisionWork(). In this function, the first action is to acquire an image  $pix[i]$  from camera  $i$  using its frame grabber board. Figure 5.3 is an example of an image with SCAMP SSV in the FOV, as it appears immediately after acquisition by camera 4.

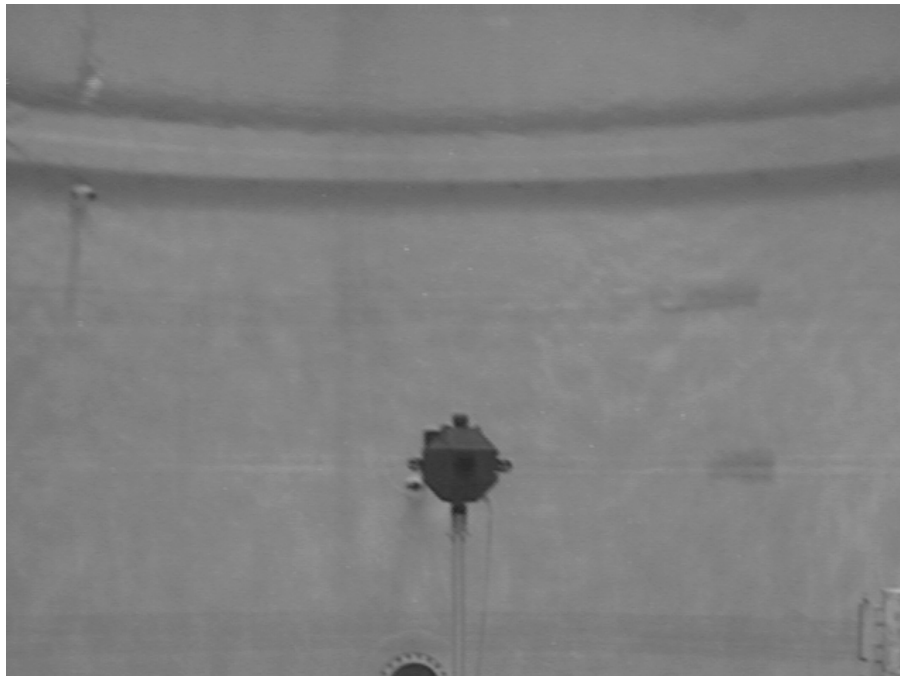


Figure 5.3 VPS image in its initial state

The image processing actions executed next are a standard series of operations used in many vision applications (see [23], and [13] for more details about its application to VPS).

Prior to operation of VPS, a set of background or static images  $bg[i]$  must be acquired. Once an image with the tracked object is acquired, the corresponding static background image is subtracted from the image. Recall that in 8-bit grayscale computer images, a pixel value of 0 corresponds to black, and a pixel value of 255 corresponds to white. After subtraction, corresponding pixels in the static and acquired images that are nearly the same will be very dark (nearly zero) in the processed image. If pixels differ greatly between the static and acquired images, which is the case in the region of the image where SCAMP SSV is found, pixels in the processed image have higher values. Put simply, the new image will be lighter in regions that differed from the background, and nearly black where the images were the same. Figure 5.4 is a sample background image for camera 4, and Figure 5.5 is the example image from above after the background has been subtracted. Note that new background images are acquired just before each test to account for lighting conditions and static object changes.



Figure 5.4 VPS background image

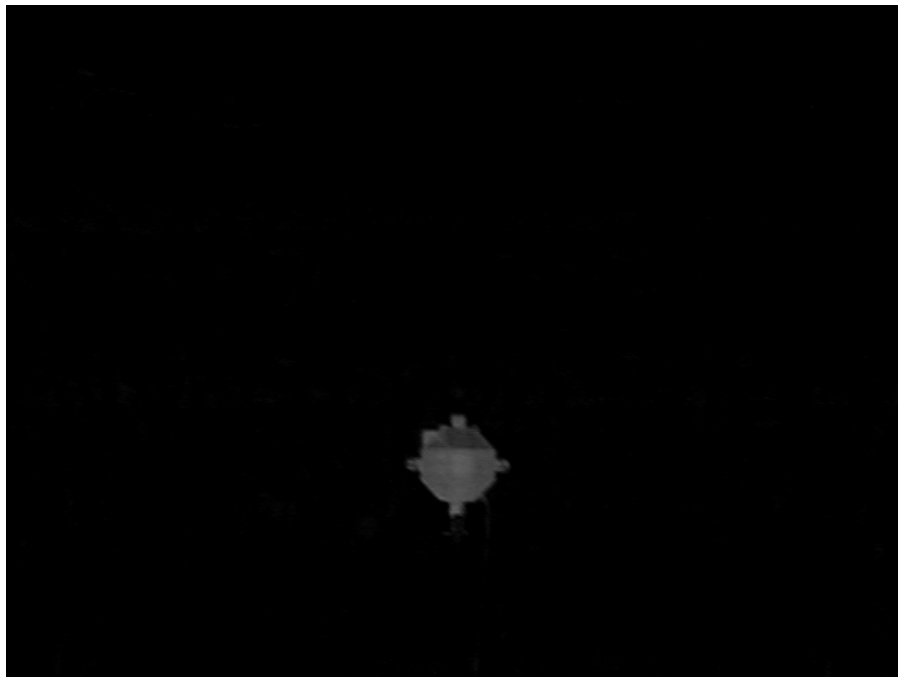


Figure 5.5 VPS image after background subtraction



Next, the image is thresholded to create a binary (all pixels are either 0 or 255) image. Any pixel that is under a certain threshold (darker than the cut-off) is set to 255 (white), and any pixel over the threshold (lighter than the cut-off) is set to 0 (black). The pixels that are set to black are considered part of the tracked object. Figure 5.6 shows the example image after thresholding.

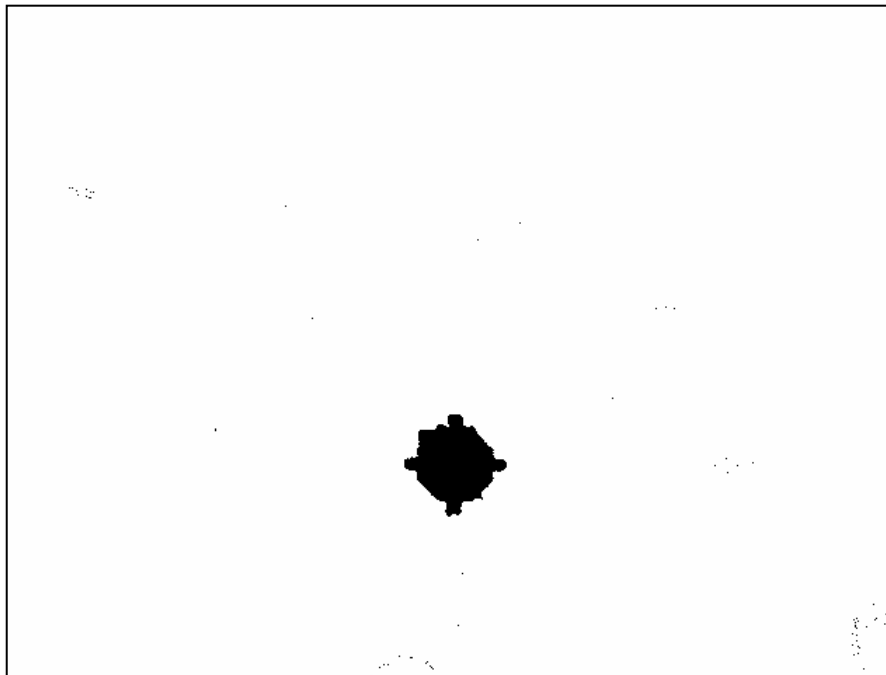


Figure 5.6 VPS image after thresholding

Ideally, the binary image created after thresholding would only contain one contiguous region of black pixels, which would correspond to SCAMP SSV. Experience shows that in addition to this region, there will be random black pixels throughout the image referred to as “salt-and-pepper” noise. These are removed by looking at each black pixel in the image, and setting it to white if in its

surrounding area there are very few other black pixels. This is accomplished by a function called `denoise()`, which, for each black pixel in a thresholded image, counts the integer number of black pixels,  $\rho$ , in its neighborhood. The neighborhood of a black pixel is defined as a square region, of dimension  $2\lambda$ , with the black pixel in the center. If  $\rho$  is less than some threshold value,  $\rho_{\min}$ , the black pixel is set to white, and left black if  $\rho$  is above the threshold. If  $\rho_{\min}$  is set too low, or  $\lambda$  too large, not all noise pixels will be eliminated. If the opposite is the case, some pixels on the edge of the tracked object could be erroneously set to white. Trial and error was used to find values of  $\rho_{\min}$  and  $\lambda$  that were appropriate for VPS.  $\rho_{\min} = 15$  and  $\lambda = 5$  were found to provide a good balance between eliminating all salt-and-pepper noise pixels, and few, if any, tracked-object pixels. Figure 5.7 shows the example image after the noise has been removed.

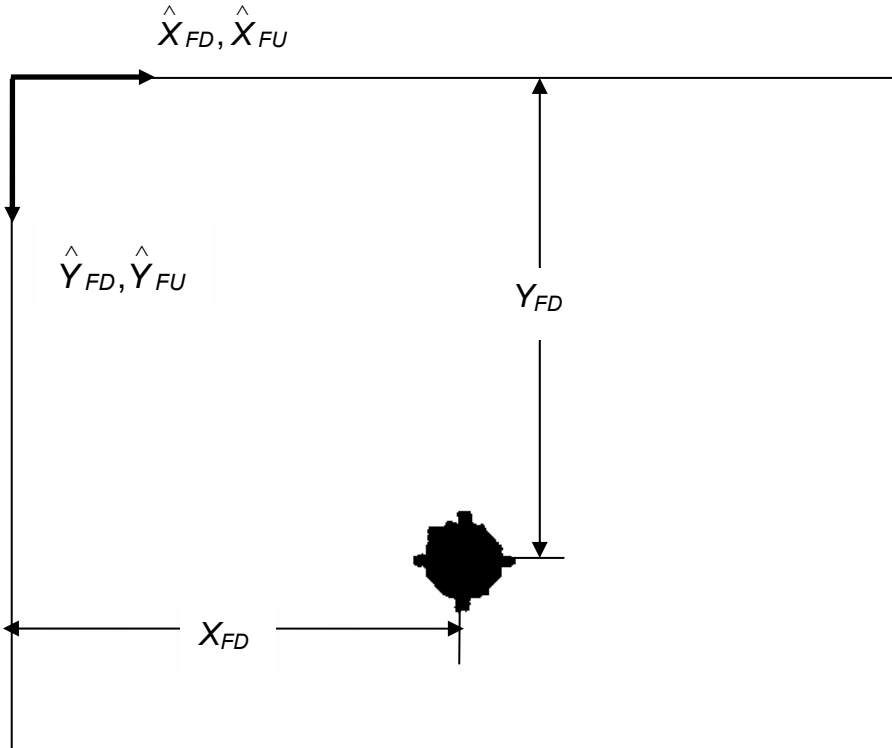


Figure 5.7 VPS image after noise removal

Once a clean binary image is available, the area of the robot, and its image centroid coordinates, all in pixels, are calculated. The area is simply the total number of black pixels in the image, while the X and Y centroid coordinates are calculated as the average pixel distance from the X and Y axes of all the black pixels. Recall that by convention, the origin of an image is at the top left-hand corner, with positive X pointing to the right and positive Y pointing down.

The centroid coordinates calculated are  $[X_{FD} Y_{FD}]$ , or the distorted digital coordinates. The EKF equations can be constructed to take into account the lens distortion, but this would require numerous additional math operations for

each measurement processed by the filter. It is more efficient to first undistort the centroid coordinates before using them in the EKF. To do this,  $[X_{FD} \ Y_{FD}]$  must first be converted into distorted real image coordinates,  $[X_D \ Y_D]$  by inverting equations (3.17) and (3.14) to form the equations

$$X_D = \frac{(X_{FD} - C_X) \cdot D_X'}{s_X} \quad (5.1)$$

$$Y_D = (X_{FD} - C_Y) \cdot D_Y \quad (5.2)$$

Now, using the distortion coefficient, the coordinates can be undistorted by applying inverted versions of equations (3.11) and (3.12)

$$X_U = X_D \cdot (1 + K \cdot (X_D^2 + Y_D^2)) \quad (5.3)$$

$$Y_U = Y_D \cdot (1 + K \cdot (X_D^2 + Y_D^2)) \quad (5.4)$$

The undistorted real image coordinates  $[X_U \ Y_U]$  can now be converted back into digital form, this time into undistorted digital image coordinates, by applying equations (3.17) and (3.14), or equation (5.1) and (5.2) in reversed form

$$X_{FU} = \frac{X_U \cdot s_X}{D_X'} + C_X \quad (5.5)$$

$$Y_{FU} = \frac{Y_U}{D_Y} + C_Y \quad (5.6)$$

After the undistorted digital centroid coordinates and image area of SCAMP SSV are ready, they are packed into a message called `vps_data` and sent to raptor. The message `vps_data` is a specially defined struct variable. Its form and contents are shown in Table 5.1.

Table 5.1 `vps_data` structure sent to raptor

Variable Type	Contents
integer	camera number
float[3]	[area, $X_{FU}$ , $Y_{FU}$ ]
struct timeval	long integer seconds, long integer microseconds

The camera number is used to ensure that each measurement is used correctly in updating the state estimate. The `timeval` struct contains the current number of seconds and microseconds that have elapsed since January 1, 1970, and is standard on both UNIX/LINUX and Windows platforms. It is important that the vision computers (running Windows 2000) and the raptor control station computer (running RedHat Linux) have synchronized clocks. Before each VPS test, the clock on both computers is synchronized to an external time source to eliminate, as much as possible, any time drift that has occurred between the computers since the last test. On the vision computers, this is done through the

w32time command. On the ReCS computer, this is done by restarting the NTP (Network Time Protocol) daemon via the command `/etc/init.d/ntpd restart`.

The message the VPS\_client receives from raptor is also a struct variable. Its form and contents are shown in Table 5.2. Only the position estimate is currently used in VPS\_client. SCAMP SSV moves slowly enough that its velocity does not need to be considered when determining whether or not to pull an image from a certain camera. In 0.125 seconds (the approximate time between the 8Hz image acquisitions for a given camera), if traveling at its estimated terminal velocity of 0.2 m/s, SCAMP SSV could be expected to translate a maximum of

$$0.125 \text{ s} \times 0.2 \text{ m/s} = 0.025 \text{ m} = 2.5 \text{ cm} \quad (5.7)$$

The “error” data member in global\_data is not currently used or assigned a meaningful value in either VPS\_client or raptor, and is implemented for potential future use. It could, for instance, be used to carry the value of the maximum or mean error covariance term (defined in Chapter 6) from the EKF.

Table 5.2 global\_data structure sent to VPS\_client

Variable Type	Contents
float[3]	position $\begin{bmatrix} \hat{X}_G & \hat{Y}_G & \hat{Z}_G \end{bmatrix}$
float[3]	velocity $\begin{bmatrix} \dot{\hat{X}}_G & \dot{\hat{Y}}_G & \dot{\hat{Z}}_G \end{bmatrix}$
float	error

## **5.2 Control Station Software (*Raptor*)**

Many contributors at the Space Systems Laboratory have developed the raptor control station software. It is important to note that the overview here will focus on those components of the software that pertain specifically to VPS. For more details related to the raptor code, see [24] and [25].

Figure 5.8 is a graphical representation of how raptor interacts with the other VPS computers and programs. It should be noted that thus far, the discussion of the VPS software has been developed referring to a non-specific “tracked object” instead of to the specific robot that was used in this research, SCAMP SSV. This made sense for the discussion of VPS\_client since it can be applied to any spheroid object (and in the future, potentially to any object). Since raptor is designed to control specifically (but not exclusively) SCAMP SSV, and the final section of this chapter is a discussion of the SCAMP SSV software itself, the “tracked object” terminology will be dropped. This should not be interpreted as meaning the VPS architecture can only be used with SCAMP SSV – given similar state estimation and communication strategies, VPS and raptor can be made to accommodate nearly any neutral buoyancy robot.

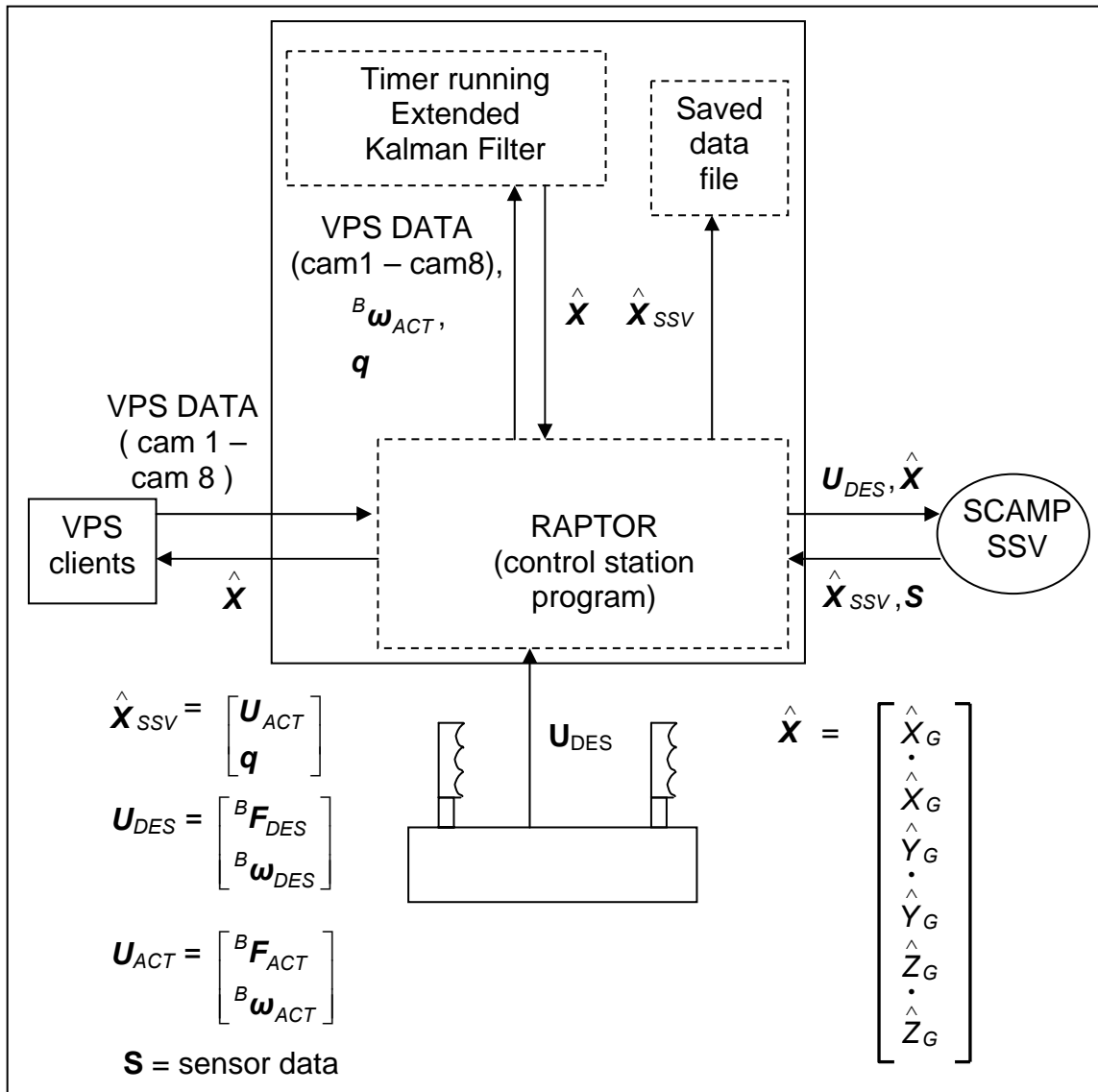


Figure 5.8 Information flow diagram for raptor

The four vision computers, each running VPS\_client, send vps\_data structures to raptor at unscheduled times, continuously and as fast as they can pull in and process images. Each time raptor receives a new measurement from a vision computer, it sends to that computer the latest inertial state estimate  $\hat{\mathbf{X}}$  for SCAMP SSV in the form of a global\_data structure. Raptor stores the vps\_data



messages in a global array. As new messages from individual cameras arrive, they overwrite the previous message from that camera in the array.

Similarly, raptor is also continuously communicating with SCAMP SSV. Its communication frequency is set at 50 Hz by SCAMP SSV's flight program. In closed loop attitude control mode, raptor sends to the robot a vector of desired inputs converted directly from pilot hand controller inputs:  ${}^B\mathbf{F}_{DES}$  are the desired body forces, and  ${}^B\boldsymbol{\omega}_{DES}$  are the desired body angular velocities. Raptor will also (in the near future) send SCAMP SSV the same inertial position and velocity estimate it sends to the vision computers,  $\hat{\mathbf{X}}$ . It receives from SCAMP SSV the robot's internal state estimate,  $\hat{\mathbf{X}}_{SSV}$ , which contains the robot's attitude quaternion  $\mathbf{q}$  and its applied inputs  $\mathbf{U}_{ACT}$ . SCAMP SSV sends other telemetry data to raptor, such as sensor readings, but these do not affect the operation of VPS. These data are represented by  $\mathbf{S}$  in Figure 5.8. Global variables within raptor store this information, and are updated every time a new message is received. If the pilot instructs raptor to do so via the GUI, it will save the data stored in these variables to a data file at approximately 100 Hz.

The EKF runs as a separate process within the raptor code. A timer is started as part of the raptor initialization to control the execution of the EKF. At a fixed frequency, this timer calls a single function, `do_VPS_filterwork()`. Each time `do_VPS_filterwork()` is called, it accesses the `vps_data` measurements stored in the global array, finds the measurements that are recent enough for use, and

uses them to update  $\hat{\mathbf{X}}$ . It also accesses data from SCAMP SSV's telemetry required in the EKF, but does not check that data's age. It assumes that because of SCAMP SSV's high communication frequency, the telemetry is sufficiently recent. The next time raptor communicates with the vision computers or with SCAMP SSV, it sends them the updated  $\hat{\mathbf{X}}$ . The EKF algorithm will be discussed further in Chapter 6.

### **5.3 SCAMP SSV Flight Software**

Although not modified for this research, the SCAMP SSV onboard software is overviewed for completeness. Figure 5.9 shows the three SCAMP SSV real-time threads that run under the VxWorks O/S.

The communications thread, running at 50Hz, receives  ${}^B\mathbf{U}_{DES}$  and  $\hat{\mathbf{X}}$  from raptor, and sends  $\hat{\mathbf{X}}_{SSV}$  to raptor. The state estimator thread, running at 25 Hz, receives raw sensor data from the magnetometer, accelerometer, and rate gyros (IMU), and (via an A/D board not shown in the diagram), computes from that data current values of the robot's state,  $\hat{\mathbf{X}}_{SSV}$ . The controller uses  ${}^B\mathbf{U}_{DES}$  and  $\hat{\mathbf{X}}_{SSV}$  to compute commands that are sent to the thrusters. In the future, the controller will be augmented to allow it to automatically control the position of SCAMP SSV as well as its orientation. For more details on the SCAMP SSV software and its evolution, see [6].

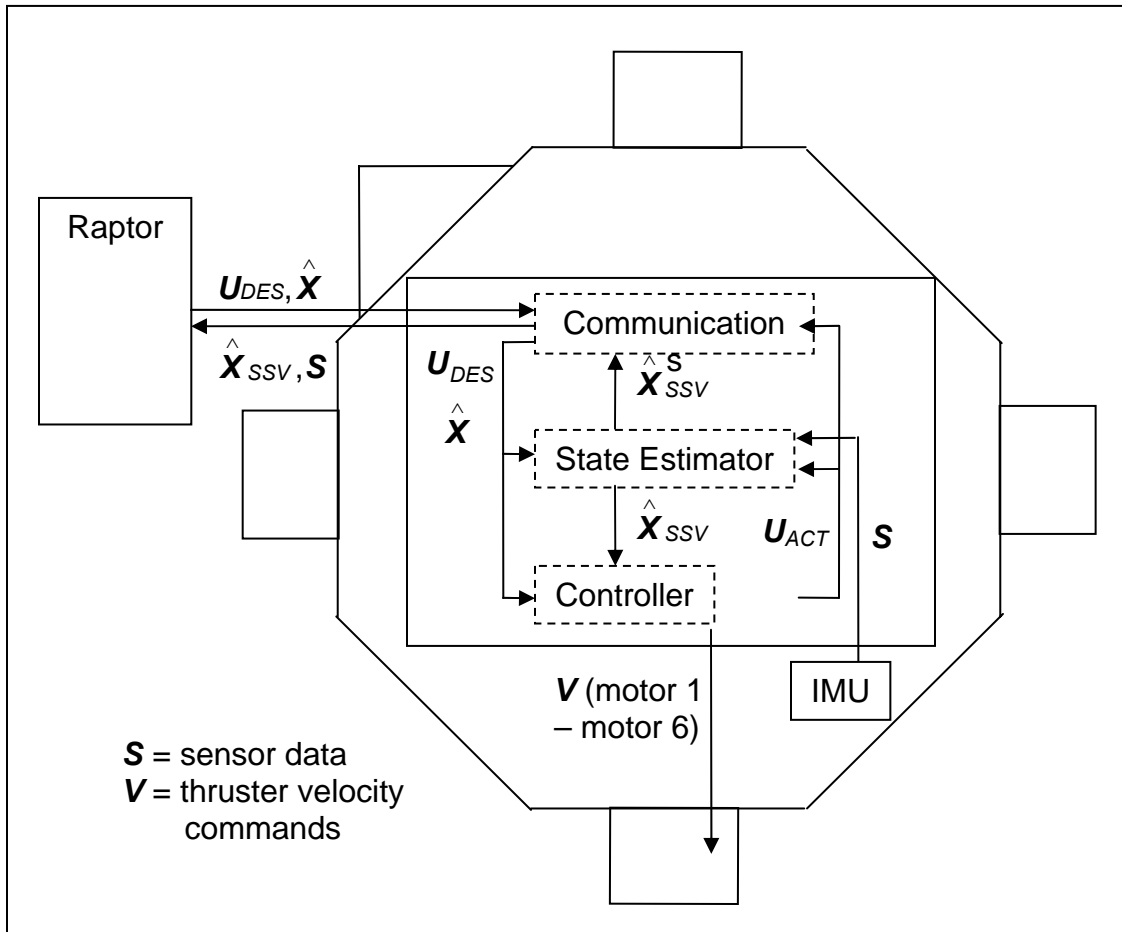


Figure 5.9 Information flow diagram for SCAMP SSV

## Chapter 6

### Optimal State Estimation and the Kalman Filter

In Section 6.1, this chapter begins with a discussion of the role of state estimation in closed-loop control strategies. Once this groundwork is laid, the idea of optimal state estimation is introduced, and the most popular linear optimal state estimation algorithm, the Kalman Filter, is discussed. The Extended Kalman Filter (EKF) is also introduced, which is a method of applying linear Kalman Filter techniques to nonlinear systems. In Section 6.2 the EKF equations, as applied to VPS, are presented, along with details on the characterization of system and measurement noise statistics.

#### 6.1 *Optimal Estimation Theory*

##### 6.1.1 Introduction to State Estimation

Consider a general set of differential equations that model a particular dynamic system, with  $\mathbf{x}$  as the state vector, and  $\mathbf{u}$  as the input control vector:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \quad (6.1)$$

The open loop block diagram for such a simple system would be as follows

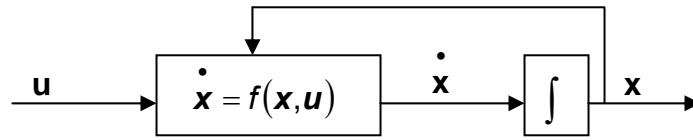


Figure 6.1: Simple open-loop block diagram

Suppose that the following control law was desired for this system:

$$u = k(\mathbf{e}) \tag{6.2}$$

where  $\mathbf{e}$  is an error vector, typically defined as the difference between the state vector and some exogenous reference vector characterizing desired behavior  $\mathbf{x}_D$ .

$$\mathbf{e} = \mathbf{x}_D - \mathbf{x} \tag{6.3}$$

Assuming the full, uncorrupted state vector  $\mathbf{x}$  was available for use, a feedback loop could be constructed as in the following block diagram:

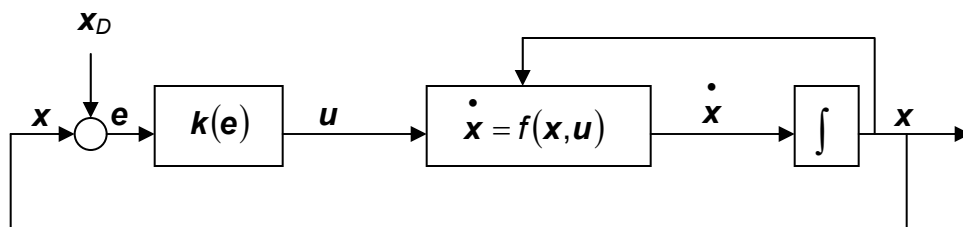


Figure 6.2 Simple closed-loop block diagram

In many real systems, knowledge of the full state vector  $\mathbf{x}$  is not available. Instead, sensors provide a vector of measurements,  $\mathbf{z}$ . Some elements of  $\mathbf{z}$  may indeed be measurements of individual state elements, while others are likely values that represent combinations of several state elements. The equations that model the relationships between the elements of  $\mathbf{x}$  and  $\mathbf{z}$  are called the sensor model, and denoted as

$$\mathbf{z} = \mathbf{c}(\mathbf{x}) \quad (6.4)$$

As can be seen in figure 6.3, an obstacle to the construction of a closed loop control system has now emerged. Because the control law requires  $\mathbf{x}$ , and only  $\mathbf{z}$  is available, there is a gap in the flow of information.

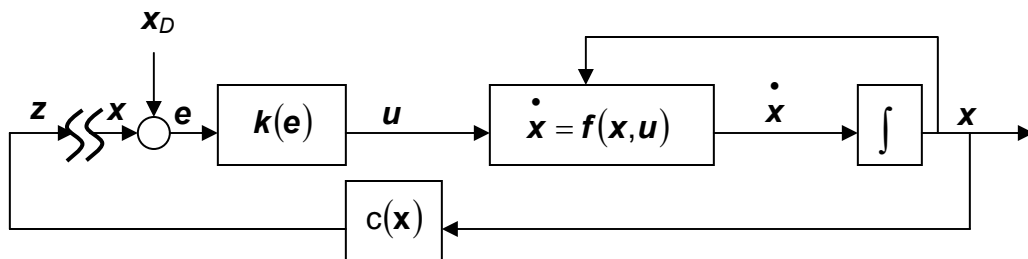


Figure 6.3 Discontinuous closed-loop block diagram

A new component, an observer  $\mathbf{g}(\mathbf{x})$ , is required in the system to bridge this gap. An observer is a computational algorithm that creates an estimate of the state vector,  $\hat{\mathbf{x}}$ . Generally, observers use the equations that comprise the system

dynamics  $f(\mathbf{x}, \mathbf{u})$  and the sensor model  $c(\mathbf{x})$ , along with the vectors  $\mathbf{z}$  and  $\mathbf{u}$  to create  $\hat{\mathbf{x}}$ . When an observer is employed, the error vector is formed using  $\hat{\mathbf{x}}$  instead of  $\mathbf{x}$ . This is illustrated in Figure 6.4. Note that the line coming from the observer, carrying the estimate  $\hat{\mathbf{x}}$  is dashed. This is to indicate that the observer does not construct  $\hat{\mathbf{x}}$  algebraically, but does so by other means (discussed later).

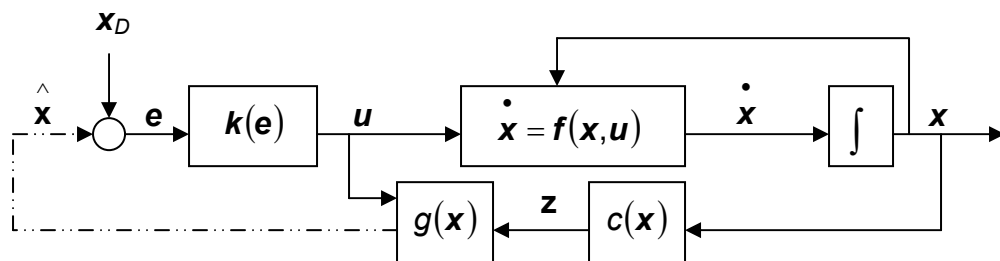


Figure 6.4 Closed-loop block diagram with observer

The discussion has thus far been for the general case where the system dynamics  $f(\mathbf{x}, \mathbf{u})$  are potentially nonlinear. The state estimation algorithms applied in VPS are based on extended techniques that start by linearizing the nonlinear dynamics to simplify the analysis. Therefore, the discussion will now leave behind the general nonlinear formulation of  $f(\mathbf{x}, \mathbf{u})$ , in favor of a linear formulation.

Assume that the dynamics of the example system, as well as the equations of the sensor model, are linear, and can be represented by the linear systems

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{u} \quad (6.5)$$

$$\mathbf{z} = \mathbf{C} \cdot \mathbf{x} \quad (6.6)$$

where  $A$ ,  $B$ , and  $C$  are constant matrices. This type of system is referred to as being Linear-Time-Invariant (LTI).

The Luenberger Observer [26] is the most common form of observer for linear systems that can be represented by (6.5) and (6.6). The Luenberger Observer combines the system dynamic equations and the sensor model equation to create a new system of differential equations that govern the evolution of the state estimate in a two-step process. First, a prediction step,  $\mathbf{A} \cdot \hat{\mathbf{x}} + \mathbf{B} \cdot \mathbf{u}$ , propagates what the derivative of  $\hat{\mathbf{x}}$  would be if the state vector  $\mathbf{x}$  matched exactly with the state estimate. In the second step, the correction or update step, the derivative of  $\hat{\mathbf{x}}$  is updated based on the discrepancy between the actual sensor measurement vector,  $\mathbf{z}$ , and the predicted measurement vector,  $\hat{\mathbf{z}}$ . In the continuous-time system, the two-step nature is not immediately apparent, since both are applied in the same equation. The two separate steps are more apparent in the discrete time formulation, which is discussed later in this chapter. The predicted measurement vector can be thought of as what the sensors would read if the state vector was exactly  $\hat{\mathbf{x}}$ . The matrix  $L$  is the estimator gain matrix,



and affects the relationship between  $\hat{\mathbf{x}}$  and the measurement residual,  $\mathbf{z} - \hat{\mathbf{z}}$ .

The state estimate  $\hat{\mathbf{x}}$  is obtained through integration of its differential equation.

Equations (6.7) and (6.8) below, along with (6.5) and (6.6), are the equations for a Luenberger Observer. Figure 6.5 shows a block diagram of the observer/controller system.

$$\hat{\mathbf{z}} = \mathbf{C} \cdot \hat{\mathbf{x}} \quad (6.7)$$

$$\dot{\hat{\mathbf{x}}} = \mathbf{A} \cdot \hat{\mathbf{x}} + \mathbf{B} \cdot \mathbf{u} + \mathbf{L} \cdot [\mathbf{z} - \hat{\mathbf{z}}] \quad (6.8)$$

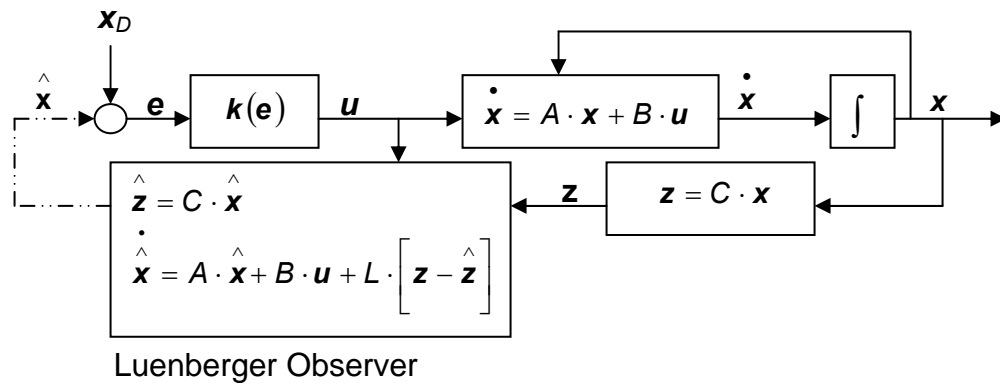


Figure 6.5 Luenberger Observer in a closed-loop system

At this point it must be pointed out that for the observer above to successfully calculate the state estimate – that is, to calculate a state estimate that converges asymptotically to the true state – certain conditions must be met. Specifically, the

matrices  $A$  and  $C$  must together be observable. This means that with the given  $A$  and  $C$  it must be possible to design  $L$  such that the matrix  $A-LC$  will be exponentially stable, or equivalently, all of the eigenvalues of the matrix  $A-LC$  will have negative real parts. For a more detailed discussion of observability, see [27].

The discussion so far has relied on two unstated assumptions. The first, that the equations in (6.1) and (6.5), for the general and the linear formulations respectively, model the system dynamics perfectly, and that there are no inputs to the systems other than those applied via the control vector  $\mathbf{u}$ . This is rarely a safe assumption. Because it is impossible to model physical systems exactly, there is some inherent uncertainty in the system model equations. Physical systems are also typically subject to process noise – unexpected, unmodeled, and often random inputs due to uncontrollable aspects of the environment in which the system exists. Wind gusts acting on an airplane are a good example of process noise. Sensor models too have limited accuracy. Typically, the sensor outputs are also corrupted by measurement noise – random or even systematic errors that alter the measurements from what they would be if the sensors were perfect.

The equations that model the dynamic system need to be expanded to take these departures from the ideal. Continuing with the linear formulation, let the vector  $\mathbf{w}$  represent both process noise, and any inaccuracies in the system

dynamic equations, and let the vector  $\mathbf{v}$  represent measurement noise and inaccuracies in the sensor model. The matrix  $G$  is simply a noise vector-scaling matrix, and is often assumed to be the unit matrix. The new system equations are

$$\dot{\mathbf{x}} = A \cdot \mathbf{x} + B \cdot \mathbf{u} + G \cdot \mathbf{w} \quad (6.9)$$

$$\mathbf{z} = C \cdot \mathbf{x} + \mathbf{v} \quad (6.10)$$

The optimal state estimator can now be introduced. Reference [27] gives the following definition:

An optimal estimator is a computational algorithm that processes measurements to deduce a minimum error estimate of the state of a system by utilizing: knowledge of system and measurement dynamics, assumed statistics of system noises and measurement errors, and initial condition information.

Unlike an observer, an estimator takes into account, through the way in which the gain matrix  $L$  is calculated, the measurement and process noise when calculating the state estimate. What makes a state estimator optimal, and the most popular algorithm for creating an optimal estimator (the Kalman Filter), are the topics of Section 6.1.2.

## 6.1.2 The Kalman Filter

A Kalman Filter is an optimal state estimator of the same general form as the Luenberger Observer. It is for use with systems that have strictly linear system dynamics and a linear sensor model. Although it is a major difference, it differs from the Luenberger Observer only in the way in which the gain matrix  $L$  is computed, and the resulting special properties this provides to the state estimate. Figure 6.6 shows the block diagram of the more realistic system model, with sensor and process noise shown.

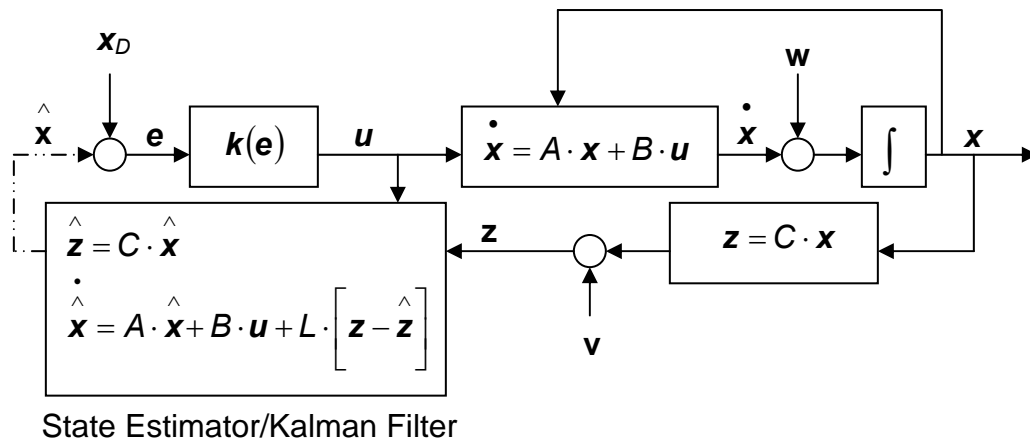


Figure 6.6 State estimator in a realistic closed-loop diagram

Before the Kalman Filter can be described, some statistical details need to be introduced. Let the vector  $\tilde{\mathbf{x}}$  be defined as the error between the state estimate,  $\hat{\mathbf{x}}$ , and the true state vector,  $\mathbf{x}$

$$\tilde{\mathbf{x}} = \hat{\mathbf{x}} - \mathbf{x} \quad (6.11)$$

Each element in the vector  $\tilde{\mathbf{x}}$  is a random variable – each can take on an infinite variety of real values. Since it is hoped that the error between the state and the estimate is small, the elements in  $\tilde{\mathbf{x}}$  hopefully will never stray far from zero. Let the symbol  $E$  represent the “expected value” of a random variable or vector of random variables. The expected value is the value that the random variable is most statistically likely to take over time. Let the matrix  $P$  be defined as the covariance matrix of  $\tilde{\mathbf{x}}$ :

$$P = E \left[ \tilde{\mathbf{x}} \cdot \tilde{\mathbf{x}}^T \right] \quad (6.12)$$

$P$  is also known as the error covariance matrix of the estimator with which it is associated. The  $i^{th}$  major diagonal element of  $P$  will have values equal to the square of the standard deviation,  $\sigma$ , of the  $i^{th}$  element of  $\tilde{\mathbf{x}}$

$$P_{i,diag} = \left( \sigma_{x_i}(t) \right)^2 \quad (6.13)$$

If all of the elements of  $\tilde{\mathbf{x}}$  for some state estimator have a normal (or Gaussian) distribution about a mean of zero, as in the equation

$$E\left[\tilde{\mathbf{x}}\right] = \mathbf{0} \quad (6.14)$$

then the estimator is referred to as unbiased. In such a case, the error covariance matrix  $P$  is diagonal.

A Kalman Filter is an unbiased, minimum variance observer for linear systems. It is a minimum variance observer because  $L$ , the Kalman Gain Matrix, is calculated such that it minimizes the following objective function

$$J(L) = \lim_{t \rightarrow \infty} \sum_i \left( \sigma_{x_i}(t) \right)^2 = \lim_{t \rightarrow \infty} \text{tr}[P] \quad (6.15)$$

The trace of the error covariance matrix  $P$  is the sum of its diagonal elements, and acts as a measure of the expected magnitude of  $\tilde{\mathbf{x}}$ . In other words, the trace of  $P$  is a measure of how close the state estimate is to the actual state vector.

Necessary to the development of the Kalman Filter equations are the assumptions that plant dynamics and measurement model are linear. It is also assumed the plant and measurement noise vectors,  $\mathbf{w}$  and  $\mathbf{v}$ , respectively, are Gaussian and zero-mean. Also required are the covariance matrices for  $\mathbf{w}$  and  $\mathbf{v}$ , referred to as  $Q$  and  $R$

$$Q = E[\mathbf{w} \cdot \mathbf{w}^T] \quad (6.16)$$

$$R = E[\mathbf{v} \cdot \mathbf{v}^T] \quad (6.17)$$

Given without proof, the equations for the Kalman Gain Matrix and the propagation of the error covariance matrix are given below. These must be used in conjunction with (6.7) and (6.8). It is assumed that  $A$ ,  $B$ ,  $C$ ,  $Q$  and  $R$  are constant matrices. The matrices  $P$  and  $L$  vary with time. For a full development of the Kalman Filter equations, refer to [28].

$$\dot{P}(t) = A \cdot P(t) + P(t) \cdot A^T + G \cdot Q \cdot G^T - L(t) \cdot R \cdot L(t)^T \quad (6.18)$$

$$L(t) = P(t) \cdot C^T \cdot R^{-1} \quad (6.19)$$

Since VPS is implemented in a digital computing environment, discrete time formulations of the Kalman Filter equations are required. The value of a variable at time  $k$  will be represented by the addition of a subscript  $k$ , as in  $\mathbf{x}_k$ . To represent the value of a variable exactly at one time step before or after time  $k$ , a subscript of  $k-1$  or  $k+1$  will be affixed, respectively. To denote the value of a variable immediately before or immediately after time  $k$  (as opposed to an entire time step before time  $k$ ), the subscript  $k(-)$  or  $k(+)$  will be affixed.

The discrete time equivalents of (6.9) and (6.10) are

$$\mathbf{x}_k = \Phi_{k-1} \cdot \mathbf{x}_{k-1} + \Lambda_{k-1} \cdot \mathbf{u}_{k-1} + \mathbf{w}_{k-1} \quad (6.20)$$

$$\mathbf{z}_k = \mathbf{C} \cdot \mathbf{x}_k + \mathbf{v}_k \quad (6.21)$$

In (6.20), the matrix  $\Phi_{k-1}$  is the state transition matrix of the dynamic system. It allows the calculation of the state  $\mathbf{x}$  at some time  $k$  in the future, given the value of the state at a previous time  $k-1$ . If the time step  $T_s$  between time  $k$  and time  $k-1$  is constant, and the system dynamics matrix  $A$  is also constant,  $\Phi_{k-1}$  is a constant matrix, and is given by the equation

$$\Phi_{k-1} = \mathbf{e}^{A \cdot T_s} \quad (6.22)$$

Given constant values for  $T_s$  and  $B$ , the matrix  $\Lambda_{k-1}$  is also constant. This matrix maps the change in  $\mathbf{x}$  caused by an input  $\mathbf{u}$  applied over a time step  $T_s$ . It is given by the equation

$$\Lambda_{k-1} = \left( \int_0^{T_s} \mathbf{e}^{A \cdot \tau} d\tau \right) \cdot B \quad (6.23)$$

The goal of the Kalman Filter is to provide a value for  $\hat{\mathbf{x}}_k$  given the previous estimate,  $\hat{\mathbf{x}}_{k-1(+)}$ , the control inputs  $\mathbf{u}$ , and measurements  $\mathbf{z}$ , that arrived during



the last time step. First, the estimate is propagated forward based on the system dynamics and the input vector  $\mathbf{u}$  by the equation

$$\hat{\mathbf{x}}_{k(-)} = \Phi_{k-1} \cdot \hat{\mathbf{x}}_{k-1(+)} + \Lambda_{k-1} \cdot \mathbf{u}_{k-1} \quad (6.24)$$

In addition to the state estimate, the error covariance matrix  $P$  must also be propagated forward with the equation

$$P_{k(-)} = \Phi_{k-1} \cdot P_{k-1(+)} \cdot \Phi_{k-1}^T + Q_{k-1} \quad (6.25)$$

where  $Q_{k-1}$  is the covariance matrix of the process noise vector at time  $k-1$ .

If the noise sources creating  $\mathbf{w}$  are assumed to be uncoupled,  $Q_{k-1}$  will be a diagonal matrix, with the diagonal elements being given by

$$Q_{k-1,i} = \sigma_{\mathbf{w},i}^2 \quad (6.26)$$

where  $\sigma_i$  is the standard deviation of the  $i^{\text{th}}$  element of  $\mathbf{w}$ .

Next, the estimate is corrected based on the measurement data with the equation

$$\hat{\mathbf{x}}_{k(+)} = \hat{\mathbf{x}}_{k(-)} + L_k \cdot \left[ \mathbf{z}_k - \mathbf{C} \cdot \hat{\mathbf{x}}_{k(-)} \right] \quad (6.27)$$

The Kalman Gain Matrix,  $L_k$ , required in (6.27), is calculated by the equation

$$L_k = P_{k(-)} \cdot \mathbf{C}^T \cdot \left[ \mathbf{C} \cdot P_{k(-)} \mathbf{C}^T + R_k \right]^{-1} \quad (6.28)$$

where  $R_k$  is the measurement noise covariance matrix

Similarly to  $Q_{k-1}$ , if the noise sources contributing to  $\mathbf{v}$  are assumed to be uncoupled,  $R_k$  will also be a diagonal matrix, with the diagonal elements being given by

$$R_{k,i} = \sigma_{v,i}^2 \quad (6.29)$$

where  $\sigma_i$  is the standard deviation of the  $i^{th}$  element of  $\mathbf{v}$ .

In (6.27),  $\mathbf{C} \cdot \hat{\mathbf{x}}_{k(-)}$  is  $\hat{\mathbf{z}}$ . If more than one measurement is available for the correction step, (6.27) can be applied as many times as necessary. In this case, each measurement would correct the already-corrected estimate, as opposed to each measurement being applied individually to the estimate as it stood after the prediction step.

The matrix  $P$  also needs to be corrected for each measurement with the equation

$$P_{k(+)} = [I - L_K \cdot C] \cdot P_{k(-)} \quad (6.30)$$

where  $I$  is the identity matrix of appropriate dimension.

### 6.1.3 The Extended Kalman Filter

As stated previously, one of the conditions necessary for the Kalman Filter equations to be valid is that both the system dynamics and the sensor model can be represented in the linear forms shown in (6.9) and (6.10). Many systems, however, cannot be represented in this way, and are of the more general form in (6.1) and (6.4).

If either  $f(\mathbf{x})$  or  $c(\mathbf{x})$  represent nonlinear vector functions, an Extended Kalman Filter must be used, which uses the linearization of the plant dynamics and sensor model about the current state estimate. In the case of VPS, the physically nonlinear dynamics of SCAMP SSV can, with accuracy sufficient for this application, be modeled as being linear. This means that the propagation equations for both the state estimate and the error covariance matrix will remain as given above. The sensor model,  $c(\mathbf{x})$ , however, is nonlinear. The state estimate correction equation changes from (6.27) to

$$\hat{\mathbf{x}}_{k(+)} = \hat{\mathbf{x}}_{k(-)} + \mathbf{L}_k \cdot \left[ \mathbf{z}_k - \mathbf{c}_k \left( \hat{\mathbf{x}}_{k(-)} \right) \right] \quad (6.31)$$

where  $\mathbf{c}_k \left( \hat{\mathbf{x}}_{k(-)} \right)$  is the value of  $\mathbf{c}(\mathbf{x})$  evaluated at  $\hat{\mathbf{x}}_{k(-)}$ . It is also equivalent to the estimated measurement  $\hat{\mathbf{z}}$ .

The equation for the Kalman Gain Matrix and the error covariance correction also change

$$\mathbf{L}_k = \mathbf{P}_{k(-)} \cdot \mathbf{C}_k \left( \hat{\mathbf{x}}_{k(-)} \right)^T \cdot \left[ \mathbf{C}_k \left( \hat{\mathbf{x}}_{k(-)} \right) \cdot \mathbf{P}_{k(-)} \cdot \mathbf{C}_k \left( \hat{\mathbf{x}}_{k(-)} \right)^T + \mathbf{R}_k \right]^{-1} \quad (6.32)$$

$$\mathbf{P}_{k(+)} = \left[ \mathbf{I} - \mathbf{L}_k \cdot \mathbf{C}_k \left( \hat{\mathbf{x}}_{k(-)} \right) \right] \cdot \mathbf{P}_{k(-)} \quad (6.33)$$

where

$$\mathbf{C}_k \left( \hat{\mathbf{x}}_{k(-)} \right) = \left. \frac{\partial \mathbf{c}_k(\mathbf{x}(t_k))}{\partial \mathbf{x}(t_k)} \right|_{\mathbf{x}(t_k) = \hat{\mathbf{x}}_{k(-)}} \quad (6.34)$$

## 6.2 Implementation of VPS Extended Kalman Filter

The state estimation task posed to VPS is to combine the measurements from the position sensors, the cameras, with the body forces exerted on the robot by the thrusters, to create an estimate of the robot's translational state, or its inertial position and velocity. The plant dynamics are fully linear, and thus amenable to

a standard Kalman Filter, but because the sensor model is nonlinear, an Extended Kalman Filter is required.

The translational state vector,  $\mathbf{x}$ , for the system was assigned as follows

$$\mathbf{x} \equiv \begin{bmatrix} X_G & \dot{X}_G & Y_G & \dot{Y}_G & Z_G & \dot{Z}_G \end{bmatrix}^T$$

Assigning  $\mathbf{x}$  in this way makes the  $A$  matrix (shown below) block diagonal, and allows the EKF (in the future) to be implemented as three separate, identical two-dimensional state estimators. The EKF is currently implemented as one six-dimensional estimator, with many of the elements of the  $A$ ,  $B$  and  $C$  matrices having a value of zero. Splitting up the EKF into three smaller EKFs would take advantage of the sparse nature of these matrices, and reduce the number of “multiply-by-zero” operations. This would make the EKF more efficient, but slightly more complex to implement. Since the speed of the EKF did not appear to be limiting its performance, the original six-dimensional architecture was retained in this development effort for the sake of simplicity.

Since SCAMP SSV moves very slowly through the water it was assumed, it was assumed that drag increases linearly with velocity instead of quadratically – at low speeds the difference will be small. This is the assumption that allows the dynamics to be modeled as being linear. The terminal velocity was estimated by measuring the time for SCAMP SSV to traverse the diameter of the NBRF in a

straight line at maximum thrust. In this test, SCAMP SSV traversed approximately 48' in 86 seconds, indicating a terminal velocity of approximately 0.2 m/s. Thus

$$F_{DRAG,i} = -C_{DT} \cdot v_i \quad (6.35)$$

where  $F_X$ ,  $F_Y$ , and  $F_Z$  are along the axes of the global reference frame. Thus, the equation of motion for SCAMP SSV in the  $X_G$  axis (the equations of motion in the  $Y_G$  and  $Z_G$  axes will be identical in form and are omitted) will be

$$\ddot{X}_G = \frac{F_X}{m} - \frac{C_{DT}}{m} \cdot \dot{X}_G \quad (6.36)$$

Necessary to the calculation of inertial control forces, and thus the propagation of the state estimate, are the instantaneous inertial attitude estimates of the robot. SCAMP SSV computes its attitude on board, using data from the magnetometer and accelerometer, both of which are tri-axial instruments. The magnetometer outputs the direction of the magnetic-North vector, while the accelerometer outputs the direction of the down vector, both in SCAMP SSV's body coordinate frame. By taking the cross product of the down and magnetic-North vectors, SCAMP SSV has access to three perpendicular inertial reference vectors (an inertial attitude reference frame) expressed in its own body coordinate frame. SCAMP SSV uses the reference frame to calculate its own relative attitude quaternion.

SCAMP SSV sends the attitude quaternion, as well as the commanded body forces, to raptor as telemetry. At each iteration, the EKF converts the current SCAMP SSV quaternion into a rotation matrix, as outlined in [10]. This matrix is the rotation matrix for transforming vectors from the body frame into the inertial (global) frame,  ${}^G_R$ . The EKF uses it to rotate the body forces into the inertial frame via the following equation

$$\mathbf{F}^G = {}^G_R \cdot \mathbf{F}^B \quad (6.37)$$

Given the forms of  $\mathbf{x}$  and  $\mathbf{u}$ , and the drag coefficient  $C_{DT}$ , the system dynamics are modeled as

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{u} + \mathbf{G} \cdot \mathbf{w} \quad (6.38)$$

where

$$\mathbf{A} \equiv \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{-C_{DT}}{m} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{-C_{DT}}{m} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & \frac{-C_{DT}}{m} \end{bmatrix} \quad \mathbf{B} \equiv \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{m} \end{bmatrix}$$

$G \equiv$  six-dimensional identity matrix

and  $\mathbf{w}$  is a zero-mean Gaussian process noise vector. The symbol  $m$  is the robot mass in air.

Applying (6.22) and (6.23) renders the following values the constant matrices

$\Phi_{k-1}$  and  $\Lambda_{k-1}$

$$\Phi_{k-1} \equiv \begin{bmatrix} 1 & \mathbf{a} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{b} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \mathbf{a} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{b} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \mathbf{a} \\ 0 & 0 & 0 & 0 & 0 & \mathbf{b} \end{bmatrix} \quad \Lambda_{k-1} = \begin{bmatrix} \mathbf{c} & 0 & 0 \\ \mathbf{d} & 0 & 0 \\ 0 & \mathbf{c} & 0 \\ 0 & \mathbf{d} & 0 \\ 0 & 0 & \mathbf{c} \\ 0 & 0 & \mathbf{d} \end{bmatrix}$$

where

$$\mathbf{a} = \frac{m}{C_{DT}} \cdot \left( 1 - e^{\frac{-C_{DT} \cdot T_s}{m}} \right)$$

$$\mathbf{b} = e^{\frac{-C_{DT} \cdot T_s}{m}}$$

$$\mathbf{c} = \frac{m \cdot \left( e^{\frac{-C_{DT} \cdot T_s}{m}} - 1 \right) + C_{DT} \cdot T_s}{C_{DT}^2}$$



$$d = \frac{1}{C_{DT}} \cdot \left( 1 - e^{\frac{-C_{DT} \cdot T_S}{m}} \right)$$

Originally, the VPS sensor model was also to be fully linear. From each captured image of SCAMP SSV, the vision computers would have computed a full 3D, global position estimate,  $[X_G \ Y_G \ Z_G]^T$ , according to the equations in Chapter 3. The sensor dynamics would have been

$$\mathbf{Z} = \mathbf{C} \cdot \mathbf{x} + \mathbf{v} \quad (6.39)$$

where

$$\mathbf{z} \equiv [X_G \ Y_G \ Z_G]^T$$

$$\mathbf{C} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

and  $\mathbf{v}$  is a zero-mean Gaussian measurement noise vector in units of length.

In order to calculate a full 3D inertial position estimate based on one image, three pieces of information need to be derived from each image – the X and Y digital image coordinates of the centroid of the robot, and the robot's image area. The range of SCAMP SSV (the value of  $Z_C$ ) is computed based on the area, and values for  $X_C$  and  $Y_C$  are calculated from (3.2) and (3.3).

Unfortunately, range calculations based on image area are notoriously inaccurate. The image area of an object in a binary image is heavily dependant on illumination. In a location of high illumination, object edges will be sharper than in a location of low illumination, and when thresholding is performed, more pixels from the object will remain in the binary image. Local illumination in the NBRF does change slightly, based on proximity to overhead lights and objects in the tank that can cast shadows. Therefore, if SCAMP SSV is imaged in two different locations with equal physical ranges from a given camera, but with unequal illumination conditions, its measured range can differ significantly. Similarly, because of skylights in the roof above the NBRF, sunlight fluctuation due to clouds or time of day also causes the apparent range of SCAMP SSV to change over time even if it is perfectly still. This temporal illumination fluctuation could be compensated for with a function that increases or decreases the brightness of each image as it is captured to bring it up to some reference brightness, but this would do nothing to correct the spatial illumination fluctuation.

Initial testing and simulations indicated that small variations in the image area of SCAMP SSV, on the order of five percent, could account for position errors of greater than 30 cm. Since this level of image object area variation was very possible, and even probable in the planned operational activities of VPS, the use of area for range calculation was abandoned.

VPS was implemented such that from each captured image that contains SCAMP SSV, only two pieces of information are extracted – the undistorted X and Y digital image coordinates of the centroid of the robot,  $[X_{FU} Y_{FU}]$ . Since only two values are obtained from one image, more than one image is required to calculate the full 3D state estimate. In this implementation, each image can be thought of as providing to the state estimator a 3D line through space, on which the center of the robot lies. Ideally, simultaneously captured images would render 3D lines that crossed each other at exactly the same 3D location – the coordinates of the robot. In reality, it is likely that none of the 3D lines will cross, so the estimator computes the location in space that comes the closest to being on all of them.

This formulation causes the sensor measurements to become nonlinear functions of the positional elements of the state vector. The sensor dynamics are now

$$\mathbf{Z} = \mathbf{c}(\mathbf{x}) + \mathbf{v} \quad (6.40)$$

where

$$\mathbf{c}(\mathbf{x}) \equiv \begin{bmatrix} X_{FU} \\ Y_{FU} \end{bmatrix}$$

and  $\mathbf{v}$  again represents a zero-mean Gaussian measurement noise vector, but this time in units of pixels.

The nonlinear functions relating the 3D inertial position of SCAMP SSV  $[X_G \ Y_G \ Z_G]^T$  to the ideal measurements  $[X_{FU} \ Y_{FU}]^T$ , are

$$X_{FU} = \frac{\left( \frac{f \cdot T_X \cdot s_X}{D_{X'}} + C_X \cdot T_Z + \left( C_X \cdot r_{31} + \frac{f \cdot r_{11} \cdot s_X}{D_{X'}} \right) \cdot X_G \right.}{T_Z + r_{31} \cdot X_G + r_{32} \cdot Y_G + r_{33} \cdot Z_G} \left. + \left( C_X \cdot r_{32} + \frac{f \cdot r_{12} \cdot s_X}{D_{X'}} \right) \cdot Y_G + \left( C_X \cdot r_{33} + \frac{f \cdot r_{13} \cdot s_X}{D_{X'}} \right) \cdot Z_G \right) \quad (6.41)$$

$$Y_{FU} = \frac{\left( \frac{f \cdot T_Y}{D_Y} + C_Y \cdot T_Z + \left( C_Y \cdot r_{31} + \frac{f \cdot r_{21}}{D_Y} \right) \cdot X_G \right.}{T_Z + r_{31} \cdot X_G + r_{32} \cdot Y_G + r_{33} \cdot Z_G} \left. + \left( C_Y \cdot r_{32} + \frac{f \cdot r_{22}}{D_Y} \right) \cdot Y_G + \left( C_Y \cdot r_{33} + \frac{f \cdot r_{23}}{D_Y} \right) \cdot Z_G \right) \quad (6.42)$$

These equations are used to compute the predicted measurement,  $\hat{\mathbf{z}} = \mathbf{c}(\hat{\mathbf{x}})$ .

Since the sensor model is non-linear, the EKF equations must be implemented.

To linearize the sensor dynamics  $\mathbf{c}(\mathbf{x})$  about a given state estimate, the partial derivatives  $\mathbf{C}(\mathbf{x})$ , as defined in (6.34), are required . These were calculated to be

$$\frac{\partial \mathbf{c}(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{C}(\mathbf{x}) = \begin{bmatrix} C_1 & 0 & C_2 & 0 & C_3 & 0 \\ C_4 & 0 & C_5 & 0 & C_6 & 0 \end{bmatrix} \quad (6.43)$$

where

$$C_1 = \frac{\partial X_{FU}}{\partial X_G} = \frac{f \cdot s_X \cdot (-r_{31} \cdot (T_X + r_{12} \cdot Y_G + r_{13} \cdot Z_G) + r_{11} \cdot (T_Z + r_{32} \cdot Y_G + r_{33} \cdot Z_G))}{D_X \cdot (T_Z + r_{31} \cdot X_G + r_{32} \cdot Y_G + r_{33} \cdot Z_G)^2} \quad (6.44)$$

$$C_2 = \frac{\partial X_{FU}}{\partial Y_G} = \frac{f \cdot s_X \cdot (-r_{32} \cdot (T_X + r_{11} \cdot X_G + r_{13} \cdot Z_G) + r_{12} \cdot (T_Z + r_{31} \cdot X_G + r_{33} \cdot Z_G))}{D_X \cdot (T_Z + r_{31} \cdot X_G + r_{32} \cdot Y_G + r_{33} \cdot Z_G)^2} \quad (6.45)$$

$$C_3 = \frac{\partial X_{FU}}{\partial Z_G} = \frac{f \cdot s_X \cdot (-r_{33} \cdot (T_X + r_{11} \cdot X_G + r_{12} \cdot Y_G) + r_{13} \cdot (T_Z + r_{31} \cdot X_G + r_{32} \cdot Y_G))}{D_X \cdot (T_Z + r_{31} \cdot X_G + r_{32} \cdot Y_G + r_{33} \cdot Z_G)^2} \quad (6.46)$$

$$C_4 = \frac{\partial Y_{FU}}{\partial X_G} = \frac{f \cdot (-r_{31} \cdot (T_Y + r_{22} \cdot Y_G + r_{23} \cdot Z_G) + r_{21} \cdot (T_Z + r_{32} \cdot Y_G + r_{33} \cdot Z_G))}{D_Y \cdot (T_Z + r_{31} \cdot X_G + r_{32} \cdot Y_G + r_{33} \cdot Z_G)^2} \quad (6.47)$$

$$C_5 = \frac{\partial Y_{FU}}{\partial Y_G} = \frac{f \cdot (-r_{32} \cdot (T_Y + r_{21} \cdot X_G + r_{23} \cdot Z_G) + r_{22} \cdot (T_Z + r_{31} \cdot X_G + r_{33} \cdot Z_G))}{D_Y \cdot (T_Z + r_{31} \cdot X_G + r_{32} \cdot Y_G + r_{33} \cdot Z_G)^2} \quad (6.48)$$

$$C_6 = \frac{\partial Y_{FU}}{\partial Z_G} = \frac{f \cdot (-r_{33} \cdot (T_Y + r_{21} \cdot X_G + r_{22} \cdot Y_G) + r_{23} \cdot (T_Z + r_{31} \cdot X_G + r_{32} \cdot Y_G))}{D_Y \cdot (T_Z + r_{31} \cdot X_G + r_{32} \cdot Y_G + r_{33} \cdot Z_G)^2} \quad (6.49)$$

Both  $c(\mathbf{x})$  and  $C(\mathbf{x})$  were calculated symbolically using Mathematica™.

### 6.3 Noise Characterization

Accurate characterization of system and measurement noise can be one of the most difficult steps in implementing an optimal state estimator. In the development of VPS, the characteristics of  $\mathbf{w}$  were estimated based on experience with the robot. The characteristics of  $\mathbf{v}$  were determined empirically for a static SCAMP SSV, and estimated for the dynamic case based on observing dynamic test data. This allowed the system and measurement noise covariance matrices,  $Q_{k-1}$  and  $R_k$ , to be computed.

#### 6.3.1 Process Noise Characterization

Using the X-axis as an example, the equation for the sum of the inertial forces on SCAMP SSV along one axis is

$$\sum F_{XG} = m \cdot \ddot{\mathbf{X}}_G + C_{DT} \cdot \dot{\mathbf{X}}_G + \delta F \quad (6.50)$$

where  $\delta F$  represents the total of unmodeled forces. These forces include disturbance forces (noise), such as those from currents in the water. They also include errors in the dynamic model equations that cause the actual force exerted by the thrusters to be different from the commanded or modeled force.

There are many potential discrepancies between the real vehicle and the model that could create unmodeled forces. The maximum single direction force exerted by one thruster pair was estimated in [6] to be 9 N. This estimate was based on thrust testing performed with several thrusters, and assumes all thrusters to perform equally. By accepting this estimate, the assumption is also accepted, even though there will be significant differences between thrusters due to wear, battery life, and alignment. Also, the force in the inertial axes is calculated by rotating the robot body forces into the global reference frame using the instantaneous robot attitude estimate. This attitude estimate will invariably be imperfect, thus corrupting the calculated inertial force estimate. The actual forces applied by the thrusters are modeled as constant over EKF time steps, which is also a departure from the actual physical system, in which thruster forces can change in a continuous manner.

These model errors exist because they are either too time consuming and difficult to characterize accurately, or impossible to characterize. Therefore, an “educated guess” was made as to their magnitudes, and different values were experimented with to find a level that gave acceptable performance. The disturbance forces are modeled as unknown inputs, and multiplied by the  $\Lambda_{k-1}$  matrix just as the control inputs are.

The kinematic model for SCAMP SSV does not suffer from uncertainty, since by definition the kinematic equations do not involve any forces, and the propagation

of the velocity is ideal. However, for  $Q$  to be full rank, the velocity members must also be assumed to have some small amount of process noise.

The  $Q_{k-1}$  matrix was assumed to be constant, and was defined as follows:

$$Q_{k-1} = (\Lambda_{k-1} \cdot \Lambda_{k-1}^T) \cdot \sigma_F^2 + E_V \quad (6.50)$$

where

$$E_V \equiv \begin{bmatrix} \varepsilon_V & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \varepsilon_V & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \varepsilon_V & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The two parameters that can be adjusted in (6.50) are  $\sigma_F$  and  $\varepsilon_V$ . The first,  $\sigma_F$ , represents the expected standard deviation of the unmodeled forces, in Newtons, incident on SCAMP SSV at any point in time. It was initially estimated to be 2 N, but during dynamic testing it was determined that  $\sigma_F = 4$  N is more appropriate, in that with the lower value, the state estimate would at times diverge unrecoverably from the true state when measurements were either very few or unavailable. The value of  $\varepsilon_V$  was set to 0.08 during testing, and was set to this level so that it would be (arbitrarily) 50 times smaller than  $\sigma_F$ .



### 6.3.2 Measurement Noise Characterization

Because the elements of the process noise vector could not be observed directly, and the dynamic model is already based on the best estimate of the robot's dynamics currently available,  $Q_{k-1}$  could only be estimated.  $R_k$ , on the other hand, can be developed empirically. SCAMP SSV was placed in the water, and firmly attached to a rigid fixture, as shown in figure 6.5. The fixture is one of the stilts upon which the calibration frame rests during use. It is rigidly bolted to a large and heavy truss that rests on the bottom of the NBRF. It has an adapter attached to the top that enables SCAMP SSV to be easily attached and released using a hose clamp.

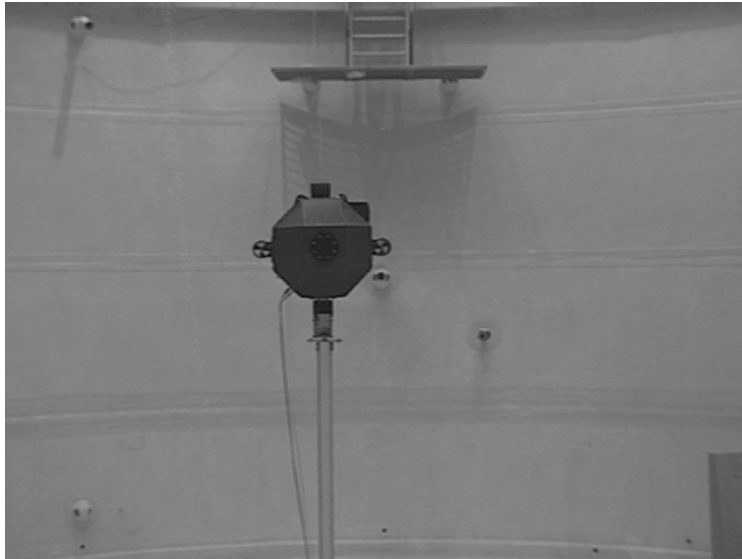


Figure 6.7 SCAMP SSV rigidly attached to fixture for sensor noise characterization tests

With the robot rendered stationary, 250 images were captured using a single camera. The images were processed to extract the undistorted image

coordinates,  $[X_{FU} Y_{FU}]$ , and area of the robot. The centroid coordinates in all 250 images would have been identical if noise was absent. The measurement noise covariance matrix was measured directly from the variance in the  $X_{FU}$  and  $Y_{FU}$  values between images. This was done using the MATLAB<sup>TM</sup> command  $\text{COV}([X_{FU} Y_{FU}])$ , where  $X_{FU}$  and  $Y_{FU}$  are vectors containing the 250 individual  $X_{FU}$  and  $Y_{FU}$  values. This process was repeated with six cameras, with SCAMP SSV in five different positions. Averaging the  $R_k$  values from the different tests, the following result was obtained:

$$R_k = \begin{bmatrix} .0121 & .002 \\ .002 & .0168 \end{bmatrix} \quad (6.51)$$

Recalling that the diagonal elements of  $R_k$  represent the square of the standard deviations,  $\sigma$ , of the measurement noise vector members, it can be seen from (6.46) that  $\sigma$  for the measurement noise in channels is estimated to be less than 0.13 pixels. This suggests that the measurement noise in both  $X_{FU}$  and  $Y_{FU}$  will be less than  $3\sigma = 0.4$  pixels for 99% of the time.

It is important to note however that this  $R_k$  matrix represents the minimum measurement noise that will be present, and is for the static case. Due to lighting changes, background objects, and the limited robustness of VPS to bad camera data, the 0.4 pixel estimate was significantly exceeded during both static and dynamic tests. For example, SCAMP SSV may be in a location or orientation that

either caused glare on one of the robot's body panels or removal of some of the robot's area due to a dark object being behind it in the background image. Both of these situations caused the apparent image area of SCAMP SSV to be reduced, which also caused a substantial shift in the estimated centroid. These sources of measurement noise will be discussed in more detail in Chapters 7 and 8. This data was not considered in the calculation of the  $R_k$  matrix. Therefore, the  $R_k$  matrix to be used for actual state estimation needs to be much larger than that calculated under static conditions. Based on measurements gathered during dynamic testing of VPS, measurement noise seemed to typically be less than approximately 4.5 pixels, so this was assumed to be close to the  $3\sigma$  value for the measurement noise, and a standard deviation of slightly greater than one-third that value was used. With  $\sigma = \sqrt{3}$ , and  $\sigma^2 = 3$ , the following  $R_k$  matrix was used in gathering the results presented in Chapter 7

$$R_k = \begin{bmatrix} 3 & .002 \\ .002 & 3 \end{bmatrix} \quad (6.52)$$

Measurements that suffer from either panel glare or background object image artifact should be rejected by the EKF. This functionality was not implemented during this research effort, but is discussed in Chapters 7 and 8.

### 6.3.3 VPS EKF Implementation on the Control Station Computer

The Extended Kalman Filter equations outlined above were incorporated into raptor, the software that runs the control station from which SCAMP SSV is piloted. As mentioned previously, the calculation of the state estimate is controlled by a timer that runs separately from the rest of the control station operations, such as communication and telemetry archiving. Figure 6.6 illustrates the EKF algorithm as it is implemented in raptor.

- 1) Initialize all global variables needed by the EKF.
- 2) Initialize and start EKF timer.
- 3) Store the current inertial attitude estimate,  $\mathbf{q}_k$ , and body forces.
- 4) Use  $\mathbf{q}_{k-1}$  and the body forces of the previous time step to compute  $\mathbf{u}_{k-1}$ .
- 5) Propagate from  $\hat{\mathbf{x}}_{k-1(+)}$  to  $\hat{\mathbf{x}}_{k(-)}$ .
- 6) Propagate from  $P_{k-1(+)}$  to  $P_{k(-)}$ .
- 7) Set the valid measurement array.
- 8) For each valid measurement  $\mathbf{z}_k$ :
  - a) Compute the current partial derivatives,  $C_k(\hat{\mathbf{x}}_{k(-)})$ .
  - b) Compute the predicted measurement,  $c_k(\hat{\mathbf{x}}_{k(-)})$ .
  - c) Compute the Kalman Gain Matrix,  $K_k$ .
  - d) Correct  $P_k$  based on  $K_k$  and  $C_k(\hat{\mathbf{x}}_{k(-)})$ .
  - e) Correct  $\hat{\mathbf{x}}_k$  based on  $\mathbf{z}_k$ ,  $c_k(\hat{\mathbf{x}}_{k(-)})$ , and  $K_k$ .
- 9) Update  $P_{k-1}$  and  $\hat{\mathbf{x}}_{k-1(+)}$ .
- 10) Update Raptor global variables for storage and communication.

Figure 6.8 Algorithm of the VPS Extended Kalman Filter

Immediately after raptor is started, all of the variables needed for the EKF are declared and initialized. Once this is done, the timer is started. The EKF is executed at a fixed frequency, because this causes  $\Phi_{k-1}$  and  $\Lambda_{k-1}$  to be constant matrices, greatly reducing the number of calculations required at each iteration of the estimator.

At each time step, the timer executes a single function, `do_VPS_filterwork()`. Each time `do_VPS_filterwork()` is called, it first accesses the SCAMP SSV telemetry data from raptor, which is updated at 50 Hz. It calculates the control forces in inertial coordinates,  $\mathbf{u}_k$ , applied to the robot by the thrusters at the current time step and stores them. Based on the inertial forces from the previous time step and the dynamic model, the EKF propagates the state estimate,  $\hat{\mathbf{x}}$ , and the error covariance matrix,  $P$ , one time step into the future.

Then the EKF accesses the `vps_data` measurements, stored in a global array. This data comes to raptor as fast as the frame grabbers can acquire images, and `VPS_client` can process and send it. The EKF looks at an array of raw VPS measurements, and selects from it only the measurements that are recent enough for use, meaning they were obtained less than one time step ago. Once the valid subset of measurements has been selected,  $\hat{\mathbf{x}}$ , starting at its value after the propagation step, is repeatedly corrected – once for each valid measurement. For each measurement used, new values of the necessary intermediate variables, such as  $K_k$ ,  $P$ , and so on, must be computed. If there are

no valid measurements – if for instance SCAMP SSV has flown into an area in the NBRF where it is not visible to any cameras – a new state estimate would still be computed, but only from the propagation based on the dynamic model and the applied control forces. No correction would take place in such a situation.

Once the state estimate has been corrected with the last measurement, the EKF updates the appropriate global variable in raptor, and then waits until the timer signals it to perform another iteration. Raptor will not send this new inertial state estimate to SCAMP SSV or to the vision computers until those systems send to raptor their own transmissions, but this communication is done completely independently from the operation of the EKF.

## Chapter 7

### Positioning Results

Static and dynamic tests were conducted in the NBRF to determine the accuracy of the EKF state estimates of VPS. For the static case, it would ideally be possible to place SCAMP SSV at several known, fixed locations in the NBRF, and then compare the position estimates that VPS provided for each location with the known truth. Similarly, in the dynamic case, it would be ideal if SCAMP SSV could be moved through a known trajectory, and the tracking properties of VPS could be observed for that trajectory. However, there is no readily available method to provide these “truth” measurements or trajectories.<sup>3</sup> This would require an accurate and independent inertial navigation system that would verify the performance of VPS, but such a system is unavailable.

To compensate for the lack of external verification measurements, the VPS cameras were partitioned into groups, supplying measurements to independent state estimators. The individual state estimates are compared over a variety of positions and trajectories for agreement. In addition to this, both static and dynamic testing was performed using all six operational VPS cameras to supply measurements to a single EKF.

---

<sup>3</sup> A railcart system was devised to provide a known linear trajectory for the previous acoustic positioning system studied for inertial navigation in neutral buoyancy. Although a possible avenue of future work, underwater use of this railcart was infeasible for this thesis due to both mechanical fixture and electronics availability and operational status.

Section 7.1 presents data taken for static estimation tasks using measurement data only. Section 7.2 presents data gathered showing the state estimate evolution under applied input forces, with measurement data absent. Section 7.3 presents dynamic state estimation results for SCAMP SSV teleoperated in the NBRF with six VPS cameras providing data to a single EKF, while Section 7.4 presents dynamic state estimation results using multiple EKFs relying on non-overlapping subsets of camera inputs.

## **7.1 Static Positioning Results**

### **7.1.1 Static Position of Hanging Ball, Double EKF**

As a simple assessment of static positioning accuracy, a water-filled ball, covered in black fabric, was suspended in the NBRF from a white rope. This black ball simulated SCAMP SSV, allowing extensive system testing without the necessity of frequently sealing the robot and checking for leaks, etc. The position of the ball could be changed easily using the crane, but only in the crane's line of motion (roughly North-South) and in the vertical ( $Z_G$ ) direction. Since the force inputs to the EKF were zero, the EKF computed the position of the black ball using the camera measurements only in what is effectively an iterative least-squares calculation. Figure 7.1 shows the black ball hanging in the NBRF, as seen from camera 2.



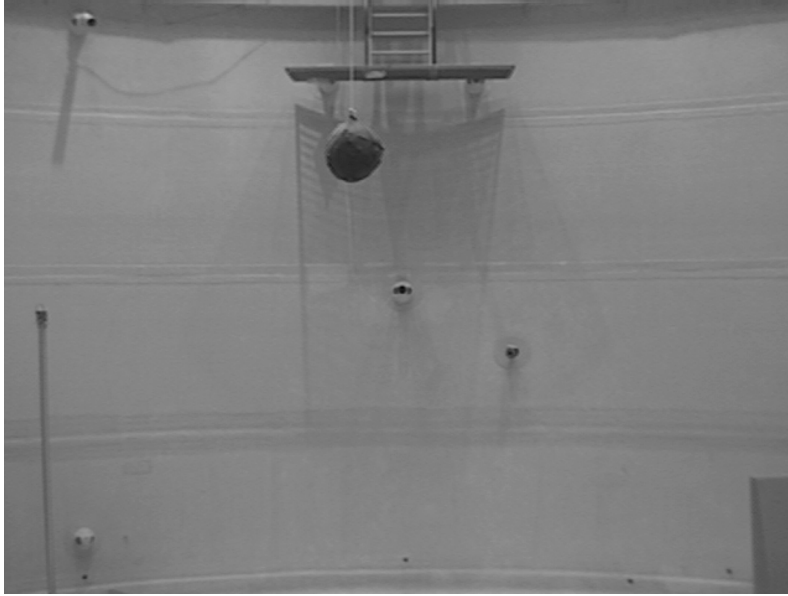


Figure 7.1 Black ball, simulating SCAMP SSV, hanging in the NBRF

The operational VPS cameras were separated into two sets, and each was associated with an independent EKF. The separated architecture is shown in Table 7.1.

Table 7.1 Double EKF Architecture

<b>Filter</b>	<b>Camera Number</b>	<b>Camera Viewing Direction</b>
EKF1	Camera 1	Middle ring, + X direction
	Camera 2	Middle ring, + Y direction
	Camera 5	Top ring, -Y and +X directions
EKF2	Camera 3	Middle ring, - X direction
	Camera 4	Middle ring, - Y direction
	Camera 6	Top ring, +Y and +X directions

Table 7.2 contains position estimates, calculated by the two estimators, for the black ball hanging in three positions in the NBRF. The positions below are

average positions over approximately 10 seconds. The  $|E|$  values represent the magnitude of the error vector between the ball positions as estimated by EKF1 and EKF2. Table 7.3 contains the standard deviation information for these tests.

Table 7.2 Double EKF static black ball position estimates, three positions

Position	EKF1			EKF2			$ E $ (m)
	X (m)	Y (m)	Z (m)	X (m)	Y (m)	Z (m)	
1	-0.1531	0.9387	-0.4689	-0.1725	0.9412	-0.4621	0.0207
2	-0.7217	0.8007	-0.0634	-0.7027	0.8196	-0.0446	0.0327
3	0.4963	1.0611	0.475	0.4734	1.072	0.474	0.0254

Table 7.3 Double EKF static black ball position standard deviations, three positions

Position	EKF1			EKF2		
	$\sigma_x$ (mm)	$\sigma_y$ (mm)	$\sigma_z$ (mm)	$\sigma_x$ (mm)	$\sigma_y$ (mm)	$\sigma_z$ (mm)
1	.019	.008	.016	.004	.012	.012
2	.028	.009	.014	.011	.021	.026
3	.007	.016	.033	.005	.087	.065

Table 7.2 shows that for this task, the position estimates calculated by the two independent estimators agree to within about 3 cm. This also may be a pessimistic agreement level, since the ball had a small periodic motion due to water currents and the flexible crane suspension. This also heavily influences the size of the standard deviations for these data. This test is using one EKF as the truth, or reference, for the other. It does not guarantee system accuracy, however, because a systematic error could cause both EKFs to calculate the same incorrect position. This does indicate, however, that the position estimation task can be performed repeatably, for the camera groupings selected.

### 7.1.2 Static Position of SCAMP SSV, Single EKF

To get static data with SCAMP SSV, the robot itself was attached to the rigid fixture (the stilt) described in Chapter 6. As with the black ball, position data was obtained for three positions. The stilt was attached to its base truss in three locations, defined by bolt hole patterns in the truss. Since the spacing of the bolt hole patterns is well known, the relative locations of the three possible SCAMP SSV positions are also known.

Table 7.4 shows the X, Y and Z coordinates of SCAMP SSV for each of the three static positions, as calculated by a single EKF, using data from all six operational cameras, and Table 7.5 contains the standard deviation data for those measurements. Table 7.6 shows two things: the known distances between the three static positions, as calculated based on the bolt hole patterns, and the distances between the static positions as calculated from the position estimates provided by VPS. The first column should be read as “the distance from position 1 to position 2”, etc. The line from positions 1 to 2 lies roughly on the global X axis, and the line from positions 2 to 3 lies roughly on the global Y axis.

Table 7.4 Static SCAMP SSV position data, using all 6 cameras

<b>Position</b>	<b>X (m)</b>	<b>Y (m)</b>	<b>Z (m)</b>
<b>1</b>	-0.2644	0.6944	1.2113
<b>2</b>	0.9155	0.8805	1.2119
<b>3</b>	1.111	0.0682	1.2127

Table 7.5 Static SCAMP SSV standard deviation data, using all 6 cameras

Position	$\sigma_x$ (mm)	$\sigma_y$ (mm)	$\sigma_z$ (mm)
1	.022	.060	.104
2	.004	.003	.006
3	.053	.008	.058

Table 7.6 Position-to-position distance values: actual and as calculated by VPS

Distance	Actual Distance (m)	VPS Distance (m)	Difference (m)
1 to 2	1.054	1.194	0.140
2 to 3	0.819	0.836	0.017
1 to 3	1.335	1.33	-0.005

It can be seen that the distances between positions 2 and 3, and between positions 1 and 3, as measured by VPS, agree well – to within 2 cm – with the physical distances. This is not the case for the distance from positions 1 to 2, which differs from the real distance by 14 cm. This is caused by an important source of error inherent to VPS. The vision processing algorithms implemented in VPS assume a dark robot on a light background. Dark objects that do exist in the background, when SCAMP SSV moves in front of them, can corrupt the centroid coordinates of the robot. This is the case with position 2 in the static SCAMP SSV tests. Figure 7.2 is an image acquired by camera 1 for static position 2. It can be seen that it is in front of a vertical member of a truss. After image processing, the image centroid of SCAMP SSV is calculated based on the dark shape shown in Figure 7.3.



Figure 7.2 SCAMP SSV in static position 2, viewed from camera 1

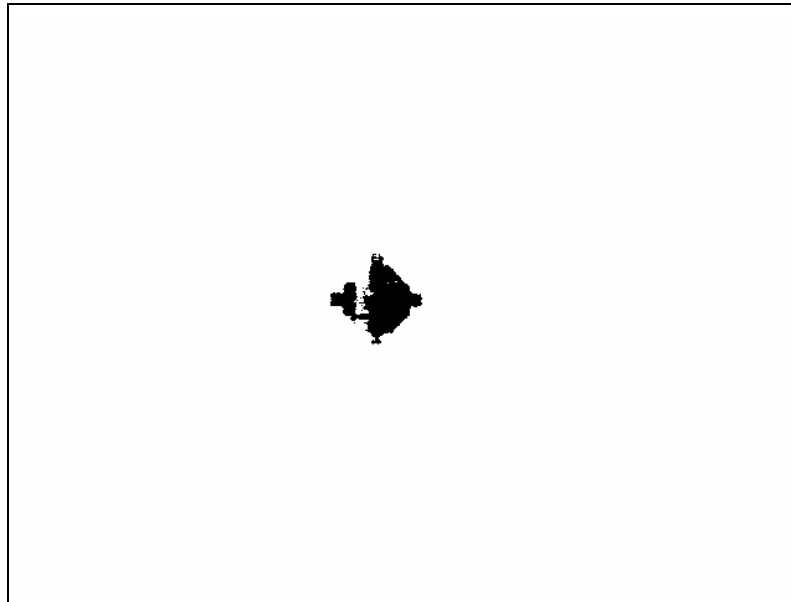


Figure 7.3 Image from Figure 7.2, after vision processing

It is clear that many pixels of the SCAMP SSV image that should legitimately be set to black have been erroneously set to white, due to the truss member in the

background. Camera 5 also suffers from a similar corruption due to another truss in its background for positions 1 and 2. The type of data corruption shown in Figures 7.2 and 7.3 could be mitigated using a technique known as convex hulling, in which an open or area can be “filled in” by a vision algorithm. This will be discussed more in Chapter 8.

Notice that the measured and actual distances from position 2 to 3 agree much better than those from position 1 to 2. This goes against intuition, since it would be thought that if the estimate of position 2 is inaccurate, any distances computed using that position would also be inaccurate. This can be explained, however, by the fact that noise coming from a camera is not introduced equally to all global axes. The measurements from each camera are in the form of  $[X_{FU}$   $Y_{FU}]$  pairs. The nonlinear relationship between a measurement set from a specific camera and the inertial position is defined by equations (6.35) and (6.36). A single camera cannot provide information about an object’s position along the camera’s optical axis, or in other words, information about range. As an example, consider a camera with an optical axis parallel to the global Y-axis. If this camera were providing extremely noisy data, this noise would not in any way corrupt the global Y-axis estimate, because the data from that camera is not used to update the Y-axis estimate. It would only corrupt the global estimates in the global X and Z axes.

### 7.1.3 Static Position of SCAMP SSV, Double EKF

Table 7.7 shows the VPS position estimates of SCAMP SSV, in the same three static positions, as computed by the two independent EKFs, and Table 7.8 contains the standard deviation data for the same tests. This data is presented, as was the double EKF black ball data, to show that VPS is capable of providing position estimates that are repeatable between different sets of cameras. The value  $|E|$  in Table 7.7 represents the magnitude of the error vector between the position estimates calculated by EKF1 and EKF2.

Table 7.7 Static SCAMP SSV position data, double EKF

Position	EKF1			EKF2			$ E $ (m)
	X (m)	Y (m)	Z (m)	X (m)	Y (m)	Z (m)	
1	-0.2155	0.677	1.2435	-0.2725	0.7008	1.2197	0.0662
2	0.8717	0.7844	1.1796	0.74	0.9321	1.1289	0.2043
3	1.1193	0.0547	1.2065	1.1229	0.08	1.2301	0.0348

Table 7.8 Static SCAMP SSV standard deviation data, double EKF

Position	EKF1			EKF2		
	$\sigma_x$ (mm)	$\sigma_y$ (mm)	$\sigma_z$ (mm)	$\sigma_x$ (mm)	$\sigma_y$ (mm)	$\sigma_z$ (mm)
1	.081	.034	.042	.005	.004	.002
2	.312	.092	.136	.161	.354	.150
3	.015	.005	.015	.005	.001	.001

Position 2, and to a lesser extent position 1, still suffer from camera noise due to dark objects in the background. Therefore, as expected, the position estimates for these positions, as calculated by EKF1 and EKF2, do not agree as closely as

the independent estimates for position 3, which did not have a dark object behind the robot.

Since both cameras 1 and 5 are on EKF1, its position estimate is likely less accurate than that of EKF2. If only the output for EKF2 is used to calculate the distance from positions 1 to 2, that distance is calculated as 1.043 m, which is only 1 cm different from the actual distance of 1.054 m. The agreement in the real position-to-position distances and those measured by VPS indicates that VPS is capable of accurate (single digit cm-scale) static position estimation, in the absence, or rejection, of noise due to dark background objects. This is a highly qualified statement, but there are many candidate noise-rejection schemes that can enhance the ability of VPS to robustly estimate static (and dynamic) position.

## ***7.2 Dynamic State Evolution with No Measurement Data***

In order to verify the operation of the state propagation step of the EKF, SCAMP SSV was sent control force commands from the hand controllers while hanging stationary from a crane above the deck of the NBRF. In this test, the SCAMP SSV computer and attitude sensors were functioning as they do in free flight, with robot attitude was being calculated onboard in real time and sent back to raptor and the EKF. This attitude information was used to convert body forces into global forces. Camera data was not sent to the EKF for this test. Since no measurement data was available for use in the update step of the EKF, the



estimator “thought” the robot was moving freely, with its motion governed by the dynamic equations and the commanded forces. The predicted motion calculated by the EKF was compared to the force input to verify qualitative agreement.

This test was performed with SCAMP SSV approximately aligned with its body X axis pointing East (in the positive inertial Y direction), its body Y axis pointing South (in the negative inertial X direction), and body Z axis pointing down (in the positive inertial Z direction). Table 7.9 presents the directions of the forces that were commanded during this test with respect to the robot body reference frame, and with respect to the global reference frame. Figure 7.4 shows the global force commands sent to the robot using the translational hand controller.

Table 7.9 Approximate force directions in SCAMP SSV body and global frames

<b>Time Span (sec)</b>	<b>Body Thrust Direction</b>	<b>Global Thrust Direction</b>
<b>9 to 14</b>	$(+) X_B$	$(+) Y_G$
<b>15 to 20</b>	$(+) Y_B$	$(-) X_G$
<b>21 to 32</b>	$(-) X_B$	$(-) Y_G$
<b>33 to 43</b>	$(-) Y_B$	$(+) X_G$

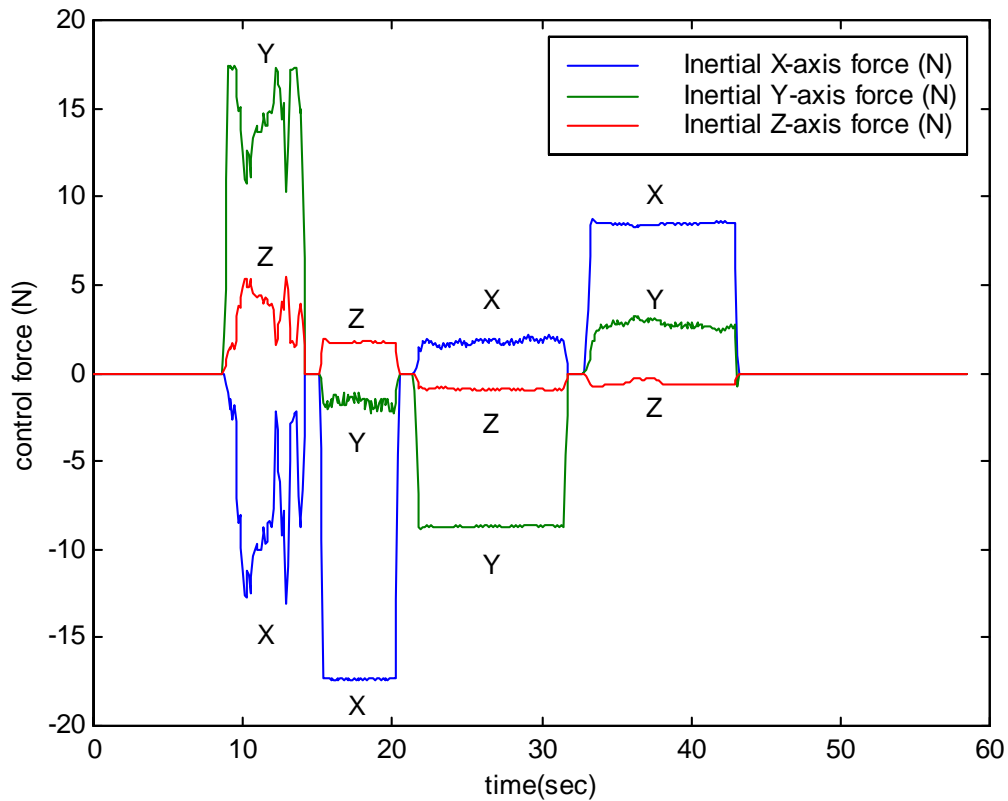


Figure 7.4 Global force inputs applied to SCAMP SSV

In this test, the robot body axes were not exactly aligned with the global axes as desired, so, for instance, forces applied in the body X direction do not result in a clean and exclusive global Y direction force, but produce a force predominantly along the global Y direction, and smaller forces along the other two global axes as well. In addition to this, it is difficult to guarantee pure single axis input to the hand controller.

Figures 7.5 through 7.10 are the position and velocity plots over time that the EKF calculated based on the force input.

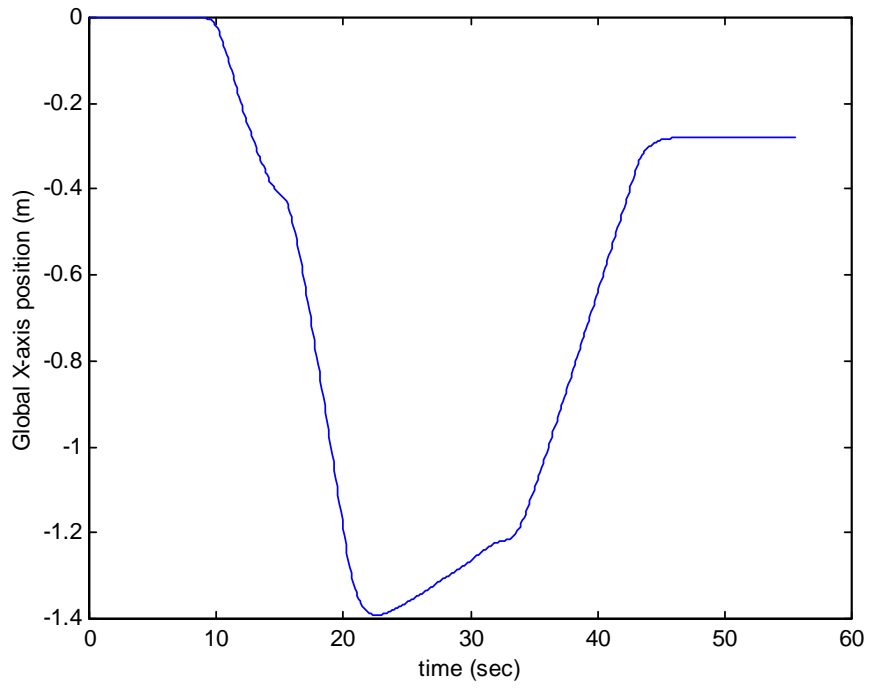


Figure 7.5 Estimated global X axis position versus time, propagation test

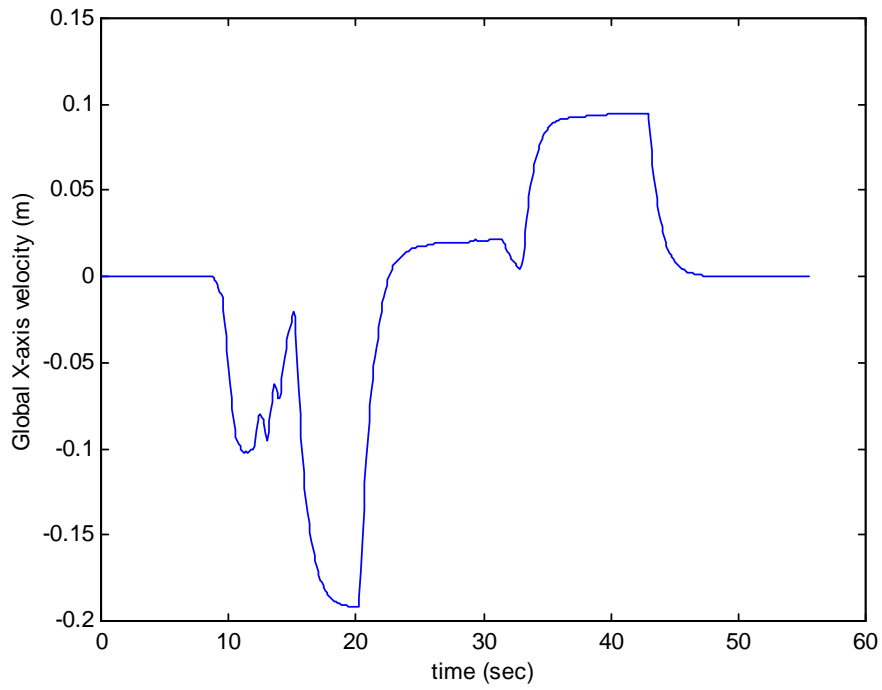


Figure 7.6 Estimated global X axis velocity versus time, propagation test

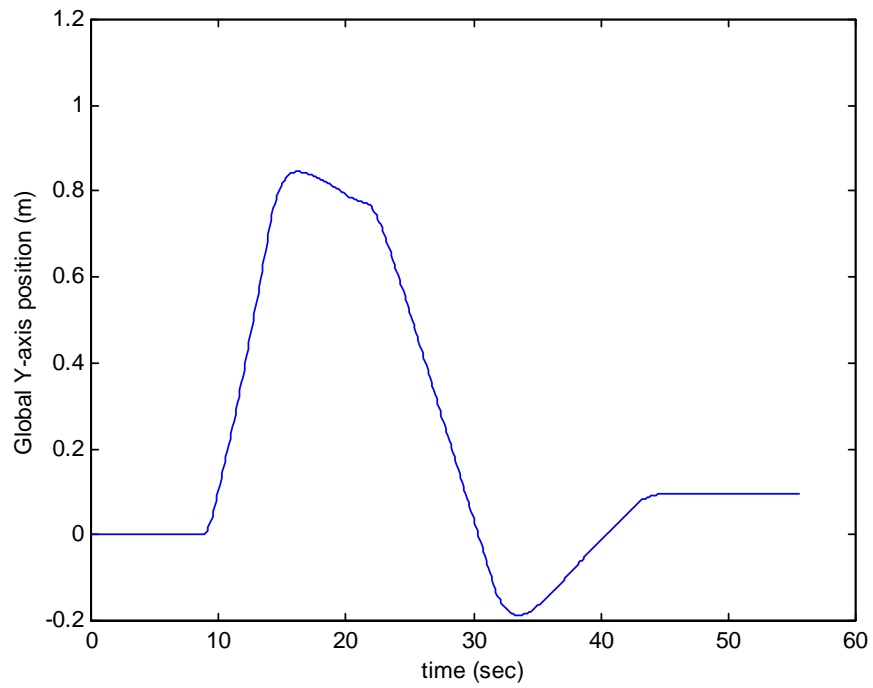


Figure 7.7 Estimated global Y axis position versus time, propagation test

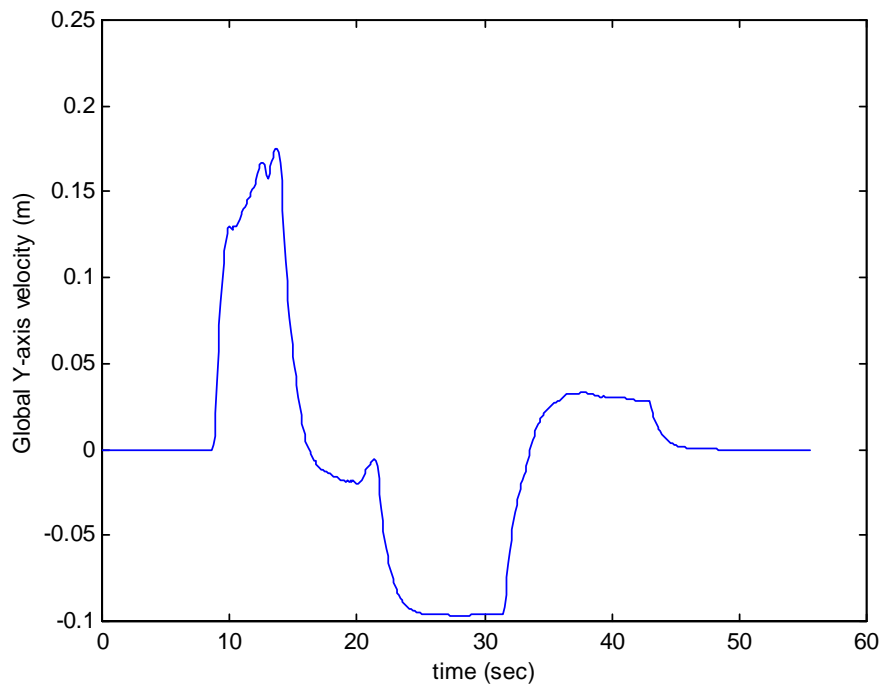


Figure 7.8 Estimated global Y axis velocity versus time, propagation test

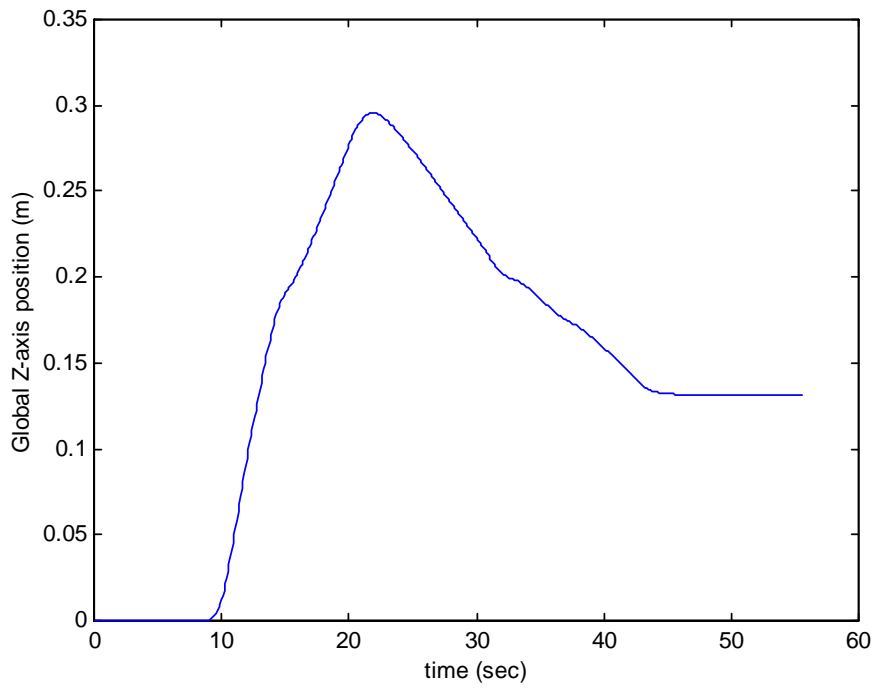


Figure 7.9 Estimated global Z axis position versus time, propagation test

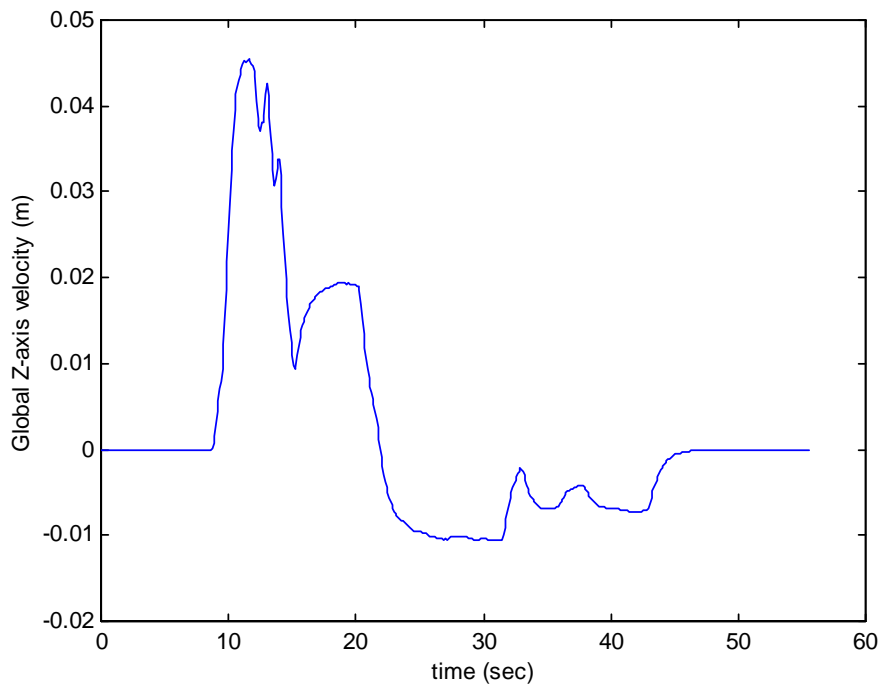


Figure 7.10 Estimated global Z axis velocity versus time, propagation test

It was expected that under full positive or negative hand controller commands (maximal force commands) in a given axis, the EKF should predict acceleration along that axis up to a constant terminal velocity, and then a diminution of the velocity over a short time to zero upon cessation of the force input. For positive thrust, terminal velocity is approximately 0.2 m/s, and for negative thrust  $-0.1$  m/s, based on the tests described in Chapter 6.

The results shown in the figures above indicate, qualitatively, that the VPS EKF is capable of propagating the state estimate based on commanded force inputs. The velocities achieved under maximum positive and negative thrusts agree with the observed terminal velocities. The slopes of the position plots agree with the sign and magnitude of the velocity plots in all axes. The results also indicate that the EKF correctly converts body axis forces into global forces. The body axis forces in Table 7.9 are converted into the correct global forces, which in turn accelerate the robot along the correct global axis. While this cannot be interpreted as an exhaustive demonstration of the accuracy of the EKF propagation, it does demonstrate that the EKF is faithful to the simple dynamic model assumed for SCAMP SSV.

### **7.3 Dynamic SCAMP SSV State Estimation, Single EKF**

For dynamic tests, SCAMP SSV was piloted in a smooth, continuous trajectory, near the center of the NBRF such that all six operational cameras had near-continuous coverage of the vehicle.<sup>4</sup> The trajectory below in Figure 7.11 was recorded in real-time by VPS, using all six operational cameras.

Figures 7.12 and 7.13 show the global X axis position and velocity, respectively, of SCAMP SSV versus time. The position plot shows smooth estimate evolution over time. There is a spike of roughly 0.3 m at approximately 57 seconds. In the noisiest regions – from 5 to 25 seconds, from 57 to 75 seconds, and 135 to 155 seconds, the oscillation of the plot is no more than 10 cm. This oscillation is shown in Figure 7.13, which is a magnified view of the 125 to 170 seconds section of Figure 7.12. Plots for the Y and Z axis positions and velocities are not shown for this test, but are for tests described in the next section.

---

<sup>4</sup> Although software was in place to reject camera data once the vehicle left one or more images planes, this software is not yet fully debugged, resulting in erroneous state estimates under certain conditions. Additionally, use of all cameras allowed the data to be split into unique subsets, providing independent state estimates that validate camera calibration and, except for possible systematic errors, EKF accuracy.

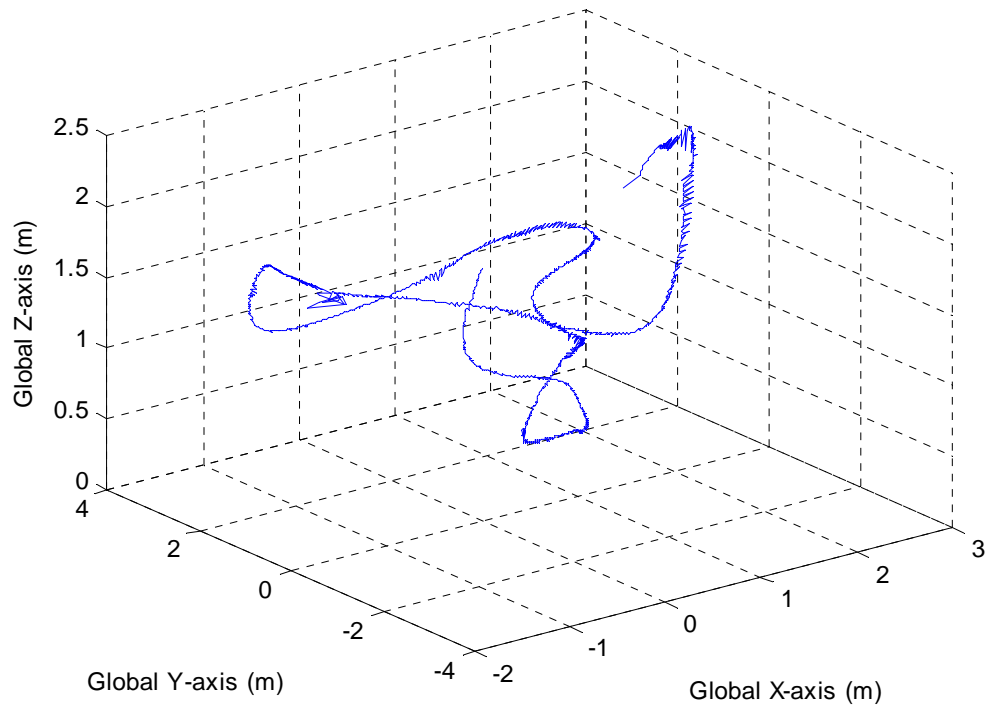


Figure 7.11 Trajectory of SCAMP SSV in 3D

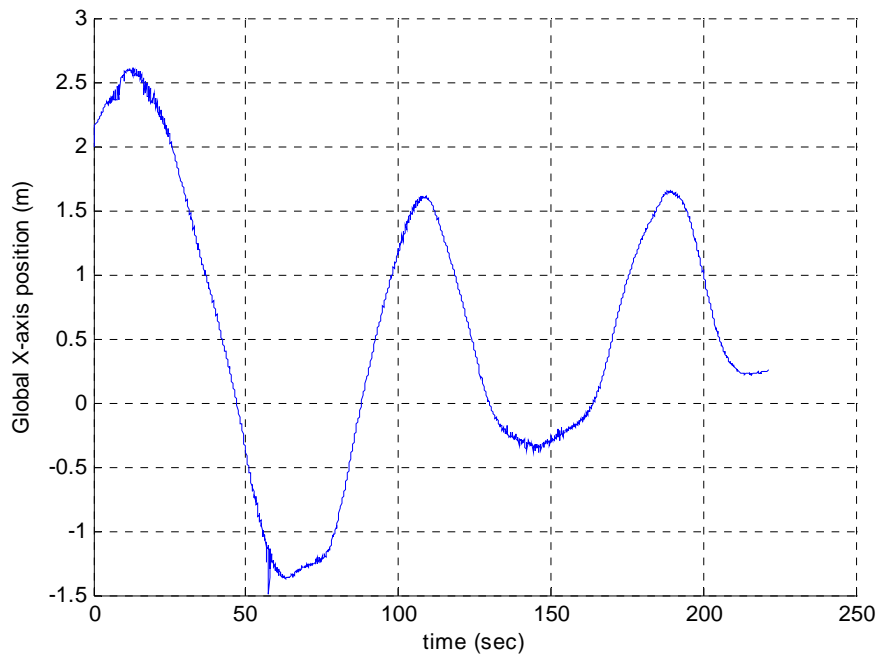


Figure 7.12 SCAMP SSV global X axis position vs. time



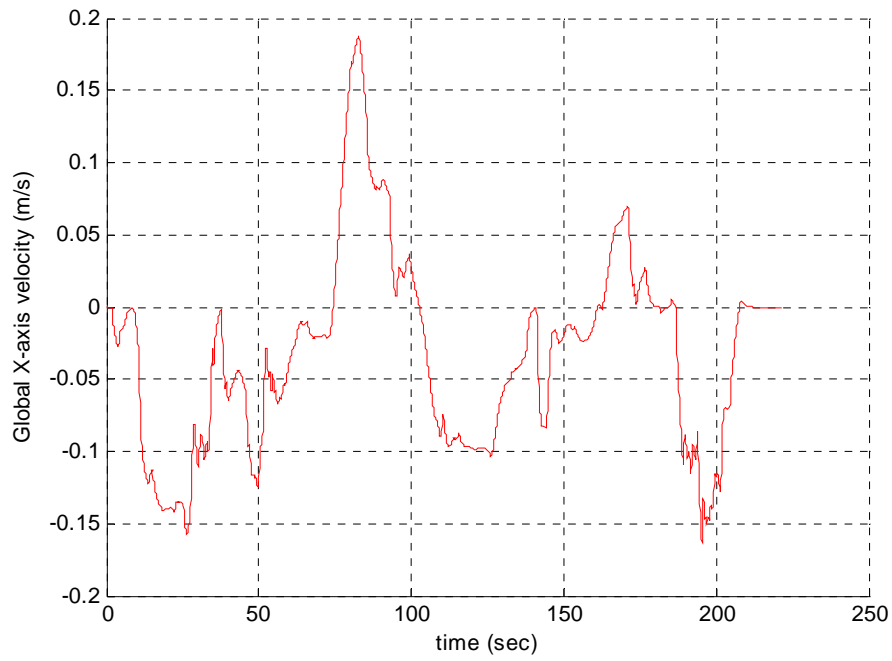


Figure 7.13 SCAMP SSV global X axis velocity vs. time

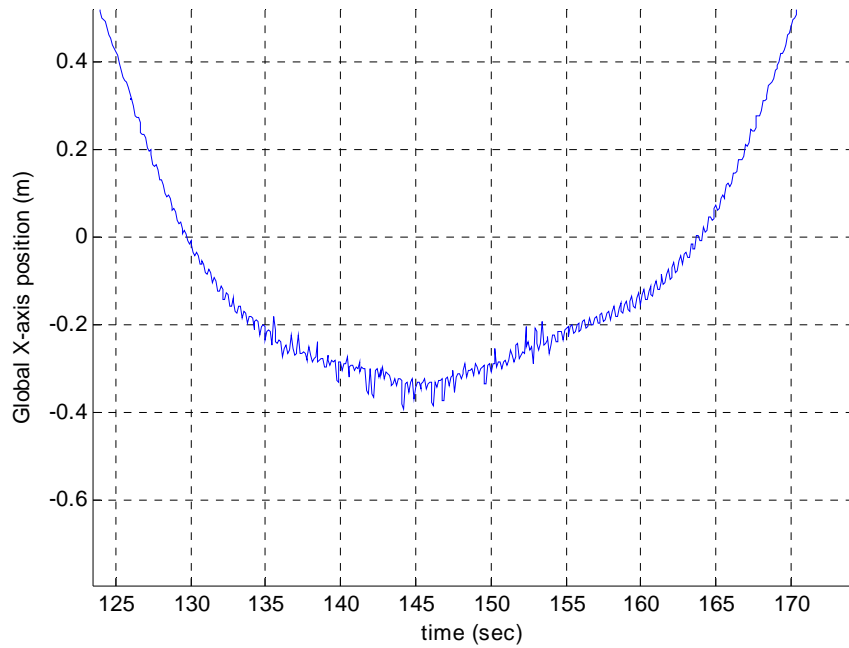


Figure 7.14 SCAMP SSV global X axis position vs. time, magnified

Figure 7.15 and Figure 7.16 are also plots of global X axis position and velocity, but this time information from the EKF error covariance matrix  $P$  is included. The first and second diagonal elements of  $P$  correspond to the global X axis position and velocity, respectively. The square root of those elements is the standard deviation,  $\sigma$ , of the X axis position and velocity. According to the EKF, there is a 99% probability that the true value of the X axis position and velocity will lie within  $\pm 3\sigma$  of its estimates of those values. The additional traces on figures 7.15 and 7.16 are the  $3\sigma$  error bounds for position and velocity.

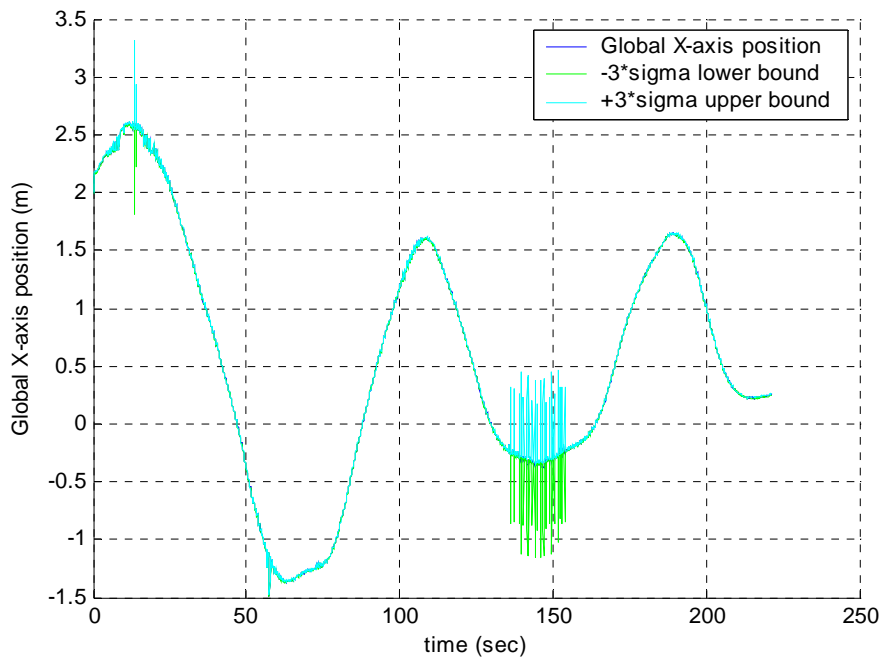


Figure 7.15 SCAMP SSV global X-axis position vs. time, with  $\pm 3\sigma$  error bounds

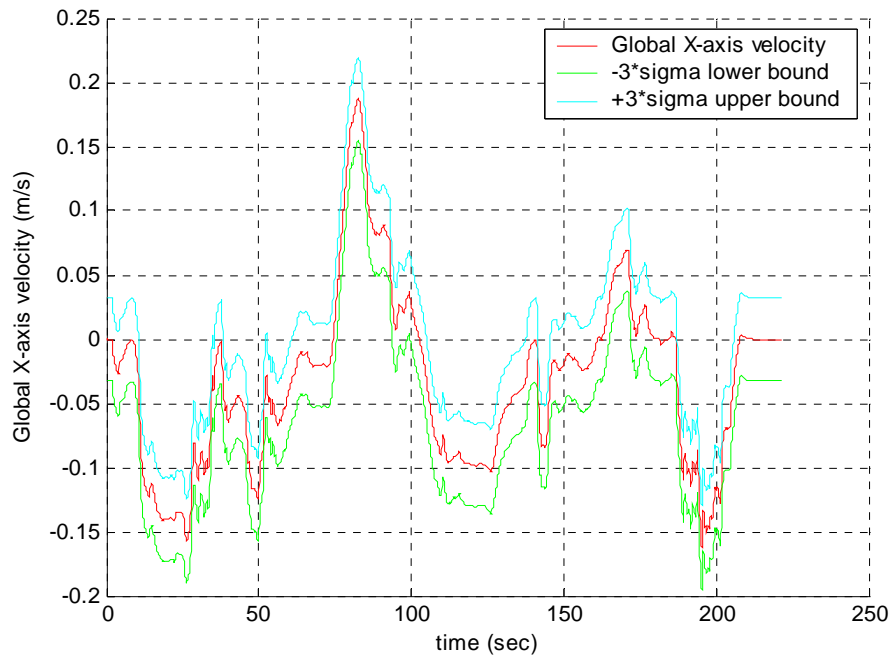


Figure 7.16 SCAMP SSV global X-axis velocity vs. time, with  $\pm 3\sigma$  error bounds

In Figure 7.15, for most of the plot the estimate and the error bounds are indistinguishable at the scale shown, so Figure 7.17 below shows a magnified view of the Figure 7.15 for the interval between 105 and 112 seconds. It can be seen that for all of that section, the error bounds are not more than 1 cm above and below the estimated value.

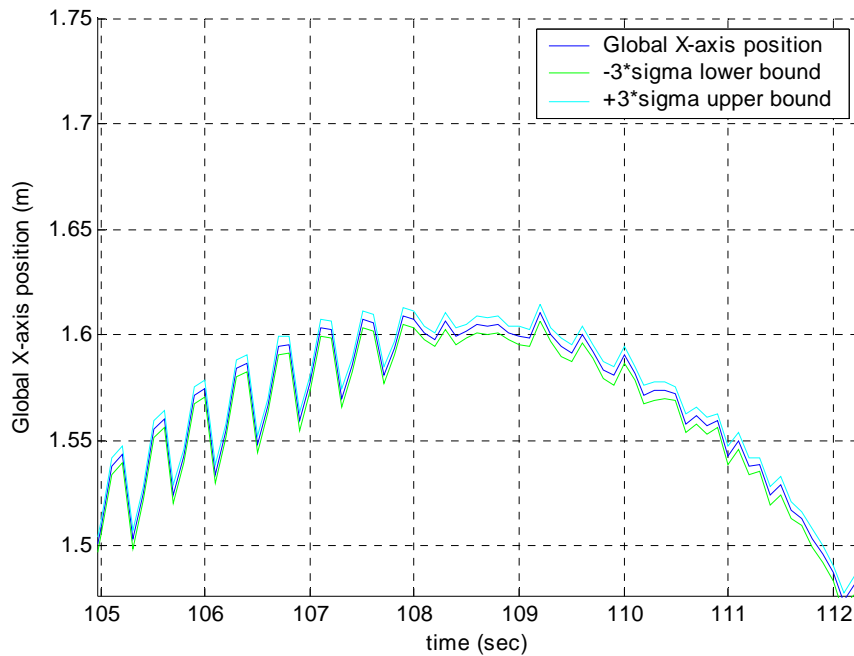


Figure 7.17 SCAMP SSV global X-axis position vs. time, with  $\pm 3\sigma$  error bounds, magnified for T=105 s to T=112 s

The error bound plot in Figure 7.15 presents some questions. The error bounds greatly expand between approximately 133 and 155 seconds, which is expected, because this region of the state estimate exhibits more noise than others, and should thus have larger error bounds. However, there are other regions of the estimate that exhibit noise for which the error bounds remain small, such as the spike that occurs at approximately 57 seconds. According to the EKF, the state estimate at these regions is just as accurate as the regions where little noise is seen. To better understand what is happening, it is necessary to look at the measurements provided by the cameras. Figures 7.18 through 7.23 show the camera measurement values, in pixels, from the six cameras during the test.

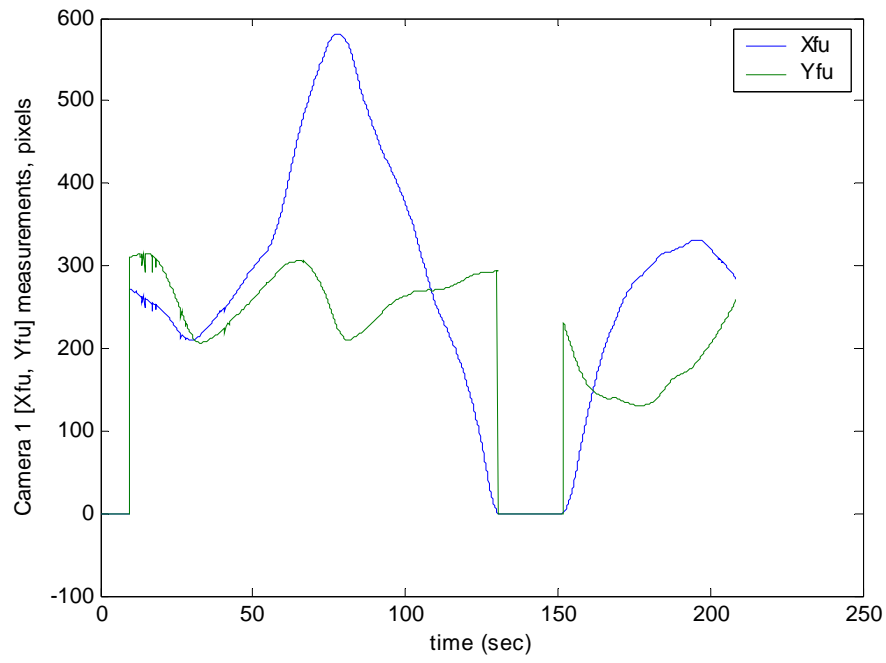


Figure 7.18 Camera 1 pixel measurements

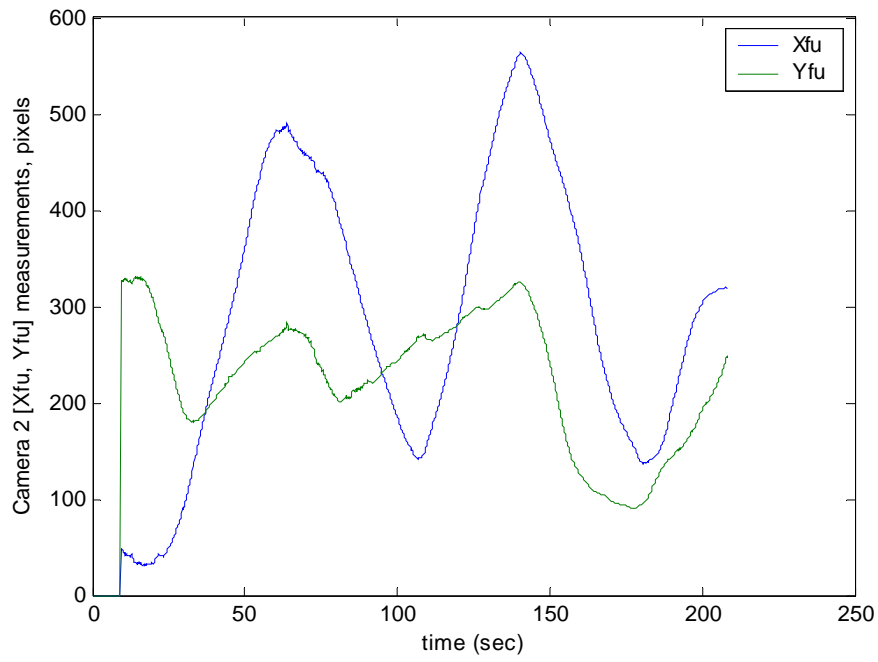


Figure 7.19 Camera 2 pixel measurements

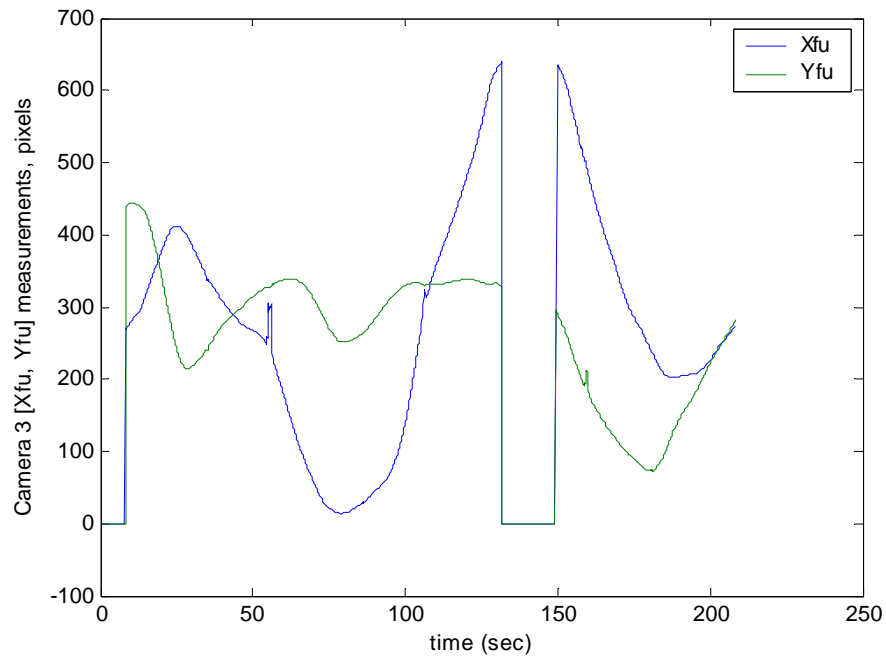


Figure 7.20 Camera 3 pixel measurements

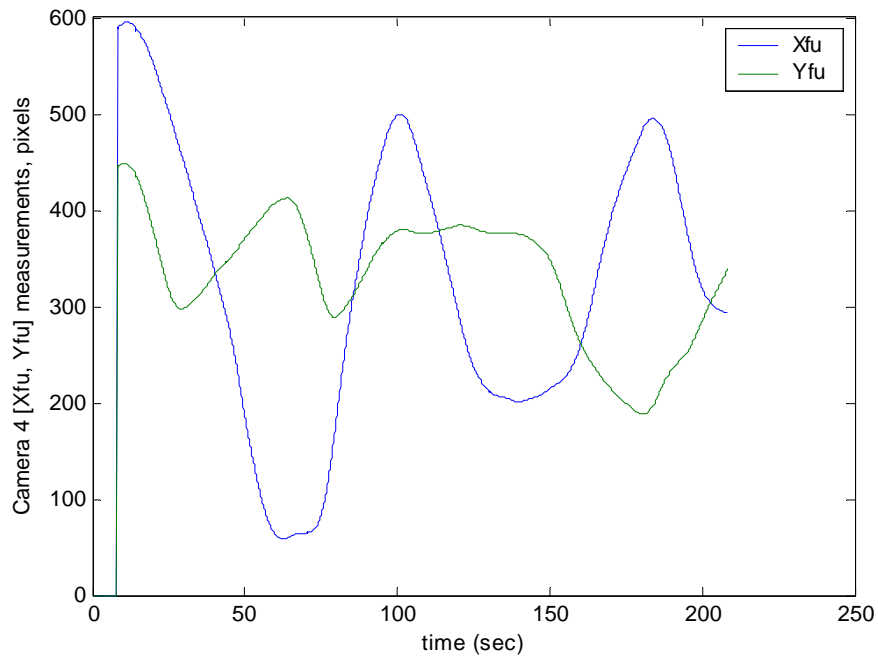


Figure 7.21 Camera 4 pixel measurements

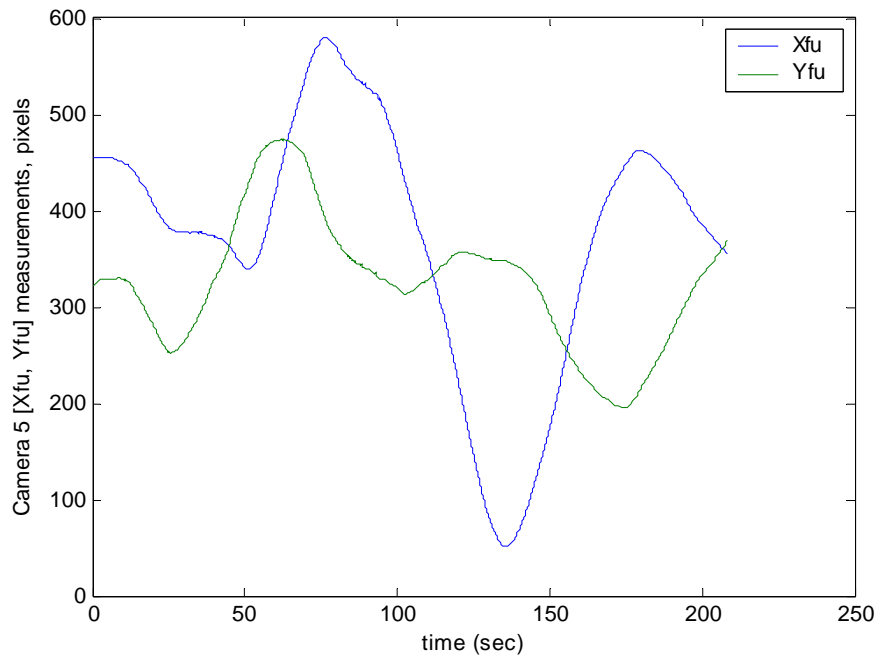


Figure 7.22 Camera 5 pixel measurements

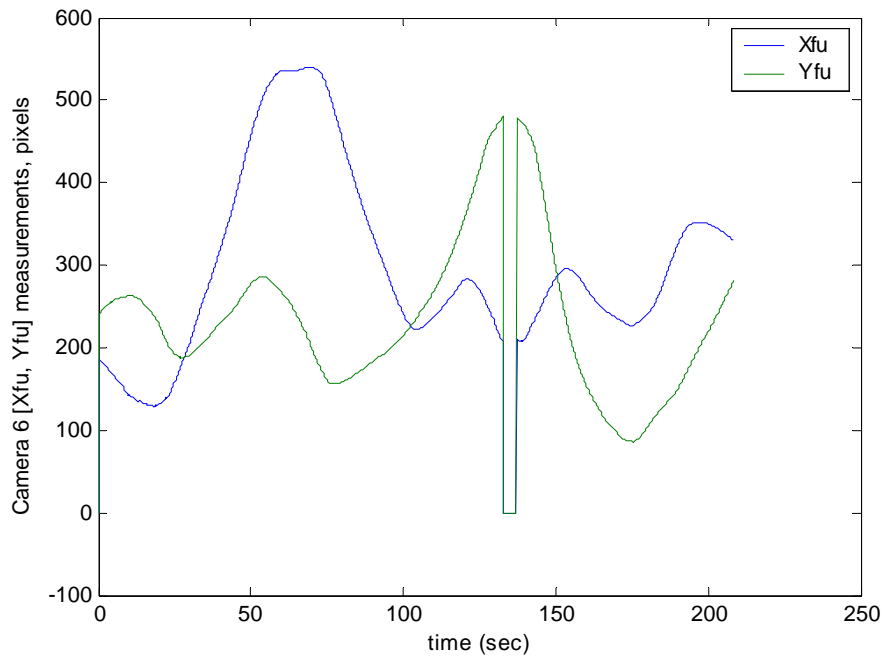


Figure 7.23 Camera 6 pixel measurements

The sections in Figures 7.18 to 7.23 where the plots of both  $X_{FU}$  and  $Y_{FU}$  drop suddenly to a value of  $-1$  occur when part of SCAMP SSV is about to leave the camera's FOV, and correspond to the limits  $(0 + R_{SSV}) < X_{FU} < (640 - R_{SSV})$  and  $(0 + R_{SSV}) < Y_{FU} < (480 - R_{SSV})$ , where  $R_{SSV}$  is the radius of SCAMP SSV, in pixels, predicted based on its current position estimate.

At approximately 10 seconds, there is a spike in the error bounds. This spike is caused by a combination of things. The data log, recorded by the EKF, indicates that at this time, the measurements from cameras 3 and 4 were not used, due to the proximity of SCAMP SSV to the edge of the FOV. Figures 7.20 and 7.21 verify this, as the  $Y_{FU}$  values at that time are approaching the limit of 480 pixels. This reduces the number of useful cameras from six to four. Since the cameras sample at approximately 8 Hz in an unsynchronized manner, and the EKF runs at 10 Hz, the EKF will have a varying number of measurements available at each iteration. Even if SCAMP SSV is in the FOV of all six cameras, at each iteration of the EKF there can be as few as one or two camera measurements available (even zero is theoretically possible, but statistically unlikely), and as many as six. The spikes in the error bounds are caused when the number of available camera measurements is further reduced from four to one and two cameras per iteration by this camera sampling/EKF frequency mismatch. Finally, also at this time, camera 1 suffers from camera noise (visible in Figure 7.18) of roughly 30 pixels, which far exceeds the predicted  $3\sigma$  measurement noise of  $3\sqrt{3}$  pixels. Since the EKF trusts the characterization of the noise statistics established in  $R$ , the



measurement noise covariance matrix, it accepts this noisy data as accurate.

This causes a large discrepancy between  $\hat{\mathbf{x}}_{k(-)}$  (the state estimate after propagation, but before any corrections) and  $\hat{\mathbf{x}}_{k(+)}$  (the state estimate after one or more corrections). The variance added by  $\mathbf{Q}_k$  in the propagation at each time step is not decreased by the application of multiple measurements, which causes the increase in the size of  $P_k$  seen in Figure 7.18.

The expansion of the error bounds between 135 and 155 seconds is likewise caused by a reduction in the number cameras providing useful data. SCAMP SSV is outside the (usable) FOV of camera 1 from 130 to 152 seconds, camera 3 from 132 to 149 seconds, and camera 6 from 133 to 137 seconds, as seen in Figures 7.18, 7.20, and 7.23, respectively. The EKF log indicates that the multiple spikes in the error bounds between 130 and 155 seconds again occur when the number of usable cameras is further reduced by the sampling/EKF iteration frequency mismatch.

A concern that arises when looking at error bounds shown in Figure 7.15 is that the error bounds do not increase in some areas of the plot where the state estimate exhibits noise, such as between 57 and 75 seconds, and between 100 and 110 seconds. It is expected that where the measurements contain more noise, the difference between the propagated state and the updated state would increase, causing an increase in the error bounds. It can be seen from Figures

7.15, and 7.18 to 7.23, that the noisy estimate sections correspond to noisy camera measurements, but why do the error bounds not grow accordingly?

This is the case because while the noise from some of the cameras at these locations in the plot exceeded that which was modeled in  $R$ , the EKF had access to a sufficient number of measurements to reduce the size of  $P_k$ . This means it increased its confidence in the estimate, but the confidence was ill founded, since it was based in part on measurements that were noisier than the EKF expected possible.

The spike in the X axis position estimate that occurs at 57 seconds is caused by the noise in the camera 3 measurement data, which is on the order of 50 pixels. The EKF accepts this extremely noisy data as being true within the statistics represented by  $R$ , and thus changes the state estimate according to the measurement. The other less noisy camera measurements further update the state estimate to be closer to reality, and reduce the magnitude of  $P_k$ , but the damage done by the noisy measurement is retained until the next EKF iteration. This indicates that further testing needs to be done to tune the values of  $R$  and  $Q$ , in order to find values that more accurately reflect the true noise statistics of the system. This also indicated that the algorithms that reject noisy data need to be improved. The error bounds should rise when data is noisy, or indicate falsely that the vehicle velocities are very large. This means that in general,  $R_k$  needs to be increased at this time, and  $Q_k$  decreased. This will place more confidence in

the dynamic model than it now receives, and less in the measurements.

Measurement rejection strategies can also be pursued to reject bad data. This work will be discussed in Chapter 8.

There are three likely sources of the error that caused the noisy estimates seen above. The first was mentioned in Chapter 6, and can be found by looking at the measurements themselves, and the regions of the images to which they correspond. In Figure 7.17 it can be seen that camera 1 provides noisy measurements approximately 15 seconds into the test. These measurements are in the region of  $X_{FU} = 260$  pixels, and  $Y_{FU} = 310$  pixels. Figure 7.24 is the background image for camera 1 used in this test. The position [260 310] is marked in the image with a white cross.

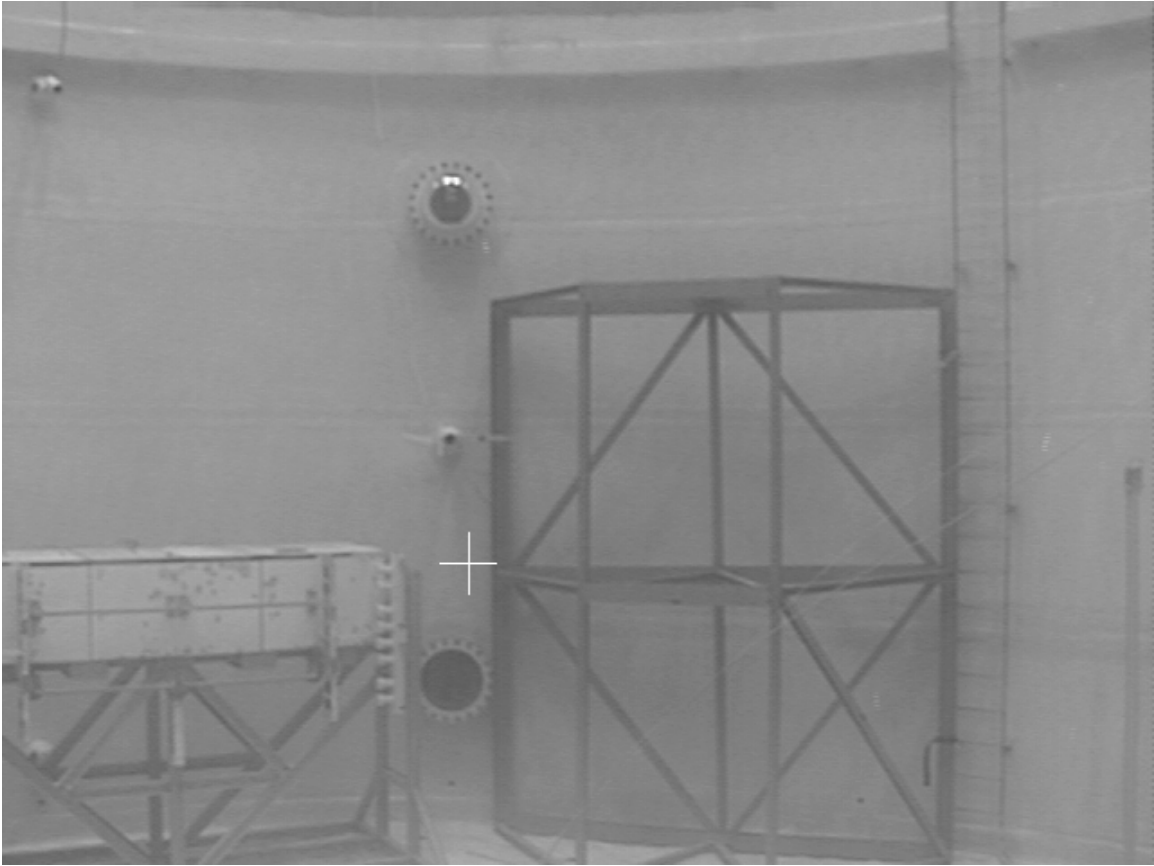


Figure 7.24 Background image for camera 1, coordinates [260 320] indicated

At the image coordinates [260 320], part of SCAMP SSV lies in front of the vertical member of truss. With SCAMP SSV in this location, part of its area will be removed from a measurement image along with the dark background object when the image is thresholded. This will not occur consistently, due to slight variations in perceived robot darkness caused by its changing attitude, and by fluctuations in illumination.

Another potential source of noise is caused directly by changes in the attitude of SCAMP SSV. Figure 7.25 shows an image taken from camera 2, during measurement noise characterization. SCAMP SSV just happens to be at such an attitude and location as to cause a glare off of some of its panels. This greatly affects its image area after thresholding, as is seen in Figure 7.26. Although it is possible that this effect caused some of the measurement noise observed in this test, images were not saved during the test so it cannot be known for certain.

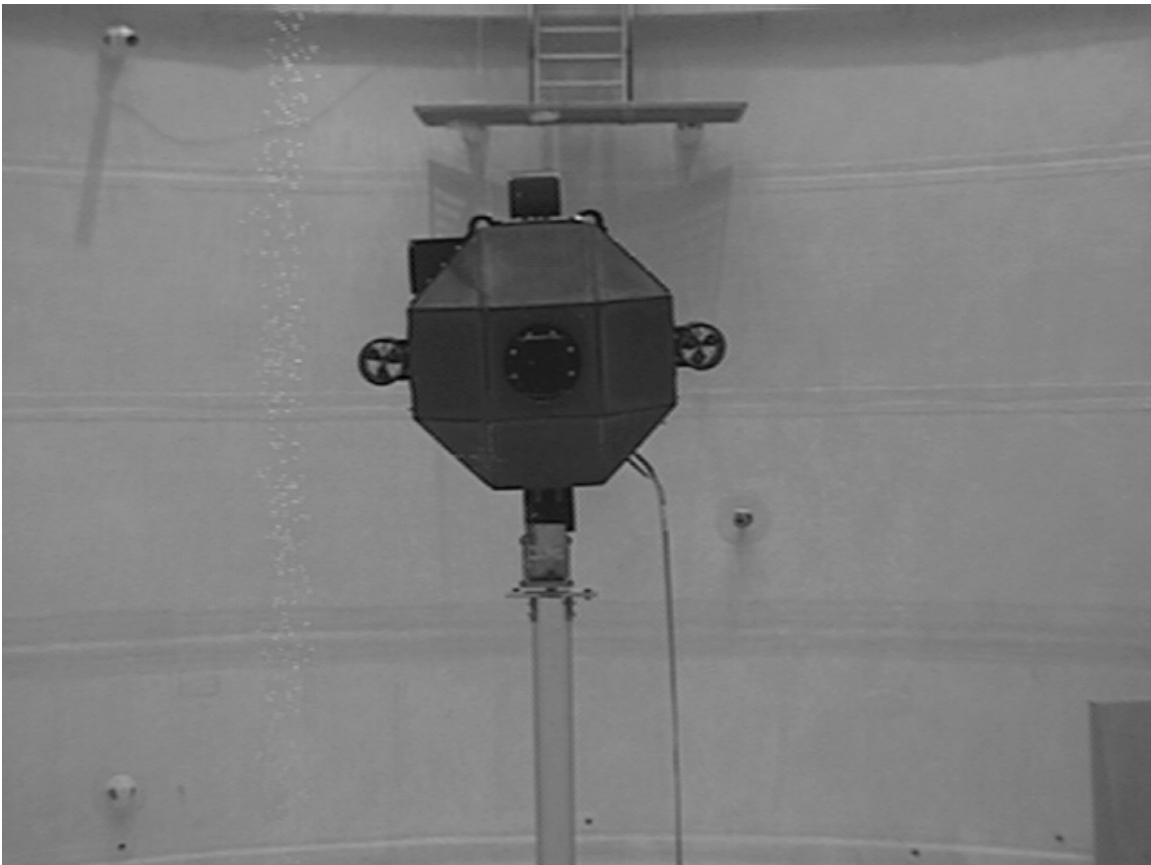


Figure 7.25 SCAMP SSV with glare on top panels, raw image

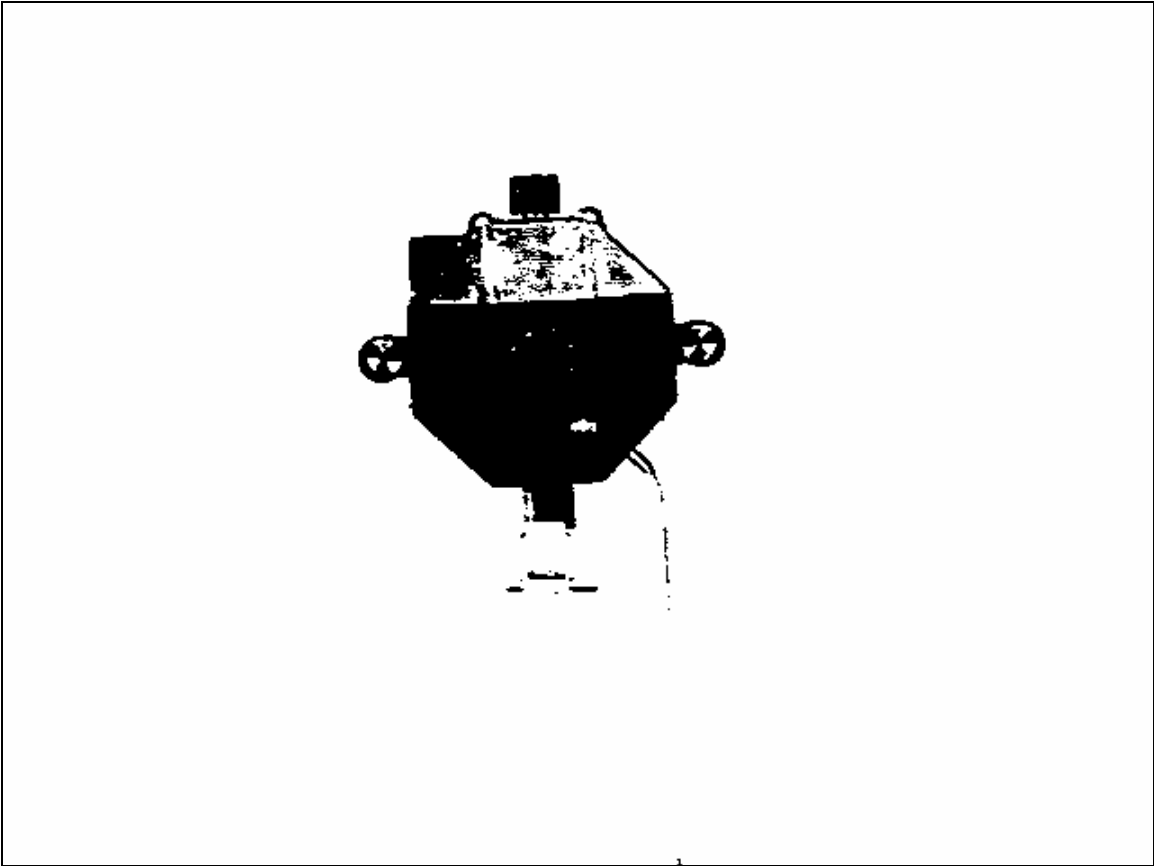


Figure 7.26 SCAMP SSV with glare on top panels, thresholded image

It is possible to apply a vision algorithm that would fill in the gaps in the image of SCAMP SSV. This technique is known as convex hulling, and is discussed briefly in Chapter 8.

The final potential source of error is glare from direct sunlight, entering through skylights and windows in the building that houses the NBRF. This glare can appear during a test, either on the NBRF wall or the vehicle itself, fluctuate significantly, and disappear in a matter of seconds, and thus is hard to until spot aside from detailed post processing of vision data. It can also appear in a

background image, and then be absent for the duration of a test, which will severely affect the camera that took the background image. Again, because images were not saved to disk during this test, this source of noise cannot be pointed to with certainty, but it can also not be ruled out. An example of this glare was captured during a previous test, and is shown below in Figure 7.27.

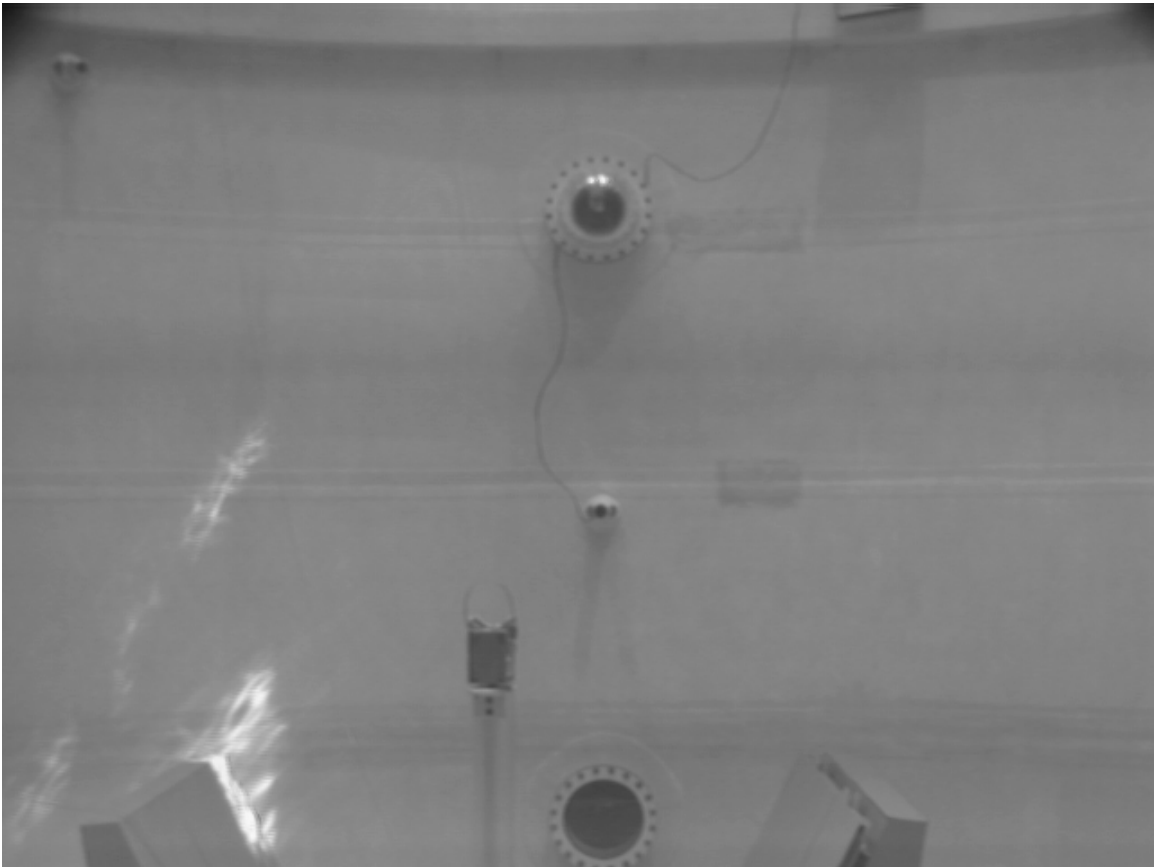


Figure 7.27 Example of glare from direct sunlight

## 7.4 Dynamic SCAMP SSV State Estimation, Double EKF

### 7.4.1 Non-Ideal Test Conditions

Next, the double EKF architecture was used to create two independent estimates of the position and velocity of SCAMP SSV while in free flight. Since each EKF uses at most three cameras to update its estimate in this test, the estimates are, as expected, noisier than when all six cameras are used. Figure 7.28 shows the 3D trajectory of SCAMP SSV as measured by EKF1, and Figure 7.29 shows the 3D trajectory as measured by EKF2.

Figures 7.30 to 7.32 show the X, Y and Z axis position estimates calculated by both EKF1 and EKF2.

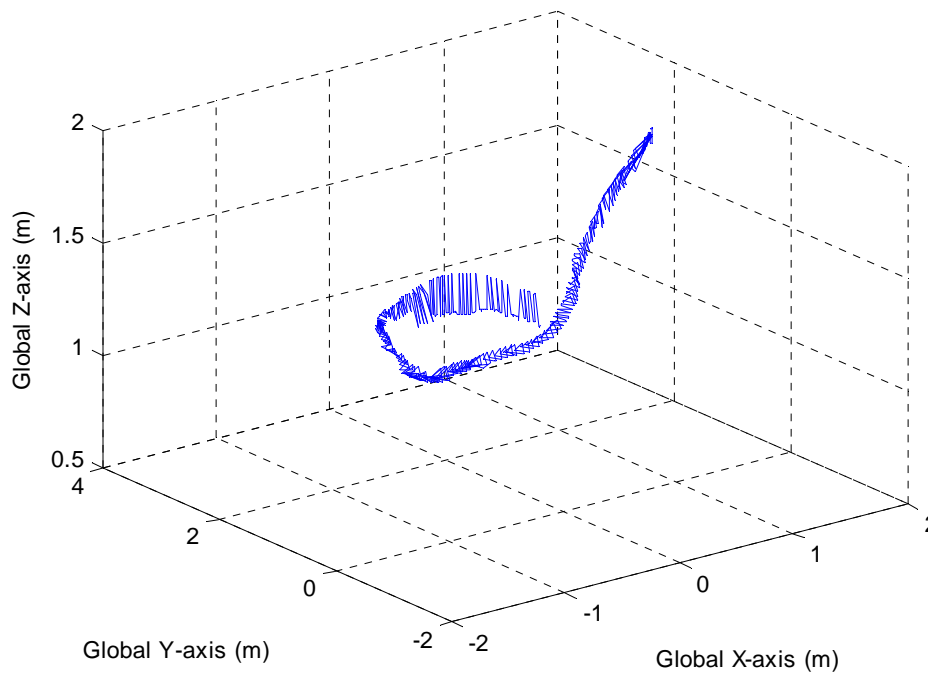


Figure 7.28 Trajectory of SCAMP SSV in 3D, measured by EKF1



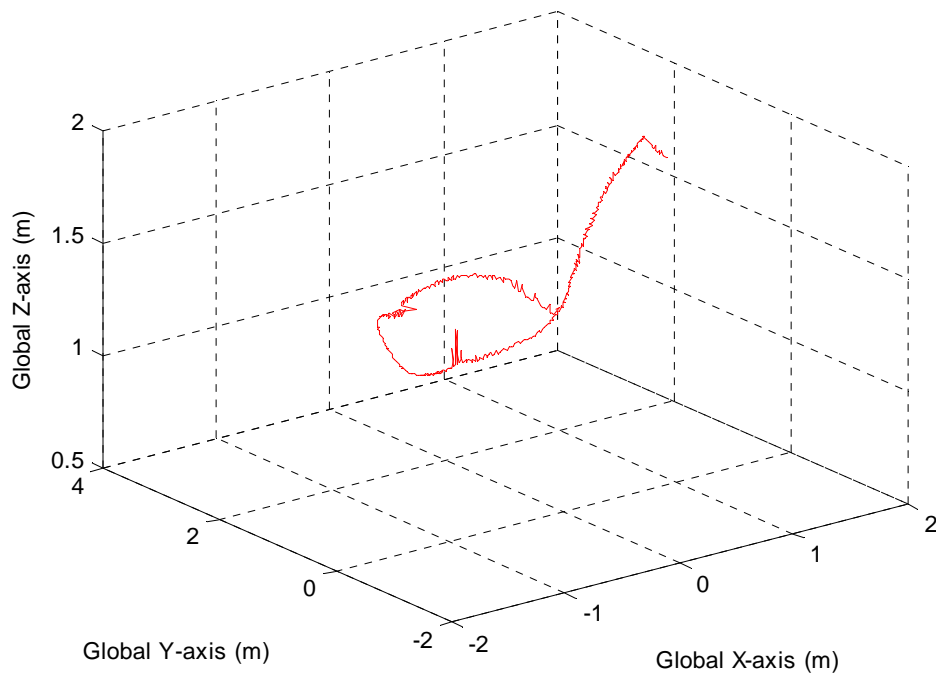


Figure 7.29 Trajectory of SCAMP SSV in 3D, measured by EKF2

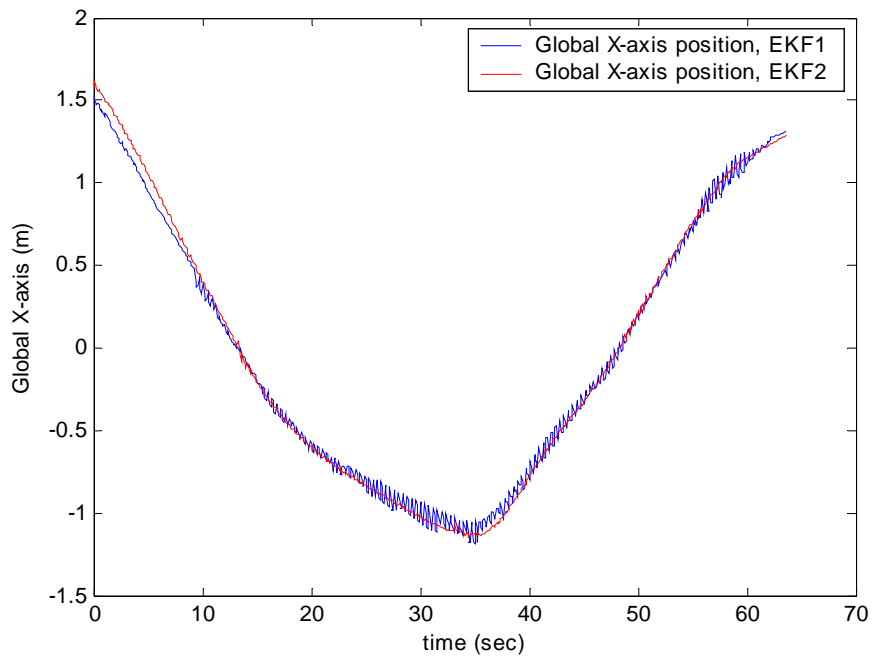


Figure 7.30 SCAMP SSV global X axis position vs. time, EKF1 and EKF2

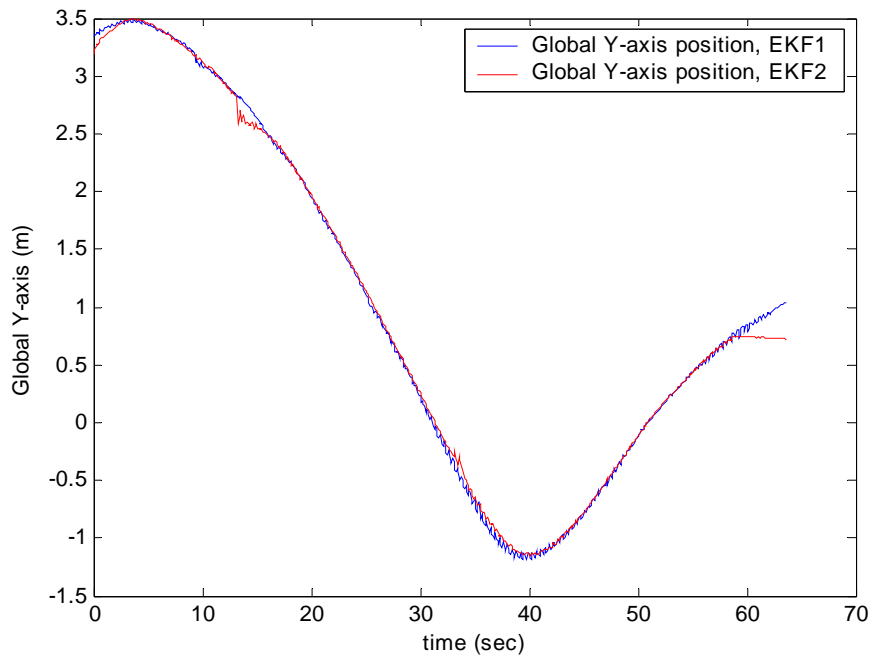


Figure 7.31 SCAMP SSV global Y axis position vs. time, EKF1 and EKF2

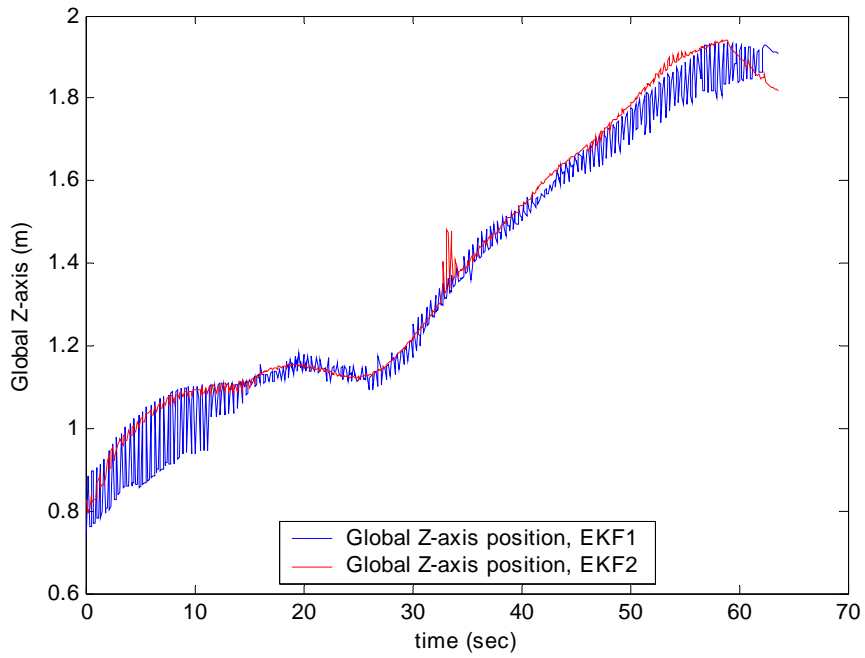


Figure 7.32 SCAMP SSV global Z axis position vs. time, EKF1 and EKF2

In the X and Y axes, the two EKF position estimates agree to within 3 cm for most of the test, and to within 10 cm for the duration of the test. The Z axis estimates vary by as much as 20 cm for portions of the test, for example from 0 to approximately 13 seconds. These regions of large discrepancies can all be traced in the EKF log files to times when data from only either one camera or no cameras was available to the EKF for updating the state estimate. The regions where two and three cameras were available for use are those where the two independent state estimate consistently agree to within 3 cm or less. Evidence of measurement noise, such as that discussed above, is seen in the plots as well, for instance at 35 seconds in Figure 7.32 and at 12 seconds in Figure 7.31.

The error bound plots for the X, Y and Z axis position and velocity estimates calculated by EKF1 are shown in Figures 7.33 to 7.38. The error bounds are, as expected, larger than those seen in the test conducted with all six cameras providing measurements to one EKF. They are small (in the position plots) only when the individual EKFs get to use two or more measurements for updating the state estimate.

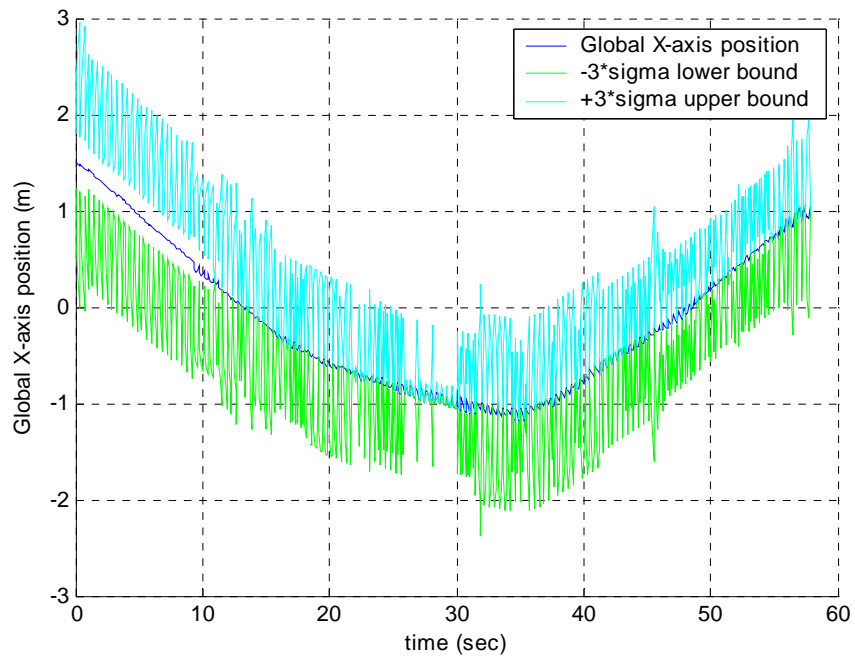


Figure 7.33 SCAMP SSV global X axis position vs. time, EKF1, with  $\pm 3\sigma$  error bounds

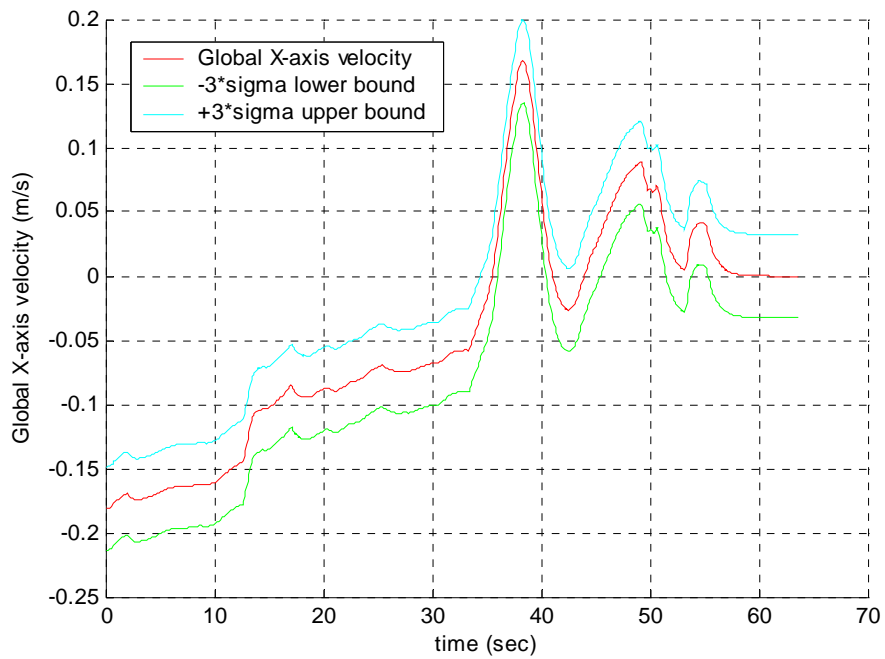


Figure 7.34 SCAMP SSV global X axis velocity vs. time, EKF1, with  $\pm 3\sigma$  error bounds

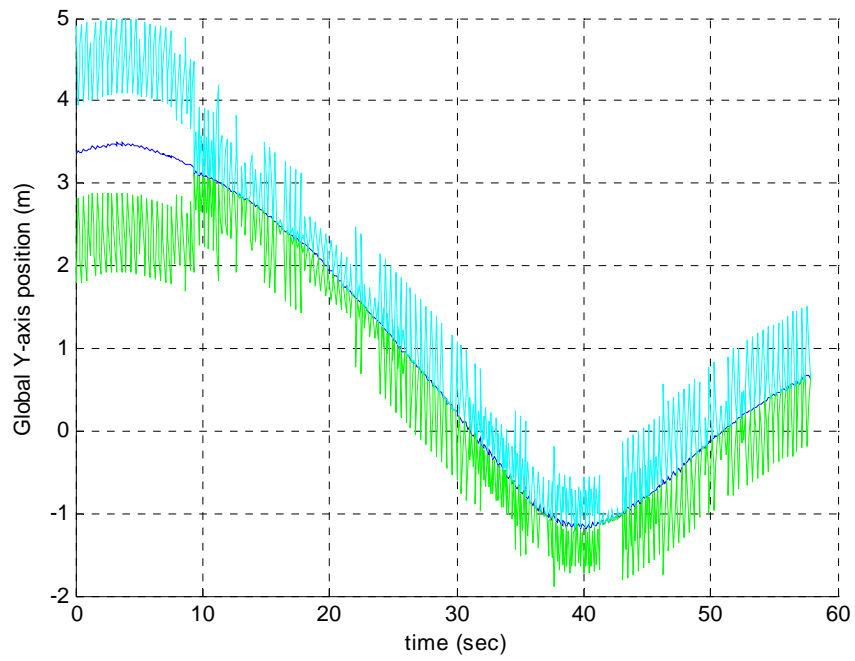


Figure 7.35 SCAMP SSV global Y axis position vs. time, EKF1, with  $\pm 3\sigma$  error bounds

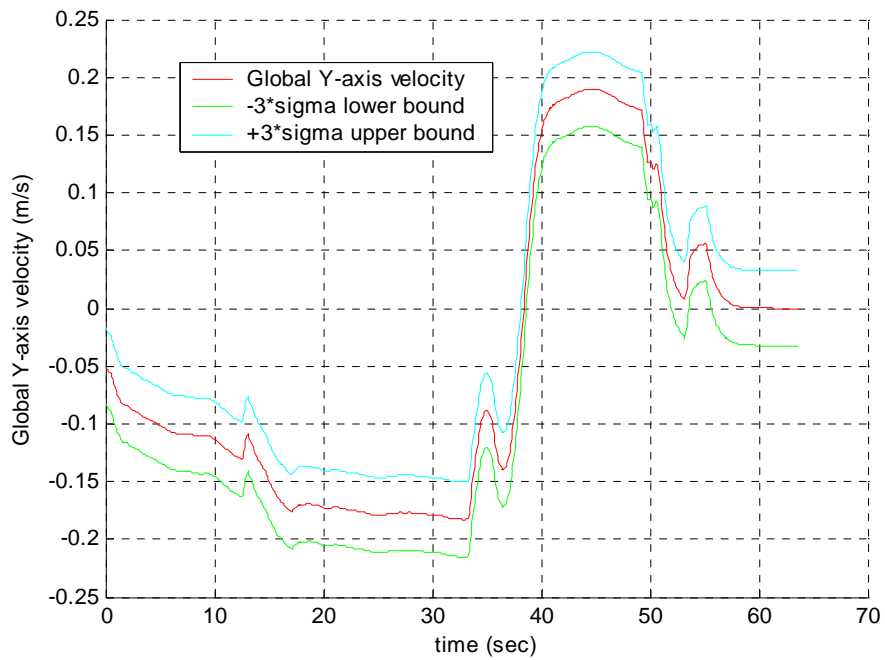


Figure 7.36 SCAMP SSV global Y axis velocity vs. time, EKF1, with  $\pm 3\sigma$  error bounds

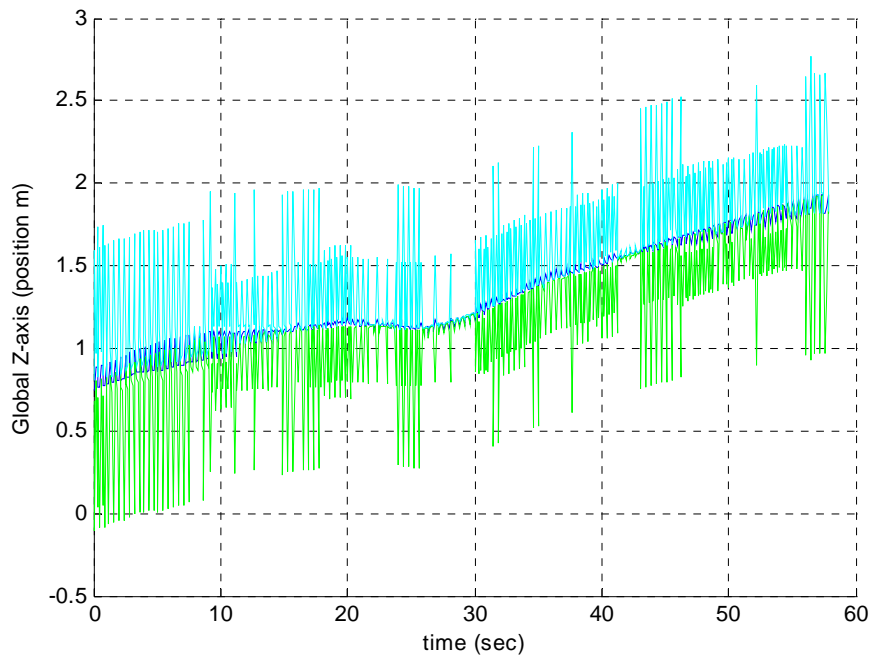


Figure 7.37 SCAMP SSV global Z axis position vs. time, EKF1, with  $\pm 3\sigma$  error bounds

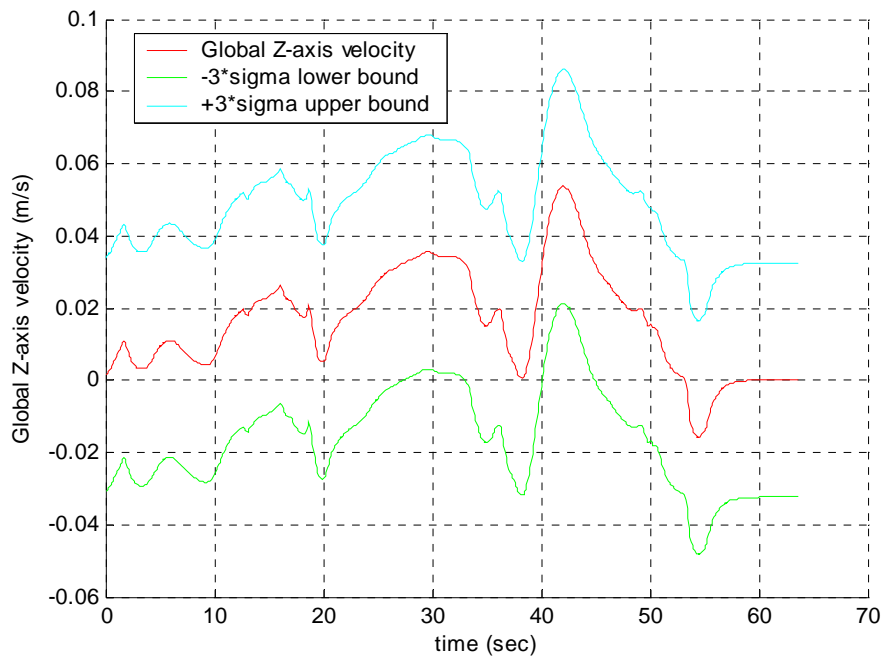


Figure 7.38 SCAMP SSV global Z axis velocity vs. time, EKF1, with  $\pm 3\sigma$  error bounds

The position plots computed by EKF2 are shown in figures 7.39 to 7.41, but due to an error in the EKF2 software, the velocity estimates were not recorded during this test and are thus unavailable for plotting. Since, like EKF1, EKF2 also had use of only three VPS cameras, at many iterations of the EKF it had between zero and two measurements with which to update the estimate. As a result, the error bounds on the state estimate are, as with EKF1 data, larger than those seen in the test where all 6 cameras were linked to a single EKF. There are, however, more periods in the EKF2 plots where the error bounds are small than in the EKF1 plots. This is due to the fact that one or more of the EKF1 cameras suffered from high amounts of noise in this test. In other words, EKF2 suffered mainly from the low number of measurements at each EKF iteration, while EKF1 suffered from this as well as noisy camera data, and is thus worse, in terms of error bounds, than EKF2. Recall that EKF1 has camera 1 and camera 5, both of which suffer from significant dark objects in their backgrounds, increasing the noise of their measurements. It is only natural for EKF1 to be noisier than EKF2 for this reason.

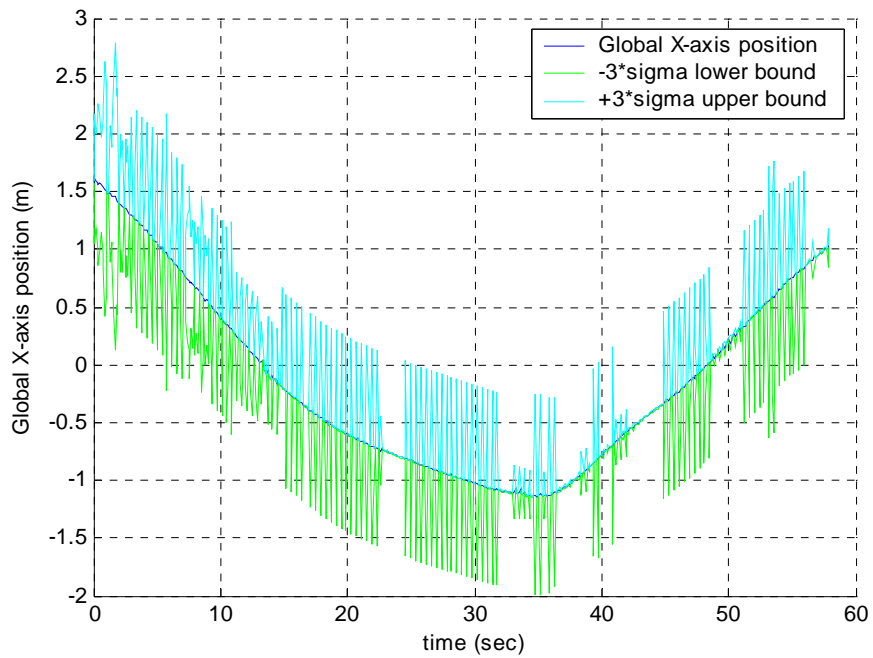


Figure 7.39 SCAMP SSV global X axis position vs. time, EKF2, with  $\pm 3\sigma$  error bounds

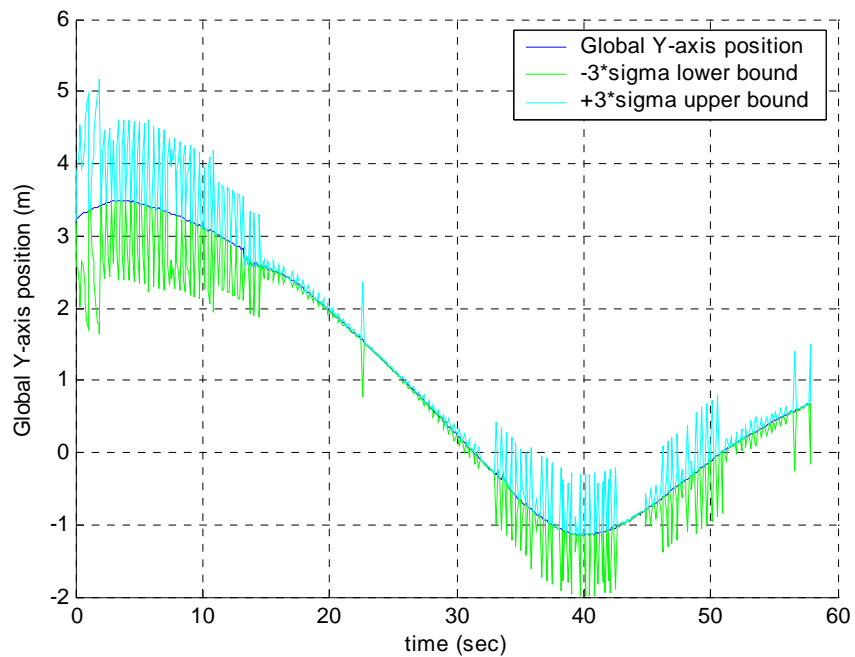


Figure 7.40 SCAMP SSV global Y axis position vs. time, EKF2, with  $\pm 3\sigma$  error bounds



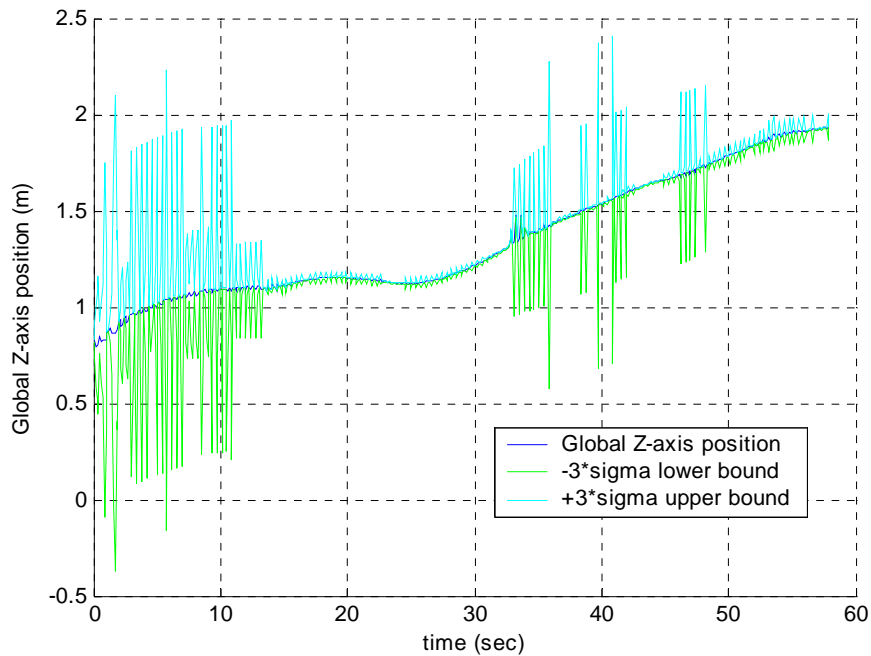


Figure 7.41 SCAMP SSV global Z axis position vs. time, EKF2, with  $\pm 3\sigma$  error bounds

These results indicate that in order to reduce the error bounds to reasonable levels (on the order of a few centimeters), three or more measurements at each EKF iteration are desirable. They also indicate that the first predicted measurement, which is based on the state estimate after propagation forward one time step, is significantly different from the first received measurement. The error covariance matrix becomes small during an EKF iteration only when the state can be updated by multiple measurements (three or more) in a single iteration. This illustrates that the accuracy with which the dynamics of SCAMP SSV are modeled needs to be improved.

## 7.4.2 Near-Ideal Test Conditions

Additional testing was conducted to better characterize the capabilities of VPS under near-ideal testing conditions, meaning with nearly constant ambient lighting and little direct sunshine. This was accomplished on a cloudy day, and the positioning results are presented below.

Figure 7.42 shows the global X axis position and velocity plots for the 576 second free-flight trajectory executed by SCAMP SSV. The Y and Z axis plots have similar performance. State estimation was performed using data from all six cameras.

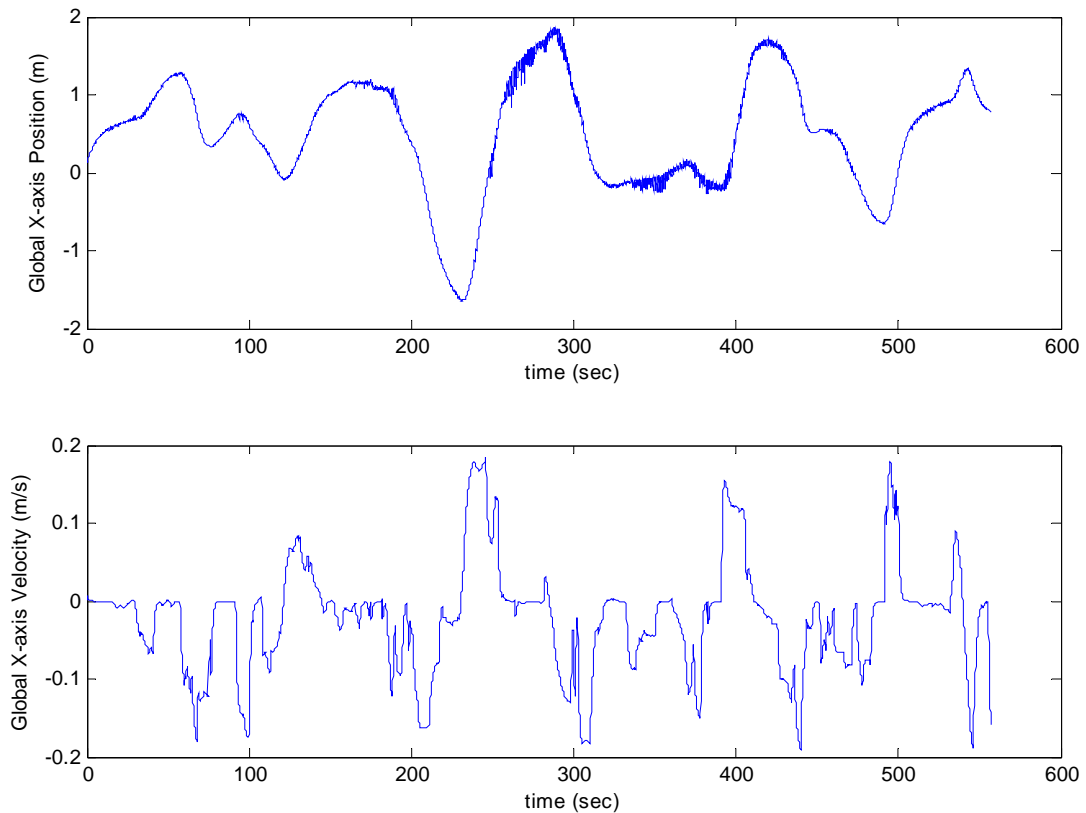


Figure 7.42 SCAMP SSV global X axis position (top) and velocity (bottom) vs. time

To compare the accuracy of data sets coming from each camera, three plots were generated for each camera. The first plot compares the image area of SCAMP SSV ( $A$ ) measured by a camera with the predicted image area,  $\hat{A}$ , both of which are measured in pixels<sup>2</sup>. The second and third plots compare the measured  $X_{FU}$  and  $Y_{FU}$  centroid coordinates with the predicted centroid coordinates,  $\hat{X}_{FU}$  and  $\hat{Y}_{FU}$ , respectively, all of which are measured in pixels. These predicted area and centroid values were computed based on the global position estimates computed by the EKF using input data from all 6 cameras. The global position estimates were transformed into position estimates in each camera's local coordinate frame, and then converted to pixel values using the equations presented in Chapter 3. The predicted centroid coordinates were generated in a similar way to  $\hat{A}$ . These plots are shown in Figures 7.43 through 7.48.

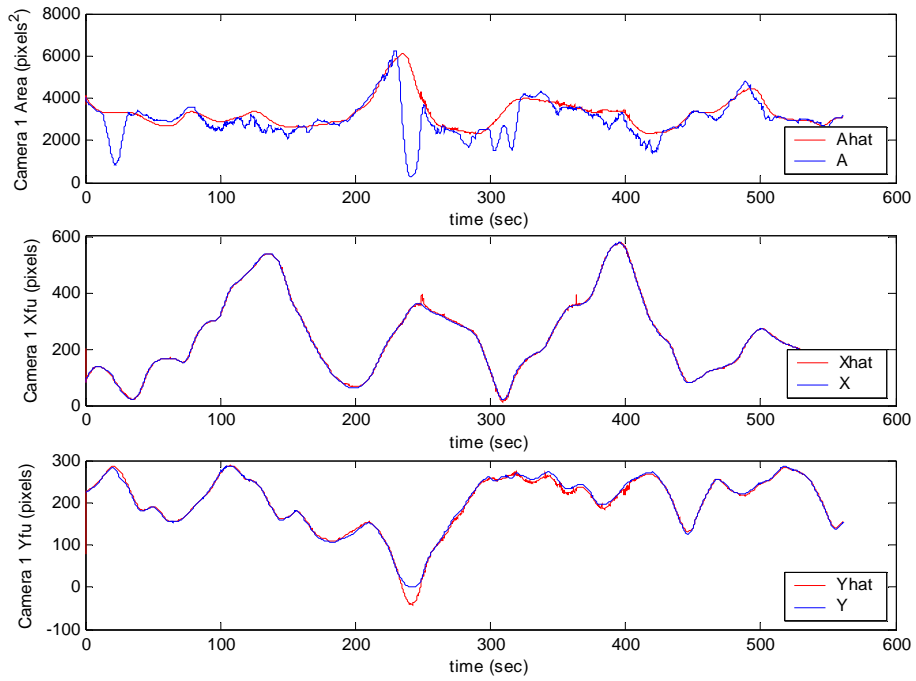


Figure 7.43 Camera 1 measured and predicted camera data

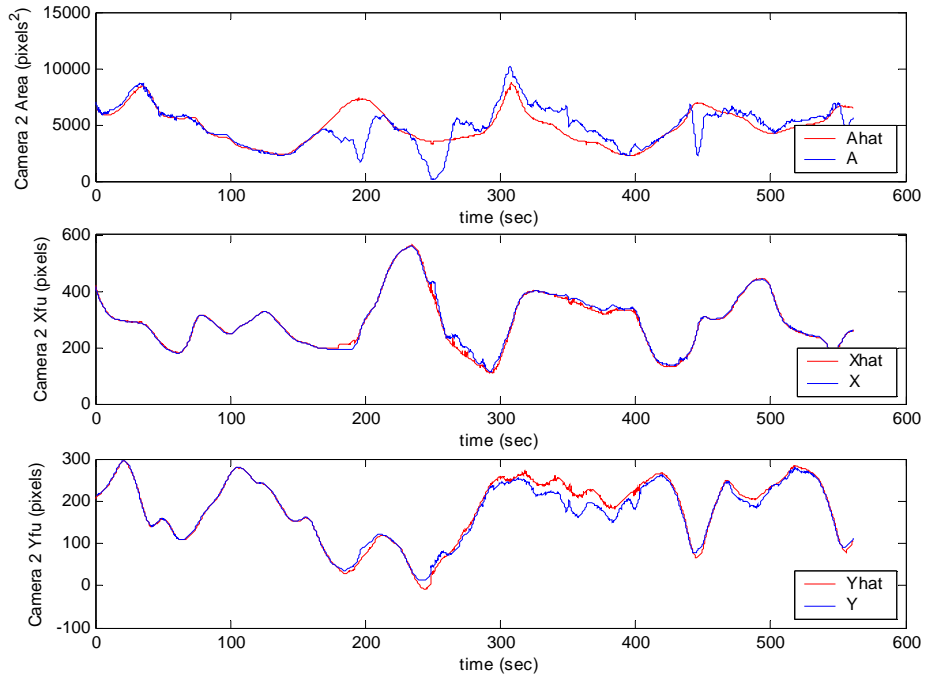


Figure 7.44 Camera 2 measured and predicted camera data

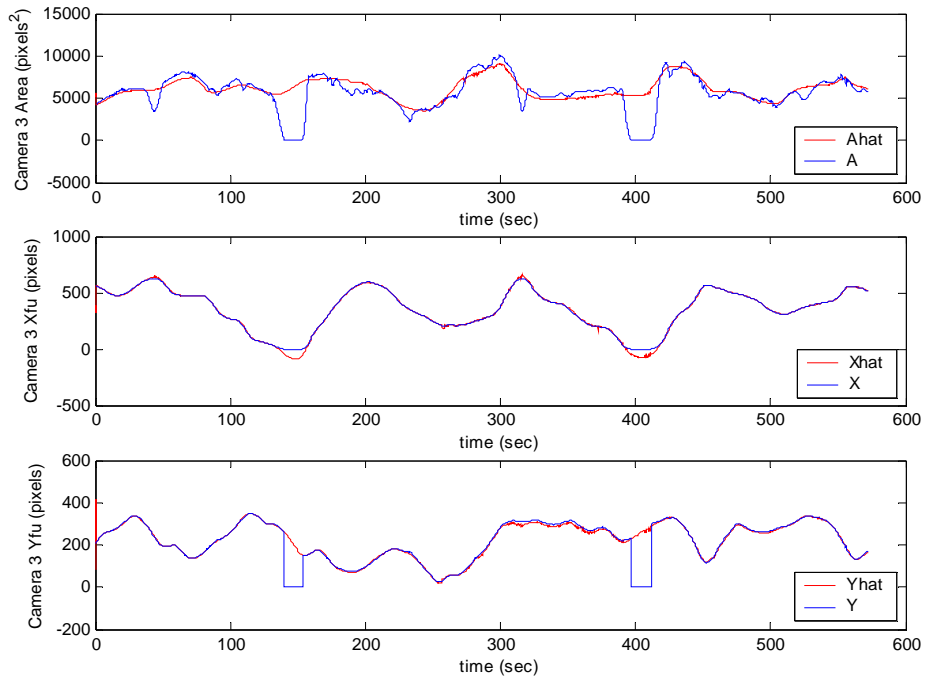


Figure 7.45 Camera 3 measured and predicted camera data

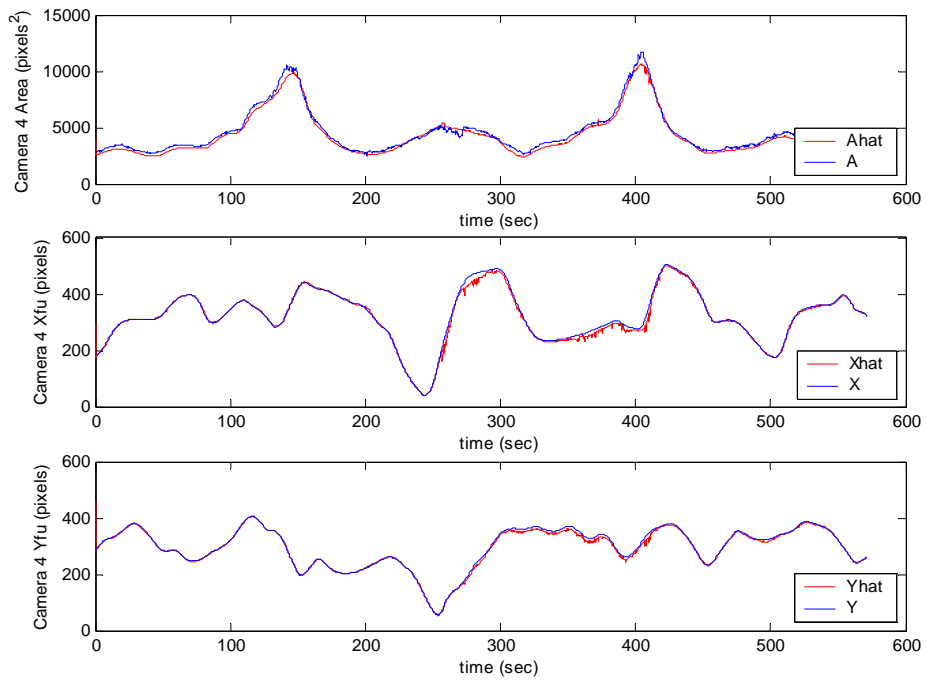


Figure 7.46 Camera 4 measured and predicted camera data

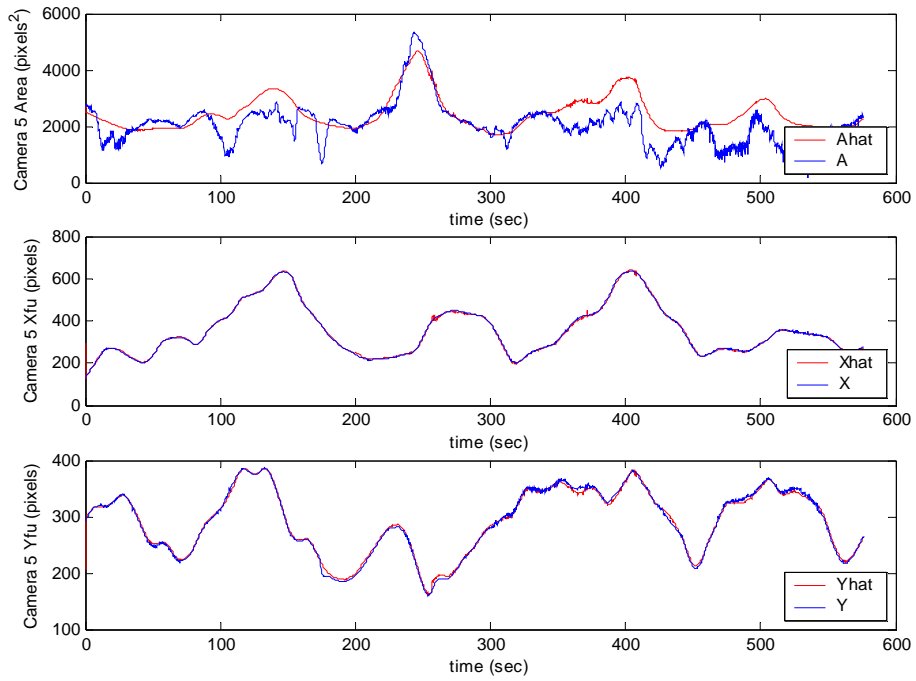


Figure 7.47 Camera 5 measured and predicted camera data

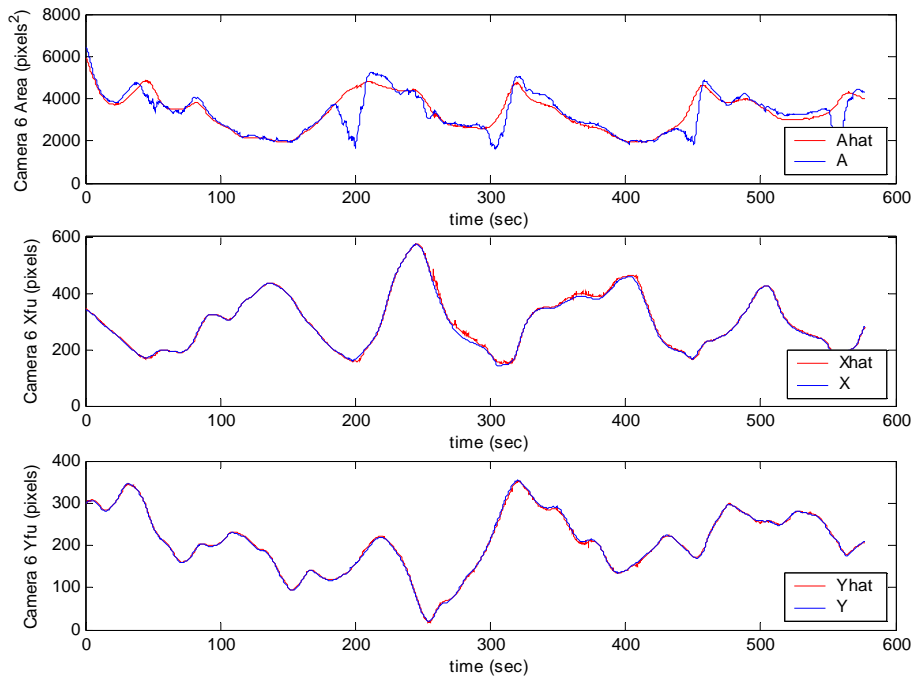


Figure 7.48 Camera 6 measured and predicted camera data

These plots clearly show why the use of object area in performing accurate vision tasks is best avoided. Note that in all of the plots, the difference between the predicted and measured area plots is much greater than the predicted and measured centroid plots, and the measured area plots often take very large excursions away from the predicted data. The measured centroid data generally matches the predicted data well, with some notable exceptions.

The most common exception is where SCAMP SSV nears the edge of the image plane. In this case, e.g. Figure 7.43 (camera 1) at t=240 seconds, as part of SSV leaves the image plane, the area appears to decrease and the centroid appears to remain further inside the image plane than would otherwise be the case. This situation has the potential to reduce EKF accuracy, since this data is not yet successfully rejected by the software.

Another notable difference between the predicted and measured centroid is when SCAMP SSV actually leaves the image plane entirely, as is the case twice in Figure 7.45 (camera 3). Although this appears to be a dramatic error, in reality this will not degrade EKF accuracy, because an area value of -1 is set whenever the vehicle is designated as outside the image plane in order to prevent this camera's data from being used.

Only one other centroid discrepancy type is present – in Figure 7.44 (camera 2) an offset is seen between the measured and predicted X-centroid value between

times 290-400 seconds. Based on the data and locations of other objects in the tank, this phenomenon appears when SCAMP SSV flies partially in front of a similarly-colored object (e.g., part of a space telescope mockup that was in the tank during this test). Part of the vehicle is indistinguishable from the background, making it appear smaller (area too low) and offset (in  $X_C$ ) from its true position.

Table 7.10 summarizes the average and standard deviation of differences between the measured and predicted camera data shown in Figures 7.43 through 7.48. On average, each camera experienced between 2-4 pixels of noise, with camera 2 noise appearing somewhat higher because of the structural interference apparent in Figure 7.44. The average difference for the area values is much higher, and ranges from 267 to 773 pixels<sup>2</sup>. Note that at the center of the tank, 1 pixel is equal to approximately 8 mm, the exact value of which depends on camera focal length and distance from the camera to the center of the tank.



Table 7.10 Deviation between predicted and measured camera data

	Area difference (pixels <sup>2</sup> )		$X_{FUc}$ difference (pixels)		$Y_{FU}$ difference (pixels)	
	average	$\sigma$	average	$\sigma$	average	$\sigma$
Camera 1	301.9	396.4	2.38	2.49	3.29	2.9
Camera 2	773.2	977.3	5.45	7.64	9.09	10.9
Camera 3	511	447.5	2.64	3.53	3.58	3.62
Camera 4	267.1	157.1	4.3	5.79	3.53	3.61
Camera 5	500.9	431.7	2.42	2.15	3.05	2.3
Camera 6	300.1	460.1	3.81	4.28	1.98	1.61

A VPS simulator was implemented in order to test the effect of removing camera data sets from the state estimation sessions. The simulator had two components. The first component sent camera data, recorded during a live SCAMP SSV flight test, to the VPS EKF at appropriate times based on timestamps associated with the camera data. The second component sent the EKF the SCAMP SSV attitude and thrust data that corresponded to the camera measurements being sent to the EKF by the first component. This allowed the EKF to “think” it was communicating normally with the cameras and SCAMP SSV when it was actually participating in post-processing simulations.

By preventing certain camera data from being sent to the EKF, different camera configurations could be tested for the same flight test. This is similar to the double EKF configuration testing presented earlier, but allows many camera configurations to be tested serially without placing the robot in the water or changing the EKF code. The state estimate data computed using all six cameras

was compared to what the state estimate data would have been for three unique 2-camera configurations. This was done by running the simulator three times, each time suppressing data from a different set of four of the six cameras. The EKF was run with camera data from cameras 1 and 2, cameras 3 and 3, and cameras 5 and 6. The global X, Y, and Z position of SCAMP SSV vs. time, as computed by the four independent camera/EKF configurations (the above three configurations plus the estimate using all 6 cameras) are shown in Figures 7.49 through 7.51.

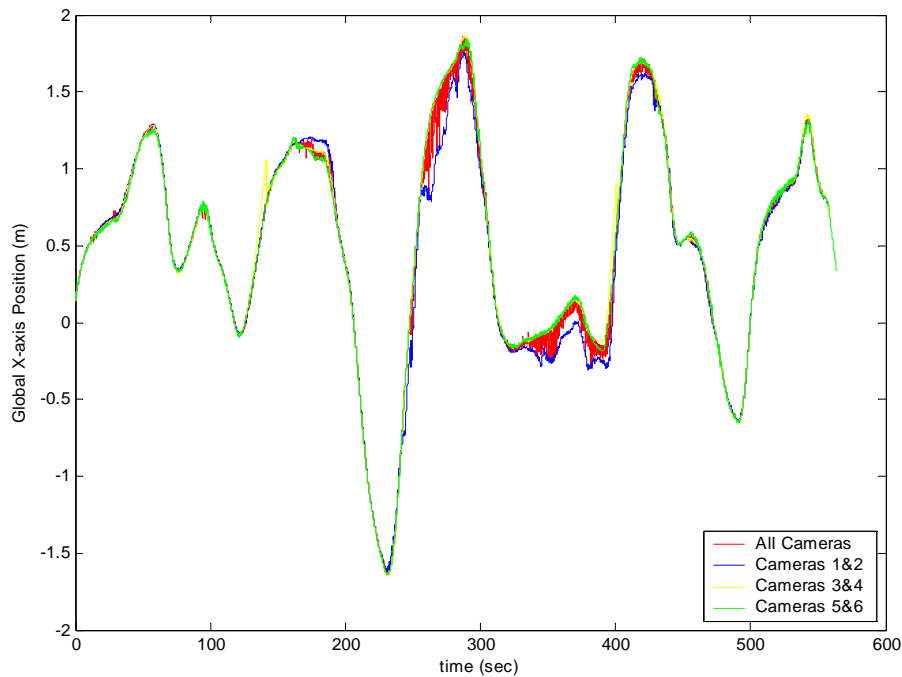


Figure 7.49 SCAMP SSV global X axis position vs. time, as computed by four EKF/camera configurations

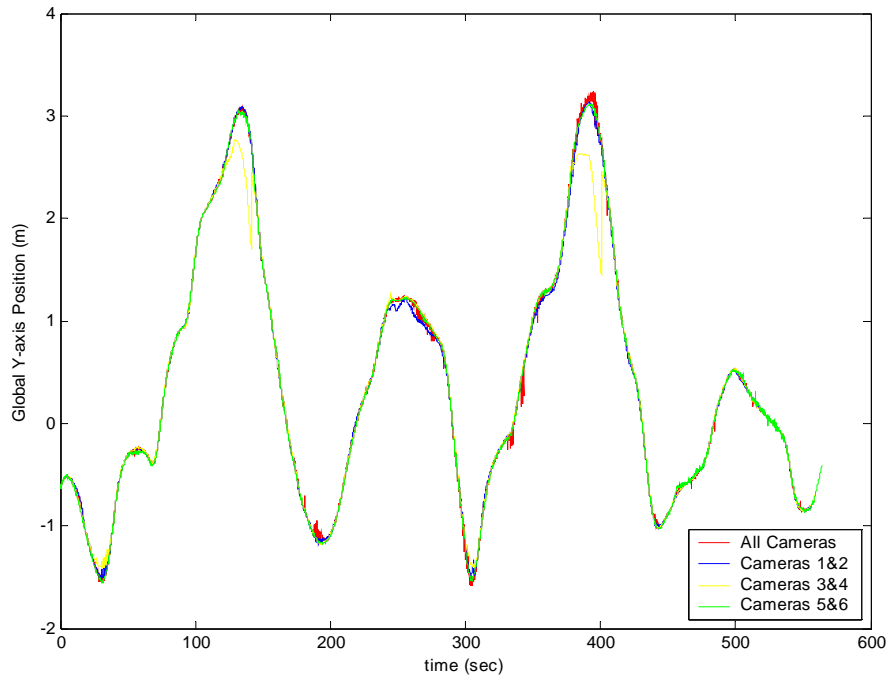


Figure 7.50 SCAMP SSV global Y axis position vs. time, as computed by four EKF/camera configurations

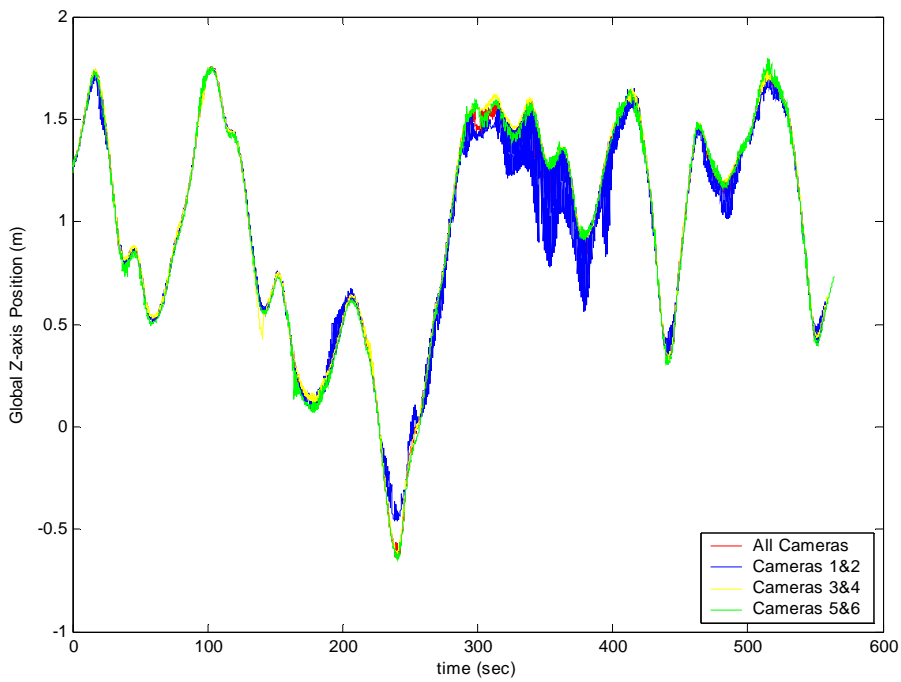


Figure 7.51 SCAMP SSV global Z axis position vs. time, as computed by four EKF/camera configurations

All of the position results are in agreement, in general, as can be seen from the plots above. The noisy camera 1-2 EKF result between times 240-300 seconds illustrates how background structures, etc. can degrade accuracy, particularly with a low level of redundancy (thus camera 2's minimal effect on the overall 6-camera state estimate). Note also the noise in the camera 3-4 EKF estimate around time 120 seconds and 400 seconds. This "drift" results from the fact that the vehicle had traveled out of the camera 3 image plane (see Figure 7.45), and camera 4 provides no  $Y_G$  data in its centroid estimate.

The X, Y, and Z velocity plots for the four unique simulated camera/EKF tests matched identically, and were thus not shown. This is likely because the  $Q_k$  elements corresponding to velocity are very small in comparison to the  $R_k$  values. This potentially caused the propagated velocity values to dominate any corrections that were made to the velocities due to the measurements. There may also be an error in the program that either generates or records the velocity elements of the state incorrectly under simulation.

Table 7.11 summarizes the differences between the EKF positional state estimates based on data from all six cameras and the estimates computed by the three independent camera/EKF configurations (camera 1-2, camera 3-4, and camera 5-6). The results in Table 7.11 include all data points, even those for which camera 2 provided erroneous data and camera 3 provided no data, and is thus a conservative indication of the agreement between the independent

camera/EKF configurations. Note that although the camera 2 errors were propagated to the overall EKF, the absence of camera 3 data that caused the discrepancies in the camera 3-4 EKF are an artifact of the pair-wise camera simulation that did not reduce accuracy in the 6-camera EKF estimates.

The camera 1-2 discrepancy accentuates the need to better reject erroneous measurements, as did previous results. However, the remaining results are very promising. First, it can be seen that with good data, the use of two cameras is actually sufficient to yield a state estimate accurate to the cm level, indicating that SCAMP SSV can be tracked almost anywhere in the tank (see coverage maps, Figures 2.3 through 2.7). Second, it can be seen that with highly-redundant measurements (from 6 total cameras in this case), even somewhat bad data (e.g., camera 2 between t=290-400 seconds) does not significantly degrade the six-camera overall state estimate. This is an important result, even with improved object detection / image rejection algorithms, because inevitably single bad data points will be used, and the real strength of the EKF is its ability to reject bad data through statistical averaging.

Table 7.11 Deviations between 2-camera EKF and 6-camera EKF simulations

EKF	Global x (m)		Global y(m)		Global z(m)	
	average	$\sigma$	average	$\sigma$	average	$\sigma$
CAM12	0.045	0.066	0.022	0.031	0.035	0.054
CAM34	0.032	0.052	0.054	0.158	0.025	0.03
CAM56	0.029	0.04	0.018	0.021	0.027	0.025

## Chapter 8

### Conclusion and Future Work

This thesis describes the design and implementation of a vision-based inertial navigation system, the Vision Positioning System (VPS), for a neutral buoyancy, space simulation robot. The three main research thrusts included: the development of an accurate and robust camera calibration technique, the implementation and characterization of an Extended Kalman Filter to combine vision and dynamic data in an optimal fashion, and the integration of hardware and software elements into a functioning system. As a result of this research, VPS is now capable of providing real-time translational state estimate information for a neutral buoyancy space simulation robot, SCAMP SSV, and through modification to the object recognition software, can be augmented to provide translational navigation data for any free-flying neutral buoyancy robot. With future additions to increase robustness to lighting changes and other noise sources, VPS provides a new and important tool to simulate close-proximity orbital robotic and spacecraft operations for a neutral buoyancy environment that will better characterize operations (target tracking) and ultimately enable autonomous vehicle operations.

## **8.1 Conclusion: VPS Calibration**

Perhaps the most significant contribution of this research was the development of procedures and equipment that allow an accurate, repeatable, and robust calibration of the VPS cameras. The accuracy of the intrinsic calibration parameters is what allows individual cameras to produce valid 2D data from which the crucial inference of 3D world information can be made. In addition to these, accurate extrinsic parameters are required to allow the combination of data from multiple cameras into one global estimate.

The two-step calibration procedure detailed in this thesis overcomes several challenges, including a relatively large calibration volume, maintaining cm-scale accuracy over long distances, inward-pointing cameras, and an underwater operating environment. This procedure provides as one of its outputs an estimate of the accuracy of four out of the five intrinsic parameters. Data on the quality of the overall calibration (intrinsic and extrinsic) of each camera is also provided by the two-step calibration procedure. According to these accuracy estimates, VPS errors near the center of the NBRF due solely to calibration errors, were not likely to exceed 1.0 cm, and in most cases were estimated to be less than 0.5 cm. This estimate of calibration accuracy was supported by the results of static and dynamic VPS data collection. For a static test, in the absence of significant camera noise due to glare, background objects, and FOV limitations, VPS was able to measure the magnitude of relative displacement of SCAMP SSV, from one position to another, to within 2 cm. In other static testing,

the position of a black spherical ball was measured simultaneously with two independent sets of VPS cameras. Again in the absence of significant camera noise, agreement between the two position estimates was typically within 2 or 3 cm for a wide variety of positions near the center of the camera FOVs. This data indicates that the VPS calibration is accurate, and also that the calibration procedure is valid.

Another significant challenge in this research effort was to achieve a state where the robot, the cameras, the VPS\_client program, the raptor/EKF software, the vision computers, and data collection personnel (a camera operator, dive crew and SCAMP SSV pilot) were all ready to work at the same time. This state was achieved during this research, and VPS system integration has now reached a level of maturity that enables its use without further infrastructure or system level work.

## **8.2 Conclusion: VPS Extended Kalman Filter**

The discrete Extended Kalman Filter equations were applied to the VPS state estimation task. In static testing with camera measurements absent, the EKF was shown to correctly (in terms of gross global direction and terminal velocity) propagate the evolution of the translational state of SCAMP SSV due to control force inputs. In other static testing, where SCAMP SSV was attached rigidly to a fixture in the FOV of the each of the VPS cameras and force data was absent,



the EKF correctly combined data from multiple cameras into one optimal (in a least squares sense) position estimate.

Dynamic testing indicated that VPS, under ideal lighting conditions or in the presence of more robust rejection of noisy data, was capable of performing its inertial navigation task. Camera noise from glare on SCAMP SSV and on the NBRF tank surfaces and from dark objects in the background corrupted some of the dynamic data, but other data collected under better conditions indicated the EKF is able to track vehicle motion during free flight. At specific times during testing with SCAMP SSV in free flight, when all six VPS cameras contained SCAMP SSV in their FOVs, and camera noise was minimal, the EKF tracked SCAMP SSV's position and velocity smoothly, with positional oscillation (noise) and  $3\sigma$  positional error bounds of less than 2 cm (the accuracy of this data cannot be reliably estimated, only inferred from static testing, as there is no way to obtain a "truth" measurement). This data is tantalizing in that it suggests the feasibility of VPS providing inertial navigation of greater accuracy than GPS, but is insufficient to declare that (single-digit or low double-digit) cm-scale accuracy can be has been achieved.

When fewer than two or three VPS cameras provided data to the EKF (as part of either the full six-camera EKF or two separate three-camera EKFs), or when camera data was corrupted by noise, the position estimate at times oscillated or spiked by as much as 0.4 m, and the positional  $3\sigma$  error bound grew to as much

as 1 m. The dynamic, free-flight state estimation data is therefore inconclusive, and indicates that further data collection is needed to prove that the EKF is functioning correctly, or to find and correct any errors or deficiencies that remain.

The EKF currently updates at 10 Hz, and because it is controlled by an independent timer, it can easily be modified to run faster if desired. While further work needs to be done to find values for the  $R$  and  $Q$  matrices that more accurately describe the noise statistics of the system, this step can be considered system refinement. The EKF infrastructure is complete at this time, and can be used for translational state estimation both for recording vehicle trajectories, and, once a controller is implemented and camera noise rejection techniques are implemented, closed-loop position control.

### **8.3 Future Work**

VPS is now capable of providing inertial navigation, with cm-scale accuracy, provided factors such as vehicle location, lighting, and background objects do not adversely impact acquired image data. It has been noted, however, that cm-level tracking performance was demonstrated over a very small volume of the NBRF, for short periods of time. VPS is not currently capable of providing consistent, robust navigation in a cluttered NBRF environment. Measures have been identified that, if taken, will serve to solve this deficiency. Other measures have also been identified that will expand the capabilities of VPS and allow it to be a more convenient and powerful research tool. These measures are organized into three main areas: hardware and operational upgrades, further testing and

parameter updates, and enhancements to VPS\_client and raptor/EKF software. The fundamental goal in the development VPS is not to do research in computer vision or state estimation, although this did occur, but to create a research tool that can be used to conduct new research in space operations simulation. Therefore, this chapter concludes with a discussion of future research applications for VPS.

### 8.3.1 Hardware and Operational Upgrades

Currently, VPS can accept data from only six of the eight available cameras. A fourth vision computer, preferably identical to the three that are now in operation, is needed for the remaining two cameras. This will obviate the need for VPS\_client software to operate on two different operating systems (Windows 98 and Windows 2000), and with two types of frame grabber boards. In addition to this, cameras 5, 7, and 8 need to be removed from the NBRF for servicing – cameras 7 and 8 for image distortion, and camera 5 in order to get a new plastic camera box. As with any camera that comes out of the NBRF, once these cameras are reinstalled, they will require intrinsic calibration via checkerboard images, and all eight cameras will require extrinsic calibration via the calibration frame.

VPS should also in the near future be implemented for use with SCAMP SSV's sister craft – SCAMP II. SCAMP II has much more control authority, and is in many ways a more robust simulation platform. This would involve very little work

– simply copying the VPS version of the raptor/EKF code to SCAMP II's control station.

There are several enhancements that need to be made to the raptor/EKF software to improve the performance and usefulness of VPS as an inertial navigation system. The first enhancements are to improve the user interface (display state data) and data acquisition capabilities. Raptor, the control station program, must be started before SCAMP SSV flight code execution. As currently implemented, the EKF control timer and EKF data logging begins immediately after raptor is initiated. Thus, the EKF log file necessarily contains useless data that is recorded between the time of raptor and SCAMP SSV start up. Of greater concern is that to stop recording EKF data, and begin a new test, the SCAMP SSV flight code and raptor must first be terminated, in that order, and then restarted in the reverse order (raptor then SCAMP SSV).

As can be imagined, this is very inconvenient. To remedy this, a button must be added to the raptor GUI by which the user can start and stop the EKF timer at will. This will also ease VPS initialization – a pilot will be able to fly SCAMP SSV to approximately the center of the tank (the VPS initial position guess), and then start the timer. The separate VPS\_client programs can be running the entire time, their data being harmlessly ignored while the EKF timer is turned off. When EKF log files are currently created, they are named according to their time of creation. This causes inconveniently long file names, and should be replaced by

a sequential file numbering system, similar to the one used by raptor to save SCAMP SSV telemetry.

A graphical display of the position of SCAMP SSV should also be added to the raptor GUI. A continuously updated 2D X-Y or 3D X-Y-Z plot indicating SCAMP SSV's current position during a test would greatly increase the usefulness of VPS to pilots, and could serve as an important component in the simulation of some space operations.

### 8.3.2 Further Testing and Parameter Updates

The values of the matrices  $Q$  and  $R$ , which define the assumptions about the process and measurement noise in the system, respectively, need to be further refined in the EKF software. The linear Kalman Filter equations render a true minimum error covariance, zero-mean, or "optimal" state estimate only if  $R$  and  $Q$  accurately represent their respective noises. While the non-linear EKF is not truly an optimal estimator, because of the linearization step, it too relies on accurate values for  $Q$  and  $R$ . The values for these matrices that are now implemented in the EKF were, fundamentally, set by intuition developed through system use and looking at logs of the camera data that was sent to the EKF. In general, the current magnitude of  $Q$  needs to decrease, meaning that too little confidence is currently placed on the dynamic model relative to the confidence placed in the measurements. Likewise, the magnitude of  $R$  should be increased, because it has been observed that camera data exhibiting much greater levels of

noise than are now modeled are possible. How much  $Q$  should be decreased and  $R$  increased should be the topic of further testing, and will depend greatly, in the case of  $R$ , on the noisy-data rejection strategies that are implemented in the EKF.

The accuracy of the dynamic model also needs to be characterized and improved. The translational drag coefficient and the maximum positive and negative thrusts are assumed equal in all axes, which is likely not true for a variety of reasons. The terminal velocity test used to estimate the value of the drag coefficient should be repeated, several times, on all axes, in both positive and negative directions. The assumption that all thrusters are equal, and that they perform at the same level as documented in [6] may not be valid. Thruster performance could be tested to verify their actual performance. Once  $Q$ ,  $R$ , and the dynamic model have been refined, a useful regimen of tests would be to fly SCAMP SSV in the NBRF, and calculate the state estimate using two separate EKFs – one that only used camera data and did not update based on propagation, and the other ignoring all camera data in favor of relying exclusively on dynamic propagation. Comparing the two state estimates would then offer insight as to how to further modify  $Q$  and  $R$ .

The test results shown in Chapter 7 do include some data where SCAMP SSV comes in and out of the FOV's of individual cameras. This was avoided as much as possible in the initial real-time dynamic testing, in order to first characterize

VPS performance in the best possible operational circumstances – that is, with SCAMP SSV visible to as many, if not all, cameras. Further testing is required to characterize VPS performance with fewer cameras having SCAMP SSV in the FOV. Static tests are required in the extreme edges of the calibrated volume, where calibration errors are likely to be higher. Also required are dynamic tests where SCAMP SSV follows trajectories that take it completely out of all camera views, to have it re-enter some FOV's at a different position. Additional software mechanisms may be required to maintain smooth state estimation under such operation.

The raptor communication software must also be changed to incorporate the translational state estimate of SCAMP SSV, as calculated by VPS, in the data buffers sent to SCAMP SSV. This will allow the implementation of a full 6DOF rotational and translational controller on SCAMP SSV, a goal that has been and remains an important target of the Space Systems Laboratory.

### 8.3.3 VPS\_client and Raptor/EKF Enhancements

As the program responsible for capturing images and distilling from them data useful to the state estimator, VPS\_client has much potential to accept modifications that will enhance the performance of VPS.

In general, it is convenient to apply all data acceptance/rejection tests in one program, and for VPS, this occurs in the raptor/EKF software. Rejecting data in

VPS\_client for some reasons, and in raptor/EKF software for others would make the software more difficult to understand, and spread data rejection records throughout several log files – one for raptor/EKF and one for each instance of VPS\_client. Therefore, all rejection of suspect camera data occurs in the EKF. But enhancements to VPS\_client could allow the raptor/EKF to accept or reject data in a more sophisticated manner.

One planned enhancement is for a function to be added to DoVisionWork() that calculates the second moments of the area of black pixels that results from the current vision processing algorithms, about the X and Y axes in the digital image,  $I_{XX}$  and  $I_{YY}$ . Since SCAMP SSV is nearly round, the ratio of  $I_{XX}/I_{YY}$  should be nearly one. An image where this ratio differs greatly from a value of one likely indicates the image suffers from noise caused by either glare or a dark background object stealing some black pixels. It could also mean that instead of SCAMP SSV, a diver or a part of a diver has been imaged. Sending the  $I_{XX}$  and  $I_{YY}$  data to the EKF along with the current data structures would be an easy way of enabling the state estimator to reject data corrupted by the most serious sources of noise that currently exist in VPS.

Another potential strategy is to “touch up” a noisy image in VPS\_client. This technique is known as “convex hulling”. If SCAMP SSV appears severely non-round in an image, or if it has significant areas of white pixels inside the black pixel area, it is possible to implement vision algorithms that would fill in the shape



based on the “good” sections of the robot’s perimeter in the image. This is more complex than the previous idea, but would allow some types of bad data be corrected and used, thus increasing the robustness of VPS. This may be an important feature if the number of dark background objects in the NBRF increases over time, as it has done in the past.

VPS\_client currently grabs and processes an entire image at each iteration, when it is only necessary to process the section of each image where SCAMP SSV could conceivably be found, based on its last known position and velocity, plus some buffer to account for errors. This “windowing” strategy would decrease the time it takes for VPS\_client to process each image since less “image” would be processed at each iteration. This would increase the camera sampling rates, and allow the EKF to receive more data if its frequency was increased. Windowing would decrease the necessity for divers to be absent from the NBRF during a test, since only the small image area around the robot would be observed, and not the entire, potentially cluttered NBRF. It would also allow multiple SCAMP-class robots to be tracked, which is an important ability if the goal of satellite formation flight simulation is to be achieved.

Windowing has long been planned for VPS – the internal VPS\_client data structures are set up to accommodate it, and a function to select the correct image region to process has been written. Once implemented, each time an image is acquired the windowing function will define four boundaries, (top,

bottom, left and right) are calculated based on the latest global position and velocity estimate of SCAMP SSV and the calibration parameters of the camera that acquired the image. These boundaries define the region of the image to be processed. Because this functionality has not been tested or debugged, however, it is not currently included in the operational code.

Before windowing can be used to track multiple vehicles, VPS\_client will require significant upgrades. Instead of one VPS data structure (containing camera number, time stamp, and image area and centroid data), it will need to store and communicate two or more structures. Likewise, it will need to keep track of the global state estimates of two or more robots. Two or more instances of the EKF would need to run in raptor, one for each of the vehicles. Cross communication between the filters would be required so that each EKF could know when to throw out data because the vehicles appeared merged in a given camera view.

Windowing would not, however, eliminate any of the measurement noise due to glare or background objects. The ultimate solution to noisy images is to abandon all of the existing vision processing algorithms in favor of one that knows the shape of SCAMP SSV, and can directly “find” it in the image, and calculate its image centroid coordinates. Object finding algorithms, such as the recognition by relation templates [28] and others [29] exist and have been implemented in the literature and would not need to be developed from scratch. This radical change in VPS strategy would, however, require significant effort to be brought

about, and would include integrating the algorithm into VPS\_client, adapting it for use with the specific shape of SCAMP SSV, and thoroughly testing its behavior with SCAMP SSV in varying range, illumination and occlusion conditions. A fortunate fact is that the existing calibration procedure, as well as the frame grabbing, communications, EKF and windowing (once it is implemented and tested) software would not need to change in the slightest if object finding was implemented. Object finding would still provide the EKF with the centroid coordinates of SCAMP SSV in the image.

Several important algorithmic changes could also be made in the EKF software. Most importantly, the techniques by which bad data is identified and rejected by the EKF could be augmented. The first improvement required is to reject data for which the difference between the actual measurement vector,  $\mathbf{z}$  (recall:  $\mathbf{z} = [X_{FU} \ Y_{FU}]$ ), and the expected measurement, predicted by the EKF by propagation of the dynamic equations,  $\hat{\mathbf{z}}$ , is too large. The definition of “too large”, as well as the criteria for turning this data rejection strategy on and off, would necessarily be the subject of development testing. During initialization, the difference between the actual and predicted measurements will naturally and correctly be large, but after EKF convergence, it should be small, if the state estimate and camera data are both “good.”

Camera data would be assumed “good” until judged “bad” because of a large discrepancy from the expected measurement. The rejection criteria could be

expressed as a threshold pixel difference between the actual and predicted measurements. Or, a measurement could be judged “bad” if it placed some number of the state variables outside of what the EKF currently judged the  $3\sigma$  error bound to be. With regards to the state estimate, “good” means “converged”, and could mean that the trace of the state estimate error covariance matrix  $P$  was below a certain level, to be determined through future testing. Data rejection would only occur after convergence, and would cease if for some reason the error bounds grew very large again. This method of identifying and rejecting noisy data is expected, if finely tuned, to provide better and more robust noise rejection than the strategy based on object image second moments ( $I_{XX}$ ,  $I_{YY}$ ) discussed above, and since they will both perform similar functions, it is unclear if both should be implemented in VPS in its final version. To make an informed decision to include or ignore either one, both strategies could be implemented and compared to one another.

It is important to note that to examine the impact of any of the proposed changes to the VPS software, a pool test would have to be conducted so that data could be collected with the new system and compared to data collected with the previous version. This is problematic, because the work associated with collecting data with SCAMP SSV in the NBRF is non-trivial, and collecting identical data from one test to another (flying in identical trajectories) is impossible. Therefore, a simulation is being developed that will send VPS camera data SCAMP SSV telemetry data, collected in a previous test, to the

raptor/EKF application exactly as if the test was being performed with the robot in the water. This will allow comparisons of different EKF designs to be made using identical input data. This will be particularly useful in developing more appropriate values of the  $Q$  and  $R$  matrices.

#### 8.3.4 Future Research Applications for VPS

Once the accuracy and robustness of VPS is improved in the manner outlined above, it will become an important tool in the hands of space systems researchers. VPS will significantly increase the fidelity with which space operations can be simulated in the neutral buoyancy environment, which is the only environment that can support large scale, long duration simulations of weightlessness in all six degrees of freedom (rotation and translation).

Likely the first research application VPS will be applied to will be the autonomous flight control of SCAMP SSV. Once position and velocity feedback are available, they can be controlled by various closed-loop control strategies. Simple tasks can be simulated at first, such as automatic position hold, and the autonomous following of predefined trajectories. Afterwards, software could be implemented on SCAMP SSV's onboard computer or raptor that could test a variety of real-time autonomous path planning strategies. Artificial "obstacles" could be defined in the software as a set of coordinates that represent hazards to be avoided. The path planner would be responsible for planning safe trajectories, while the

closed-loop control system would be responsible for calculating and executing thrust commands to follow those trajectories. Algorithms that identify and correct for failure events, such as thruster or sensor failures, could also be simulated. And finally VPS could allow SCAMP SSV to autonomously interact with humans while simulating human-robot teams working in space.

Inertial navigation also enables neutral buoyancy simulations of autonomous rendezvous and docking operations. Initially this would only be possible with one SCAMP-class vehicle docking to a stationary target, but once VPS is enhanced with windowing and is capable of tracking multiple vehicles, true close-proximity spacecraft navigation and guidance in 3D could be simulated. This is a very important topic of research since collision between spacecraft is usually catastrophic, and can be difficult to ensure against given the non-intuitive, 3D nature of rendezvous and docking dynamics.

Finally, the simulation of satellites in formation flight is another research activity that is enabled by VPS. The efficacy of relative navigation strategies and technologies could be measured using inertial position and velocity measurements provided by VPS. Neutral buoyancy robots could simulate satellites navigating by way of GPS, and follow-the-leader formation control strategies could be evaluated.

## Appendix

This appendix contains two tables, A1 and A2, which in turn contain raw and synthetic target-to-target measurements for the calibration frame and are described in more detail in Chapter 4.

Table A1 Raw target-to-target measurements

	Ball 2	Ball 3	Ball 4	Ball 5	Ball 6	Ball 7	Ball 8	Ball 9	Ball 10	Ball 11	Ball 12	Ball 13	Ball 14	Ball 15	Ball 16	Ball 17	Ball 18	Ball 19	Ball 20
Ball 1	1569	594	1156	1586	1902	1711	1314	0	0	1904	0	1724	2257	2552	2084	1752	2227	2622	2437
Ball 2	0	1913	1601	1171	0	1257	1725	1710	987	0	1510	2537	1999	1753	2356	2549	2400	2089	2391
Ball 3	0	0	946	1573	2096	2087	1645	779.5	1401	1974	1293	1262	2005	2610	2080	1489	1874	2521	2375
Ball 4	0	0	0	758	1585	2293	2182	1006	927	1907	1705	1614	1458	2435	2527	2067	1740	2287	2591
Ball 5	0	0	0	0	958	2146	2314	0	615	1668	1835	2060	1246	2061	2636	2429	1837	1996	2564
Ball 6	0	0	0	0	0	1593	2088	1730	814	1249	1723	2491	1592	1489	2437	2609	2135	1726	2309
Ball 7	0	0	0	0	0	0	827	1792	1646	1165	973.5	2447	2500	1576	1507	2075	2487	2069	1739
Ball 8	0	0	0	0	0	0	0	1520	1817	0	647.5	2042	2624	2057	1231	1561	2400	2357	1737
Ball 9	0	0	0	0	0	0	0	0	960	1314	932	849	1384	1914	1562	1090	1174	1758	1690
Ball 10	0	0	0	0	0	0	0	0	0	1075	1289	1711	1095	1548	2036	1941	1484	1531	1966
Ball 11	0	0	0	0	0	0	0	0	0	0	1025	1832	1479	658	1281	1689	1481	924	1063
Ball 12	0	0	0	0	0	0	0	0	0	0	0	1503	1993	1657	1017	1166	1765	1820	1365
Ball 13	0	0	0	0	0	0	0	0	0	0	0	0	1628	2322	1640	780	984.5	1968	1734
Ball 14	0	0	0	0	0	0	0	0	0	0	0	0	0	1659	2326	2091	871.5	1188	1999
Ball 15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1646	2189	1744	699	1197
Ball 16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	994	1800	1731	679
Ball 17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1415	2006	1317
Ball 18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1189	1509
Ball 19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1173



Table A2 Synthetic target-to-target measurements, with discarded measurements highlighted

	Ball 2	Ball 3	Ball 4	Ball 5	Ball 6	Ball 7	Ball 8	Ball 9	Ball 10	Ball 11	Ball 12	Ball 13	Ball 14	Ball 15	Ball 16	Ball 17	Ball 18	Ball 19	Ball 20
Ball 1	1570	592	1155	1586	1902	1711	1314	1057	1389	1904	1140	1725	2258	2551	2086	1751	2227	2621	2436
Ball 2	0	1913	1601	1170	586.1	1256	1723	1707	985.4	1374	1509	2537	1999	1754	2357	2551	2399	2089	2389
Ball 3	0	0	946	1573	2098	2086	1645	778.8	1401	1973	1293	1261	2005	2610	2081	1489	1874	2521	2373
Ball 4	0	0	0	756.2	1584	2293	2182	1006	927	1908	1706	1614	1456	2435	2526	2067	1742	2287	2591
Ball 5	0	0	0	0	957.4	2146	2314	1362	614.2	1667	1835	2059	1245	2061	2635	2429	1838	1996	2564
Ball 6	0	0	0	0	0	1593	2087	1729	813.6	1250	1723	2491	1591	1489	2438	2609	2137	1731	2309
Ball 7	0	0	0	0	0	0	827.5	1792	1647	1165	973.3	2447	2499	1576	1506	2075	2487	2070	1739
Ball 8	0	0	0	0	0	0	0	1520	1817	1498	646.9	2041	2625	2057	1231	1561	2400	2357	1738
Ball 9	0	0	0	0	0	0	0	0	960.6	1314	932.2	848.2	1385	1914	1562	1089	1173	1756	1690
Ball 10	0	0	0	0	0	0	0	0	0	1075	1289	1711	1093	1548	2035	1941	1484	1531	1966
Ball 11	0	0	0	0	0	0	0	0	0	0	1026	1832	1479	657.6	1281	1689	1482	923.7	1063
Ball 12	0	0	0	0	0	0	0	0	0	0	0	1502	1993	1657	1017	1164	1765	1820	1365
Ball 13	0	0	0	0	0	0	0	0	0	0	0	0	1627	2322	1640	779.4	984.8	1968	1732
Ball 14	0	0	0	0	0	0	0	0	0	0	0	0	0	1659	2326	2091	871.9	1188	1999
Ball 15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1646	2189	1744	698.3	1195
Ball 16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	994.5	1799	1732	679.1
Ball 17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1414	2006	1316
Ball 18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1189	1506
Ball 19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1172



## Bibliography

- [1] B. N. Agrawal and R. E. Rasmussen. Air bearing based satellite attitude dynamics simulator for control software research and development. Proceedings of SPIE – The International Society for Optical Engineering, V4366, p 204 – 214, 2001.
- [2] Aircraft Operations Division, National Aeronautics and Space Administration. *JSC Reduced Gravity Program User's Guide*. Houston, Texas, 2000.
- [3] S. Odenbach. Drop tower experiments on thermomagnetic convection. *Microgravity Science and Technology*, v 6, n 3, p 161 – 163, September 1993.
- [4] H. B. Brown and J, M, Dolan. Novel gravity compensation system for space robots. ASCE Specialty Conference on Robotics for Challenging Environments. Albuquerque, New Mexico, February 26 – March 2, 1994.
- [5] G. Creamer, S. Hollander. The spacecraft robotics engineering and control laboratory. <<http://www.nrl.navy.mil/content.php? P=02REVIEW207>> (version current May 13, 2004). Naval Research Laboratory.
- [6] Lisa Shahina Hossaini. The design and analysis of a second generation free flying underwater camera platform. Master's thesis, University of Maryland College Park, 2000.
- [7] Karl Gerard Kowalski, Applications of a three-dimensional position and attitude sensing system for neutral buoyancy space simulation, Master's Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 1989.
- [8] Corinne Claire Ségalas. A three-dimensional acoustic positioning system for neutral buoyancy simulation. Master's thesis, University of Maryland College Park, 2001.
- [9] Philip J. Churchill. Position and attitude station-keeping of a free-flying telerobotic vehicle. Master's thesis, University of Maryland College Park, 1993.
- [10] Elliot Kaplan. *Understanding GPS: Principles and Applications*. Kaplan, 1996.
- [11] H. Stewart Cobb. GPS pseudolites: theory, design, and applications. PhD Dissertation, Stanford University, 1997.

- [12] Ella M. Atkins, Jamie A. Lennon, and Rhiannon S. Peasco. Vision-based following for cooperative astronaut-robot operations. 2002 IEEE Aerospace Conference. Big Sky, Montana, March 9-16, 2002.
- [13] Rhiannon S. Peasco. A vision-based underwater three-dimensional positioning system. Master's thesis, University of Maryland, College Park, 2002.
- [14] T. Williams and S. Tanygin, "On-orbit engineering test of the AERCam Sprint Robotic Camera Vehicle," *Advances in the Astronautical Sciences*, 99(2), 1001-1020, 1998.
- [15] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, v. RA-3(4), August 1987.
- [16] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison Wesley, New York, 1989.
- [17] T. A. Clarke, J. F. Fryer. The development of camera calibration methods and models. *Photogrammetric Record*, 16(91): pp 51-66. 1998.
- [18] Janne Heikkilä, Olli Silvén. A four-step camera calibration procedure with implicit image correction. *IEEE Computer Vision and Pattern Recognition Conference*, San Juan, Puerto Rico, 1997.
- [19] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. *IEEE Proceedings of the Seventh International Conference on Computer Vision*. Vol. 1, p. 666 – 673. Kerkyra, Greece.
- [20] The Space Systems Laboratory. The use and calibration of the NBRF vision positioning system, SSL internal document, in progress.
- [21] [http://www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/) > (version current May 13, 2004).
- [22] <http://www-2.cs.cmu.edu/~rgw/TsaiCode.html> > (version current May 13, 2004).
- [23] Hiroshi Yamano, Hideo Saito. Robot motion tracking system with multiple views. *Intelligent Robots and Computer Vision XX: Algorithms, Techniques, and Active Vision*. Proceedings of the SPIE Vol. 4572 (2001).
- [24] Phillip Churchill. ReCS Software 5/18/93. Space Systems Laboratory internal document #93-002, 1993.

- [25] Will Miller, Ella Atkins. Raptor: a linux based ground station software for the SCAMP neutral buoyancy vehicles. Space Systems Laboratory internal document. 2002.
- [26] Peter Dorato, Chaouki Abdallah, Vito Cerone. *Linear-Quadratic Control: An Introduction*. Prentice Hall, 1998.
- [27] Arthur Gelb (editor). *Applied Optimal Estimation*. MIT Press, 1974.
- [28] David A. Forsyth, Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, New Jersey, 2003.
- [29] D.M. Gavrila and L.S. Davis. 3-D model-based tracking of humans in action: a multi-view approach. Proceedings CVPR '96, 1996 IEEE Computer Society Conference, 18-20 June 1996.