

12-26-2013

Real-Time, Multiple Pan/Tilt/Zoom Computer Vision Tracking and 3D Positioning System for Unmanned Aerial System Metrology

Daniel D. Doyle

Follow this and additional works at: <https://scholar.afit.edu/etd>

Recommended Citation

Doyle, Daniel D., "Real-Time, Multiple Pan/Tilt/Zoom Computer Vision Tracking and 3D Positioning System for Unmanned Aerial System Metrology" (2013). *Theses and Dissertations*. 507.
<https://scholar.afit.edu/etd/507>

This Dissertation is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact richard.mansfield@afit.edu.



**Real-Time, Multiple, Pan/Tilt/Zoom, Computer Vision Tracking, and 3D
Position Estimating System for Small Unmanned Aircraft System Metrology**

DISSERTATION

Daniel D. Doyle, Lieutenant Colonel, USAF

AFIT-ENY-DS-13-D- 08

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A.
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government.

AFIT-ENY-DS-13-D-08

**Real-Time, Multiple, Pan/Tilt/Zoom, Computer Vision Tracking, and
3D Position Estimating System for Small Unmanned Aircraft System
Metrology**

DISSERTATION

Presented to the Faculty

Graduate School of Engineering

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Daniel D. Doyle, BS, MS

Lieutenant Colonel, USAF



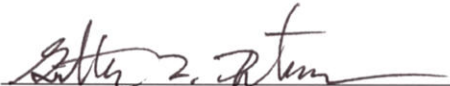

September 2013

DISTRIBUTION STATEMENT A.
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED


**Real-Time, Multiple, Pan/Tilt/Zoom, Computer Vision Tracking, and
3D Position Estimating System for Small Unmanned Aircraft System
Metrology**

Daniel D. Doyle, BS, MS
Lieutenant Colonel, USAF

Approved:

 Jonathan T. Black, PhD (Chairman)	<u>4 Nov 2013</u> Date
 Richard G. Cobb, PhD (Member)	<u>1 Nov 2013</u> Date
 Gilbert L. Peterson, PhD (Member)	<u>1 Nov 2013</u> Date
 Alan L. Jennings, PhD (Member)	<u>6 Nov 2013</u> Date

Accepted:

 Adedeji Badiru, PhD Dean, Graduate School of Engineering and Management	<u>13 Nov 2013</u> Date
--	----------------------------

Abstract

The study of the flight kinematics of Unmanned Aircraft Systems (UASs) continues to be an important field of research for understanding and developing state-of-the-art nano/micro air vehicles. Analyzing various flapping-wing UASs using photogrammetry shows potential given the ability to use images for constructing a three-dimensional (3D) representation of a given object. Development of a photogrammetry/metrology system using Computer Vision (CV) tracking and 3D point extraction would provide an avenue for collecting free-flight data crucial in understanding the aerodynamic and aeroelastic behavior of flapping wing UASs; thus far not achieved. Free-flight data would provide a link between empirical and theoretical development thereby increasing knowledge of various flight profiles of flapping-wing UASs. The focus of this work is to develop a scalable system capable of real-time tracking, zooming, and 3D position estimation of a UAS using multiple cameras.

Real-time, state-of-the-art photogrammetry systems use retro-reflective markers to obtain object poses and/or positions over time. Computer Vision (CV) methods capable of performing photogrammetry tasks provide the potential for non-contact measurements that avoid altering the test subject. Related work includes using Pan/Tilt/Zoom (PTZ) cameras in combination with or without static cameras; however, little has been done in the area of metrology and multiple PTZ cameras. Contributions include the development of a real-time CV-tracking algorithm for PTZ cameras, using optical flow and known camera motion, in order to capture a moving object. The developed implementation is versatile, allowing the ability to test other CV methods with a PTZ system and known camera motion. Further, digital zoom is applied to each camera to simulate PTZ cameras for tracking purposes. Utilizing known camera poses, the UAS's 3D position is estimated and focal lengths are calculated for filling the image to a desired amount. This system is tested against truth data obtained using a commercial photogrammetry system. Automatically capturing high-resolution, high frame-rate images of small, free-flight flappers is obtainable using the system developed in this work.

Acknowledgements

1 Peter 5:6 Humble yourselves therefore, that He may exalt you in due time.

I have enjoyed these years of study (and countless frustrations) and feel extremely privileged and blessed. Now that my research is coming to an end, I know that I am going to miss it. This time of study has sparked in me a fervor for research, and I'm thankful to all of those involved in getting me here.

My deepest thanks go to Dr. Jon Black, for taking me under his wing, giving me a challenge, the tools, and the flexibility to accomplish it. I also would like to thank Dr. Alan Jennings for his willingness to wrestle through problems with me, for continually challenging me to understand the theory behind the application, and for providing me with the constructive criticism I needed in order to develop a worthwhile product.

I dedicate this research to my wife. She is the best wife a man could ask for. I have enjoyed her constant support which has allowed me to finish my research on time. I also had the chance to appreciate all that she did and does as a mother. She demonstrates the importance of being a full-time, stay-at-home mom. May God bless her and the work she does for our country.

Finally, thank you, Lord Jesus, for being my hope and my salvation. I am constantly inspired by You and Your Word. Thank You for Your wisdom and understanding and for creating in me an exceeding desire to learn, to create, to inspire, and to be inspired.

Daniel D. Doyle

Table of Contents

	Page
Abstract	iii
Acknowledgements	iv
List of Figures	ix
List of Tables	xiv
Abbreviations	xvi
I. Introduction	1-1
1.1 Nano/Micro Unmanned Aircraft Systems	1-1
1.2 Non-Contact Measurement Systems	1-2
1.2.1 Photogrammetry and Videogrammetry	1-3
1.2.2 Computer Vision	1-4
1.3 Motivation for Developing a Pan-Tilt Computer Vision System Photogrammetry and Videogrammetry Analysis	1-4
1.4 Research Focus	1-5
1.5 Document Layout	1-6
II. Background	2-1
2.1 Overview	2-1
2.2 Computer Vision Object Detection	2-2
2.2.1 Detection, Classification and Learning/Recognition	2-2
2.2.2 Representation, Features and Detection Categories	2-4
2.3 CV Object Tracking	2-15
2.3.1 Tracking	2-15
2.4 Computer Vision (CV) Using Multiple Cameras	2-17
2.4.1 Target/Sensor Cases	2-17
2.4.2 Data Fusion Core Techniques	2-17
2.4.3 Core Techniques Explored through Recent Work	2-19
2.4.4 Pan/Tilt Unit (PTU)/PTZ Applications	2-21
2.5 Summary	2-24
III. Design Considerations	3-1
3.1 Research Platform, Devices, and Software	3-1
3.1.1 Hardware	3-1
3.1.2 Lighting	3-4
3.1.3 Software	3-4
3.2 Assumptions	3-6
3.3 Objectives/Exit Criteria	3-7

	Page	
3.3.1	Each camera zooms in/out and falls within specified thresholds	3-8
3.3.2	Each camera pans and continues to track the test subject	3-8
3.3.3	Each camera tilts and continues to track the test subject	3-8
3.3.4	Each camera pans/tilts/zooms and continues to track the test subject	3-8
3.3.5	Considered Objectives (not as part of the exit criteria)	3-8
IV.	Selection of a Computer Vision Tracking Technique	4-1
4.1	Overview	4-1
4.2	Images	4-2
4.3	Feature Detection and Matching	4-2
4.3.1	Scale-Invariant Feature Transform (SIFT)	4-3
4.3.2	Speeded-Up Robust Features (SURF)	4-3
4.3.3	Fast Library for Approximate Nearest Neighbors (FLANN)	4-4
4.3.4	Homography, Random Sample Consensus (RANSAC), and other Matching Phenomenon	4-4
4.3.5	Covariance Tracking	4-5
4.4	Motion Detection and Tracking	4-7
4.4.1	Gaussian Mixture Model (GMM)	4-7
4.4.2	Optical Flow	4-9
4.4.3	Point Selection	4-9
4.4.4	Point Averaging	4-10
4.5	Segmentation and Tracking	4-10
4.5.1	Mean Shift (MS)	4-10
4.6	Experiments	4-12
4.6.1	Training	4-12
4.6.2	Data Sets	4-13
4.6.3	Discussion	4-19
4.7	Summary and Selection	4-21
V.	Algorithm Development and Error Analysis	5-1
5.1	Overview	5-1
5.2	System Components	5-2
5.2.1	Equipment used	5-2
5.2.2	Background and Test Subject	5-2
5.3	Algorithm Development	5-3
5.3.1	Background Motion Estimation	5-3
5.3.2	Approximation Accuracy	5-7
5.3.3	Tracking Process	5-8
5.3.4	Point classification	5-8
5.3.5	Kalman filter	5-12

	Page
5.3.6 Pan/tilt control	5-13
5.4 Experiments and Results	5-14
5.4.1 Point estimation performance	5-14
5.4.2 Test subject tracking and timing	5-17
5.4.3 Comparison testing	5-20
5.5 Summary	5-24
VI. Employing a Multiple, Real-Time, Pan/Tilt/Zoom Camera Object Tracking, and 3D Position Estimating System	6-1
6.0.1 Overview	6-1
6.1 System Components	6-2
6.1.1 Equipment used	6-2
6.2 Theoretical Overview	6-2
6.2.1 3D Position Estimation	6-3
6.2.2 Ideal Focal Length Estimation	6-4
6.3 Parallelized Tracking Using a PTZ Camera	6-5
6.3.1 Parallel Process Development	6-5
6.4 Zoom control	6-7
6.5 Experiments and Results	6-8
6.5.1 Relating the “Two-camera, PTZ system” to Truth Data	6-8
6.5.2 Experiments	6-9
6.5.3 Flight Profile Examination	6-11
6.5.4 Focal Length Estimation	6-17
6.5.5 Additional Design Considerations	6-18
6.6 Conclusions	6-19
VII. Discussion, Conclusion, and Future Work	7-1
7.1 Flapping Wing Research	7-1
7.2 Conclusions	7-2
7.3 Limitations of the Computer Vision (CV) Pan/Tilt/Zoom (PTZ) System	7-5
7.4 Future Work	7-5
7.4.1 Pose Estimation	7-5
7.4.2 Computer-Controlled Mechanical Zoom	7-6
7.4.3 Image Registration	7-6
7.4.4 Data Fusion	7-7
7.4.5 Velocity control for Pan/Tilt Unit (PTU)	7-7
7.4.6 Tracking of a Flapping-Wing Unmanned Aircraft System (UAS)	7-7
7.5 Summary	7-8

	Page
Appendix A. Fusion Background	A-1
A.1 Introduction	A-1
A.2 Fusion	A-2
A.2.1 Data Fusion	A-3
A.2.2 Sensor Fusion	A-5
A.2.3 Mult-Sensor Data Fusion	A-6
A.2.4 Image Fusion	A-7
A.2.5 Other Types of Fusion	A-8
A.3 Fusion Models	A-8
A.3.1 Joint Directors of Laboratories (JDL) Data Fusion Model	A-9
A.3.2 Observe, Orient, Decide and Act (OODA) Loop . . .	A-16
A.3.3 Data, Information, Knowledge, Wisdom (DIKW) Pyra- mid	A-17
A.3.4 Rasmussen Information Processing Hierarchy	A-19
A.3.5 The Intelligence Cycle	A-21
A.3.6 Waterfall Model	A-21
A.3.7 Omnibus Model	A-21
A.3.8 Situation Awareness Reference Model	A-23
A.3.9 Visual Data Fusion Model	A-24
A.3.10 Unified Data Fusion (λ JDL) Model	A-24
A.3.11 Perceptual Reasoning Machine (PRM)	A-25
A.3.12 Dasarathy Model	A-25
A.3.13 Distributed Network of Autonomous Modules [89, Hen- rich and Kausch, 2004]	A-26
A.3.14 Other Fusion Models	A-27
Bibliography	28

List of Figures

Figure		Page
1.1	Examples of flapping-wing UASs and biological counterparts. Images collected using an image search on micro air vehicles, 8/7/2013, from the following websites: sciencedaily.com, mlpc.com, glue.umd.edu, what-is-this.com, af.mil, wpafb.af.mil, deskeng.com, avinc.com, 3me.tudelft.nl, micro.seas.harvard.edu, and robobees.seas.harvard.edu.	1-5
2.1	A time-line of the technology and techniques used in this work. . . .	2-1
2.2	Object representations [133].	2-4
2.3	General object detection hierarchy.	2-6
2.4	General object tracking hierarchy.	2-16
3.1	Dell M6600 Precision	3-1
3.2	PTU-D46-17	3-2
3.3	Flea 3 USB 3.0 Camera	3-3
3.4	Fujinon 2.8 ~ 8mm vari-focal lens.	3-4
3.5	High performance lights.	3-4
3.6	Bescor LED-500 dimmable studio light kit.	3-5
4.1	The MS process.	4-12
4.2	Clipboard with attached texture and selected tracking point (blue). . .	4-13
4.3	Clipboard with retained keypoints (multi-colored circles), contour points (corners of the green lines), and selected tracking point (solid green circle).	4-13
4.4	SIFT, SURF and covariance tracking Root Mean Square (RMS) pixel error.	4-15
4.5	GMM RMS pixel error.	4-15
4.6	Pyramidal Lucas-Kanade (PyrLK) Optical flow and MS RMS pixel error.	4-15
4.7	Algorithm tracking speed.	4-15

Figure		Page
4.8	SIFT, SURF and covariance tracking RMS pixel error.	4-18
4.9	GMM RMS pixel error.	4-18
4.10	PyrLK Optical flow and MS RMS pixel error.	4-18
4.11	Algorithm tracking speed.	4-18
5.1	Equipment used.	5-3
5.2	Background used for performance testing.	5-4
5.3	The test subject is a helicopter that acts as a pendulum for tracking; green squares represent the initial placement of the optical flow points from image, I_{i-1} and the large, red circle represents the calculated pixel centroid, $(x, y)_{pix}$	5-5
5.4	The mean error of the small-angle approximation (left) and linear approximation (right).	5-7
5.5	Closed-form solution versus the max error of the small-angle approximation (left) and linear approximation (right).	5-8
5.6	Flowchart and steps for real-time tracking of a single, moving object.	5-9
5.7	Interpreting optical flow points in a current frame as background, moving, or noise.	5-11
5.8	Account of features (using background estimation with a moving object threshold) given equal degrees of travel versus pixel distance.	5-17
5.9	Account of features (using background estimation with a moving object threshold) given a diagonal movement versus pixel distance. Note: a movement of $\psi = 3.1^\circ, \theta = 3.1^\circ$ is actually 4.4° of travel . . .	5-18
5.10	A moving test subject captured by a static camera with previous points shown as green rectangles, \mathbf{g}_{i-1} , current optical flow points as white, closed circles, \mathbf{g}_i , estimated points as green, open circles, $\hat{\mathbf{g}}_i$, and removed outliers shown as red, closed circles.	5-19
5.11	A moving test subject captured by a moving camera with previous points shown as green rectangles, \mathbf{g}_{i-1} , current optical flow points as white, closed circles, \mathbf{g}_i , estimated points as green, open circles, $\hat{\mathbf{g}}_i$, and removed outliers shown as red, closed circles.	5-20

Figure		Page
5.12	A sequence of frames, approximately 0.24 seconds apart for one period, of a swinging test subject with the center of the image identified by a white diamond, previous points shown as green rectangles, \mathbf{g}_{i-1} , current optical flow points as white, closed circles, \mathbf{g}_i , estimated points as green, open circles, $\hat{\mathbf{g}}_i$, and removed outliers shown as red, closed circles. . .	5-21
5.13	For comparison purposes, the Continuously Adaptive Mean Shift (CAMShift) algorithm was used to determine the object centroid. The color histogram (right) is made up of the components within the ellipse of an image in the test set (left).	5-22
5.14	For comparison purposes, good matches were used to provide the moving object centroid using a point average. The training image (right) shows feature locations that were matched to an image in the test set (left).	5-23
5.15	Affine transformation of the training contour (right) onto the moving object (left).	5-23
6.1	Equipment used.	6-2
6.2	Parallel process parameters and camera initialization (left) and tasks (right) using digital zoom.	6-6
6.3	Checkerboard with established coordinate system (x-axis, red; y-axis, green, and z-axis, blue circle into the checkerboard) based on point to point correspondences and known checkerboard square size.	6-7
6.4	The test subject used to obtain data using the “two-camera, PTZ system” and the Vicon camera system.	6-10
6.5	X-axis testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system” with a point-to-point (P2P) relationship. Note: The ‘Time of Capture’ relates to the image sequence shown in Fig. 6.13.	6-12
6.6	Y-axis testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system” with a point-to-point (P2P) relationship. Note: The ‘Time of Capture’ relates to the image sequence shown in Fig. 6.14.	6-13
6.7	Z-axis testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system” with a point-to-point (P2P) relationship. Note: The ‘Time of Capture’ relates to the image sequence shown in Fig. 6.15.	6-14

Figure		Page
6.8	Below camera-level testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system” with a point-to-point (P2P) relationship. Note: The ‘Time of Capture’ relates to the image sequence shown in Fig. 6.16.	6-15
6.9	Error versus the average distance of the moving object from the cameras. Color and circle size is used to distinguish larger pan/tilt angles (top), and a box plot (bottom) is used to show the median (red line), 25 th and 75 th percentiles as the bottom and top edges of each box, whiskers as extreme points, and outliers plotted individually.	6-17
6.10	Camera 1 focal length vs. Vicon lower/upper/ideal focal length.	6-19
6.11	Camera 2 focal length vs. Vicon lower/upper/ideal focal length.	6-20
6.12	Generic moving object tracking using “two-camera, PTZ system”.	6-21
6.13	X-axis testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system”.	6-22
6.14	Y-axis level testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system”.	6-23
6.15	Z-axis level testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system”.	6-24
6.16	Below camera-level testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system”.	6-25
7.1	Example output using the POSIT algorithm on a helicopter with five known model coordinates.	7-6
A.1	JDL data fusion model (1992).	A-10
A.2	Revised JDL data fusion model (1998).	A-11
A.3	Revised JDL data fusion model (2004).	A-12
A.4	JDL user data fusion model 2002 [19].	A-14
A.5	JDL user data fusion model 2005 [20].	A-15
A.6	Information fusion situation assessment reference model [22].	A-16
A.7	Observe, Orient, Decide and Act (OODA) Loop.	A-17
A.8	Data, Information, Knowledge and Wisdom (DIKW) Pyramid.	A-18

Figure		Page
A.9	Rasmussen information processing hierarchy [36].	A-20
A.10	Waterfall fusion process model [45].	A-21
A.11	Omnibus model (Bedworth, O'Brien).	A-22
A.12	Model of situation awareness [47].	A-23
A.13	Perceptual reasoning machine [21, Kadar, SPIE02].	A-25
A.14	Data feature decision model (Dasarathy, B., 1994).	A-26

List of Tables

Table		Page
1.1	Nano/Micro Specifications [3].	1-1
2.1	background subtraction (BS) methods and performance analysis [96].	2-11
2.2	Object Tracking Category Definitions	2-17
2.3	Pan/tilt/zoom camera applications.	2-23
2.4	Related work summary of pan/tilt system resolution and operating frame-rate.	2-24
3.1	Computer Specifications	3-1
3.2	PTU Specifications	3-2
3.3	Camera Specifications	3-3
3.4	Lens Specifications	3-4
3.5	PTU CV Software Components	3-5
3.6	Tracking system test scenario objectives.	3-8
4.1	Static Background Algorithm Performance.	4-16
4.2	Dynamic Background Algorithm Performance.	4-19
5.1	Pan/Tilt/Zoom (PTZ) tracking with background subtraction	5-10
5.2	Approximate Pixel Movement and Speed for Pan or Tilt Angle, Θ , with Focal Length = 1,076 pixels.	5-15
5.3	Optical flow error for a stationary background with varying movements of pan, $\Delta\psi$, and tilt, $\Delta\theta$	5-16
5.4	Average algorithm time (in milliseconds) for varying image resolutions versus number of features used.	5-17
5.5	Timing of image capture, algorithm and pan/tilt movement over 999 iterations	5-19
5.6	General performance characteristics of object detection/tracking sys- tems for a set of 999 images at 656x524 resolution.	5-24

Table		Page
6.1	Specific tests with objectives and pass/fail.	6-11
6.2	Root mean square (RMS) error associated with specified tests.	6-16
7.1	Fulfillment of exit criteria.	7-2
A.1	Comparison of definitions for Fusion and Data Fusion.	A-4
A.2	Comparison of definitions for Fusion and Image Fusion.	A-8
A.3	Review of levels and their respective roles regarding the JDL-user model [22].	A-15
A.4	Rasmussen Information Processing Hierarchy and the JDL Data Fusion Model.	A-20

Abbreviations

3D three-dimensional.....	7-1
AFSOC Air Force Special Operations Command.....	1-1
AI Artificial Intelligence.....	2-7
ANN Artificial Neural Network.....	2-14
API Application Programming Interface.....	3-4
BS background subtraction.....	2-10
CAMShift Continuously Adaptive Mean Shift.....	32
CFD Computational Fluid Dynamics.....	7-1
CPI Color Probability Image.....	2-20
CPU Central Processing Unit.....	5-19
CUDA Compute Unified Device Architecture.....	3-5
CV Computer Vision.....	7-1
DOF Degrees of Freedom.....	3-6
DF Data Fusion.....	7-7
EKF Extended Kalman Filter.....	2-18
EM Expectation-Maximization.....	2-7
FLANN Fast Library for Approximate Nearest Neighbors.....	5-21
FOV Field of View.....	7-1
FPGA Field Programmable Gate Array.....	2-15

fps frames per second	7-3
GLOH Gradient Location Orientation Histogram	2-6
GMM Gaussian Mixture Model	4-1
GPU Graphics Processing Unit	7-1
HIL human-in-the-loop	4-20
HOG Histogram of Oriented Gradients	2-6
HSI hue, saturation, and intensity	2-13
HSV hue, saturation, and value	2-13
IMMF Interacting Multiple Model Filter	2-19
JPDA Joint Probabilistic Data Association	2-18
KLT Kanade-Lucas-Tomasi	2-6
KF Kalman Filter	7-7
MHT Multiple Hypothesis Testing	2-18
MRF Markov Random Field	2-8
MS Mean Shift	4-1
NBC Naive Bayes Classifier	2-14
PCA Principal Component Analysis	2-7
PDA Probabilistic Data Association	2-18
PDF Probability Density Function	2-10
PF Particle Filter	2-18

PSO Particle Swarm Optimization	2-19
PyrLK Pyramidal Lucas-Kanade.....	5-1
PTU Pan/Tilt Unit	7-3
PTZ Pan/Tilt/Zoom.....	7-1
RANSAC Random Sample Consensus.....	4-4
RMS Root Mean Square	4-13
SDK Software Development Kit.....	3-4
SGD Stochastic Gradient Descent.....	2-15
SIFT Scale-Invariant Feature Transform.....	4-1
SUAS Small Unmanned Aircraft System.....	3-9
SURF Speeded-Up Robust Features	5-21
SVD Singular Value Decomposition	2-7
SVM Support Vector Machines	2-14
SVSF Smooth Variable Structure Filter.....	2-19
UAS Unmanned Aircraft System.....	7-1
UKF Unscented Kalman Filter.....	2-18
USAF United States Air Force.....	3-9

Real-Time, Multiple Pan/Tilt/Zoom Computer Vision Tracking and 3D Positioning System for Unmanned Aerial System Research

I. Introduction

1.1 Nano/Micro Unmanned Aircraft Systems

Flying robotic systems known as Unmanned Aircraft Systems (UASs), micro air vehicles, or drones take to the skies and perform complex activities previously performed by humans such as takeoff, loitering, target acquisition and landing. Developing a system capable of capturing and measuring these and other complex activities of Small Unmanned Aircraft Systems (SUASs) is the focus of this research.

The Air Force Special Operations Command (AFSOC) is designated as the SUAS lead, and they, in turn, have designated four major subclasses that better define SUAS [3]. These subclasses are:

1. Nano/Micro
2. Man-portable
3. Multi-mission
4. Air-launched

The focus of this research deals with the tracking and measuring of Nano/Micro UAS in free-flight, specifically, for flapping-wing kinematics that are difficult to test in a laboratory. Nano/Micro UAS characteristics are listed in Table 1.1. For the remainder of this research, SUAS and UAS are used interchangeably.

Table 1.1: Nano/Micro Specifications [3].

Weight	0-20 lbs
Operating Altitude	<1200 ft (Above Ground Level)
Speed	100 kts

The Unmanned Aircraft System (UAS) flight plan describes Nano/Micro UAS as “aircraft capable of conducting a variety of indoor and outdoor reconnaissance sensing missions” [3]. Developing aircraft capable of these activities, hovering, dropping turns, backward flight, wing billowing, etc., requires an in-depth understanding of biological counterparts. This development requires a system capable of capturing high-resolution measurements during free-flight testing which is the purpose of this work.

Smaller, flapping-wing UASs, often referred to as micro air vehicles, are typically 150 mm or less in dimension and weigh no more than 90 g [7]. Due to the small-scale and speed involved with these vehicles, novel systems are needed to measure forces and motion without disrupting flight. Understanding how flapping-wing UASs fly and being able to optimize their performance are challenges for the metrology community. Evaluating flight characteristics seen in everyday life from bugs and birds would enhance our ability in developing UASs.

The Air Force Institute of Technology and the Air Force Research Laboratory have worked on advancing flapping-wing mechanisms through flight control research, wing design and analysis of dynamics [6, 37, 40, 62, 77]. There are many other facilities devoted to developing and understanding the mechanics of flapping-wing flight, including: the Harvard Microrobotics Lab [104], the Korean Advanced Institute of Science and Technology (KAIST) Smart Systems and Structures Lab [75], and Georgia Institute of Technology’s Intelligent Control Systems Laboratory [99].

1.2 Non-Contact Measurement Systems

The purpose of non-contact measurement systems is to observe the normal behavior of an object without affecting the object’s performance. Contact measurement methods pose many challenges when working with UASs. These methods consist of, but are not limited to, using strain gages, paint, retro-reflective markers, etc. which typically alter vehicle performance either by weight or other restrictions to lightweight devices (i.e., 90 g or less).

Non-contact measurement systems provide the means of gaining a truer understanding of the aeroelastics of billowing membranes (coupled with fluid-structures interaction) of biological or UAS counterparts. Non-contact displacement measurement methods in-

clude laser vibrometry, Lidar/laser range finders, capacitance measurement, interferometry and photo/videogrammetry. Laser vibrometry is the preferred method for measuring non-contact vibrations [85]. However, only a single point is measured at a time, and alignment must be maintained through the test only measuring very small deflections (i.e., no flapping). Videogrammetry uses multiple, synchronized images or stereo pairs and typically retro-reflective markers or projected targets for determining surface shapes capturing shape and motion from a single test [62]. Projected targets are not practical for use with free-flying test articles, and adding retro-reflective markers negates non-contact measurement.

Using a camera allows for more freedom of movement. However, the subject must stay in the fields of view with little to no occlusion. Incorporating a pan/tilt camera with real-time tracking will allow for the camera to move so that the subject remains in its Field of View (FOV). Adding this capability expands the effective capture volume, without a sacrifice in resolution of the subject, for better flight data over a larger set of flight profiles. The ideal system would have a fast response (high frame-rate and low-latency for faster vehicles), high resolution (for better discrimination of the vehicle) and work with general dynamic objects (not require selection or training).

1.2.1 Photogrammetry and Videogrammetry. “Photogrammetry is the process of deriving the three-dimensional (3D) coordinates of object points based on their two-dimensional location in a set of photographic images” [85]. Videogrammetry is an extension of photogrammetry by applying the same principles to a set of images in time.

Obtaining sets of two-dimensional points from an image uses various techniques such as target-based and texture-based point extraction. Software systems are able to readily identify locations in a given set of images with discrete man-made targets (fiducials) through the use of Computer Vision (CV) techniques for point correspondences. Texture-based photogrammetry uses randomly varying intensities of a small set of pixels as a target for determining correspondences between images. Recent work with biomimetics analysis uses texture-based photogrammetry for non-rigid surface measurement of flexible structures [62]. The work analyzes flapping-wings of a statically-mounted UAS and specifically determines that texture-based photogrammetry is a significant improvement over the state-of-the-art

due to the high density of points generated, giving accurate measurements of billow and other large deformations.

1.2.2 Computer Vision. Many modern technologies incorporate CV techniques for identifying objects to tracking in real-time [116]. A literature search reveals hundreds of methods for resolving the tracking and recognition problems. Various forms of CV techniques include point detectors, segmentation, background modeling and classification as given by the survey conducted by Yilmaz [133]. An example of each given technique can be found in [35, 80, 93, 139]. Computational libraries, such as in [26], provide hundreds of CV algorithms while conferences continue to see authors with faster and more robust algorithms for various tracking scenarios. Each method has its advantages and disadvantages towards recognizing and tracking various objects in different environments. Determining which algorithm is best-suited to specific scenarios is the challenge.

Image registration techniques are improving and provide ways of determining complex, non-rigid motion [52]. Photogrammetry continues to evolve by using updated or improved CV algorithms for finding point correspondences. This work combines the work of multiple systems for the purpose of tracking, capturing and performing videogrammetry operations on UASs.

1.3 Motivation for Developing a Pan-Tilt Computer Vision System

Photogrammetry and Videogrammetry Analysis

Security systems typically use CV techniques associated with Pan/Tilt/Zoom (PTZ) systems to track faces, people and vehicles such as described in [27, 48, 86]. This work constitutes a new use for CV PTZ systems in the area of 3D point estimation of flapping-wing UASs. The benefits of designing and developing such a system is instrumental in researching various small, free-flying flapping-wing mechanisms and organisms. Tracking and measuring these mechanisms or organisms (e.g., moths) under operational conditions is key in analyzing and understanding their free-flight kinematics, in turn, providing designers the tools necessary in developing UASs for indoor and outdoor reconnaissance missions (see Fig. 1.3).



Figure 1.1: Examples of flapping-wing UASs and biological counterparts. Images collected using an image search on micro air vehicles, 8/7/2013, from the following websites: sciencedaily.com, mlpc.com, glue.umd.edu, what-is-this.com, af.mil, wpafb.af.mil, deskeng.com, avinc.com, 3me.tudelft.nl, micro.seas.harvard.edu, and robobeeseas.harvard.edu.

Another motivation for such a system deals with reduced cost and complexity. Current non-contact measurement systems require large investments for lasers, multiple infrared cameras, separate PTZ cameras, software/hardware interfaces, lighting, etc. Developing a comprehensive CV-based system for tracking and measuring would eliminate much of the overhead of commercial hardware/software packages.

1.4 Research Focus

This research details the development of a PTZ CV system for tracking and measuring a moving object of interest. The first part of this work tests various types of CV algorithms (i.e., Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), Mean Shift (MS), Pyramidal Lucas-Kanade (PyrLK) optical flow, covariance, and Gaussian Mixture Model (GMM)) against a static and dynamic background for narrowing and identifying appropriate algorithms for the Pan/Tilt Unit (PTU) CV system.

The focus then shifts towards experimental testing of the CV algorithm and comparing it to existing methods for the purpose of tracking a moving object against a moving background using a single camera. Lastly, two PTZ cameras are used to track a moving object, determine its 3D position, and command a desired focal length based on the size of the test subject. The results are compared to data obtained using a turn-key, photogrammetry system.

Demonstration of these techniques paves the way for analyzing various, single UASs in a controlled environment. Capturing a UAS using multiple cameras and obtaining high-resolution, high frame-rate images would enable state measurements of not only the vehicle as a whole, but other visible points of interest as well.

This research provides a real-time, multiple PTZ camera system capable of capturing high frame-rate, high-resolution images of free-flight, flapping-wing UASs. High frame-rate is quantified as greater than 280 frames per second (fps). Resolution is quantified by setting a desired outcome and maintaining a zoom within an upper and lower zoom threshold of the UAS within the camera's FOV. The design is a field-portable CV PTZ tracking system expandable to a multi-camera setup. Additional cameras make the system suitable for indoor and outdoor photogrammetry purposes.

1.5 Document Layout

Chapter **II** provides an extensive literary review of object detection, tracking, stereo-vision and 3D extraction methods used in CV. It also reviews related research in PTU and PTZ CV systems laying the foundation for Chapter's **IV-VI**. The basic assumptions, the objectives, and the exit criteria are provided in Chapter **III**. In Chapter **IV**, selected CV algorithms are reviewed for tracking in a static and dynamic environment. Comparisons are made based upon accuracy, speed and memory. Based on the work in Chapter **IV**, an algorithm is developed for tracking a moving object against a dynamic background and performance aspects of the algorithm are tested in Chapter **V**. Furthermore, Chapter **V** explores the developed algorithm by tracking a moving object, analyzing the results, and comparing its performance against existing methods. Chapter **VI** represents the culmination of the work performed by adding another camera to the system, employing parallel

implemenation of CV-tracking, performing digital zoom, and estimating the 3D position of the moving object. In addition, several experiments are recorded and compared to truth data obtained using a turn-key, photogrammetry system. Lastly, Chapter **VII** provides a discussion of why the techniques and the CV PTZ camera system developed are relevant, the conclusions, future work, and the summary.

II. Background

2.1 Overview

Measuring free-flight, flapping-wing Unmanned Aircraft Systems (UASs) typically requires altering the vehicle so that the measurement system is able to track and measure the vehicle. These measurement systems often use lasers or retro-reflective markers for measuring the test subject. This work seeks to enable a measurement system with the capability to “see” or track the free-flight, flapping-wing UAS using Computer Vision (CV) techniques.

This chapter details the definitions and various methods used in CV. Computer vision is the capability of a computer to perceive objects [102]. Perception is defined as “the ability to see, hear, or become aware of something through the senses” [87]. Computer vision is about describing the world that humans see “in one or more images and to reconstruct its properties, such as shape, illumination, and color distributions” [116]. Therefore, CV is the implementation of “seeing” through an imaging sensor, and “becoming aware” through the ability to reconstruct world properties.

Figure 2.1 provides a rough time-line of the development of the CV techniques and applicable technologies used in this work [9, 23, 116]. Advances in image interpretation, control techniques and learning also make CV a popular area of research as demonstrated by the increasing number of technical papers on the subject.

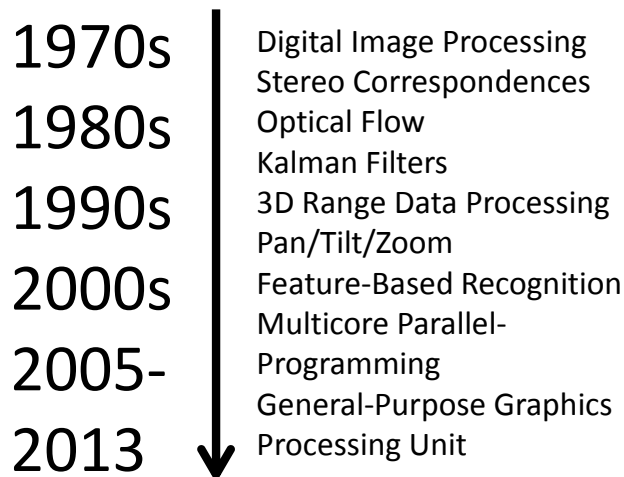


Figure 2.1: A time-line of the technology and techniques used in this work.

2.2 Computer Vision Object Detection

Detection is defined as “the action or process of identifying the presence of something concealed” (i.e., identifying an object in an image), and the term ‘object’ refers to “a material thing that can be seen and touched” [87]. Putting these two definitions together gives the following definition:

Object Detection - the action or process of identifying the presence of a material thing that can be seen and touched

Object detection can be broken down into components such as representation, features, shape, appearance, classification, etc. This section presents a hierarchy of object detection as a way to frame the plethora of techniques available. However, this is not all inclusive as the field varies widely with methods used. The focus of this section is to compartmentalize most object detection techniques into broad, mutually exclusive or overlapping subcategories and to form the basis for how objects are represented and what features are used.

2.2.1 Detection, Classification and Learning/Recognition. Detection starts with identifying that something or someone exists. The outlined methods of point detectors, segmentation, background modeling and supervised/unsupervised classifiers are those methods that determine existence. The term object recognition is often confused with object detection as it uses many of the object detection methods specified to “recognize” objects in an image. To complicate matters further, each of the four categories described use similar feature-based and dynamic-based components for detection. Some of these include frame-differencing where two or more frames are used to derive interest points, segments, background from foreground, etc.

2.2.1.1 Detection. We previously defined object detection as the action or process of identifying the presence of a material thing that can be seen and touched. Visualizing objects using a computer does not allow the ability to touch the object. Therefore, aligning the definition with the use of image sensors gives: the extraction of information from an image (or a sequence of) frames, processing of the information, and determining whether the information contains a particular object and its exact location in the image [74].

2.2.1.2 *Classification.* Classification is defined as the action or process of classifying something according to shared qualities or characteristics [87]. For example, examining an output y , say an animal, and arranging y into a particular class, say a dog, performs one level of classification. Another example would be identifying an object through a series of successive states until a given name is reached such as:

$$\text{Object} \succ \text{Animate} \succ \text{Animal} \succ \text{Mammal} \succ \text{Dog} \succ \text{Labrador} \succ \\ \text{Chocolate(Brown)} \succ \text{Male} \succ \text{Name} = \text{“Palmer”}.$$

The classification problem can be extremely long and computationally intensive in specifying every minute detail of a particular object of interest. The problem resorts to determining what level of classification is required and whether particular details of interest are needed even after an object is classified (e.g., color, eye color, type, sex, size, etc.). Therefore, feature generation, also referred to as feature extraction, must be used wisely to select the minimum amount of features required in order to best represent that which is needed thereby reducing computational costs and storage.

2.2.1.3 *Learning/Recognition.* Classification plays an important role in the learning process which results in being able to recognize. Recognition is the action or process of recognizing, in particular the identification of a thing or person from previous encounters or knowledge [87]. For example, the dog, Palmer, was classified from previously being introduced or by other means of identification. Distinctions have been made both visually and by other information such as knowing that Palmer lives next door (i.e., if a dog next door is a Labrador, then it is likely “Palmer”). Now the level of classification can be extended from:

$$\text{Object} \succ \dots \succ \text{Palmer}$$

and Palmer would be considered recognized. Making these distinctions is important in order to tackle the object detection and learning problems presented in CV. Levels of detail associated with particular problems result in determining what algorithms are effective for a given object and environment. Therefore, care must be taken in selecting object detection methods; specifically, how the object is represented and what features should be used to obtain the selected representation.

2.2.2 Representation, Features and Detection Categories.

2.2.2.1 *Representation.* Yilmaz, et al. [133] provided a foundation for describing the various representations through shapes and appearances. These representation and definitions are given in Figure 2.2.

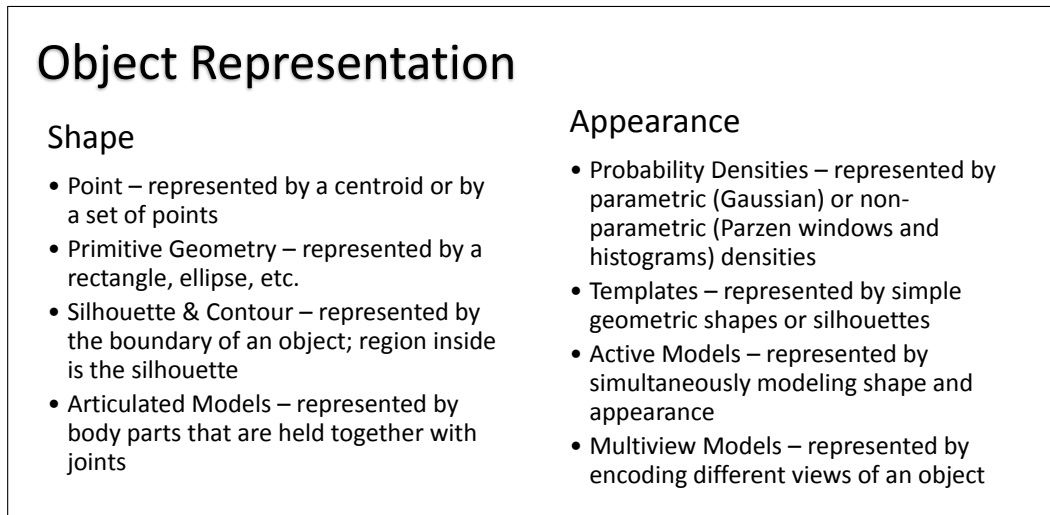


Figure 2.2: Object representations [133].

Depending on the problem, selecting an appropriate representation provides the means for detecting and tracking an object in an image.

2.2.2.2 *Features.* Features are used to represent an object in an image. For example, in order to represent an object using points or a set of points in an image, a feature is required for determining the location of that point or set of points.

The features and definitions are as follows [133]:

- Color - apparent color of an object is influenced by two physical factors: 1) spectral power distribution of the illuminant; 2) the surface reflectance properties of the object
- Edges - strong changes in image intensities defining object boundaries; less sensitive to illumination changes
- Optical Flow - dense field of displacement vectors defining the translation of each pixel; computed using the brightness constancy of corresponding pixels in consecutive frames [Horn and Schunk, 1981]
- Texture - measure of the intensity variation of a surface which quantifies the properties such as smoothness and regularity; less sensitive to illumination changes

The most popular feature for representing an object is color; however, many applications use a combination of features [133].

2.2.2.3 Detection Categories. There are combinations of various subcategories of both detection and tracking techniques. For example, a combination of background subtraction and shape for object detection could be used while using a variant of the Kalman Filter (KF) for tracking. Therefore, generalizations are made for components of object detection into one of the following subcategories: point-based, segmentation-based, motion-based (background modeling), or learning-based (supervised classifiers)(see Fig. 2.3). Yilmaz, et al. [133] specifically describe these four categories as point detectors, segmentation, background modeling, and supervised classifiers. Common object detection methods use information in a single frame such as the point-based and segmentation-based methods. The motion-based and learning-based detectors make use of temporal information computed from a sequence of frames [133]. This literature review provides recent research in the four subcategories (see Fig. 2.3).

Point Detectors: Recent Work and Applicability

Point detectors often take into account multiple attributes of their surroundings through local patches, magnitudes and gradients, intensities, color, etc [116]. The information about a point is contained in a descriptive vector, also called a descriptor, which is commonly high-dimensional (i.e., typically greater than or equal to 64 dimensions for some

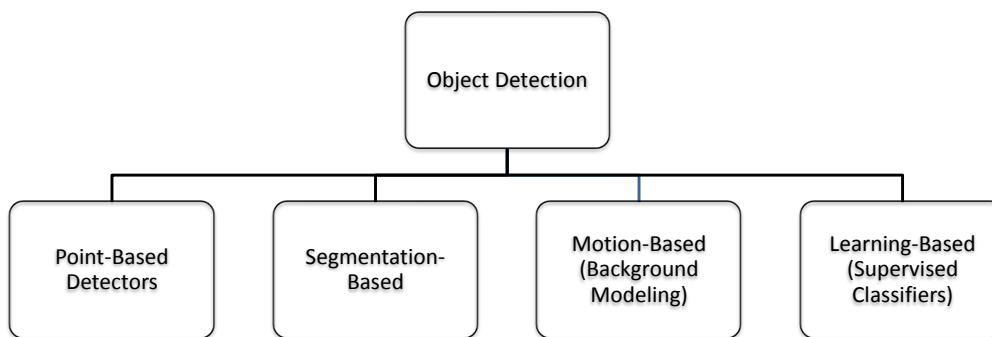


Figure 2.3: General object detection hierarchy.

of the more robust feature detectors). Examples of point detectors are Moravec’s interest point detector [Moravec 1979], Harris interest point detector [Harris and Stephens 1988], Kanade-Lucas-Tomasi (KLT) detector [Shi & Tomasi 1994], and Scale-Invariant Feature Transform (SIFT) detector [80, 133]. More recent point detectors include the Histogram of Oriented Gradients (HOG) [93], Gradient Location Orientation Histogram (GLOH) [88], contrast-normalized patches [Brown, Szeliski, and Winder 2005], covariance patches or regions [121], DAISY [118], and the Speeded-Up Robust Features (SURF) detector [15, 116]. Edges are found by observing “boundaries between regions” with different attributes [116]. Edge detectors are composed of filters or gradient operations that bring out the edges. Canny and Sobel are two types of edge detectors. These detectors are not only suitable for point detecting, but they are “well-suited to describing object boundaries and man-made objects” for segmentation as well [116]. Given a single point location in an image, its local surroundings are used to represent information towards detecting similar points in successive or unrelated frames.

Many approaches for point detection use a combination of methods and/or image processing developments. The CV community continues to expand and elaborate on one another’s ideas to develop improved algorithms. This section explains some of the latest developments.

Viola and Jones [124] made use of the integral image, similar to the summed area table, to speed up processing. Others have developed new algorithms using this technique [15, 121].

The integral image eliminates the need to compute a multi-scale image pyramid reducing initial image processing significantly.

Another method for reducing computational time includes a procedure commonly used in Artificial Intelligence (AI) known as Principal Component Analysis (PCA). Using an entire dataset for detection is time consuming and thus PCA is used to extract the most effective components, the eigen components through Singular Value Decomposition (SVD), thereby reducing the dimensionality of the data. Of course, careful selection of how many principal components to keep is based upon user-selection or other adaptive parameters. Ke and Sukthankar [70] applied PCA to SIFT and produced results that were more accurate and significantly faster than the standard SIFT local descriptor.

Dalal and Triggs [93] introduced the HOG algorithm for human detection. This paved the way for others to foster point detector development by applying various learning techniques. Wijnhoven, et al. [128] developed a stochastic gradient descent method for training datasets for object detection using HOG descriptors in order to take advantage of its simplicity and good performance. Tuzel, et al. [120] use HOG descriptors for learning on Lie Groups and detecting objects in various poses.

Developed in 2010, DAISY was inspired by SIFT [80] and GLOH [88] by using an Expectation-Maximization (EM) algorithm “to compute dense depth and occlusion maps from wide baseline image pairs”. DAISY’s local descriptor is developed by performing sums of Gaussian convolutions and is more efficient computationally for dense-matching purposes [118].

Matching several points on a flapping wing from one frame to the next can be extremely difficult. Being able to distinguish these points, especially in the presence of blur due to high flapping frequencies and ability to capture focused images, poses many challenges. Using point matching techniques as described here for determining wingtip location is critical towards the success of a Pan/Tilt Unit (PTU)/Pan/Tilt/Zoom (PTZ) CV system. SIFT [80], SURF [15] and covariance [121] are the point detecting methods tested in Chapter **IV** for the post-processing activities of determining wingtip location for a given image.

Segmentation, Recent Work and Applicability

Image segmentation aims to partition, or segment, objects or regions from the background. Szeliski [116] defines image segmentation as the task of finding groups of pixels that “go together”. Some popular segmentation methods include: active contours, level sets, graph-based merging, Mean Shift (MS), texture and intervening contour-based normalized cuts, and binary Markov Random Field (MRF) solved using graph cuts [116]. Thresholding is another simpler way of segmenting images. Thresholding segments objects in an image by using features such as color and intensity; however, the challenge is in selecting an optimum threshold in order to obtain objects of interest [107].

Two problems are typically addressed when developing segmentation algorithms [Shi and Malik 2000] [133]:

1. Criteria for a good partition - measuring the goodness of an image partition
2. Method for achieving efficient partitioning - reducing computational complexity of a good partition

The MS clustering approach uses a joint spatial and color-space approach $[x, y, l, u, v]$ where $[x, y]$ is pixel coordinate and $[l, u, v]$ is the color representation. The graph-cut approach takes the pixel of an image and partitions pixels into regions based on weighted edges. The total weight between edges is called a cut and are typically calculated using color, brightness or texture similarity. The goal is to find partitions that minimize a cut [133]. One form of an active contour is called *snakes* by its inventors (Kass, Witkin and Terzopoulos, 1988). It is an “energy-minimizing, two-dimensional spline curve that evolves (moves) towards image features such as strong edges” [116]. Rather than observing objects in regions, the *snakes* separate regions by lines or edges. For example, the outline of a coffee mug captured by an edge detector followed by a snake active contour would change according to the pose of the coffee mug.

Fukunaga and Hostetler [49] proposed the MS procedure in 1975, and it re-emerged in 1995 with Cheng’s work, “Mean Shift, Mode Seeking and Clustering” [34]. Comaniciu, et al. [34] provide a method for detecting a large variety of objects with different color/texture patterns with the use of a metric derived from the Bhattacharyya coefficient. This is further

developed by using an isotropic kernel for target representation and localization. Ma, et al. [83] seek to improve upon the color-based MS tracking algorithm by integrating color and motion cues. An adaptive approach attempts to account for large variations in pose, appearance, and background color similarity. Motion cues are determined through histogramming successive frame differences and experiments show robustness of the integration algorithm over the traditional MS tracker. Zivkovic, et al. [140] use an ellipse and its appearance by histogram features for object detection. Yilmaz [132] introduces the use of asymmetric kernels to the mean shift tracking framework “to reduce the estimation bias due to a better representation of the underlying density.”

A region merging technique of segmentation combines adjacent regions of pixels based on average color differences and specific thresholds. This process segments the image into pieces called *superpixels* and is useful for pre-processing images for faster and more robust stereo matching, optic flow and recognition [116]. Fulkerson, et al. “identify and localize object classes in images” using this pixel-level categorization and exceed previously published state of the art. A method called *quick shift* is used for superpixel extraction, and classification is performed using a bag-of-features approach using SIFT descriptors on regions defined by superpixels [50]. Arbelaez, et al. [8] state that their contour detection and image segmentation surpass state-of-the-art algorithms using their techniques. An extensive evaluation shows that their algorithm, the globalized Probability of boundary-Oriented Watershed Transform-Ultrametric Contour Map (gPb-OwT-UCM), outperforms the region merging, MS, multi-scale normalized cuts and segmentation by weighted aggregation algorithms.

Segmentation techniques often take more time when operating with the whole image. However, providing a search region with these techniques locates and segments fast moving objects for tracking in real-time. The MS algorithm is one of the real-time tracking techniques tested in Chapter IV.

Background Modeling and Optical Flow: Recent Work and Applicability

Background modeling seeks to establish a means for separating foreground pixels from background pixels. A temporal sequence of frames is typically used to establish background from foreground pixels and to provide some robust processes for interpreting the background.

In a dynamic background, the attempt to establish a model is difficult, and therefore, background modeling is most commonly used with static cameras. One might assume that any movement in the image would be readily identifiable; however, various image characteristics prevent the task of background subtraction (BS) and background modeling this simple approach. Piccardi [96] provides an examination of background modeling with the following list (from simple to complex) along with a brief summary of each:

- Running Gaussian average - based on ideally fitting a Gaussian Probability Density Function (PDF) on the last n pixel's values and updating using a cumulative average with each new frame. Speed and low memory requirement are advantages of this approach.
- Temporal median filter - use of the median value of the last n frames as the background model. It has been argued that sub-sampling with respect to the original frame by a factor of ten will still provide for an adequate background model. The main disadvantage of a median-based approach is that its computation requires a buffer with the recent pixel values. It also does not account for a rigorous statistical description and does not provide a deviation measure for adapting the subtraction threshold.
- Gaussian Mixture Model (GMM) - multi-valued background model able to cope with multiple background objects; an image model providing a description of both foreground and background values.
- Kernel density estimation (KDE) - an approximation of the background PDF provided by the histogram of the most recent values classified as background values. The histogram, as a step function, may provide poor modeling of the true, unknown PDF so the KDE guarantees smoothness and continuity of the histogram.
- Sequential KD approximation - the mean shift (MS) vector is an effective gradient-ascent technique able to detect main modes of the true PDF directly from the sample data with a minimum set of assumptions. The sequential KD approximation looks to make use of this by finding a low mean integrated squared error.
- Co-occurrence of image variations - exploitation of spatial co-occurrence of image variations. Neighboring blocks of pixels belonging to the background should experience

similar variations over time (however, false detections occur with borders of distinct background objects). Instead of working with pixel resolution, blocks of $N \times N$ pixels are treated as an N^2 -component vector trading off resolution with better speed and stability. The advantage is that the model is based on variations making the model more robust to illumination changes.

- Eigenbackgrounds - based on an eigenvalue decomposition, but applied to the whole image instead of blocks. Avoids the tiling effect of block partitioning. It can be subject to variations improving its efficiency; however, there is uncertainty in what should be part of the initial images and model updating over time.

In addition, Piccardi [96] provides methods and performance analysis in Table 2.1. The speed and memory column variables of Table 2.1 are described as follows: n_s is the number of samples (n_s is sub-sampled from the full sample set, n), m is the number of Gaussian distributions used; $m + 1$ is the number of modes of the approximated PDF, $(8n/N^2)$ is where n accounts for searching the nearest neighbors amongst the n variations and N^2 spreads the cost over the pixels in a block; (nK/N^2) is where n is the number of variations in the training model and K their dimension, and M is the number of the best eigenvectors.

Table 2.1: BS methods and performance analysis [96].

Method	Speed	Memory	Accuracy
Running Gaussian average	$O(1)$	$O(1)$	Low/Medium
Temporal median filter	$O(n_s)$	$O(n_s)$	Low/Medium
Mixture of Gaussians	$O(m)$	$O(m)$	High
Kernel density estimation (KDE)	$O(n)$	$O(n)$	High
Sequential KD approximation	$O(m + 1)$	$O(m)$	Medium/High
Co-occurrence of image variations	$O(8n/N^2)$	$O(nK/N^2)$	Medium
Eigenbackgrounds	$O(M)$	$O(n)$	Medium

Optical flow is “the distribution of apparent velocities of movement of brightness patterns in an image”. Pixel velocity is represented by two components and are determined by pixel content change. The rate of change of image brightness and smoothness are used as constraints [60]. This method performs motion analysis per pixel [116].

Zivkovic [139] uses a pixel-level approach of BS along with an efficient adaptive algorithm using Gaussian mixture probability density. GMM parameters and number of components of the mixture are constantly updated for each pixel. Camplani and Salgado [28] use GMM BS for object detection in a multi-camera scheme. Adaptation strategies linked to processing requirements, scene activity, and available resources are used and processing requirements are dynamically tuned to achieve desired results. Vijverberg, et al. [123] propose the use of motion vectors to develop a consistent motion mask (CMM) combined with a background segmentation technique. They also develop a Block-Target Association (BTA) algorithm to assign blocks of pixels to targets for tracking occluded areas. State-of-the-art target detection is not achieved, but reduced resource constraints and acceptable performance is attained. “Motion vectors provide good features for target detection and tracking with limited additional costs” [123]. Bhaskar, et al. [18] propose using a GMM to cope with multi-modal background distributions. Cluster GMM gives better performance than the pixel-level GMM and so the authors pursue cluster GMM-based symmetric alpha stable (Cluster BS- $S\alpha S$) distributions. “Cluster BS reduces clutter while $S\alpha S$ distributions help with handling the dynamic illumination changes in a scene which models moving backgrounds better due to a heavy tail” [18]. Experiments show that Cluster BS- $S\alpha S$ maximizes the proportion of correctly classified pixels and minimizes misclassification compared to the pixel-based GMM. Another BS scheme uses a combination of the output of all sensors to obtain a more accurate and more robust estimate of moving and stationary pixels. Entropy of separation helps determine thresholding parameters. Bondzulich, et al. [24] report that, like other BS algorithms, their algorithm is susceptible to: 1) poor performance when noisy or low-contrast imagery is available from both sensor streams; 2) excessive sensitivity to platform motion or general background motion; and 3) low sensitivity to small targets in the video frame.

Bouget [25] uses the Lucas-Kanade optical flow equation with an image pyramid representation to find a pixel’s location via a pixel displacement vector. Large pixel motions are achieved while keeping the integration window “relatively small”. Bauer, et al. [14] use the local Lucas-Kanade method with the global Horn-Schunk technique to produce a dense and robust optical flow field. Baker, et al. [12, 13] propose a new set of “benchmarks and evaluation methods for the next generation optical flow algorithms” by contributing data to

test different aspects of optical flow algorithms. These include (1) sequences with non-rigid motion where the ground-truth flow is determined by tracking hidden fluorescent texture, (2) realistic synthetic sequences, (3) high frame-rate video used to study interpolation error, and (4) modified stereo sequences of static scenes. Sun, et al. [115] use this data to discover that “classical” flow formulations perform well with modern optimization techniques such as median filtering. They find that applying a median filter to intermediate flow values during incremental estimation and warping produces the most significant improvements, the heuristic improves the accuracy while increasing the energy of the objective function, and information about image structure and flow boundaries can be incorporated into a weighted version of the non-local term to prevent over-smoothing across boundaries. Liu [78] presents *Human-Assisted Motion Annotation* to obtain more accurate motion estimates than automated methods, *Motion Magnification* useful for “visualizing and measuring small, subtle deformation of objects of interest”, *Analysis of Contour Motions* allowing for completion of contours, and *SIFT flow* to establish correspondence between scenes. Jin-Jie, et al. [63] propose using color images in optical flow, specifically the hue, saturation, and intensity (HSI) or hue, saturation, and value (HSV) color space, along with a weighted matrix to obtain a vector field. It is asserted that their proposed method is superior to the traditional optical flow methods supported via experimental results.

Background modeling and optical flow methods are quite popular. The GMM proposed by Zivkovic [139] provides a useful tool for determining motion within a static scene. This method is used to perform real-time, automatic object detection when a UAS moves through the center of the image. The Pyramidal Lucas-Kanade (PyrLK) optical flow technique is tested in Chapter IV and then is used for real-time tracking in Chapters V-VI of this research.

Supervised Classifiers, Recent Work and Applicability

More recent developments in feature tracking use learning algorithms with classifiers that recognize objects in an image [116]. Supervised classifiers refer to using learning algorithms that have the presence of an outcome variable to guide the learning process to train classifiers [58]. Recent work is described below.

- Neural Networks

El-Bakry, et al. [42] use feature-based classification and object detection using efficient neural networks. The purpose is to reduce computational steps required during the search process. Focus is primarily on face detection and recognition. First, a face must be detected in an image, and then it can be recognized. Weight normalization versus sub-image normalization requires fewer computational steps and is done off-line before the search process with neural networks. Their work also goes to the length of stating that incorrect equations have been used in a previous work regarding the multilayer perceptron (MLP) algorithm for fast face detection specifically for cross-correlation between the input image and the weights of the neural networks. Faster face detection is reported by applying cross-correlation in the frequency domain between the probe image and the normalized input weights of the neural networks.

- Artificial Neural Network (ANN), Support Vector Machines (SVM), and Naive Bayes Classifier (NBC)

ANNs, SVMs, and Bayesian statistics and Dempster-Shafer theory of evidence are referred to as different kinds of classification and inference methods. Classification rates for all of the methods vary given different datasets and tuning of each algorithm. Starzacher and Rinner [111] demonstrate effective feature fusion using ANNs, SVMs, and NBCs for select scenarios and tuning on an embedded test platform. Given appropriate parameters and addition of heterogeneous features and decisions, increased classification rates are possible.

- AdaBoost Learning

The AdaBoost-based algorithm is considered one of the most efficient algorithms in object detection where boosting is the most widely used ensemble method [74] [102]. The AdaBoost algorithm generates hypotheses by successively re-weighting the training examples [102]. AdaBoost utilizes a small number of weak-classifiers, which are then used to construct cascades of strong classifiers. The combination of the strong classifiers in a cascade results in high accuracy rates and computational efficiency. The weak classifiers most commonly used with AdaBoost are Haar-like features; fixed-size images which contain a small number of black and white rectangles. Kyrkou and Theocharides [74] incorporate the AdaBoost-based algorithm along with Haar-like

features on a Field Programmable Gate Array (FPGA) to detect objects in large images (up to 1024x768 pixels) with frame-rates that can vary between 64 – 139 frames per second (fps) for various applications.

- Stochastic Gradient Descent Learning

Wijnhoven, et al. [128] use the appearance and spatial information of the HOG classifier for detection. The sliding window classifier presented in [93] is considered here due to its simplicity and good performance with respect to pedestrian detection. In each image, positions are classified as either object or background. It is shown in this work that there are speedups of two to three orders of magnitude in computation time in using the proposed Stochastic Gradient Descent (SGD) learning method.

- Learning Lie Groups

Tuzel, et al. [120] present a model that can accurately detect objects in various poses through learning the motion model on the Lie Algebra. The appearance of the object is described by several orientation histograms. A binary tree model is used against a training set for describing motion vectors on the Lie Algebra. The motion group is learned and reduces estimation errors with several experiments demonstrating superior performance.

- P-N Learning

Kalal, et al. [67] present a novel algorithm using P-N learning which refers to a learning process guided by positive (P) and negative (N) constraints which restrict the labeling of the un-labeled set. It is demonstrated that “an accurate object detector can be learned from a single example and an un-labeled video sequence.” The learning runs on-line and has been compared with relevant tracking algorithms for which they state that their algorithm achieves state of the art.

2.3 CV Object Tracking

2.3.1 Tracking. Object tracking is easily confused with object detection because of its reliance upon the other. Detection is required in order to track something. The term track means “to follow the course or trail of (someone or something), typically in order to find them or note their location at various points” [87]. The following definition is used:

Object Tracking - to follow the course or trail of a material thing that can be seen and touched, typically in order to find its location at various points

Tracking is often associated with radar and sonar in military applications. CV-related applications are becoming ubiquitous with the ability to detect and classify objects in images with increased accuracy. After detection of an object (i.e., the information obtained using the representations and features described previously), measurements are taken and predictions are made based upon tracking algorithms. These object tracking algorithms are broken down into three major categories: point tracking, kernel tracking and silhouette tracking (see Fig. 2.4) [133]. Definitions for the three sub-categorizations of object tracking are presented in Table 2.2.

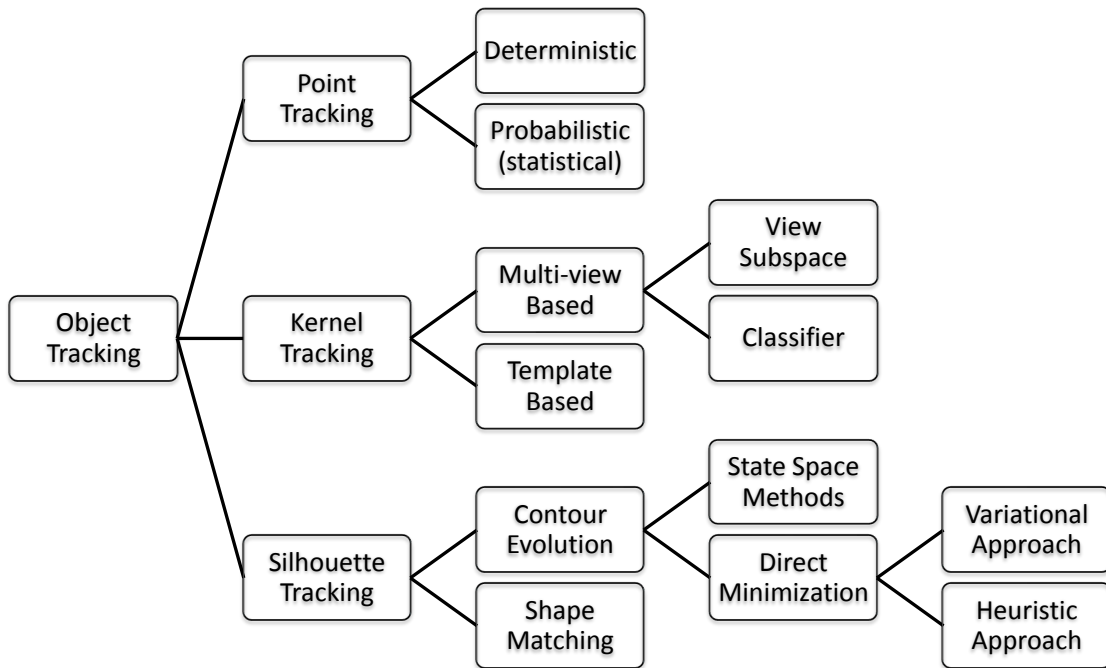


Figure 2.4: General object tracking hierarchy.

Object tracking algorithms are typically used to filter, predict or smooth the location of the object. Applying the KF is a prime example of filtering the object location based on measured and predicted locations.

Table 2.2: Object Tracking Category Definitions

Point Tracking	Objects detected in consecutive frames are represented by points - requires detection of object in every frame
Kernel Tracking	Refers to the object shape and appearance - usually in the form of parametric transformation such as translation, rotation, and affine
Silhouette Tracking	Performed by estimating the object region in each frame - uses information encoded inside the object region

2.4 Computer Vision (CV) Using Multiple Cameras

2.4.1 Target/Sensor Cases. The sensors in a CV system deal specifically with imaging sensors, however, other sensors, such as laser, radar, and sonar, may be considered when concerned with an object detection and tracking problem. Similarly, motion and object identification are required in order to gain structural characteristics with a PTU system. Below is a list of cases to consider when determining object detection and tracking for a given problem [36]:

- Single target, single sensor
- Single target, multiple sensors
- Multiple targets, multiple sensors

The methods covered in this background deal with two of the above cases. In developing a PTU or PTZ CV system for measuring structural characteristics of an UAS, two cases apply: single target, single sensor or single target, multiple sensors. Sensors in a CV system are imaging sensors. Multiple cameras are dealt with in this section and specifically, combining image sensors to better interpret or acquire an object’s position is the focus. Combining data from many sensors deals with the area of data fusion and techniques described in the following section.

2.4.2 Data Fusion Core Techniques. Though data fusion is a broad, complex topic, in essence it is “combining data to refine state estimates and predictions” [113]. Accurate measurements are required to obtain useful data for processing and understanding and data fusion provides methods for reducing noise and error in sensing devices. Appendix A contains a thorough background on fusion and fusion models. Many of the methods used

for CV object tracking also employ similar methods such as the KF. Fusion methods are explored here to refine estimates based on multiple sensor detection and tracking, specifically, to gain better correspondence estimates between images during post-processing activities.

The core techniques used in object tracking vary from using linear-quadratic estimation to probabilistic methods. These are as follows [36]:

- Kalman Filter
- Probabilistic Data Association
- Joint Probabilistic Data Association
- Multiple Hypothesis Testing

There are generally four types of tracking scenarios for motion and error measurement of the model:

1. Linear, Gaussian
2. Nonlinear, Gaussian
3. Linear, non-Gaussian
4. Nonlinear, non-Gaussian

The KF assumes a linear dynamical system with Gaussian distributions for error and measurement terms. The KF is often referred to as a linear-quadratic estimation problem. Multiple variants of the KF include its nonlinear counterparts such as the Extended Kalman Filter (EKF) which uses a Taylor series expansion for linearization, and the Unscented Kalman Filter (UKF) which uses a deterministic sampling approach to improve upon the means and covariances of the underlying system. The Particle Filter (PF) generalizes the KF by providing the optimal Bayesian estimate through the use of particles. These particles are randomly placed to provide a distribution of the space. Probabilistic Data Association (PDA) is used in multi-sensor, single target tracking and Joint Probabilistic Data Association (JPDA) and Multiple Hypothesis Testing (MHT) are used in multi-sensor, multi-target tracking. These methods use probabilistic approaches of combining or fusing data. Similar methods are actually used to interpret the data before fusion as well.

Single sensors have their weaknesses and one of them is the lack of redundancy in case of failure. Combining two or more heterogeneous sensors can be advantageous for multiple reasons. One of them already mentioned is redundancy and another reason includes using each sensor's accuracy to give a more accurate, more complete result [44, 71].

See Appendix A for more information on data fusion and data fusion models.

2.4.3 Core Techniques Explored through Recent Work. The KF and its variants were discussed in the previous section. This section provides more details of filters, some of their applications, and other methods relating to the KF and PF.

Kirchmaier, et al. [71] use Particle Swarm Optimization (PSO) to fuse audio data with Continuously Adaptive Mean Shift (CAMShift) for directional information and position in performing three-dimensional (3D) object tracking. The PSO tracking system has the advantage that a movement prediction can be considered, and the tracking will not fail with abrupt movements, as the particles not belonging to this subset provide other possible locations.

Gadsden, et al. [51] propose using the Smooth Variable Structure Filter (SVSF) [Habibi, 2007] for fusion and object tracking. An Interacting Multiple Model Filter (IMMF) combined with the SVSF estimation strategy is presented as an effective method for non-linear target tracking.

The algorithm developed in [105] adaptively selects an appropriate filtering technique such as the KF and different types of nonlinear filters such as the EKF, the UKF, and the PF. It does so based upon a sliding window of the Normalized Innovation Squared (NIS). A measure of filter performance is shown to be sufficient to adaptively switch between algorithms based upon the current state of the system. Adaptively selecting appropriate filtering methods reduces computational costs and root mean square error [105].

Bae, et al. [11] introduce a tracking algorithm based on the morphological operation and dynamic filtering called the Highest Probability Data Association Filter. High computational costs are often associated with PDA, JPDA, and MHT approaches due to their consideration of all possible events. This method is used for real-time, cluttered environments. An advantage is the ability to retrieve temporarily hidden targets using their

track-termination and re-detect method. One limitation of their method is that a long-term period of a hidden target cannot be retrieved.

The KF can be applied to all of the above cases, however, significant degradation in performance occurs for those cases that are nonlinear, non-Gaussian. The PF has the ability to model non-Gaussian, nonlinear distributions more effectively than the KF [94, 95, 101]. But the PF is most notably burdened with re-sampling of particles as well as particle degeneracy.

Nelson and Roufarshbaf [94] use a PF-like approach by approximating the Bayesian inference solution using a tree-search approach for target tracking. All possible sequences of target states are mapped to a path in a search tree and each path represents a possible sequence of target states. The objective is to avoid the problems of degeneracy, resampling and elimination of possible likely paths as experienced with a PF.

Rincon, et al. [101] propose a double-tracking strategy using PF and UKF techniques for reliability, position and velocity measurements. The JPDA method estimates target player positions on a football field, and makes final decisions based upon all of the information [101]. Joint Probabilistic Data Association (JPDA) uses color and motion probabilities to generate a Color Probability Image (CPI). A GMM generates each object's distribution and the importance function acts as a mask by extracting the main colors of the object and detecting them in the full image or in the zone surrounding the estimated position. The multi-camera situation uses the UKF solves for a difficult situation, such as occlusions and noise. The UKF is used against the plan in which all of the measurements from each camera are mapped. The plan is a map of the football field, and a homographic matrix for each camera is used to transform points from an image to the coordinate system of the plan. A height estimator is also used with respect to the number of pixels that represent the average height of a person at a location in the image. The main contributions include the computation complexity reduction through the use of an integral image and the CPI mask, the extension of the UKF to multi-camera applications, and the feedback procedure, which increases the robustness and accuracy of both tracking systems.

Lampinen, et al. [110] introduce the Rao-Blackwellized PF for multi-target tracking. They first assume that “finite sets of motion models can adequately address target tracking

issues” and select the IMM-F due to its performance. Variable rate techniques allow for changes in motion models. The kinematic state remains fixed for a period, but not necessarily the measurement period, and different types of motion are allowed. Adaptive segment lengths are determined via the measurements along with their durations.

Pei, et al. [95] use color information and PCA features to detect an object or objects. Tracking performance is unstable only using color information and using multiple features increases tracking efficiency [95]. Specifically, the PF is tuned to the multiple features with a likelihood measurement. Experimental results show the efficiency of the algorithm which reduces the effects of illumination and effects of color properties between foreground and background [95].

Mohammadi, et al. [90] present the problem of tracking moving objects in an environment observed by multiple, un-calibrated cameras with overlapping Field of Views (FOVs). Adequate coverage is addressed, in addition to observing critical areas, to provide robustness against occlusion. Understanding an object’s history of actions in an environment is necessary to establish correspondences between different views. Technical problems in multiple cameras include [90]: installation, calibration, object matching, switching, data fusion, and occlusion handling.

2.4.4 PTU/PTZ Applications. Contributions using PTZ cameras continue to increase as real-time object detection and tracking algorithms improve. Several works discuss efficiently tracking and maintaining an object of interest in real-time assuming camera motion. Camera motion can either be undesired, requiring image stabilization [109], or desired, such as for tracking or building a background mosaic [17]. The principle involved is the same: finding the transform from an image to a reference [33]. Transforms are typically found between frames by fitting or matching point correspondences [29], phase correlation [10] or block matching [69]. An alternate technique is to use a projective texture as a reference [33]. The use of the known camera motion is surprisingly not used more often. Though the exact model is developed [68], often simplified models are used, such as the small angle approximation [92]. Algorithms that do not require known camera motion have the advantage that they can be used with hand-controlled cameras or other moving mounts. However, estimating motion comes at the cost of increased error, increased processing time,

the object appearing smaller in the image (because the background must be viewed) and requiring texture, or the lack of texture, on the background and foreground.

An alternative to compensating for tracking the camera's motion is to use one or more static cameras to direct the moving camera. This has the advantage that moving objects are simpler to detect with a static camera. Applications include controlling a remote vehicle [32, 84] or following sporting events [53]. The cost of adding a static camera or cameras is that the FOV must cover the range of the moving camera for this alternative to be effective increasing setup time and calibration. However, stationary, smart cameras can be used to self-calibrate with fast setup and reasonable accuracy [108]. The work presented here uses a single pan/tilt camera to track a moving object.

As processing power has developed, new and existing tools have been applied in real-time, and have been used for both tracking an object within a static camera's view or tracking an object with a moving camera.

Heuristics can be used to design detectors for increased accuracy or speed. General-purpose feature descriptors can be used with training data, tuning or selection for a wide variety of objects. Some specific applications using a pan/tilt camera include tracking the color on a ball [5, 54], human positioning/tracking [59, 72, 131], faces [4, 5, 72], and moving vehicles/ships [4, 29, 76, 138]. Various methods are used in these applications for detecting objects such as MS and its variant CAMShift [54, 59, 72, 76, 138], background modeling using a GMM [59, 76], skin color segmentation [5], training/learning an object's appearance using PCA and sum of squared difference (SSD) [4], and template-based detection using SURF [29]. The general-purpose methods often require selection, a pre-defined capture area (i.e., alert zone), training, and/or tuning. The cost of using application specific algorithms include decreased system versatility, increased setup/calibration time, and limitations on feature changes such as illumination, object pose, shape, size, etc.

Features can also be non-specific, such as motion-based features, which are useful in general-purpose applications. General-purpose (motion-based) tracking allows for more flexibility in handling multi-purpose problems and allows tracking of an object of any size or shape [92]. This is well-suited for a metrology system capable of tracking a multitude of objects without having the concern of training, or changes in illumination or appear-

ance associated with application specific methods. Short duration/sudden events are easily accommodated. Operator delay is removed since object selection and re-selection are not required.

A single PTZ camera, multiple PTZ cameras, and a combination of static camera(s) with PTZ camera(s) have been broadly categorized into three areas of research: 1) calibration, 2) tracking, and 3) 3D localization. These categories are shown with various techniques in Table 2.3. Note that each are dependent upon each other, 3D localization requires tracking and tracking assumes a calibrated PTZ system. Table 2.3 provides a sampling of the variety of techniques in the area of CV PTZ cameras.

Table 2.3: Pan/tilt/zoom camera applications.

	PTZ Camera	Multiple PTZ Cameras	Static Camera(s) & PTZ Camera(s)
Calibration			
intrinsics as a function of zoom	[129]		
extrinsics as a function of rotation center		[31]	
using SURF features to create a mosaic image		[134]	
using SIFT features with displacement constraint	[61]		
using SIFT features to create a mosaic image	[136]		
using Givens rotations and an image pair	[65]		
using geometric restrictions (motion re-calibration)	[117]		
intrinsics and extrinsics using corners and Nelder-Mead simplex optimization		[30]	
using bundle adjustment	[64]		
using statistically optimal error function		[65]	
Tracking			
using CAMShift with background/foreground properties	[76]		
using CAMShift with a Kalman Filter	[54] [138] [5] [98]		
using CAMShift and window differencing technique	[135]		
using region covariance feature matching	[137]		
using background compensation and 1-D feature matching/outlier rejection	[114]		
using background subtraction (classify post-processing)			[59]
using background subtraction with mosaic	[10] [69] [136]		
using upper body and face detector	[72]		
using complementary features with particle filtering	[4] [39]		
using Multiple Hypothesis Testing		[27]	
using planar structure from motion	[119]		
3D Localization			
localize using reactive zoom*; perspective and affine	[119]		
localize using LUT for rectification transformations		[73]	
localize using a mosaic	[130]	[125]	[134]

It is sought to compare related work using standard system characteristics such as image resolution and frame-rate. In addition, many CV algorithms require training and

design for occlusion handling. Therefore, Table 2.4 provides a comparison of related work by resolution, frame-rate, training and occlusion handling of related PTZ systems.

Table 2.4: Related work summary of pan/tilt system resolution and operating frame-rate.

Ref.	Resolution (width x height)	frame-rate (pan/tilt delay)	Training	Occlusion Handling
[17]	320x240	11 fps (unknown)	No	No
[33]	Not available	22 fps (25 ms)	Yes	No
[29]	600 x 800	~30 fps (unknown)	Yes	Partial
[10]	Not available	8 fps (unknown)	No	No
[69]	320 x 240	5 fps (unknown)	Yes	No
[92]	Not available	unknown (unknown)	No	Partial
[98]	320 x 240	Not available	No	Partial
[131]	640 x 480	30 fps (unknown)	No	No
[4]	1024 x 768	20-25 fps (~65 ms ^a)	Yes	Full
[138]	320 x 240	25 fps (unknown)	No	Partial
[76]	768 x 576	25 fps (unknown)	No	Partial
[54]	320 x 240	10 fps (unknown)	No	Partial
[72]	352 x 255	20-25 fps (100 ms)	No	Partial
[5]	Not available	unknown (unknown)	No	Partial

^aAkhloufi [4] performed parallelized CV-tracking; delay based on the same pan/tilt model used in this research

2.5 Summary

This background provided an overview of object detection, object tracking, CV with multiple cameras, and PTU/PTZ applications . A basis is formed for understanding these activities, what methods are presently available, and what challenges may be encountered.

The PTZ applications, tracking and multiple cameras reviewed in this background form a basis for further work in developing a PTZ CV system for performing state measurements of an in-flight, flapping UAS. This research furthers Magree’s work [85] by incorporating CV techniques to automatically track a UAS, determine its 3D position, and to command a desired focal length for obtaining high-resolution, high frame-rate images.

The common CV object detection techniques include MS, optical flow, SIFT, SURF, and various forms of background subtraction. The data fusion technique commonly used is the KF. Therefore, these CV object detection techniques are explored in Chapter **IV**, and the KF is implemented as part of the work described in Chapters **V** and **VI**.

III. Design Considerations

This chapter presents the design considerations of the Computer Vision (CV) Pan/Tilt/Zoom (PTZ) system. First, a description is provided of the equipment and software used for this work. This is followed by the assumptions made for this work. Lastly, the objectives and exit criteria are provided as a foundation for the remaining chapters.

3.1 Research Platform, Devices, and Software

This section details the software and devices used to develop the CV tracking and measurement system.

3.1.1 Hardware. A single computer, Dell M6600 Precision (see Fig. 3.1) is used to control the Pan/Tilt Units (PTUs) and cameras. Table 3.1 provides the computer specifications.

Table 3.1: Computer Specifications

Manufacturer:	Dell [©]
Model:	M6600 Precision
Processor:	Intel [®] Core [™] i7-2820QM @ 2.30 GHz
Installed memory (RAM):	16.0 GB
Graphics Card:	NVIDIA Quadro 4000M
Dedicated Graphics Memory:	2.0 GB RAM
Processor Cores:	4
System type:	Windows 7, 64-bit Operating System



Figure 3.1: Dell M6600 Precision

The PTU specifications are listed in Table 3.2. Figure 3.2 shows an image of the PTU.

Table 3.2: PTU Specifications

Units:	2
Manufacturer:	FLIR [®] Systems, Inc.©
Model:	PTU-D46-17
Max Payload Weight (nominal):	6 lbs
Weight:	3 lbs (not including 8oz controller)
Position Resolution (°) ^a :	0.05143 (1/2 step), 0.0123 (1/8 step)
Max/Min Pan Speed (°/sec):	300 / 0.0123
Max/Min Tilt Speed (°/sec):	300 / 0.0123
Pan Range (°):	±180
Tilt Range (°):	+31/-80
Height (in)	5.2
Operating Voltage (VDC):	9-30
Operating Temperature (°C):	-20 to +60

^aPan-tilts support 1/2, 1/4, and 1/8 microstep modes as well as autostep mode, all software configurable. Quoted resolutions are for 1/8 step mode. Firmware default is 1/2 step. Torque and speeds are somewhat reduced at 1/4 and 1/8 step modes.

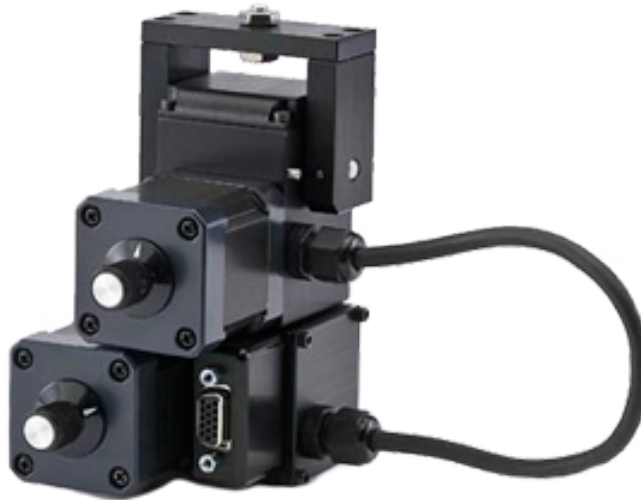


Figure 3.2: PTU-D46-17

The specifications for the camera are listed in Table 3.3. Figure 3.3 shows an image of the cameras used for the PTU CV system.

Table 3.3: Camera Specifications

Units:	2
Manufacturer:	PointGrey Research, Inc. ©
Product Name:	Flea [®] 3 USB 3.0
Model Number:	FL3-U3-13S2C/M-CS ^a
Imaging Sensor:	Sony IMX035 CMOS, 1/3", Mono/Color Rolling Shutter
Max Resolution:	1328x1048 (1.3 Megapixels)
Max frame-rate:	120 frames per second (fps)
Pixel Size:	3.63 μ m
Transfer Rate:	5 Gbit/sec
Digital Interface:	USB 3.0

^a“C/M” indicates both color and monochrome sensor options are available for this model number.



Figure 3.3: Flea 3 USB 3.0 Camera

Attached to the cameras are lenses specified in Table 3.4 and shown in Fig. 3.4.

Table 3.4: Lens Specifications

Units:	2
Manufacturer:	Fujinon - FUJIFILM Holdings America Corporation [©]
Model:	LENS-30F2-V80CS
Focal Length:	2.8 ~ 8mm
Angle of View ($H \times V$):	100°00' \times 73°45' WIDE
Angle of View ($H \times V$):	35°03' \times 26°18' TELE
Operation:	Manual zoom, focus and iris
Iris Range:	F1.2 ~ Close



Figure 3.4: Fujinon 2.8 ~ 8mm vari-focal lens.

3.1.2 Lighting. Two sets of lighting were used. The laboratory, where the commercial photogrammetry system was used, consisted of a 3 x 3 pattern of fluorescent lights and 6 bays of high performance lights arraigned in a 2, 1, 1, 2, configuration. These high performance lights have a luminance of $\sim 28,000$ lumens for $\pm 90^\circ$ (see Fig. 3.5).



Figure 3.5: High performance lights.

For the portable setup outside the laboratory, the lighting used consisted of two Bescor Video Accessories Ltd. LED-500, dimmable studio lights (see Fig. 3.6).

3.1.3 Software. The C++ compiler and program development environment used is Microsoft Visual Studio 2010. The CV Application Programming Interface (API) is an open BSD-licensed Software Development Kit (SDK) called OpenCV. Point Grey Research,



Figure 3.6: Bescor LED-500 dimmable studio light kit.

Inc. uses their own API called FlyCapture 2.0. An SDK is also provided by NVIDIA®™ in order to take advantage of parallel computing and the Graphics Processing Unit (GPU). This takes form in the Compute Unified Device Architecture (CUDA) toolkit. The PTU is controlled by another API provided by FLIR Systems, Inc. Software components are listed in Table 3.5.

Table 3.5: PTU CV Software Components

Windows edition:	Windows 7 Professional
Compiler:	Microsoft Visual Studio 2010
CV API:	OpenCV (opencv.willowgarage.com)
Camera API:	FlyCapture2 (www.ptgrey.com)
NVIDIA GPU API:	CUDA Toolkit 4.2 (www.nvidia.com)
PTU API:	cpi-v1.09.23 (www.flir.com)

3.2 Assumptions

There are many aspects involved in developing a portable, application-based system utilizing CV PTZ cameras. Several assumptions are made concerning the test subject and the environment, and several given and assumptions are made concerning the design of the PTZ camera system. Many of the given for the PTZ camera system are made based upon the research platform, devices, and software used for this research (see Sec. 3.1).

First, here are some assumptions made about the test subject:

- test subject size ranges between 75-475 mm
- test subject exterior is non-reflective and has texture or patterns that can be used with CV algorithms
- test subject has known geometry or model points that can be used for pose analysis
- test subject is unknown to the CV PTZ system
- test subject is capable of moving with 6 Degrees of Freedom (DOF)
- test subject flight dynamics are capable of being modeled (i.e., able to use Kalman filtering or other filtering methods for improved tracking)
- test subject is capable of being fitted with retro-reflective markers for tracking using a commercial photogrammetry system
- test subject may be flown in the area necessary for obtaining focused, high-resolution images

Next, here are some assumptions made about the environment:

- environment is a 7.6 m x 7.6 m x 2.4 m room
- environment has sufficient lighting for high frame-rate cameras to acquire images for feature-based matching
- environment contains a system for recording truth data of a test subject capable of 6 DOF movement
- environment is static during testing (i.e., no large movements besides the test subject)

Lastly, here are some assumptions made about the PTZ camera system:

- PTZ camera system has two cameras for tracking the test subject
- PTZ camera system uses digital zoom versus mechanical zoom
- PTZ camera system during digital zoom operation does not go below each camera's focal length or beyond six times the camera's focal length
- PTZ camera system allows each camera to zoom independently according to the test subject position to a desired zoom setting
- PTZ camera system tracks a fully visible test subject (i.e., the CV algorithm does not require occlusion handling development)
- PTZ camera system is capable of capturing the test subject upon entrance into each camera's Field of View (FOV)
- PTZ camera system should be able to process images in less computational time than that required by the PTU
- PTZ camera system allows for easy setup in a variety of locations (not during test)

3.3 Objectives/Exit Criteria

The desire is to have a system capable of performing state-of-the-art videogrammetry of an Unmanned Aircraft System (UAS). Keeping this in mind along with given and assumptions, the system should be capable of fulfilling a given test scenario using a test subject. Visually tracking the test subject for ≥ 10 seconds and comparing it to truth data is considered fulfillment of a given test scenario. The operations of the test scenario, their designed objectives, and exit criteria are listed in Table 3.6.

The test scenario in Table 3.6 was designed to show fulfillment of PTZ operations with each camera while tracking a test subject. Specifics for fulfillment of the exit criteria are provided in more detail based on the given and assumptions and the desire to design a system capable of capturing images suitable for photogrammetry. For each test subject operation, the CV PTZ camera system results must be compared to truth data to meet the exit criteria.

Table 3.6: Tracking system test scenario objectives.

test subject Operation	Objective	Exit Criteria ^a
Flying towards cameras	tests digital zooming out	Each camera zooms out and falls within specified thresholds
Flying away from cameras	tests digital zooming in	Each camera zooms in and falls within specified thresholds
Flying horizontally from left to right or right to left	tests movement of pan	Each camera pans and continues to track the test subject
Flying vertically from ground towards the ceiling or vice versa	tests movement of tilt	Each camera tilts and continues to track the test subject
Combining three of the above listed movements ^b	tests a combination of PTZ	Each camera pans, tilts, and zooms and continues to track the test subject

^aComparing the CV PTZ camera system to truth data is required for accomplishment of exit criteria.

^bThe test scenario could be fulfilled by moving the test subject in a circular pattern such that the pan, tilt, and zoom operations are required for each camera.

3.3.1 Each camera zooms in/out and falls within specified thresholds. The camera’s focal length, f , is considered THE minimum focal length. Therefore, digital zoom operations do not go below this value and should not be taken into consideration when evaluating exit criteria. The rate at which the camera zooms must be sufficient to maintain a lower and upper threshold of $\pm 20\%$ over an interval ≥ 10 seconds.

3.3.2 Each camera pans and continues to track the test subject. Measuring pan movement $> \pm 10^\circ$ while visually tracking a test subject over an interval ≥ 10 seconds is sufficient to meet the objective and exit criteria.

3.3.3 Each camera tilts and continues to track the test subject. Measuring tilt movement $> \pm 5^\circ$ while visually tracking a test subject over an interval ≥ 10 seconds is sufficient to meet the objective and exit criteria.

3.3.4 Each camera pans/tilts/zooms and continues to track the test subject. Fulfilling the previous four objectives over an interval of ≥ 10 seconds while visually tracking the test subject fulfills the objective and exit criteria.

3.3.5 Considered Objectives (not as part of the exit criteria). Other objectives considered during this research include:

- Processing the CV algorithm should be greater than 30 fps in order to provide the PTU ample time for processing and executing commands,
- Digital zoom should be based on the distance of the test subject from the camera,
- Incorporation of parallel-processing and GPU-based algorithms,
- Development of a state-of-the-art PTZ camera system capable of tracking any UAS or moving object faster than any other general-purpose, CV PTZ system,
- Development of a versatile, parallel implementation platform, capable of examining the performance of other CV algorithms, and
- Design of an easy setup, only requiring a calibration board, enabling a versatile, portable CV PTZ system.

The United States Air Force (USAF) Flight Plan [3] states that Small Unmanned Aircraft Systems (SUASs) are less than 20 lbs. The design considerations are suitable for vehicles less than 20 lbs, but are not suitable for vehicles unable to fly within the given environment. This allows for the smaller, flapping-wing UASs that are typically 150 mm or less in dimension and weigh no more than 90 g [7]. In addition, closed-loop control and wings are being developed for these UASs [37, 77]; however, larger, flapping-wing UASs will likely be required for initial implementation to accommodate the electronics and actuating devices. The assumptions laid out in this chapter are suitable for these smaller, flapping-wing UAS specifications, in addition to accommodating larger, flapping-wing UASs capable of flying within the specified environment.

IV. Selection of a Computer Vision Tracking Technique

4.1 Overview

This chapter compares algorithms to determine object detection and tracking algorithms that best track a moving object using a Pan/Tilt Unit (PTU) camera in a static and dynamic background. The selected Computer Vision (CV) algorithms compared in this chapter are the trained Scale-Invariant Feature Transform (SIFT) [80], Speeded-Up Robust Features (SURF) [15], and covariance methods, a Kernel object tracker based on the Mean Shift (MS) algorithm [35], the Gaussian Mixture Model (GMM) [139] and the Pyramidal Lucas-Kanade (PyrLK) optical flow method [25]. The matching algorithm used in this work for the trained feature generation algorithms, SIFT and SURF, is the Fast Library for Approximate Nearest Neighbors (FLANN) [91]. Initial testing is performed on a sequence of images using a stationary camera. Further testing is performed on a sequence of images such that the PTU camera is moving in order to capture the moving object. Comparisons are made based upon accuracy, speed and memory.

Many algorithms exist for detecting and tracking objects, and the field continues to grow and increase in both complexity and speed. These algorithms were chosen based on availability, programmability and popularity. Advantages and disadvantages are sought by analyzing moving objects in both static and dynamic backgrounds for determining structural characteristics of biological organisms or small, lightweight Unmanned Aircraft System (UAS). Real-world videos with user-provided locations provide comparison data for experimentation and testing of the hypothesized metrics and methods. The goal of this chapter is to compare the effectiveness of six image/video processing techniques on real videos in the following two categories: Stationary camera, moving single object and PTU camera, moving single object.

For the first category, the experiment consists of following a clipboard with a static camera. The experiment in the second category consists of following the clipboard using a PTU camera. Organization of this chapter consists of a brief description of the algorithms used along with added matching techniques, the experiments conducted and summary and conclusions.

4.2 Images

This section defines the parameters used throughout the remainder of this chapter. Define I as a two-dimensional image or frame and images in a video sequence with the subscript i to indicate the specific frame examined (i.e., I_i). The two-dimensional points that make up the pixel coordinates in an image are the pair of values:

$$\mathbf{x} = (x, y) \in \mathbb{R}^2 \quad (4.1)$$

The following represents the contents of an image i at location \mathbf{x} :

$$I_i(\mathbf{x}) = I_i(x, y) \quad (4.2)$$

The upper left corner pixel coordinate indicates the origin $(0, 0)$; where x increases positively to the right and y increases positively downwards. Therefore, the lower right pixel coordinate is $(columns - 1, rows - 1)$ where *columns* and *rows* represent the width and height of the image in pixels, respectively. Real-world images were captured in 1280×1024 frames as 8-bit, gray-scale images and resized to a 640×512 frame using bilinear interpolation to speed up processing. A grayscale image has one component and has values ranging from $0 - 255$ where 0 represents black and 255 represents white.

4.3 Feature Detection and Matching

Let d be a feature vector referred to as a descriptor of m dimensions. Furthermore, let d_{ij} represent the j^{th} descriptor of an i^{th} image. Define T as the training set such that $I_i \subset T$ where I_i is an image in the training set for $i = 1 \dots n$, where n is the number of images in the training set. Each image, I_i , contains a set of descriptors, $D_i = \cup_{j=1}^{l_i} d_{ij}$, where l_i is the number of descriptors in I_i . Let D contain the union of descriptors for all training images, $D = \cup_{i=1}^n D_i$. Therefore, D represents the training dataset of descriptors of n images. Similarly, define the query image as Q , and its descriptors as d_j for $j = 1 \dots o$; where o is the number of descriptors in image Q .

Let M contain the set of matches between the union of descriptors, D , and the descriptors in Q . Therefore, M contains the matches between specific training image descriptors

and query image descriptors. These matches are performed based on the FLANN matching algorithm discussed in Sec. 4.3.3.

4.3.1 Scale-Invariant Feature Transform (SIFT). There are many benefits to using SIFT in tracking. It detects features that are invariant to image scale and rotation, and are shown to provide robust matching across a substantial range of affine distortion, change in three-dimensional (3D) viewpoint, addition of noise, and change in illumination [80]. The major stages of computation for feature generation include: scale-space extrema detection, keypoint localization, orientation assignment and computing the keypoint descriptor. Thousands of keypoints may be generated in an image. Default parameters from Lowe’s paper are used in this investigation in obtaining the keypoints and descriptors for matching. The next step is to collect the information needed to identify our object of interest. This is done per training image through a process of visualizing all of the keypoints detected by SIFT, manually excluding those keypoints not on the object of interest, computing their descriptors and storing them in $D_i \subseteq D$.

4.3.2 Speeded-Up Robust Features (SURF). Bay, et al. [15] claim SURF is less sensitive to noise than SIFT in addition to requiring less computational time which makes SURF an attractive feature detection algorithm. Some structural characteristics considered in SURF are a Hessian threshold, the number of octaves and octave layers, and whether it is upright. The developers of the SURF algorithm used the approach of a very basic Hessian-matrix approximation based on good performance in accuracy [15]. The Hessian threshold provides a minimum value for the determinant of the Hessian matrix. Thus, by setting a larger Hessian threshold, fewer keypoints are obtained. A real-world experimental parameter of 400 was set for the Hessian threshold in order to obtain more discriminate keypoints since a larger selection of keypoints is available with highly textured images. A notable difference in SURF vs. SIFT is that an integral image is used to speed up processing and instead of down-sampling, up-scaling images is achieved at constant cost. As for whether it is upright, this parameter declares whether or not rotation-invariance of the keypoint is important. It is assumed in our experiments that rotation-invariance is important.

In SIFT, the gradient distribution was used for defining descriptors, and in SURF, the distribution of a first-order Haar wavelet response in the x and y direction is used. Another interesting difference between the two feature detection algorithms is that SIFT records 128 dimensions whereas SURF records 64 reducing time for detecting and matching features. The descriptor is formulated to provide regions of 4×4 wavelet response in the x and y direction along with providing intensity changes. Each 4×4 region is normalized to provide invariance to a bias in illumination and invariance to contrast [15]. Similar to SIFT, training is done per training image through a process of visualizing all of the keypoints detected by SURF, manually excluding those keypoints not on the object of interest, computing their descriptors and storing them in $D_i \subseteq D$.

4.3.3 Fast Library for Approximate Nearest Neighbors (FLANN) . The matching algorithm used with SIFT and SURF is called FLANN [91]. This is a library of nearest neighbor algorithms from which FLANN automatically determines the best algorithm and parameter settings given a dataset. A variety of settings can be made using FLANN. One of the first parameters to select from is the type of search to perform. There is a linear, brute-force search, a kd -tree search, a k -means search, a composite search which uses both kd -tree and k -means, and an auto-tuned search which selects one of the previously mentioned searches and tunes the parameters. Auto-tuning of the parameters for algorithm selection was used in this work. Using Muja and Lowe’s paper as a basis for performing algorithm selection, it can be presumed that using k -means would be appropriate for datasets no greater in the range of 100,000 points. It is also noted that performance improves for Muja and Lowe’s dataset using up to 20 random trees. However, this is at the expense of memory overhead. Real-time processing is preferred, and in the auto-tuning parameters more importance is given to time. The collection D is added and trained based on the auto-tuning parameters in matching descriptors to the query descriptors in Q . After going through this process, the matches are stored in M for confirmation.

4.3.4 Homography, Random Sample Consensus (RANSAC), and other Matching Phenomenon. To increase the probability of a correct match, the collection of matches, M , goes through a process known as RANSAC for determining a homographic transformation, known as a homography, that best maps points in the training image to points in the

query image [26]. A random set of four points, each associated with a descriptor, d_{ij} , is contained in $I_i \subseteq T$. Then the matches in M describe four point matches in image Q from which to determine a homography. The transformation determines whether points in the selected training image fall within a specified pixel distance, ϵ , of the points in the query image. This pixel distance is set to 2 for our experiments. The transformation providing the most matches determines the homography used to project points from one image to the next. In our case, image $I_{selected}$ is the training image selected based upon having the most matches within the training set. Image Q is a query image for determining whether our object of interest is present.

Other methods included in this work to add robustness to the matching process consist of: contour intersection checks, improper contour lengths and reduced search spaces. The first of these determines whether or not the contour of the object crosses any other contour lines during transformation. Failure to meet this criterion excludes the transformation and the next transformation is tested until the criterion is met. It is assumed that the unobstructed, full object is present in the image at all times. Thus, if all transformations are exhausted, then our algorithm uses the four closest matching points in M . The next method checks to see if contour lengths double in length or half the length from the training image thereby eliminating poor matches. The last assistive method reduces the search window saving computational time. If positive matches exceed a specified threshold, then the search window is reduced using previously matched contour points for a search area that is $2 \cdot width \times 2 \cdot height$; where *width* and *height* are calculated using a minimum area rectangle about the contour points.

The tracking point, tp , is determined through an established tracking point from a training image in T with the calculated homography, H , for mapping the tracking point in the image selected, $tp_{I_{selected}}$, into the query image, Q . This is shown in Eq. 4.3.

$$tp_Q = H \cdot tp_{I_{selected}} \quad (4.3)$$

4.3.5 Covariance Tracking. Covariance tracking uses features described by $d \times d$ covariance matrices. The features consist of pixel coordinates, intensity and gradients

formed from a $M \times N$ rectangular region made up of k feature vectors describing each pixel. Feature vectors, \mathbf{f}_k , are described as follows:

$$\mathbf{f}_k = [x \ y \ I(x, y) \ I_x(x, y) \ I_y(x, y) \ \dots].$$

Covariance matrices are used to find a minimum covariance distance between the model and the estimated location. The covariance matrix is setup using spatial and image features as described above. Equation 4.4 shows the covariance matrix, \mathbf{C}_R , with respect to the feature vectors, \mathbf{f}_k , and its mean, μ_R .

$$\mathbf{C}_R = \frac{1}{MN} \sum_{k=1}^{MN} (\mathbf{f}_k - \mu_R)(\mathbf{f}_k - \mu_R)^T \quad (4.4)$$

The minimum distance is found using Eq. 4.5.

$$\rho(\mathbf{C}_i, \mathbf{C}_j) = \sqrt{\sum_{k=1}^d \ln^2 \lambda_k(\mathbf{C}_i, \mathbf{C}_j)} \quad (4.5)$$

where $\{\lambda_k(\mathbf{C}_i, \mathbf{C}_j)\}$ are the generalized eigenvalues of \mathbf{C}_i and \mathbf{C}_j computed from

$$\lambda_k \mathbf{C}_i \mathbf{x}_k - \mathbf{C}_j \mathbf{x}_k = 0 \quad k = 1 \dots d \quad (4.6)$$

The image is searched to find the best matching region which determines the location of the object in the current frame [121].

4.3.5.1 Model Update. Tuzel, et al. [120] add a model update strategy to covariance tracking using a Lie Algebra. Using different poses of an object, an intrinsic mean is calculated and used as the model for determining an estimated object location. The calculated covariance matrices are considered Lie Groups requiring a mapping to the Lie Algebra. A Lie Algebra, \mathbf{c}_t , is formed by the mapping of Lie Group covariance matrices, \mathbf{C} and \mathbf{C}_t by

$$\mathbf{c}_t = \log(\mathbf{C}^{-1} \mathbf{C}_t) \quad (4.7)$$

where $\{\mathbf{C}_t\}_{t=1 \dots t}$ is a collection of Lie Group covariance matrices. Alternatively, the mapping of the Lie Algebra, \mathbf{c} , back to the Lie Group, \mathbf{C} , is presented in Eq. 4.8.

$$\mathbf{C} = \exp(\mathbf{c}) \tag{4.8}$$

A select number, t , of varying poses and resulting covariance matrices are averaged to produce the true (intrinsic) mean for searching an image for the minimum distance. For more details in implementing the algorithm see [120].

4.3.5.2 Local Search. A local search is implemented to reduce the computational time. Two times the rectangular region, $M \times N$, is used as a search area for the object [122]. The local search prevents a detected object from moving or being detected outside this specified distance. The disadvantage to local search is that if the object is lost, then the tracker requires the user to manually select the correct region.

Tyagi, et al. [122] also provide results using the covariance tracker with a steepest descent procedure speeding up computational time. This method showed promise, but the results obtained in this work did not track the test subject beyond selection.

4.4 Motion Detection and Tracking

4.4.1 Gaussian Mixture Model (GMM). GMM is associated with background subtraction such that an incoming frame I_i is subtracted from $I_{(i+1)}$ to highlight pixels that are not considered background. A probability density function is then typically used for each pixel in determining whether a pixel from the incoming image, $I_{(i+1)}$, is part of the background or foreground. The next advancement introduced variances of pixel intensity levels which is considered a single Gaussian model [139]. Motion detection was advanced further by using a GMM with a fixed number of components. Zivkovic [139] presents an improved algorithm for constantly adapting the parameters and number of components for

each pixel using an on-line procedure. The GMM with M components is represented in Eq. 4.9.

$$\hat{p}(\vec{x}|\mathcal{X}_T, BG + FG) = \sum_{m=1}^M \hat{\pi}_m \mathcal{N}(\vec{x}; \hat{\mu}_m, \hat{\sigma}_m^2 I) \quad (4.9)$$

where $\hat{\mu}_1, \dots, \hat{\mu}_m$ are the estimates of the means and $\hat{\mu}_1, \dots, \hat{\mu}_m$ are the estimates of the variances that describe the Gaussian components. The background model is approximated using the first B clusters in Eq. 4.10.

$$\hat{p}(\vec{x}|\mathcal{X}_T, BG) \sim \sum_{m=1}^B \hat{\pi}_m \mathcal{N}(\vec{x}; \hat{\mu}_m, \sigma_m^2 I) \quad (4.10)$$

Then B is determined by a threshold, c_f specified in Eq. 4.11 and is used as a measure of the maximum portion of the data that can belong to foreground objects without influencing the background model.

$$B = \operatorname{argmin}_b \left(\sum_{m=1}^b \hat{\pi}_m > (1 - c_f) \right) \quad (4.11)$$

Update equations and further explanation can be found in [139]. Some of the basic parameters that can be set for the selected algorithm include shadow detection, the length of the history used, and the threshold that describes whether a pixel is well described by the background. The more detailed parameters consist of the number of mixtures, the background ratio, the variance of newly generated components, a complexity reduction parameter, a shadow detection value and a shadow threshold value. Shadow detection was turned off since moving objects were often seen as shadows and, in turn, pixels were then turned gray. Turning off the shadow detection helped to fine tune what was seen as moving in a given scene. Shadows are considered as a foreground pixel if given a specified pixel value.

For the cases explored in this work, intuitively, it can be assumed that the static background (static cameras) would provide excellent results using GMM whereas a dynamic background (moving cameras) would cause this algorithm to capture everything.

4.4.2 *Optical Flow.* The pyramidal implementation of the Lucas-Kanade algorithm PyrLK [25] compares two images, the first image, $I_i(\mathbf{x})$, and the second image, $I_{i+1}(\mathbf{x})$, to determine a pixel's velocity at a given time step. This velocity is then used to determine the location of the pixel in $I_{i+1}(\mathbf{x})$. Smaller motions are less demanding in terms of an integration window versus larger motions requiring larger integration windows. For example, the sample point under observation is $\mathbf{u} = [u_x, u_y]^T$ in $I_i(\mathbf{x})$. If the velocity, $\mathbf{d} = [d_x, d_y]^T$, pushes the sample point outside the integration window of $\mathbf{v} = \mathbf{u} + \mathbf{d}$ in $I_{i+1}(\mathbf{x})$, then the tracked point will no longer be tracked. There are two ways to completely lose points. One is for the point to go outside the image, and the second occurs when the pixels surrounding the tracked point varies beyond a set threshold between images I_i and I_{i+1} [25]. We seek to minimize a residual function, ϵ , shown in Eq. 4.12. The image velocity, \mathbf{d} , is the vector that minimizes this function in the image neighborhood of size $(2w_x + 1) \times (2w_y + 1)$ (i.e., the integration window).

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} (I(x, y) - I_{i+1}(x + d_x, y + d_y))^2 \quad (4.12)$$

A point or set of points must be provided for the algorithm to track. Incidentally, using keypoints as detected by SIFT or SURF would provide good points to track. However, for this work, a window is selected about the test subject and equally spaced points are placed within this window for tracking. The integration window we use for this experiment is a 15 pixel x 15 pixel window. The maximum pyramid level setting has been set to three based on default values given in the OpenCV library [26]. The convergence parameter contains two values, a residual pixel value as a minimum threshold and the maximum number of iterations for finding a sample pixel. The minimum threshold has been set to 0.03 pixel as recommended in [25]. It takes on average five iterations to reach convergence and 20 as the maximum.

4.4.3 *Point Selection.* Optical flow requires that point selection be made for tracking. First, a region about the desired tracking point, tp , is selected establishing points to observe through a sequence of frames. Let these observation points be called flowpoints. Optical flow estimates pixel location of the flowpoints from I_i to $I_{(i+1)}$, so the point(s) must

always be within the confines of the image space of I_{i+1} . This is to say that once the points go outside the bounds of the image, the points no longer exist (i.e., any occlusion of the object of interest requires a means of re-selecting and/or establishing flowpoints). Therefore, flowpoints must be selected again for observation.

4.4.4 Point Averaging. Calculation of the tracking point used for error analysis is determined by summing up the foreground (FG) point locations for GMM and flowpoints for optical flow in $I_{(i+1)}$ and dividing them by the total number of FG points or flowpoints. These are expressed in the following equations:

$$tp = \frac{1}{n_{FG}} \sum_{i=1}^{n_{FG}} \mathbf{x}_{FG_i} \quad (4.13)$$

$$tp = \frac{1}{n_f} \sum_{i=1}^{n_f} \mathbf{x}_{f_i} \quad (4.14)$$

where x_{FG} is the pixel coordinate of a FG pixel, and n_{FG} is the number of FG pixels and x_f is the pixel coordinate of a flowpoint, and n_f is the number of flowpoints. The tracking point is used to compare with truth data.

4.5 Segmentation and Tracking

4.5.1 MS. Mean shift is a segmentation algorithm based on region distribution. Comaniciu, et al. [35] refine the MS process down to using a kernel such as the Epanechnikov kernel as part of calculating the color distributions about a region. Note that color space is used as the feature space for analysis in this work. The Epanechnikov kernel is

$$k(x) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)(1-x) & \text{if } x \leq 1 \\ 0 & \text{otherwise} \end{cases} \quad (4.15)$$

where c_d is the volume of the unit d -dimensional sphere.

The target model and candidate distributions are shown in Eq.'s 4.16 and 4.18 with corresponding normalization constants (see Eq. 4.17 and 4.19). These normalization constants ensure that each distribution sums up to one.

$$\hat{q}_u = C \sum_{i=1}^n k(\|\mathbf{x}_i^*\|^2) \delta[b(\mathbf{x}_i^*) - u] \quad (4.16)$$

where $u = 1 \dots m$ and contains the number of histogram bins, m , representing the color space, and δ is the Kronecker delta function for determining whether the histogram bin at the pixel location \mathbf{x}_i is equal to u and the normalization constant is

$$C = \frac{1}{\sum_{i=1}^n k(\|\mathbf{x}_i^*\|^2)}. \quad (4.17)$$

$$\hat{p}_u(\mathbf{y}) = C_c \sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y} - \mathbf{x}_i}{h}\right\|^2\right) \delta[b(\mathbf{x}_i) - u] \quad (4.18)$$

where

$$C_c = \frac{1}{\sum_{i=1}^{n_h} k(\|\mathbf{x}_i^*\|^2)} \quad (4.19)$$

and y is the pixel location of the target center.

The Bhattacharyya coefficient was originally used in the MS tracker; however, it was determined in [35] that its implementation was not necessary and only provided for minimal improvement. Instead, the kernel is primarily used in seeking out the maximum mode of the candidate distribution with respect to the target distribution.

Further details and algorithm implementation can be found in [35]. A 33 pixel x 33 pixel window is used for the static case and a 80 pixel x 80 pixel window is used for the dynamic case in the following experiments. The larger window was chosen for the dynamic case to help capture large movements of the camera. The stopping criterion threshold value is set to 0.01 with a maximum iteration value of 100. Comaniciu, et al. [35] state that no more than 20 iterations is required unless sub-pixel accuracy is desired. The experiments used a stopping criterion threshold value of 0.01 and an increased iteration value of 100 to improve accuracy.

Meanshift Process Diagram

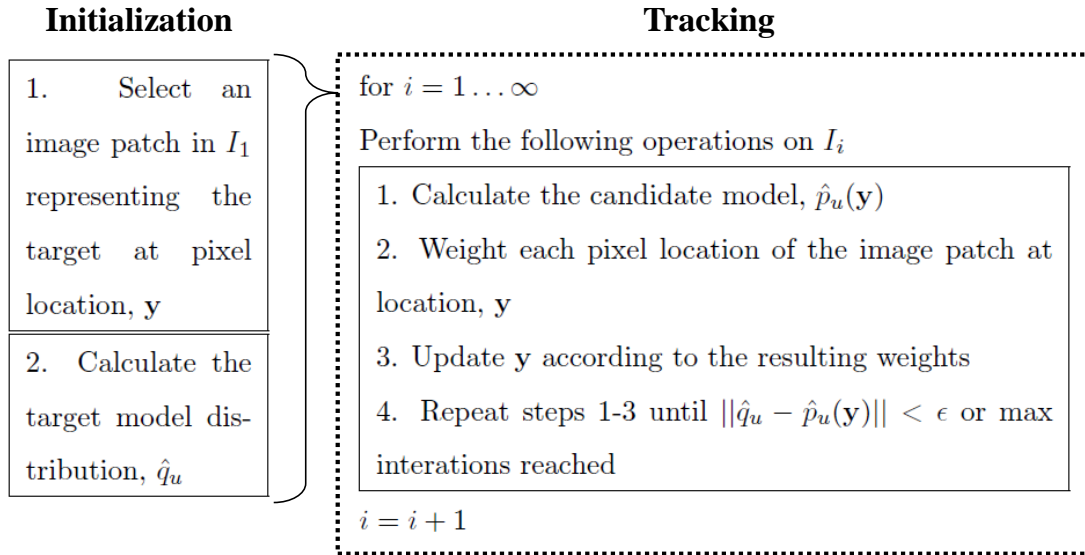


Figure 4.1: The MS process.

A point or region must be selected for providing the MS algorithm with a target distribution as with optical flow. If there is partial or full occlusion of the detected object, then it is very likely that the target distribution has to be re-selected in order to continue tracking. The process is shown in Fig. 4.1. Re-selection requires performing the initialization step again and setting $i = 1$ (see Fig. 4.1).

4.6 Experiments

4.6.1 Training. A set of training frames with feature points comprised the input used for FLANN matching as described earlier. Additional parameters were stored per image to include the contour points of the object and the tracking point for SIFT and SURF. Our object of interest is a clipboard with a texture attached to it and an identified tracking point (see Fig. 4.2 as an example of one pose with a selected tracking point). The training frames were composed of a clipboard against a white background at various poses. Figure 4.3 shows an example of the parameters stored: retained features (i.e., the multi-colored circles), contour points (i.e., the intersection of the green lines), and a tracking point). The

retained features make up the descriptors used for matching. Contour points provide the means for determining whether a proper transformation was made, and the tracking point provides us with a means of calculating the Root Mean Square (RMS) pixel error.

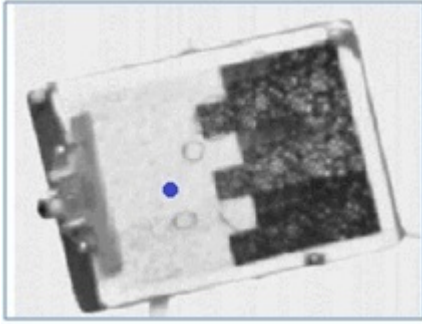


Figure 4.2: Clipboard with attached texture and selected tracking point (blue).

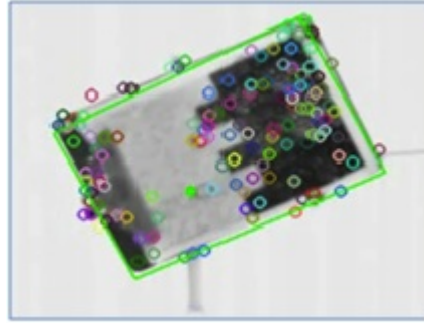


Figure 4.3: Clipboard with retained keypoints (multi-colored circles), contour points (corners of the green lines), and selected tracking point (solid green circle).

Training for the covariance tracker consisted of using 40 images from each dataset for determining the intrinsic mean of the covariance matrix used as the target model. In each training image, the tracking point, tp , is selected, a covariance matrix is calculated for a given patch about the tp , and the Lie Algebra model update is performed.

4.6.2 Data Sets. The selected algorithms were compared using two different data sets. The data sets each consist of one thousand images. The first data set contained the clipboard from Figure 4.2 being moved around by a person against a static background. The second data set contained a clipboard from Figure 4.3 being moved around by a person in a dynamic background. A Pan/Tilt/Zoom (PTZ) camera is used; however, movement of this camera is controlled by another commercial system. Each algorithm uses the same sets of images providing focus on how the algorithms handle both static and dynamic backgrounds. Tracking, algorithm speed, and memory are the performance characteristics for determining

effectiveness. We also look at occlusion handling and user input. Truth sets for both the static and dynamic backgrounds were established manually for error calculations.

Each algorithm relays a tracking point per image which allows calculation of the RMS error given the truth sets. RMS pixel error is given per frame by the following equation:

$$\text{RMS pixel error} = \sqrt{(x_{tp} - x_{truth})^2 + (y_{tp} - y_{truth})^2} \quad (4.20)$$

where (x_{tp}, y_{tp}) is the image pixel location of the tracking point calculated by the algorithm, and (x_{truth}, y_{truth}) is the pixel location of the tracking point that was established manually. Timing each algorithm consists of calculating the processing time required per image. Memory score for each algorithm is based on a high, medium or low scoring based on the number of operations required and amount of storage required per image to provide a tracking point.

4.6.2.1 Static Background. One thousand frames are examined in order to analyze the performance characteristics of each algorithm. The first 220 frames represent the clipboard at rest. Frames ranging from 1-490, 515-610, 620-800, 810-865 and 935-1000 are fully observable and contain a dynamic object of interest (i.e., the clipboard). The remaining frames vary between fully observable, partially occluded and full occlusion. Occlusion causes a loss of points for optical flow requiring a restart or selection of flowpoints, but error also increases for the remaining algorithms as well during these intervals. Error from occlusions is seen clearly in Figures 4.4, 4.5 and 4.6 starting at frame 490, 610 and 860. The clipboard is only partially occluded for frames 800-810. Partial occlusion has little effect on the feature detection algorithms and the PyrLK optical flow and MS methods. The optical flow and MS methods were reset to the tracking point after the noted occlusions. These are marked with dashed lines in Figure 4.6.

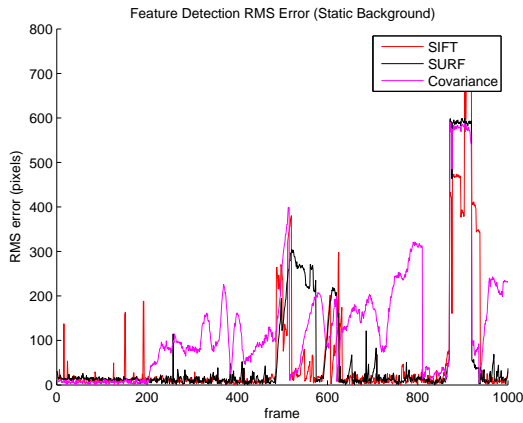


Figure 4.4: SIFT, SURF and covariance tracking RMS pixel error.

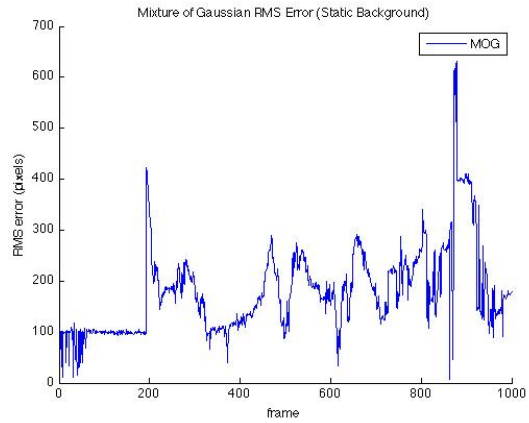


Figure 4.5: GMM RMS pixel error.

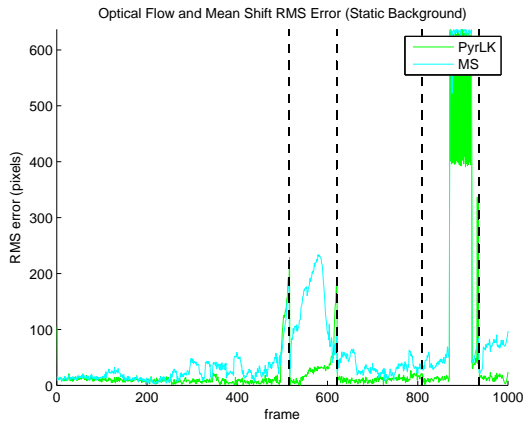


Figure 4.6: PyrLK Optical flow and MS RMS pixel error.

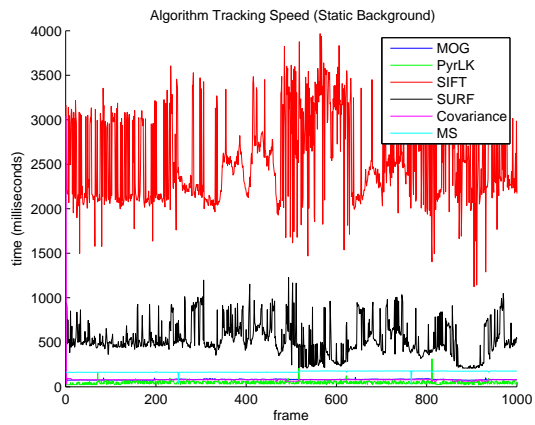


Figure 4.7: Algorithm tracking speed.

GMM performs well in determining the location of a moving object in a static scene. The large inaccuracies shown in Figure 4.5 come from measuring other moving objects within the given image set (i.e., measurements include the person moving the clipboard). Without incorporating matching schemes, the algorithm picks up on all movement, and

therefore, the tracking point is skewed based upon this movement. The person moving the clipboard throughout the images creates additional moving pixels counted in the point average and therefore the RMS pixel error is much higher. Each algorithm calculation time in milliseconds is plotted for each frame in Figure 4.7.

Table 4.1 provides the average RMS pixel error with standard deviations, the average speed with standard deviations and a subjective assessment on memory requirements based on computations required to obtain the tracking point. This assessment details tracking performance of a fully observable object of interest. Therefore, the occluded portions of the static image set were removed for purposes of determining the results in Table 4.1. The most accurate algorithm is the optical flow PyrLK method and also has the smallest standard deviation. The least accurate algorithm is the GMM due to measuring additional moving objects (i.e., the person moving the clipboard). The fastest algorithm is also the optical flow PyrLK method while the slowest algorithm is SIFT by more than 50 times. The lowest amount of processing is inextricably linked to memory. Optical flow PyrLK requires the least amount of processing whereas SIFT requires the most. As detailed above, the feature matching algorithms detect keypoints, calculate descriptors of varying dimensions and then go through several other matching schemes for robustness giving them a high mark for memory. The covariance tracker gets a medium mark due to the memory and processing required in calculating a model used for matching, and the MS algorithm received a low/medium memory rating due to the increased convergence required and tracking window.

Table 4.1: Static Background Algorithm Performance.

Method	Accuracy (Pixels)	Speed (ms)	Memory
SIFT	23.7 ± 51.1	2560.6 ± 484.3	High
SURF	33.7 ± 62.9	549.0 ± 181.0	High
Covariance Tracker	100.3 ± 84.8	81.0 ± 99.0	Medium
GMM	166.8 ± 78.6	81.2 ± 4.5	Low
PyrLK Optical Flow	13.0 ± 20.4	47.2 ± 22.1	Low
MS	33.2 ± 30.6	280.1 ± 14.9	Low/Medium

Tyagi, et al. [122] report that their gradient descent (GD) covariance tracking algorithm gives a minimum error ranging from 9.68 – 10.95 pixels over a sequence of around 100K frames captured at a resolution of 640x480. Table 4.1 shows that the covariance

tracker is the second worst tracker and requires major tuning in order to obtain the accuracies seen in [122] and [121]. The MS algorithm provides increased accuracy and reduced computational time in comparison to SURF.

4.6.2.2 Dynamic Background. The dynamic image set also consists of one thousand frames. The first 115 frames represent the clipboard at rest. Frames ranging from 115-185 and 195-1000 are fully observable and contain a dynamic object of interest (i.e., the clipboard) against a dynamic background. Additionally, the PTZ camera system moved in a non-smooth manner such that the object moved in large pixel displacements between images.

SIFT and SURF show positive tracking with sporadic miscalculations and mismatches whereas the covariance tracker needs some direction (see Fig. 4.8). It is much harder to identify where, if any, occlusion occurs in the dynamic background. The first 115 frames, the object is not moving. The covariance tracker does not perform well with large movements and the feature space does not contrast the object from the background sufficiently. The GMM tracker is the second worst because there is nothing moving in the images for it to be able to track. Again, note that Figure 4.10 shows dashed lines indicating re-selection of flowpoints about the tracking point. Figure 4.11 shows that the feature detection algorithms take much longer than the motion detection algorithms. The segmentation algorithm, MS performs similarly to the covariance tracker in time.

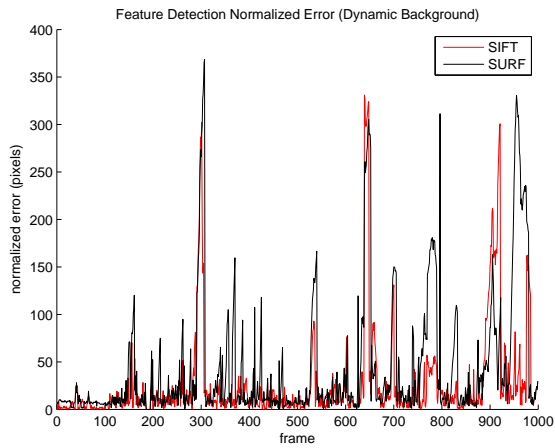


Figure 4.8: SIFT, SURF and covariance tracking RMS pixel error.

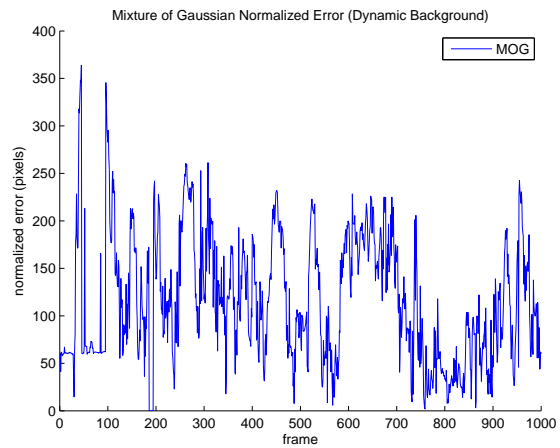


Figure 4.9: GMM RMS pixel error.

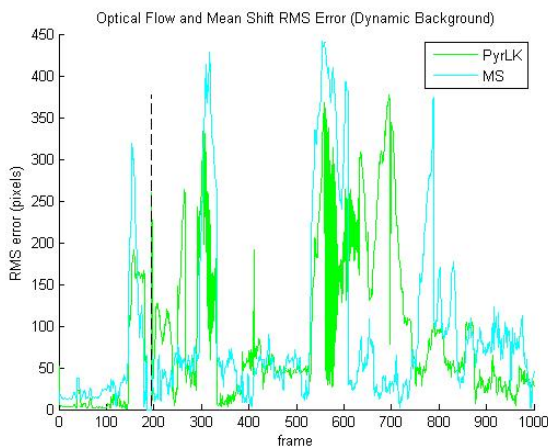


Figure 4.10: PyrLK Optical flow and MS RMS pixel error.

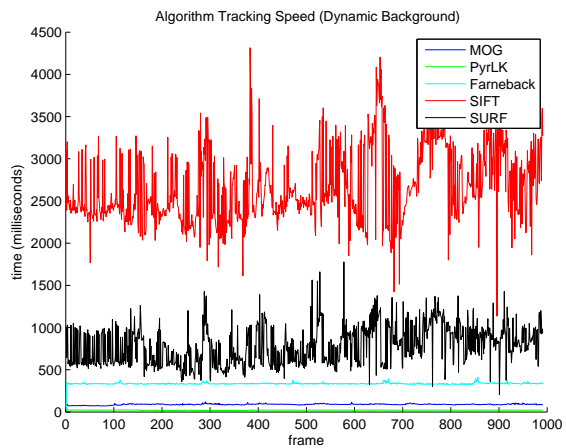


Figure 4.11: Algorithm tracking speed.

Table 4.2 provides the average RMS pixel error and speed with standard deviations for the selected algorithms as well as a subjective assessment of memory required based on computations performed to achieve the tracking point.

Table 4.2: Dynamic Background Algorithm Performance.

Method	Accuracy (Pixels)	Speed (ms)	Memory
SIFT	26.6 ± 51.4	2668.4 ± 438.6	High
SURF	43.9 ± 68.1	788.9 ± 233.1	High
Covariance Tracker	169.3 ± 120.5	274.0 ± 14.2	Medium
GMM	117.5 ± 66.3	89.7 ± 6.8	Low
PyrLK Optical Flow	90.4 ± 91.7	28.1 ± 10.8	Low
MS	94.5 ± 106.0	269.3 ± 13.8	Medium

In Table 4.2, it is seen that the most accurate algorithm is SIFT and only degrades in performance from the static case by 2.9 pixels followed by SURF which degrades in performance by 10.2 pixels. Comparing the GMM plots of Figures 4.5 and 4.9, the dynamic background shows more variation, but reduced error and standard deviation. It is still 3.4 times greater in error than SIFT, but the speed is desirable. Optical flow and MS performance degrades dramatically. Of note is that both of these algorithms have high standard deviations meaning that the accuracy includes those times when each algorithm loses the target and collects spurious results. Re-selection and other means could help in reducing error for making these algorithms more useful in dynamic settings. PyrLK optical flow is the fastest algorithm again and is measured at 28.1 milliseconds followed by GMM at 89.7 milliseconds. Again, SIFT is the slowest algorithm at 2,668.4 milliseconds; 94 times slower than PyrLK optical flow.

4.6.3 Discussion. PyrLK optical flow represents the most accurate tracking point and fastest algorithm in a static background. SIFT provides the most accurate tracking point for the dynamic background and is second in accuracy for the static background. The PyrLK optical flow algorithm was chosen for further development.

4.6.3.1 Advantages. Feature detection algorithms provide matching targets automatically given that the algorithms know what to look for (i.e., training data is used to identify the object of interest). These algorithms also find the target in the midst of partial occlusion if the search is performed on the whole image. These algorithms also provide more accurate results between static and dynamic backgrounds. Training of the covariance

tracker is also much simpler than SIFT and SURF only requiring a single covariance matrix per training image.

GMM provides the ability to find any moving pixel in a given scene. Given the current scenarios, GMM would have performed very well if the clipboard was the only object moving in a static background. In addition, the algorithm's speed and adaptive nature of determining foreground and background pixels may be taken advantage of in other ways such as identifying areas of high activity or relative motion in a dynamic background.

Optical flow is fast and in the case of the static background, the PyrLK method is also very accurate. Under the right settings where illumination is constant, the frame-rate is sufficient to maintain pixel movement to within the window of integration, and no object occlusion occurs, optical flow is a viable real-time tracking algorithm in both static and dynamic settings.

Similarly, MS is a viable real-time tracking algorithm under the right conditions. Some human-in-the-loop (HIL) may be required to direct the target distribution at times. The sequence of images for each dataset only contained grayscale images, and so increased accuracy would likely result in converting or capturing color images. However, even under the current testing conditions, MS provided the third most accurate results beating the second most accurate, SIFT, in time by more than 9 times.

4.6.3.2 Disadvantages. SIFT and SURF recognizably have higher computational costs based on the time required to perform object detection. Although this chapter shows speeds for these algorithms that would not make real-time tracking possible, parallel computing provides the potential for making this a reality. Other disadvantages include preprocessing requirements such as training, and the time it takes to get rid of poor matches and adding additional matching schemes. The covariance tracker requires more matching schemes for robustness as that performed with the previous feature trackers; however, this would increase the algorithm's complexity and computational costs.

In a dynamic scene, GMM does not differentiate between background and moving object making it very difficult to pinpoint an object of interest. More time is required to

perform the operations with more movement in a scene. Increased tuning and use of other algorithmic techniques are required to make this a more viable instrument of choice.

PyrLK and MS require HIL to start the tracking process by placing flowpoints and target distribution. Points are lost quickly with no illumination constancy and object movement greater than the assigned integration window for optical flow. MS loses the object when similar color distributions are encountered. In addition, object occlusion requires HIL selection of new flowpoints and target distributions for observation.

4.7 Summary and Selection

This chapter examined three feature detection, two motion detection, and one segmentation algorithm in a static and dynamic background captured by a PTZ camera. The advantages of the feature detection algorithms allows detection of objects even after occlusion occurs in addition to lower tracking error between static and dynamic backgrounds. It is also noted that the motion detection and segmentation algorithms provide fast, real-time performance as long as the algorithms know where to look. Synergistic effects are possible through a combination of the selected algorithms. Based on the static and dynamic testing, the PyrLK optical flow algorithm was selected for developing a real-time, CV algorithm using known camera motion as described in Chapter V.

V. Algorithm Development and Error Analysis

5.1 Overview

The algorithm presented in this chapter leverages existing methods in a novel combination for general-purpose tracking. It uses known camera motion with exact pixel displacement [68], good features to track [106], and optical flow tracking [25]. Processing uses Graphics Processing Unit (GPU)-based “good features” to track detector¹ (e.g. corners) and a GPU-based implementation of the Pyramidal Lucas-Kanade (PyrLK) optical flow algorithm² [26]. “Good features” are detected in an image and selected as optical flow points. The *background motion* is calculated for each of these points in order to extract the *derived motion*. Optical flow points with a *derived motion* less than a threshold are considered background. Moving points can then be used for Computer Vision (CV)-based tracking. The resulting system is able to track a single, generic moving object where no training or selection is required.

The method presented in this chapter uses Kanatani’s derivation of pixel position between images taken from different positions of rotation about the camera center [68]. Murray, et al. similarly use Kanatani’s relationship of knowing the pan/tilt motion in supporting a motion tracking system to compensate for the background through image subtraction, edge detection, morphological filtering and thresholding [92]. Their technique employs a motion-energy method vs. the optic flow tracking employed in this work.

This work is novel in that it is suited for any textured moving object with the application intent of tracking a free-flight, flapping-wing Unmanned Aircraft System (UAS). Due to the speed of UASs, cameras must pan, tilt, and zoom to track the motion at a desired resolution, and CV algorithms must run at least at 30 frames per second (fps). Pan/tilt cameras have the advantage that the view is not fixed, but every motion of the camera causes apparent motion of points in the image which are actually static. This apparent motion, termed *background motion*, can be calculated by Kanatani’s relationship [68]. *Background motion* must be known to determine the actual motion of moving points, termed *derived motion*.

¹OpenCV 2.4.2 Library - GoodFeaturesToTrackDetector_GPU

²OpenCV 2.4.2 Library - PyrLKOpticalFlow (GPU)

This work provides the analysis and experimental results of Kanatani’s relationship with optical flow tracking using an active camera, and its implementation with GPU-based optical flow for real-time, general-purpose tracking. The analytical results show the induced error with respect to geometric approximations. The experimental results use the static feature points with measured camera motion to determine the error introduced by optical flow techniques. Known motion and inertial measurement have been used to compensate for motion [92,109], but using commanded motion with optical flow for exact derived motion and background classification has not been seen to date [41]. GPU-based feature detection and tracking is an advancement beyond the methods presented in Chapter **II**.

The system components are described in Sec. 5.2. The background motion estimation is provided in Sec. 5.3.1. The algorithm development is provided in Sec. 5.3. The experiments and results are provided in Sec. 5.4. The summary is described in Sec. 5.5.

5.2 System Components

5.2.1 Equipment used. Figure 5.1 shows the hardware used in this chapter. The algorithm development and error analysis were performed outside of the laboratory and so the lighting used consisted of two Bescor Video Accessories Ltd. LED-500, dimmable studio lights. Chapter **III** provides the specifications. The requested image size for the experiments is 656x524 with a requested shutter speed of 1.02 ms. The focal length is 1,076 pixels.

5.2.2 Background and Test Subject. A portion of the background used for the initial performance testing is provided in Fig. 5.2.

The test subject is a UAS (see Fig. 5.3). The large, red circle indicates the calculated pixel centroid, $(x, y)_{pix}$, and the green rectangles represent the detected features, \mathbf{g}_i . For system debug, the UAS was tethered. The length of the pendulum is approximately 1.2 m (47 in), and so the dynamics of the system are known using

$$T = 2\pi\sqrt{\frac{L}{g}} \quad (5.1)$$

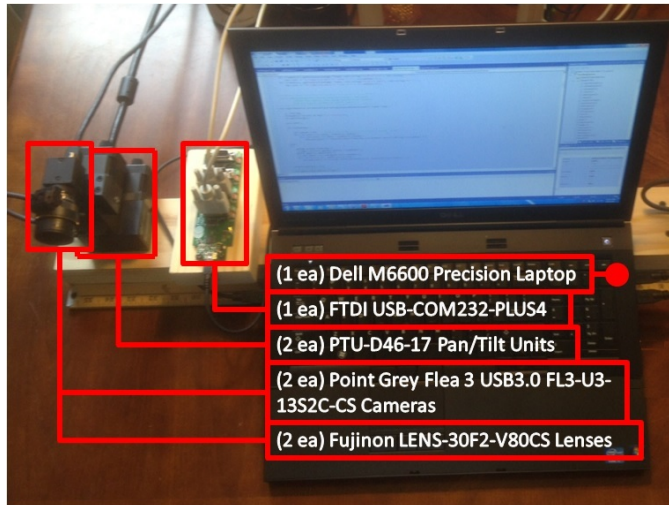


Figure 5.1: Equipment used.

where T is the period, g is gravity, and L is the pendulum length. A period of 2.2 seconds was determined experimentally over ten cycles which agrees with the theoretical period. Given the period, the average velocity of the test subject is calculated using twice the length of the arc divided by 2.2 seconds.

5.3 Algorithm Development

First, an overview of the theory for background motion estimation is presented along with approximations. Then the steps of the tracking process are presented. Next, point classification is covered to distinguish points as moving, background or noise. Point classification is followed by a description of the Kalman filtering process to increase tracking performance. Lastly, pan/tilt control provides how a pixel position is interpreted and what is sent to the pan/tilt controller.

5.3.1 Background Motion Estimation. The motion of the Pan/Tilt/Zoom (PTZ) camera is modeled here for determining where background optical flow points should theoretically move based on PTZ camera movement. It is assumed that the rotation is about the camera physical center. A change in the camera coordinate system due to pan and tilt motions leads to *background motion* in an image. This *background motion*, is the relative background due to the camera's changing coordinate system based on pan/tilt motion.



Figure 5.2: Background used for performance testing.

Equation (5.2) provides the rotation matrix for pan/tilt motions where the z -axis represents the principal axis, the x -axis is positive, right and the y -axis is positive, down. Pre-multiplying the rotation matrix, \mathbf{R} , to a set of points from the camera's previous coordinate system, \mathbf{X}_{i-1} , then gives the new locations of the points in (5.3) as seen by the camera.

$$\mathbf{R}(\psi, \theta) = \begin{bmatrix} \cos(\psi) & \sin(\psi) \sin(\theta) & \sin(\psi) \cos(\theta) \\ 0 & \cos(\theta) & -\sin(\theta) \\ -\sin(\psi) & \cos(\psi) \sin(\theta) & \cos(\psi) \cos(\theta) \end{bmatrix} \quad (5.2)$$

$$\mathbf{X}_i = \mathbf{R}(\psi_i, \theta_i)^\top \mathbf{X}_{i-1} \quad (5.3)$$

where \mathbf{X}_i is the three-dimensional (3D) location of the point with respect to the camera coordinate system, i is the frame number in a sequence of images, ψ_i and θ_i are the change in pan and tilt angles, respectively, from the $(i-1)^{th}$ to the i^{th} coordinate system.

Using the focal length of the camera to provide proportional points on the image plane with respect to three-dimensional coordinates provides the ability to determine optical flow motion inputs of image coordinates (x, y) with respect to the principal axis, delta pan angle, ψ_i , and delta tilt angle, θ_i . The pinhole camera model in Eq. 5.4 provides the means for

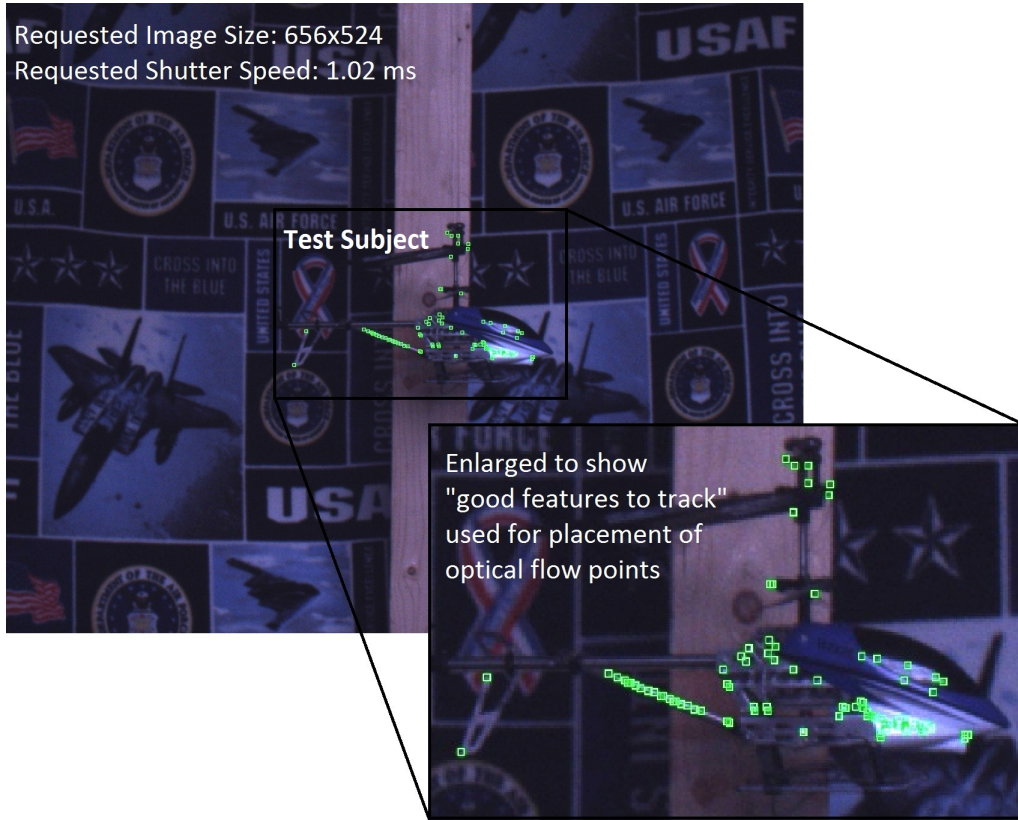


Figure 5.3: The test subject is a helicopter that acts as a pendulum for tracking; green squares represent the initial placement of the optical flow points from image, I_{i-1} and the large, red circle represents the calculated pixel centroid, $(x, y)_{pix}$.

determining the image plane coordinates of a set of points from 3D points by using the mapping to eliminate dependence upon the 3D coordinates.

$$(X, Y, Z)^{\top} \mapsto (fX/Z, fY/Z)^{\top} = (x, y)^{\top} \quad (5.4)$$

$$(\hat{x}, \hat{y})_i^{\top} = f \left(\frac{X_i}{Z_i}, \frac{Y_i}{Z_i} \right)^{\top} \quad (5.5)$$

5.3.1.1 Closed-form solution. Substituting the transformed equations from (5.3) into (5.5) results in equations (5.6) and (5.7) which provide the estimated background location of a point $(\hat{x}, \hat{y})_i$ on the image plane given an input of focal length, delta pan and

tilt angles, ψ_i and θ_i , and image coordinates $(x, y)_{i-1}$. This relationship was first developed by Kanatani [68].

$$\hat{x}_i(f, \psi_{i-1}, \theta_{i-1}, x_{i-1}, y_{i-1}) = f \left(\frac{x_{i-1} - f \tan(\psi_i)}{x_{i-1} \tan(\psi_{i-1}) \cos(\theta_{i-1}) - y_{i-1} \frac{\sin(\theta_{i-1})}{\cos(\psi_{i-1})} + f \cos(\theta_{i-1})} \right), \quad (5.6)$$

$$\hat{y}_i(f, \psi_{i-1}, \theta_{i-1}, x_{i-1}, y_{i-1}) = f \left(\frac{x_{i-1} \sin(\psi_{i-1}) \tan(\theta_{i-1}) + y_{i-1} - f \cos(\psi_{i-1}) \tan(\theta_{i-1})}{x_{i-1} \sin(\psi_{i-1}) - y_{i-1} \tan(\theta_{i-1}) + f \cos(\psi_{i-1})} \right). \quad (5.7)$$

5.3.1.2 Small-angle/first-order approximation. Small movements are made and captured from one frame to the next in the range of $0^\circ - 3.1^\circ$ for the experiments given in Sec. 5.4. Utility in using approximations is sought to understand the viability of making such assumptions. The small-angle approximation simplifies the closed-form solution from having to use basic trigonometric functions by neglecting any second or higher-order terms. The following substitutions are made: $\sin(\Theta) \approx \Theta$, $\cos(\Theta) \approx 1$, and $\tan(\Theta) \approx \Theta$ where Θ is an arbitrary angle in radians. The small-angle/first-order approximation is as follows:

$$\hat{x}_i = f \left(\frac{x_{i-1} - f\psi_{i-1}}{x_{i-1}\psi_{i-1} - y_{i-1}\theta_{i-1} + f} \right), \quad (5.8)$$

$$\hat{y}_i = f \left(\frac{y_{i-1} - f\theta_{i-1}}{x_{i-1}\psi_{i-1} - y_{i-1}\theta_{i-1} + f} \right). \quad (5.9)$$

5.3.1.3 Simplified, linear approximation. The small-angle/first-order approximation can be further simplified by assuming f is the dominant term in the denominator of (5.8) and (5.9). This gives a simplified, linear approximation of:

$$\hat{x}_i = x_{i-1} - f\psi_{i-1}, \quad (5.10)$$

$$\hat{y}_i = y_{i-1} - f\theta_{i-1}. \quad (5.11)$$

5.3.1.4 Summary of Proposed Models. The models above are proposed in hopes that computational efficiency may be found and utilized according to application

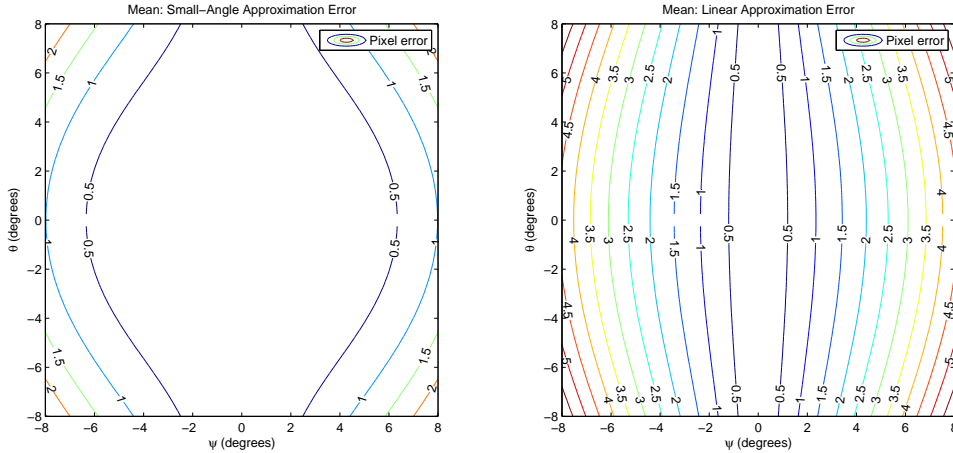


Figure 5.4: The mean error of the small-angle approximation (left) and linear approximation (right).

specific needs. Applications requiring faster processing times, such as embedded microcontrollers, may determine that one of the above approximations is more appropriate.

5.3.2 Approximation Accuracy. Sensor bias may be a problem in any sensor-based application and must be calibrated for application accuracy. Theoretical accuracy of the small-angle/first-order approximation and the simplified, linear approximation compared to the closed-form solution provides insight into application-specific needs. Figures 5.4 and 5.5 show the mean and max differences of the small-angle and linear approximations with respect to the closed-form solution. Note that the rotational matrix, R , in Eq. 5.2 applies pan and then tilt explaining why axis error values are different. For a mean set of points distributed uniformly, the small-angle approximation provides accuracy to within a pixel for $\psi \leq 8^\circ, \theta = 0^\circ$. The linear approximation provides accuracy of a pixel for $\psi \leq 2.3^\circ, \theta = 0^\circ$. Varying the tilt angle for these approximations reduces the theoretical accuracy according to Fig. 5.4. Asymmetric error is seen in Fig. 5.4 and 5.5 primarily due to the closed-form solution's application of pan and then tilt operations in the rotation matrix.

Throwing out the assumption of uniform point placement, the worst case error due to the models for a given ψ and θ is shown in Fig. 5.5 for the small angle approximation and the linear approximation. The maximum approximation pixel error is greater with tilt angle, θ , for the small angle/first-order approximation; however, the error is at least four

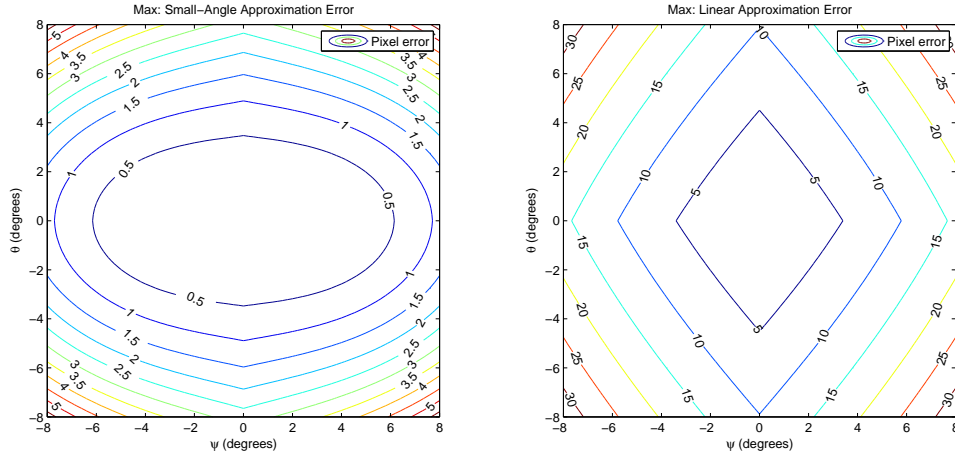


Figure 5.5: Closed-form solution versus the max error of the small-angle approximation (left) and linear approximation (right).

times smaller than the linear approximation (e.g., $\psi = 4^\circ$, $\theta = 2^\circ$, the small-angle give 1/2 and the linear gives 7). The simplified, linear approximation is useful in applications requiring less than 2.3° of pan/tilt between images.

5.3.3 Tracking Process. Figure 5.6 and Table 5.1 show a flowchart and steps for real-time tracking of a single, moving object, respectively. The system rate consists of the time required to capture an image, process the image, and move the pan/tilt unit. Real-time tracking for this paper consists of image capture and image processing speeds ≥ 30 fps and system rates ≥ 15 fps (i.e., image capture and image processing speed $\leq 0.03\bar{3}$ ms and pan/tilt movement $\leq 0.03\bar{3}$ ms for a combined system rate $\leq 0.06\bar{6}$ ms).

5.3.4 Point classification. In the process of detecting “good features”, \mathbf{g}_i for point classification, a boundary of 50 pixels from the image edge is omitted to effectively make a 656x524 image, a 556x424 image for analysis. Omitting the edge facilitates better overlap to match “good features” from the previous image, I_{i-1} , to the current image, I_i . Figure 5.7 shows three scenarios used to classify optical flow points in a given frame as background, moving, or noise. Given a pan/tilt movement, the midpoint between the previous and

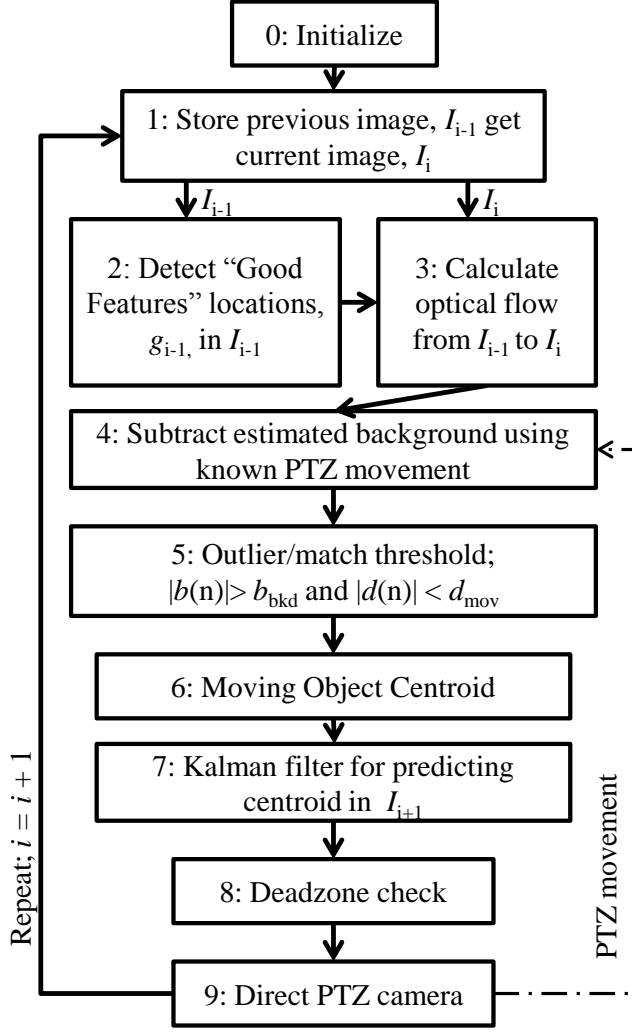


Figure 5.6: Flowchart and steps for real-time tracking of a single, moving object.

estimated feature locations,

$$m_i(n) = \frac{\hat{g}_i(n) + g_{i-1}(n)}{2}, \forall n \in \mathbf{g}_i \quad (5.12)$$

is used to establish the moving object threshold for noise determination. An optical flow point, $g_i(n)$, lying within the estimated feature location/background threshold radius, b_{bkd} , is considered background, otherwise a point within the moving object threshold, d_{mov} , is considered a dynamic point (see Fig. 5.7).

Table 5.1: Pan/Tilt/Zoom (PTZ) tracking with background subtraction

Step	Procedure
0	Initialize parameters and capture first image: $(x, y)_{ref} = (0, 0)$, $(x, y)_{rel} = (0, 0)$, and image, I_0 ; $i = i + 1$
1	Store previous image, I_{i-1} and get current image I_i
2	Detect (x, y) coordinates of “good features”, \mathbf{g}_{i-1} , in I_{i-1}
3	Calculate optical flow from I_{i-1} to I_i such that \mathbf{g}_{i-1} gives \mathbf{g}_i
4	Convert relative pan/tilt units, $(x, y)_{rel}$, to ψ and θ and estimate background pixel movement, $\hat{\mathbf{g}}_i$, using \mathbf{g}_{i-1} , f_i and f_{i-1} Subtract the midpoint, $m_i(n) = \frac{\hat{g}_i(n) + g_{i-1}(n)}{2}$ from $g_i(n)$ to give M displacements, $d_i(n) = \ g_i(n) - m_i(n)\ \forall n \in \mathbf{g}_i$ Subtract $\hat{\mathbf{g}}_i$ from the optical flow movement to give M displacements, $\mathbf{b}_i : b_i(n) = \ g_i(n) - \hat{g}_i(n)\ \forall n \in \mathbf{g}_i$
5	Points in $\mathbf{b}_i : b_i(n) > b_{bkd}$ and $\mathbf{d}_i : d_i(n) < d_{mov}$ are considered moving $\forall n \in \{\mathbf{g}_i\}$
6	Determine the pixel centroid, $(x, y)_{pix}$ and magnitude, $m(x, y)$, of remaining n in \mathbf{g}_i
7	Convert $(x, y)_{pix}$ and $m(x, y)$ measurements to the pan/tilt coordinate system and use a Kalman filter for predicting $(x, y)_{rel}$ for I_{i+1}
8	Convert $(x, y)_{rel}$ to $(x, y)_{pix}$ and apply deadzone check functions $dz(x, x_{dz})$ and $dz(y, y_{dz})$ If $(x, y)_{pix}$ is within a rectangular deadzone, then $(x, y)_{ref}$ remains the same Otherwise, $(x, y)_{ref}$ is adjusted based upon the deadzone, the scale factor, k , and $(x, y)_{pix}$
9	Convert $(x, y)_{pix}$ to $(x, y)_{rel}$ Update the reference pan/tilt coordinates, $(x, y)_{ref} = (x, y)_{ref} + (x, y)_{rel}$, and direct the pan/tilt camera
10	Repeat starting at step 1

The Euclidean norm between an estimated background location, $\hat{g}_i(n)$, and the feature point, $g_i(n)$, is used to establish a set of points, \mathbf{b}_i , for classifying background points. Points falling within the background threshold radius, b_{bkd} , are classified as background using

$$b_i(n) = \|g_i(n) - \hat{g}_i(n)\|, \quad (5.13)$$

$$b_i(n) > b_{bkd} \quad \forall n \in \{\mathbf{g}_i\}$$

where n is the index of the points in the set of matched features, and b_{bkd} is the background threshold radius for background classification.

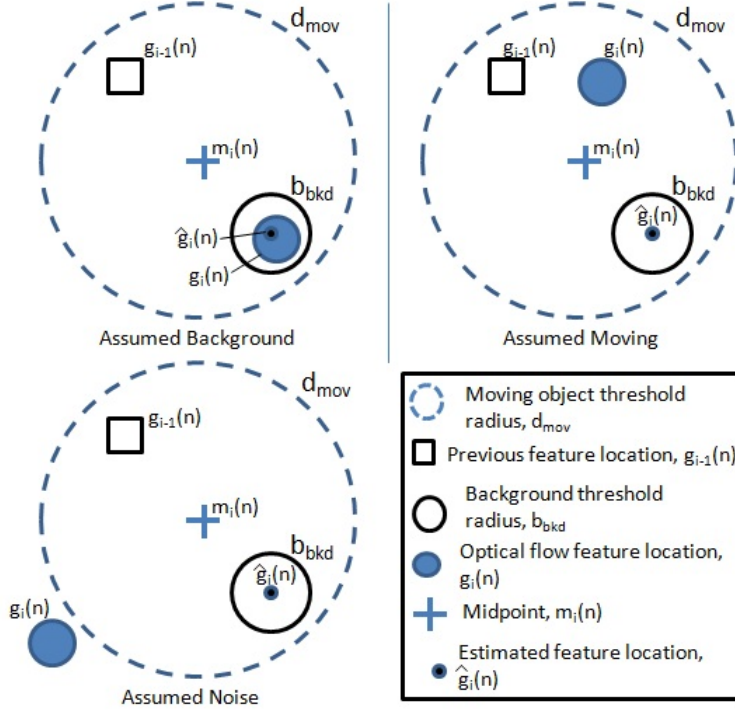


Figure 5.7: Interpreting optical flow points in a current frame as background, moving, or noise.

The Euclidean norm between the *midpoint*, $m_i(n)$, and the feature point, $g_i(n)$, is used to establish a set of points, \mathbf{d}_i , for classifying points as moving or noise. Points falling outside the moving object threshold radius, d_{mov} , are classified as noise and all remaining feature locations are classified as moving using

$$d_i(n) = \|g_i(n) - m_i(n)\|, \quad (5.14)$$

$$d_i(n) < d_{mov} \quad \forall n \in \{\mathbf{g}_i\}$$

where n is the index of the points in the set of matched features, and d_{mov} is the moving threshold radius for classifying points as moving or noise. As a basic check for bad matches, any point that has a $b_i(n) \leq b_{bkd}$ or $d_i(n) \geq d_{mov}$ is considered background or noise, respectively, and removed. The rest of the points are assumed to be accurately tracking the moving object or are classified as background using Eq. 5.13 and 5.14.

The centroid of the moving points should roughly correlate with the object's center. This assumes a symmetric distribution of “good features” across the UAS. The mean of the current locations, \mathbf{g}_i , of the remaining points gives the centroid in pixels,

$$centroid = (x, y)_{pix} = \frac{1}{M} \sum_{n=1}^M g_i(n), \quad (5.15)$$

where M is the number of moving points. The mean of the current displacement vectors, \mathbf{d}_i , of the remaining points estimates the objects translation,

$$velocity = (x, y)_{vel} = \frac{1}{M} \sum_{n=1}^M d_i(n). \quad (5.16)$$

5.3.5 Kalman filter. The Kalman filter is used to predict the movement of the UAS. The Kalman filter model is given by

$$\begin{aligned} \mathbf{X}_i &= \mathbf{A}\mathbf{X}_{i-1} + \boldsymbol{\nu}_{i-1} \\ \mathbf{Z}_i &= \mathbf{C}\mathbf{X}_i + \boldsymbol{\mu}_i \end{aligned} \quad (5.17)$$

where \mathbf{X}_i is the state vector, \mathbf{Z}_i is the measurement vector, \mathbf{A} is the state transition matrix, \mathbf{C} is the measurement matrix. The state and measurement noise, $\boldsymbol{\nu}_{i-1}$ and $\boldsymbol{\mu}_i$, respectively, are assumed to be Gaussian random variables with zero mean. Their probability density functions are $N[0, \mathbf{Q}_{k_i}]$ and $N[0, \mathbf{R}_{k_i}]$, where the covariance matrix \mathbf{Q}_{k_i} and \mathbf{R}_{k_i} are referred to as the transition noise covariance matrix and measurement noise covariance matrix.

A constant acceleration system model is assumed with the model given by

$$X_i = X_{i-1} + \dot{X}_{i-1}\Delta t + \frac{1}{2}\ddot{X}_{i-1}\Delta t^2 \quad (5.18)$$

where Δt is sample time and \mathbf{X} represents horizontal and vertical state vectors.

Therefore, two separate Kalman filters were incorporated, one for horizontal motion and one for vertical motion, such that the state vectors look like $\mathbf{X}_{i,x} = [x, \dot{x}, \ddot{x}]^\top$ and $\mathbf{X}_{i,y} = [y, \dot{y}, \ddot{y}]^\top$, respectively. The resulting transition matrix is given in Eq. 5.19.

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \quad (5.19)$$

Both position and velocity are measured in a given image, I , so the measurement matrix is given in Eq. 5.20.

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad (5.20)$$

The noise covariance matrix, $\mathbf{Q}_{k_{i-1}}$, was set to the identity matrix multiplied by 1×10^{-4} and the measurement noise covariance, \mathbf{R}_{k_i} , was set to the identity matrix multiplied by 1×10^{-3} based on visual effects of reduced jitter. Experimental results were sufficient using these settings; however, minimal adjustments were made so further tuning based on objective testing would likely provide better settings.

5.3.6 Pan/tilt control. The goal of the pan/tilt controller is to track the object near the center of the image. If the object were not moving, then the pan/tilt unit would simply need to point at the previous location. Because the object is moving, the Kalman filter is used to predict the next location, so that the pan/tilt unit will point to the anticipated next location. Note that since the Kalman filter predicts the object location in pixels, it is independent of the pan/tilt controller, so long as the object remains in the image. Therefore, as long as the measurements, $(x, y)_{pix}$ and $(x, y)_{vel}$, are accurate, the system will be stable. The pan/tilt controller parameters controlled the abruptness of motion based on user selection. The user-selected parameters for the pan/tilt unit were set to the pan/tilt unit's maximum speed and acceleration, $300^\circ/\text{sec}$ and $300^\circ/\text{sec}^2$, respectively (see Chapter **III** for the pan/tilt specifications); the base speed was set to half of the maximum speed, $150^\circ/\text{sec}$. This was done in order to achieve the requested pan/tilt movement as fast as possible for a given iteration.

A deadzone is implemented based on an acceptable bound of the object in the image (i.e., the object is close enough to the center of the image that movement of the pan/tilt unit is not required). If the predicted point is more than x_{dz} or y_{dz} away from the image center, then the pan/tilt unit is commanded to move the appropriate axis so that the predicted point will lie on the deadzone boundary for a scale factor of $q = 1$. For $q < 1$, the predicted point falls outside the deadzone boundary causing a slower tracking response. The deadzone function is

$$dz(x, x_{dz}) = \begin{cases} q(x + x_{dz}) & x < -x_{dz} \\ 0 & |x| < x_{dz} \\ q(x - x_{dz}) & x \geq x_{dz} \end{cases} \quad (5.21)$$

for the x direction. The values used in the experiments are $x_{dz} = 50$, $y_{dz} = 50$, and $q = 0.8$. The result of the deadzone calculation for each independent direction is then converted to pan/tilt units, added to the current position, and then sent to the pan/tilt controller:

$$\begin{aligned} x_{\text{pan direction}} &= dz(x, x_{dz})c_x + x_{\text{pan reference}} \\ y_{\text{tilt direction}} &= dz(y, y_{dz})c_y + y_{\text{tilt reference}} \end{aligned} \quad (5.22)$$

where c_x , c_y represent the conversion constants from pixels to pan/tilt units in the x and y directions, respectively.

5.4 Experiments and Results

5.4.1 Point estimation performance. The experiments in this section use the closed-form solution of the *background motion*. Initial experiments contained a feature-filled background as was shown in Fig. 5.2.

It is assumed that the object is no closer than 1 m with a system rate of 15 fps. Using optical flow background estimation and the previous assumptions, Table 5.2 gives approximate pixel movements and object speed for a given pan or tilt angle, Θ , with focal length equal to 1,076 pixels used to interpret scale of results.

An object moving 1.2 m/s at 1 m distance will travel 90 pixels at a system rate of 15 fps if the camera is not moving. As the camera tracks the object, the background points will move 90 pixels while the object will appear stationary. Because the threshold to identify

Table 5.2: Approximate Pixel Movement and Speed for Pan or Tilt Angle, Θ , with Focal Length = 1,076 pixels.

$\Theta(^{\circ})$	1.0	1.5	2.1	2.6	3.1	3.7	4.2	4.7
Pixel Movement	20	30	40	50	60	70	80	90
Speed (m/s) ^a	0.3	0.4	0.6	0.7	0.8	1.0	1.1	1.2

^a Speed relates to an image plane 1 m from the camera with a system rate of 15 fps.

erroneous matches is based on the midpoint of \mathbf{g}_{i-1} and $\hat{\mathbf{g}}_i$, a minimum threshold would be 45 pixels and so this is used for d_{mov} (see Fig. 5.7).

Testing included the use of *only* the feature-filled background. Because all of the points are known to be static, any difference in optical flow points from *background motion* is an error in the method. Horizontal, vertical and diagonal camera movements at varying increments with one thousand features per frame were used. Table 5.3 shows the mean with standard deviation in parentheses for each test along with the number of frames used.

The large errors of some points unduly influenced the results of Table 5.3, so a classification system (see Sec. 5.3.4) was used to identify points that could be ruled out as incorrect matches.

Using the background estimation with the moving object threshold, Fig. 5.8 shows the percentage of features accounted for with respect to the estimated distance in pixels for a movement of 40 pixels.

Since larger errors are seen with diagonal movements as shown in Fig. 5.8, a combination of ψ and θ movements is used as a means for understanding performance limitations of the system. Therefore, Fig. 5.9 shows the test scenario for diagonal movements with the number of features accounted for versus distance. Given $\psi \leq 1^{\circ}$, $\theta \leq 1^{\circ}$, 99% of the placed features are accounted for to within 4 pixels. Figures 5.8 and 5.9 were used to come up with an appropriate background threshold, b_{bgd} , to account for points based on selected movements. Matches further than 8 pixels away were very spread out, resulting in limited advantage of a b_{bgd} larger than 8 pixels for the tested motions. Ideally, the smallest threshold should be used, since it creates a minimum speed that a moving object would be considered stationary. At 15 fps, 8 pixels would correspond to a speed of 0.1 m/s at 1 m.

Table 5.3: Optical flow error for a stationary background with varying movements of pan, $\Delta\psi$, and tilt, $\Delta\theta$.

Θ^a ($^\circ$)	Vertical $\Delta\psi = 0,$ $\Delta\theta = \Theta$ x,y pixels ^c (σ_x, σ_y pixels) ^d	Horizontal $\Delta\psi = \Theta,$ $\Delta\theta = 0$ x,y pixels (σ_x, σ_y pixels)	Diagonal $\Delta\psi = \Theta,$ $\Delta\theta = \Theta$ x,y pixels (σ_x, σ_y pixels)	Diagonal $\Delta\psi = \Theta,$ $\Delta\theta = -\Theta$ x,y pixels (σ_x, σ_y pixels)	frames ^b (#)
-3.1	-3.1, 17.8 (37.0, 39.2)	-47.1, 6.3 (51.6, 36.8)	-49.8, 28.6 (68.9, 55.7)	-36.9, -25.6 (53.4, 51.8)	16
-2.6	-1.3, 9.5 (23.9, 30.1)	-33.8, 7.0 (43.3, 34.2)	-35.8, 23.7 (60.1, 49.8)	-26.4, -17.0 (48.1, 43.0)	19
-2.1	-0.8, 3.5 (20.2, 19.1)	-10.5, 1.7 (33.4, 22.7)	-20.8, 13.6 (45.7, 40.9)	-18.1, -11.3 (38.6, 34.9)	24
-1.5	-0.4, 0.8 (11.1, 15.0)	-2.3, -0.3 (21.5, 12.3)	-5.1, 2.8 (25.0, 22.3)	-6.7, -4.8 (24.3, 23.3)	33
-1.0	-0.7, -0.7 (13.1, 2.2)	0.0, -0.4 (12.0, 8.6)	-0.6, -0.7 (13.0, 8.1)	-0.8, -0.1 (13.8, 12.5)	49
-0.5	-0.1, -0.4 (3.9, 1.2)	0.2, -0.1 (5.5, 4.1)	-0.1, -0.3 (3.9, 1.9)	-0.1, 0.2 (5.9, 5.5)	99
0.5	0.5, -0.2 (6.3, 6.4)	-0.3, 0.0 (0.6, 1.7)	-0.1, 0.1 (6.5, 6.6)	0.0, -0.4 (2.7, 1.7)	99
1.0	-0.9, 0.0 (15.4, 12.3)	-0.9, -0.5 (9.8, 9.7)	-0.3, -0.3 (12.5, 13.7)	0.0, -0.9 (10.8, 8.5)	49
1.5	-0.7, -0.8 (14.7, 14.6)	2.2, -0.7 (16.5, 14.2)	6.9, -5.2 (25.0, 23.6)	7.2, 3.9 (25.4, 26.0)	33
2.1	-1.6, -3.7 (22.2, 23.3)	7.4, -0.7 (21.7, 18.4)	15.6, -16.0 (39.6, 40.2)	15.3, 9.5 (30.7, 31.7)	24
2.6	-1.7, -11.2 (23.3, 34.5)	32.5, 2.8 (35.4, 33.3)	31.2, -25.0 (49.9, 48.6)	23.2, 16.2 (36.9, 37.8)	19
3.1	-2.6, -22.2 (33.4, 44.7)	42.8, 2.6 (38.5, 36.2)	41.5, -31.3 (56.2, 53.9)	34.5, 23.2 (50.5, 47.8)	16

^aThe angle of movement; each column uses this value to determine the amount of pan and tilt movement

^bConstitutes the frames used for a given Θ (e.g., a diagonal movement, where $\Theta = 3.1^\circ$, is approximately 4.4° of pan/tilt movement per frame equating to approximately 70° of pan/tilt movement over the course of the test)

^c x,y pixels represents the global mean over a given number of frames.

^d σ_x, σ_y represents the standard deviation over a given number of frames.

Table 5.4 shows average algorithm processing times in milliseconds over 4000 iterations for varying number of features used and image resolution. A resolution of 656x524 with 250 features leaves 14.5 ms for camera and pan/tilt control for real-time tracking at 30 fps.

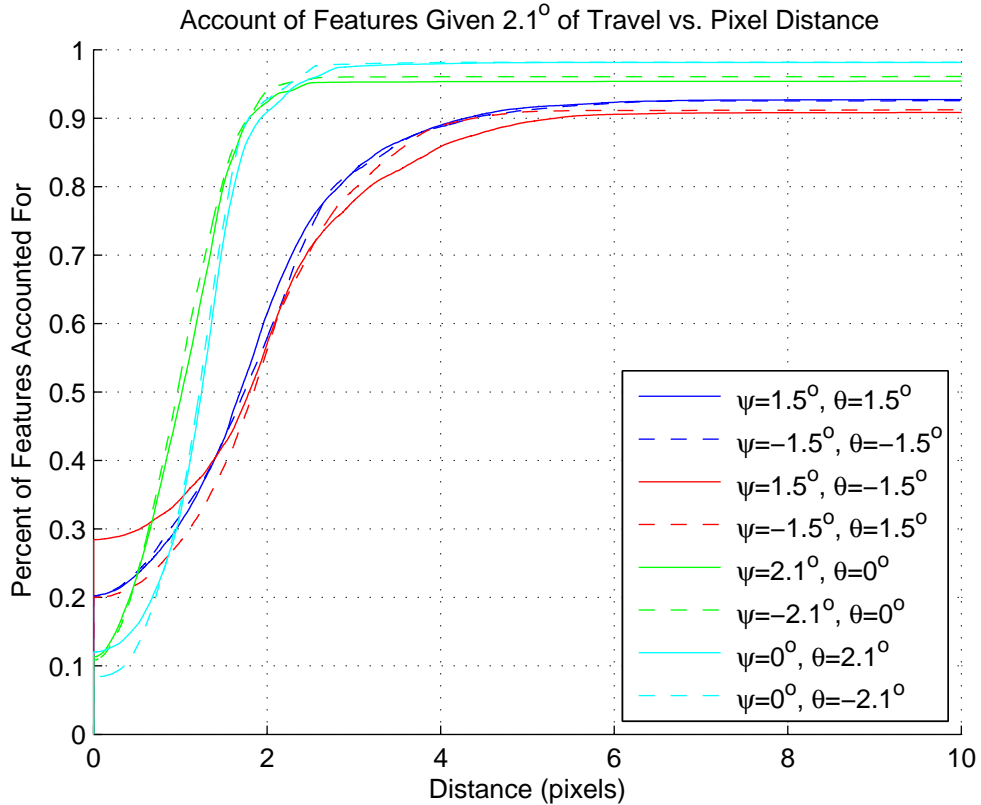


Figure 5.8: Account of features (using background estimation with a moving object threshold) given equal degrees of travel versus pixel distance.

Table 5.4: Average algorithm time (in milliseconds) for varying image resolutions versus number of features used.

# of features	Image Resolution (width x height)			
	328x262	656x524	984x786	1312x1048
250	14.0	18.8	21.5	30.6
500	19.6	24.1	27.3	38.0
750	22.1	30.9	33.3	45.8
1000	21.8	36.4	39.4	52.0

The camera used for these experiments achieved an average time to capture and process an image of 3.4 ms for the requested image resolution of 656x524.

5.4.2 *Test subject tracking and timing.* Experiments were conducted by swinging the test subject in the range of 30° – 40° from rest. This provides a maximum test subject

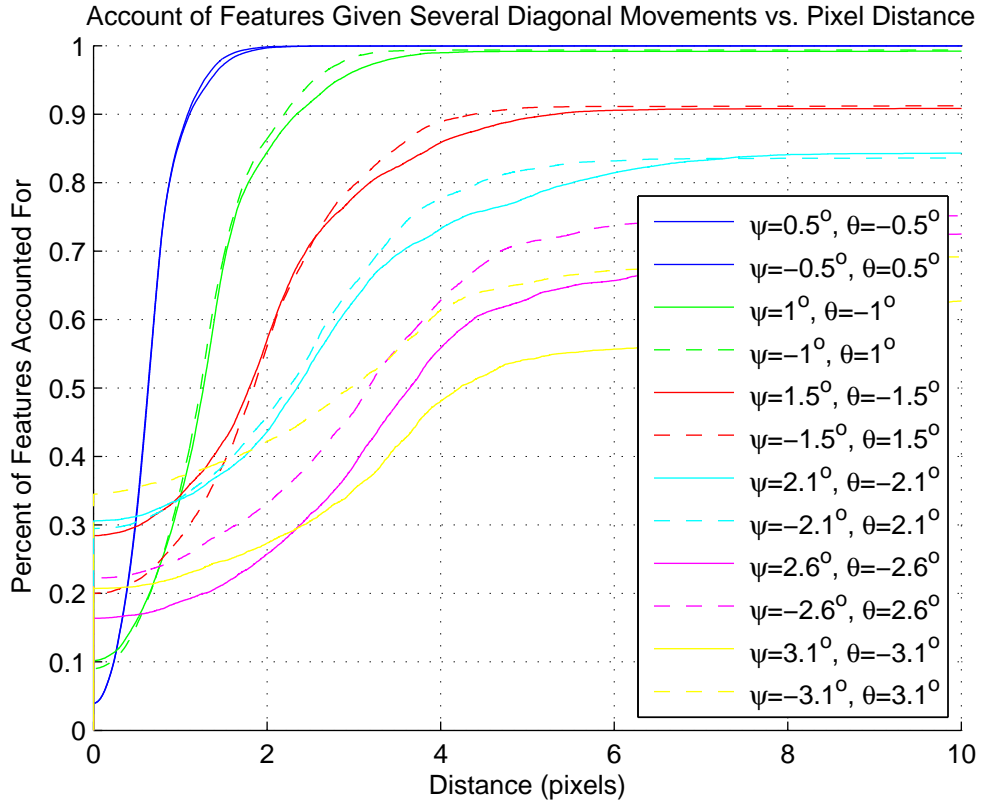


Figure 5.9: Account of features (using background estimation with a moving object threshold) given a diagonal movement versus pixel distance.
 Note: a movement of $\psi = 3.1^\circ, \theta = 3.1^\circ$ is actually 4.4° of travel

velocity of 1.1 – 1.5 m/s. The complete system, consisting of image capture, algorithm, and pan/tilt movement; tracked the test subject at a system rate of 6 – 18 fps.

The test subject scenario shown in Fig. 5.10 shows the various workings of the algorithm for a moving test subject and static camera. Points that lie within b_{bkd} are eliminated and shown as red, closed circles. In particular, it is seen that those points on the doorknob are correctly identified as background motion and are eliminated from the centroid calculation. The white, closed circle points show the matched optical flow points on the moving test subject for determining the pixel centroid and velocity vector.

The second scenario shows the algorithm at work for a moving test subject, moving camera (see Fig. 5.11). Again, those points near the doorknob are eliminated as they are identified as stationary, global points. Despite movement of approximately 49 pixels, the red points near the doorknob in Fig. 5.11 are still eliminated *background motion*.



Figure 5.10: A moving test subject captured by a static camera with previous points shown as green rectangles, \mathbf{g}_{i-1} , current optical flow points as white, closed circles, \mathbf{g}_i , estimated points as green, open circles, $\hat{\mathbf{g}}_i$, and removed outliers shown as red, closed circles.

Table 5.5: Timing of image capture, algorithm and pan/tilt movement over 999 iterations

Procedure	Average	Minimum	Maximum
Image Capture	3.4 ms	3.2 ms	3.4 ms
Algorithm	12.7 ms	10 ms	17 ms
Pan/Tilt Movement	65.0 ms	43 ms	159 ms
TOTAL	81.0 ms	56.2 ms	179.4 ms

Figure 5.12 provides a sequence of frames, approximately 0.24 seconds apart for one period, showing the tracking of the moving (i.e., swinging) test subject with a moving camera. Note that both upper and lower, left images show eliminated points in red due to the lack of motion of the test subject during transition.

In order to achieve faster processing than that shown in Table 5.4, the number of features used was 200 and the effective area was reduced to 456x324 image resolution. Timing of the processes involved was determined using the Central Processing Unit (CPU) clock. An average of 999 iterations for image capture, image processing, and pan/tilt movement is shown in Table 5.5.

The results show that a system rate of 6-18 fps per camera may be realized for the entire process. The slowest part of the process consisted of directing the pan/tilt unit at a minimum of 43 ms and a maximum of 159 ms for the test given. Larger movements caused



Figure 5.11: A moving test subject captured by a moving camera with previous points shown as green rectangles, \mathbf{g}_{i-1} , current optical flow points as white, closed circles, \mathbf{g}_i , estimated points as green, open circles, $\hat{\mathbf{g}}_i$, and removed outliers shown as red, closed circles.

larger delays in pan/tilt movement in order for the unit to complete the directed motion. The algorithm operated at a minimum of 10 ms and a maximum of 17 ms (using 200 features and an effective image area of 456x324), and the camera operated at a minimum of 3.2 ms and a maximum of 3.4 ms for capturing a 656x524 image. Faster processing is possible by not waiting for the pan/tilt unit to complete directed tasks, but the error would not be compensated for. This would require determination of pan/tilt controller information such as the time required to process commands, the time associated with initiation of movement, and the time required before being capable of accepting new commands.

5.4.3 Comparison testing. The proposed algorithm requires no training or matching of features to templates. It is believed that the technique developed in this chapter captures faster, single moving objects better than existing general-purpose methods. Training for specialized tracking is required in [32], [4], and [29]. A color histogram representation is required in [72], [138], and [76] and often fails with large shifts in motion or objects of similar colors. Specialized tracking of skin color for face detection is used in [5]. It is sought here to provide further comparisons in order to give a general idea of realizable performance characteristics by other algorithms with generic assumptions.



Figure 5.12: A sequence of frames, approximately 0.24 seconds apart for one period, of a swinging test subject with the center of the image identified by a white diamond, previous points shown as green rectangles, \mathbf{g}_{i-1} , current optical flow points as white, closed circles, \mathbf{g}_i , estimated points as green, open circles, $\hat{\mathbf{g}}_i$, and removed outliers shown as red, closed circles.

Comparisons were made using a set of 999 images captured using the optical flow background estimation. The timing between images corresponds with Table 5.5 and accuracy is determined through a combination of theoretical pendulum movement and user verification.

The comparison algorithms chosen were accessible via the OpenCV library [26]. The chosen comparison algorithms are optical flow, Continuously Adaptive Mean Shift (CAMShift) and GPU Speeded-Up Robust Features (SURF) with Fast Library for Approximate Nearest Neighbors (FLANN) matching. Since these algorithms vary in user requirements, tuning and utility, it is sought here to provide the reader a general idea of performance characteristics and comparability of the techniques developed in this chapter and not, necessarily, optimized techniques.

Merely using optical flow requires the user to specify how many points to track and where to start. Due to large shifts in the background, optical flow points are quickly lost and re-selection is required to continue tracking. Therefore, it was sought to correct these large displacements due to camera movement using phase correlation (e.g., phase correlation was used in [10] to provide a rough estimate of displacement). However, the lack of background features and the size of the moving object in the given test was detrimental in maintaining optical flow points. Therefore, phase correlation was not used in the experiments. Both the moving object size and background features are critical in using phase correlation.

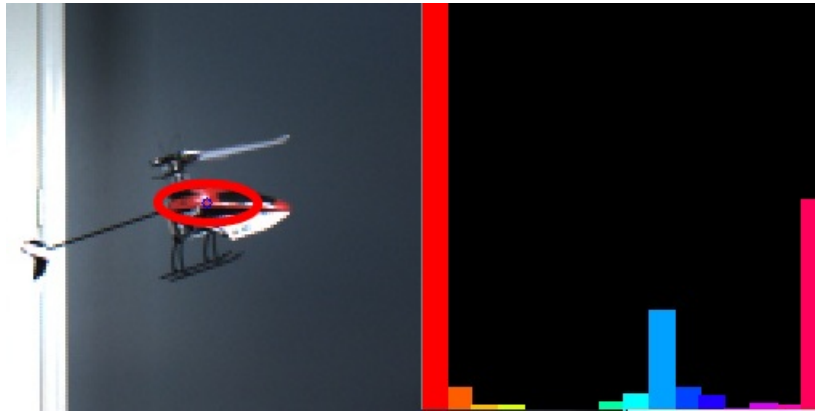


Figure 5.13: For comparison purposes, the CAMShift algorithm was used to determine the object centroid. The color histogram (right) is made up of the components within the ellipse of an image in the test set (left).

The CAMShift algorithm generally tracked well at slower speeds where the object was within the range of a user-specified kernel. Tracking ceased where the local maximum was found to be a background point that eventually left the screen upon pan/tilt movement. Using a background with the same color as the moving object of interest would most likely degrade the tracking of the object and would require user re-selection of the area of interest.

The GPU SURF feature detection was used due to its speed along with matching using the FLANN. System timing, using solely the GPU SURF detection algorithm without matching, provided an average processing speed of around 45 ms. Note that this time is already greater than that achieved in [29] using OpenCL for the entire process. Therefore, in fairness, the process could be optimized to possibly perform better tracking and timing than given here. Training images consisted of twenty images with varying camera parameters of the helicopter on a white background as shown in Fig. 5.14 (right). Convex hulls were generated about the helicopter using the SURF features for each image for transformation operations (see Fig. 5.15). Matches were rank ordered based upon a minimum distance associated with a match. Matches within 20% of the best match were used to determine the moving object's centroid. From preliminary testing, the threshold of 20% was used subjectively to eliminate outliers based upon examination of match distances for the given test. The closest match was not always found to be on the moving object and therefore false matches were often identified.

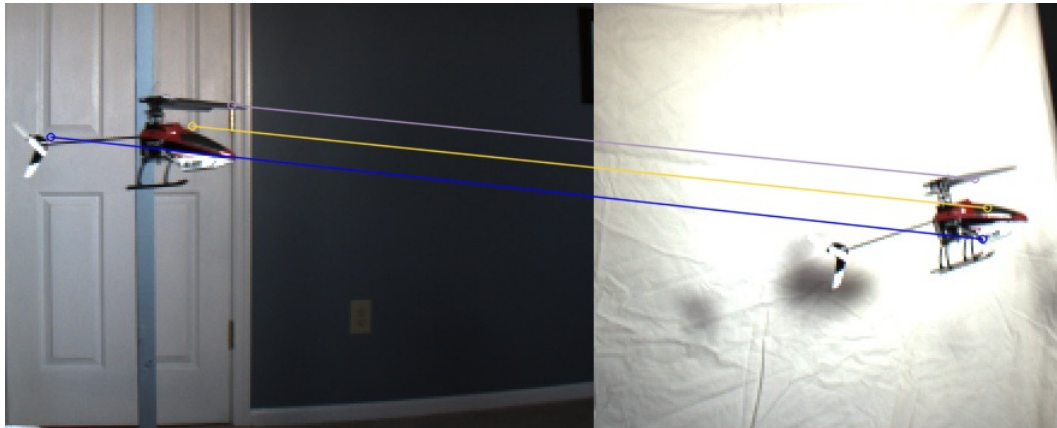


Figure 5.14: For comparison purposes, good matches were used to provide the moving object centroid using a point average. The training image (right) shows feature locations that were matched to an image in the test set (left).

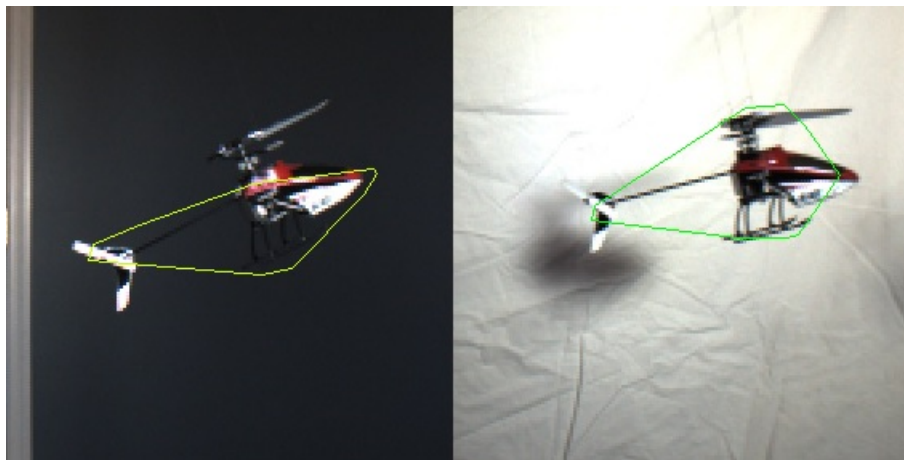


Figure 5.15: Affine transformation of the training contour (right) onto the moving object (left).

An advantage of the technique described in this chapter is that it tracks a single, moving object despite changing features and background while others require similarity in features between frames. For example, the proposed algorithm would be able to track a UAS from any orientation regardless of features whereas feature-based trackers require training or minimal transient feature characteristics.

Table 5.6 shows some performance characteristics of the technique developed in this chapter (GPU optical flow background estimation) along with the selected algorithms. Again, these results provide a general idea of performance characteristics related to those of the comparisons described in Chapter II.

Table 5.6: General performance characteristics of object detection/tracking systems for a set of 999 images at 656x524 resolution.

Algorithm	Training / Selection	Timing / (System rate ^a)	Pixel error / (# re-selects)
GPU Optical Flow Background Estimation	- / -	12.7 ms / (12 fps)	36 ± 34 / -
Optical flow	- / Selection	22.4 ms / (11 fps)	60 ± 65 / (46)
CAMShift	- / Selection	22.5 ms / (11 fps)	43 ± 95 / (23)
GPU SURF	Training / -	56.4 ms / (8 fps)	56 ± 51 / -

^aSystem rate includes the average times for image capture and pan/tilt operation.

5.5 Summary

A real-time tracking algorithm capable of 30 fps using GPU-based algorithms for a pan/tilt camera was presented. The theory and approximations were provided to estimate *background motion* for optical flow background subtraction. The experiments with the static background showed minute benefit in classifying points further than 8 pixels from the predicted location as background. The larger this threshold, the larger the lower limit on the speed of the moving object. Actual tracking experiments showed that greater angles, $\geq 4.4^\circ$, could be achieved by tracking at speeds of ≥ 1.2 m/s with the system operating between 6-18 fps.

An algorithm was developed consisting of point classification, Kalman filtering and pan/tilt control where point classification consisted of whether points were background, motion or noise. The magnitude of the camera movement was varied to show accuracy of the estimation of the background motion by comparing it with the measured optical flow. Results show algorithm processing times for various image resolutions and number of features which shows how performance can scale to achieve desired frame-rates. In addition, the results show that the technique developed in this chapter requires no training, initialization/operator selection, or pre-defined capture area (i.e., alert zone). The novel combination captures free-flying, single moving objects better than existing general-purpose methods.

The CV-tracking method was tested, analyzed and compared in this chapter. Chapter **VI** employs the developed method with two PTZ cameras in order to track a UAS, estimate its 3D position, and set desired camera focal lengths based on the size of the UAS.

VI. Employing a Multiple, Real-Time, Pan/Tilt/Zoom Camera Object Tracking, and 3D Position Estimating System

6.0.1 Overview. The goal of this work is to develop a system capable of tracking a single moving object (specifically a UAS) with Pan/Tilt/Zoom (PTZ) cameras, estimating its three-dimensional (3D) position, and estimating ideal camera focal lengths based on desired object size resolution. This work is novel in the application of tracking all textured moving objects (or general-purpose) with the intent of tracking an indoor Unmanned Aircraft System (UAS). This chapter presents the experimental results of a system that can: 1) track a moving object using two or more cameras, 2) estimate its 3D position, and 3) estimate ideal camera focal lengths based on desired object size resolution leading to a solution that is easily expandable to a multiple camera, photogrammetry system. In addition, the methodology for estimating *background motion*, using Kanatani's relationship on optical flow tracking, is developed for pan, tilt and zoom operations. Furthermore, a parallel implementation is presented with Graphics Processing Unit (GPU)-based optical flow for real-time, general-purpose tracking. The experimental results show the root mean square error with respect to data obtained using a Vicon Motion System, Ltd[®] system. A parallel implementation of GPU-based feature detection, tracking and zooming is an advancement beyond the methods presented in Chapter II.

Some possible applications include: tracking and testing of flexible vehicles and structures without the use of tracking aids, tracking or monitoring vehicles, to include satellites for intercepting space debris, incoming/departing aircraft from a flight tower, and obtaining high-quality images of UASs, satellites, debris, intruders, aircraft, etc.

The system components are described in Sec. 6.1. The theoretical overview for estimating *background motion* based on camera pan, tilt, and zoom operations and the basis for determining 3D position and ideal focal length is presented in Sec. 6.2. A parallel implementation of Computer Vision (CV)-tracking using a PTZ camera is presented in Sec. 6.3. Zoom control is presented in Sec. 6.4. Finally, the experiments and results are shown in Sec. 6.5 followed by the conclusions in Sec. 6.6.

6.1 System Components

6.1.1 Equipment used. In comparison to the equipment used in Chapter **V**, an additional camera and pan/tilt unit have been added (see Fig. 6.1). The testing was performed in the laboratory with the commercial photogrammetry system and so the lighting consisted of the fluorescent and high performance lights. Chapter **III** provides the specifications. The requested image size for the experiments is 656x524 with a requested shutter speed of 1.02 ms. The focal length is 748 pixels.

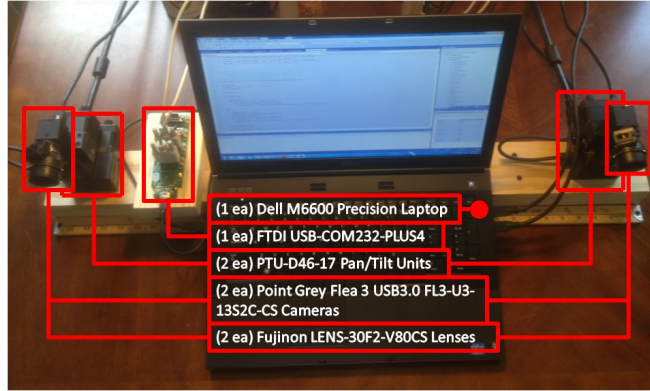


Figure 6.1: Equipment used.

6.2 Theoretical Overview

Extending the relationship developed in Chapter **V**, Sec. 5.3.1, to accommodate a change in focal length gives

$$(\hat{x}, \hat{y})_{i-1}^\top = f_{i-1} \left(\frac{X_{i-1}}{Z_{i-1}}, \frac{Y_{i-1}}{Z_{i-1}} \right)^\top \quad (6.1)$$

and

$$(\hat{x}, \hat{y})_i^\top = f_i \left(\frac{X_i}{Z_i}, \frac{Y_i}{Z_i} \right)^\top . \quad (6.2)$$

Utilizing Eq. (5.3), (6.1) and (6.2) results in equations (6.3) and (6.4) incorporating the dependence upon the change in focal lengths before and after movement, f_{i-1} and f_i , respectively.

$$\begin{aligned} & \hat{x}_i(f_i, f_{i-1}, \psi_{i-1}, \theta_{i-1}, x_{i-1}, y_{i-1}) \\ &= f_i \left(\frac{x_{i-1} - f_{i-1} \tan(\psi_i)}{x_{i-1} \tan(\psi_{i-1}) \cos(\theta_{i-1}) - y_{i-1} \frac{\sin(\theta_{i-1})}{\cos(\psi_{i-1})} + f_{i-1} \cos(\theta_{i-1})} \right), \end{aligned} \quad (6.3)$$

$$\begin{aligned} & \hat{y}_i(f_i, f_{i-1}, \psi_{i-1}, \theta_{i-1}, x_{i-1}, y_{i-1}) \\ &= f_i \left(\frac{x_{i-1} \sin(\psi_{i-1}) \tan(\theta_{i-1}) + y_{i-1} - f_{i-1} \cos(\psi_{i-1}) \tan(\theta_{i-1})}{x_{i-1} \sin(\psi_{i-1}) - y_{i-1} \tan(\theta_{i-1}) + f_{i-1} \cos(\psi_{i-1})} \right). \end{aligned} \quad (6.4)$$

6.2.1 3D Position Estimation. Let the world coordinate system be established such that camera poses are known with respect to the world coordinate system. The j^{th} camera position in the world coordinate system, $P_{0,cam_j}(x, y, z)$, may be determined using its *initial* pose, \mathbf{R}_{0,cam_j} and T_{0,cam_j} , by

$$P_{0,cam_j} = \mathbf{R}_{0,cam_j}^\top (P_0 - T_{0,cam_j}) \quad (6.5)$$

where P_0 is the origin in the world coordinate system, R_{0,cam_j} is the *initial* j^{th} camera orientation from the world coordinate system to the j^{th} camera's coordinate system, and T_{0,cam_j} is the *initial* j^{th} camera translation vector from the world to the j^{th} camera coordinate system.

As the j^{th} camera tracks the moving object, the change in pan and tilt angles from \mathbf{R}_{0,cam_j} are given by Ψ and Θ , respectively. Therefore, the rotation matrix of the camera including pan and tilt angles is given by

$$\mathbf{R}_{obj_j} = \mathbf{R}_j(\Psi, \Theta) * \mathbf{R}_{0,cam_j} \quad (6.6)$$

where $\mathbf{R}_j(\Psi, \Theta)$ represents the rotation matrix based upon the j^{th} camera's pan and tilt angles, and \mathbf{R}_{0,cam_j} represents the *initial* orientation of the j^{th} camera with respect to the world coordinate system.

Now the centroid of the moving object in the image needs to be accounted for from \mathbf{R}_{obj_j} . Knowing the image focal length, f , principal point, (p_x, p_y) , and the centroid of the moving object, (x, y) , a point, P_{obj_j} , can be established from the j^{th} camera location, P_{0,cam_j} , by

$$P_{obj_j} = P_{0,cam_j} + \mathbf{R}_{obj_j} \begin{bmatrix} x - p_x \\ p_y - y \\ f \end{bmatrix}. \quad (6.7)$$

Given N camera locations, $P_{0,cam_j}(x, y, z)$ and the rays for the moving object's centroid, $P_{obj_j}(x, y, z)$, the intersection, $P_{int}(x, y, z)$, may be found using [126]

$$P_{int} = \left(\sum_{j=1}^N n_j n_j^\top \right)^{-1} \left(\sum_{j=1}^N n_j n_j^\top P_{0,cam_j} \right) \quad (6.8)$$

where $P_{obj_j}(x, y, z) - P_{0,cam_j}(x, y, z)$ are normalized and given by $n_j^\top = [n_{j_x}, n_{j_y}, n_{j_z}]$ and $n_j n_j^\top$ is a 3×3 matrix.

The distance to the camera is then found by determining the Euclidean norm

$$d_{cam_j} = \|P_{int} - P_{0,cam_j}\|. \quad (6.9)$$

6.2.2 Ideal Focal Length Estimation. Assuming that the size of the moving object is known, by similar triangles the physical object size, o_{act} , divided by the distance to the camera, d_{cam} , is approximately equal to the focal length, f (pixels), divided by the object size in an image, o_{img} (pixels),

$$\left(\frac{o_{act}}{d_{cam}} \right) \approx \left(\frac{o_{img}}{f} \right) \quad (6.10)$$

where o_{act} is the Euclidean norm of the object width and height in millimeters, and o_{img} is the Euclidean norm of the object width and height in pixels.

After establishing the approximate relationship, rearranging Eq. 6.10 gives the ability to change models by only changing the actual model size, o_{act} . Using a new model size,

$o_{act_{mdl}}$, allows the ability to determine the new model size in pixels, $o_{img_{mdl}}$, and, in turn, set a desired ratio, r , based on the image width by

$$o_{img_{mdl}} = o_{act_{mdl}} \left(\frac{f}{d_{cam}} \right) \quad (6.11)$$

$$r = o_{img_{mdl}} / (\text{image width}).$$

where r equals the ratio of the model in pixels with respect to the image width in pixels.

Setting the desired ratio, r , determines how much of the image frame should contain the object. This gives the ability to determine what focal length is required for a given distance. Solving for f in Eq. 6.11 with the desired image size gives an ideal focal length,

$$f_{idl}(i) = r(\text{image width}) \left(\frac{d_{cam}(i)}{o_{act_{mdl}}} \right) \quad (6.12)$$

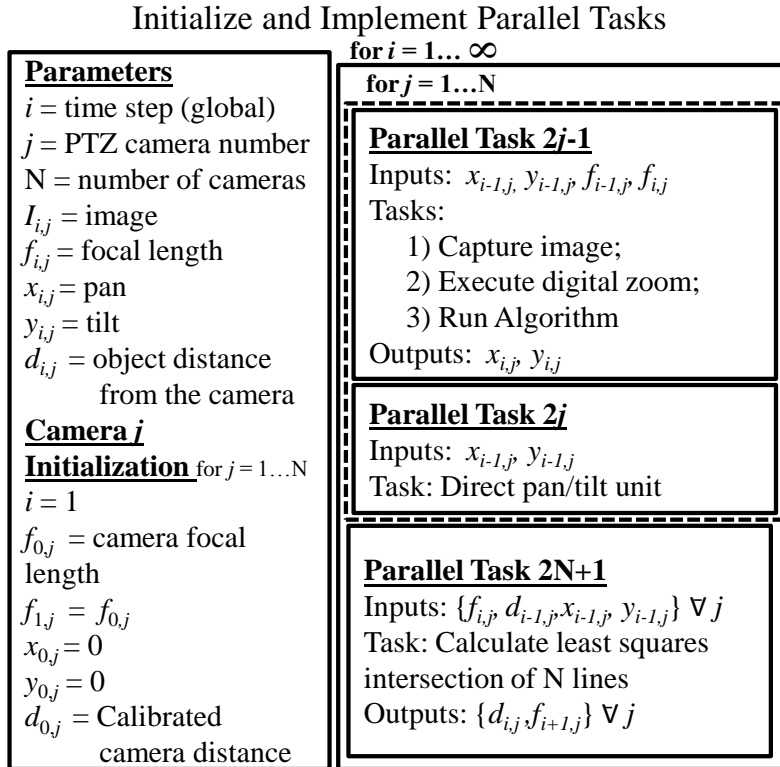
where $d_{cam}(i)$ is the distance to the object from the camera at a given time step, i .

6.3 Parallelized Tracking Using a PTZ Camera

6.3.1 Parallel Process Development. Image capture, algorithm computation, panning, tilting and zooming operations all take time, but are not completely dependent upon each other. Parallelizing the tracking process with a PTZ camera allows algorithm computation while the PTZ camera operates. Those components requiring serial operation include capturing the image, performing a digital zoom operation on the image, and running the tracking algorithm on the digitally zoomed image. While these operations are taking place, the pan/tilt unit can process a previous iteration's commands and move the pan/tilt unit to the commanded position.

The parallel process parameters and initialization steps are shown in Fig. 6.2 (left). The initialization is unique from other steps in that a number of cameras, N , can be set up for parallel operation. The variables provided in initialization are provided as inputs to parallel tasks. Note that operations within each parallel task are performed serially with exception to GPU-based functions within the CV algorithm. Figure 6.2 (top, right) shows two parallel tasks, associated with the j^{th} camera and pan/tilt unit with digital zoom,

enclosed in a dotted, black line to show those tasks repeated for N cameras. Figure 6.2 (bottom, right) shows the last parallel task that utilizes data from each camera, calculates the least squares intersection of N lines, and then provides object distances and focal lengths for each camera. The inputs for all of the parallel tasks require their outputs from the previous step. For example, let $i = 2$, and $N = 2$, then Parallel Task 1 and 3 have inputs $x_{1,1}, y_{1,1}, f_{1,1}, f_{2,1}$, and $x_{1,2}, y_{1,2}, f_{1,2}, f_{2,2}$, to find $x_{2,1}, y_{2,1}$ and $x_{2,2}, y_{2,2}$, respectively, then Parallel Task 2 and 4 have inputs $x_{1,1}, y_{1,1}$, and $x_{1,2}, y_{1,2}$, respectively, and Parallel Task 5 has inputs $f_{2,1}, d_{1,1}, x_{1,1}, y_{1,1}, f_{2,2}, d_{1,2}, x_{1,2}$, and $y_{1,2}$ to find $d_{2,1}, d_{2,2}, f_{3,1}$, and $f_{3,2}$. All of the inputs for the parallel tasks are provided by the previous step.



Note: Parallel tasks enclosed in the dotted black line require N parallel implementations (e.g., for N=2 cameras would give 2 parallel implementations of the enclosed tasks plus the last parallel task for a total of five parallel tasks for one iteration, i).

Figure 6.2: Parallel process parameters and camera initialization (left) and tasks (right) using digital zoom.

The system described in Sec. 6.1 does not contain cameras with mechanical zoom. However, the parallel tasks may be easily adjusted by removing the digital zoom from

‘Parallel Task $2j-1$ ’ and adding it to the ‘Parallel Task $2j$ ’ (see Fig. 6.2 (top, right)). Also, mechanical zoom could be accomplished using a separate parallel task if the hardware is independent of the pan/tilt unit. Another parallel task consideration is that of displaying the images obtained by the system. Depending on the computer system and number of cameras, the parallel task could be implemented as another subtask in the ‘Parallel Task $2N+1$ ’ (see Fig. 6.2 (bottom, right)).

Note that a CV algorithm of choice may be used in ‘Parallel Task $2j-1$ ’, subtask 3) Run Algorithm (see Fig. 6.2 (top, right) providing the ability to test other CV methods with a PTZ system using known camera motion.

6.4 Zoom control

Zoom was accomplished by using a priori knowledge of the moving object and its environment. Specifically, the size of the moving object was specified along with the pose of each camera. In order to establish an environment for simple zoom estimation, a calibration board was used with known square sizes to set up the world coordinate system. The Levenberg-Marquardt optimization was used to find a pose that minimizes the sum of squared distances between observed projections given the camera matrix and distortion coefficients¹ [26].

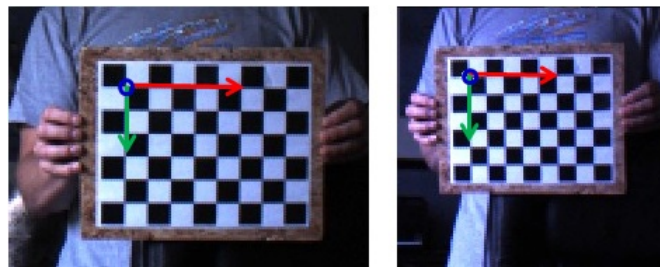


Figure 6.3: Checkerboard with established coordinate system (x-axis, red; y-axis, green, and z-axis, blue circle into the checkerboard) based on point to point correspondences and known checkerboard square size.

Point to point correspondences are found based on a given checkerboard size and a sub-pixel corner detection algorithm. The correspondences between images provides the relation for identifying the camera rotation and translation matrices for setting up the world coordinate

¹OpenCV 2.4.3 Library - solvePnP

system. Once the coordinate system is in place and the camera locations have been identified, approximating the distance is trivial in that the least squares approximation may be used to find a 3D point with respect to the newly established coordinate system (see Sec. 6.2.1).

Once an intersection is found, the process begins for identifying the appropriate zoom based on object distance from each camera (see Sec. 6.2.2). In order to eliminate noise, a low-pass filter is used:

$$f_{cmd,i} = -k\Delta t(f_{cmd,i-1} - f_{idl,i}) + f_{cmd,i-1}, \quad (6.13)$$

$$\text{with } 0 < k\Delta t < 1$$

where f_{cmd} is the commanded focal length, k is the filter time constant affecting rate of change, Δt is the change in time, and f_{idl} is the ideal focal length using Eq. 6.12.

The commanded focal length is also what is used to determine the scaling provided to the digital zoom calculation. The digital zoom calculation simply uses bilinear interpolation for resizing the image to the commanded scale,

$$scale = \begin{cases} \frac{f_{cmd}}{f_0} & f_{cmd} > f_0 \\ 1 & otherwise \end{cases} \quad (6.14)$$

where f_0 is the actual focal length of the camera without scaling. For $scale \leq 1$, the commanded focal length, f_{cmd} , is set equal to the actual focal length, f_0 .

6.5 Experiments and Results

6.5.1 Relating the “Two-camera, PTZ system” to Truth Data. The system was designed such that each camera operates independently in addition to making it simple to add more cameras. Initial testing of the system described in this chapter began with relating it to the established coordinate system of the Vicon Motion Systems Ltd.^{®2} system referred to simply as Vicon. Relating systems was done by placing four retro-reflective markers on

²Vicon California, 5419 McConnell Avenue, Los Angeles, CA 90066; (310) 306-6131, email: info@vicon.com

the checkerboard used to calibrate and establish the world coordinate system for the “two camera, PTZ system”. Rotation and translation values of the checkerboard in the Vicon coordinate system were recorded in order to place the Vicon points into the world coordinate system established by the “two camera, PTZ system”. The following relation provides Vicon points in the “two camera, PTZ system” world coordinate system:

$$P_{new} = R_{ckbd}^{\top}(P_{Vicon} - T_{ckbd}) \quad (6.15)$$

where P_{new} is a point in the “two camera, PTZ system” coordinate system, R_{ckbd} is the rotation matrix of the checkboard, P_{Vicon} is the point as recorded by Vicon, and T_{ckbd} is the translation matrix from the Vicon origin to the “two camera, PTZ system” origin.

The flight paths were aligned using a combination of visualizing vehicle trajectories in 3D space and considering visually plausible root mean square error between systems of the observed flight path. Vicon data was considered truth and used to determine the root mean square error of the “two camera, PTZ system”. In addition, a bicubic interpolation was used to fill in any missing Vicon points due to occlusion of the retro-reflective markers.

6.5.2 Experiments. The algorithms are evaluated for tracking generic moving objects and the performance is quantified in Chapter V. Using a single camera with no zoom, objects were tracked at approximately 1.2 m/s at a distance of 1 m. As an extension to this work, several new experiments were performed to test the tracking algorithm, the 3D position estimation, and digital zoom of a moving object using two cameras. The 3D position estimation was restricted to a 7.6 m x 7.6 m x 2.4 m room using a Vicon camera system for providing accurate truth data (i.e., 3D position and time) of the moving object with error at the sub-millimeter level.

In order to obtain accuracy measurements, the Vicon camera system was used to identify and compare position of the vehicle. Several vehicles were used in the experiments to demonstrate the generic moving object capability. However, only the Blade^{®3} mQX quadcopter was used in obtaining Vicon truth data (i.e., position vs. time) for error analysis (see Fig. 6.4). The quadcopter’s maximum base dimension is 250 mm and maximum

³Horizon Hobby, Inc., 4105 Fieldstone Road, Champaign, IL 61822; (217)352-1913

Test Subject (Quadcopter)

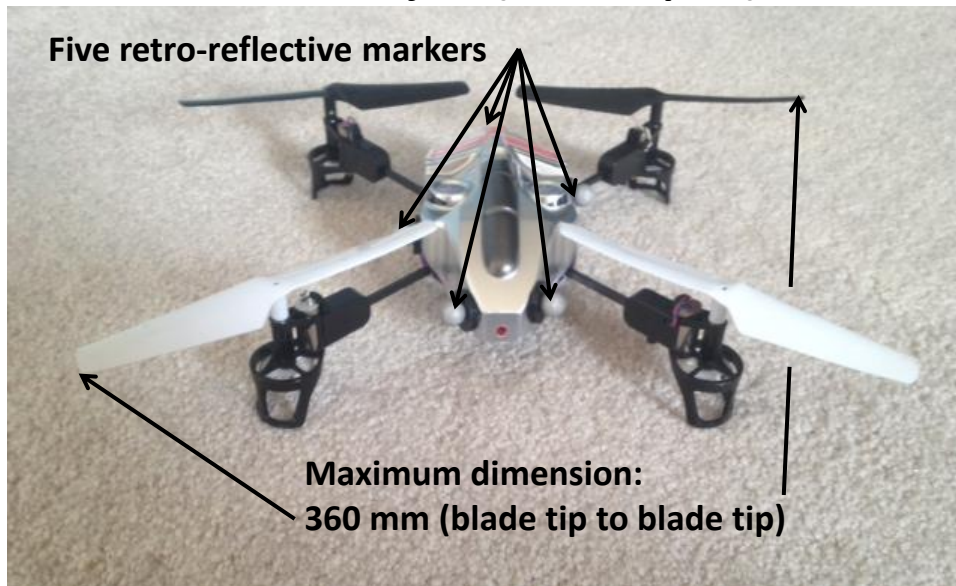


Figure 6.4: The test subject used to obtain data using the “two-camera, PTZ system” and the Vicon camera system.

dimension from blade tip to blade tip is 360 mm. Five retro-reflective markers were placed on the vehicle in order for the Vicon camera system to track the quadcopter using the 10 specialized cameras.

The experiments, objectives and subjective pass/fail criteria are listed in Table 6.1. The subjective pass/fail criteria is based solely on whether or not the PTZ cameras visually tracked the quadcopter 200 out of 275 recorded test frames (i.e., visually tracked $> 70\%$ of the test); which equated to ≥ 10 seconds of flight time. Note that the 3D position estimation error increased exponentially as the projection lines of the cameras were near parallel so this data was removed from error calculation; however, visual tracking still persisted. Further data reduction occurred when trying to line up the data obtained from the two systems showing decreased test times and frames as a result (see Table 6.2).

Above camera-level testing failed due to the bright lights used to enable minimum shutter speeds of $\approx 1ms$. Due to the nature of optical flow and approximating pixel location using brightness constancy, the algorithm performs poorly against bright lights. It would pass if the lights were filtered out or otherwise moved out of the way. The predicted points

Table 6.1: Specific tests with objectives and pass/fail.

#	Test	Objective	Pass/Fail
1	x-axis testing	verify system positioning/test the ability to track flight and determine depth primarily in the x-axis	Pass
2	y-axis testing	verify system positioning/test the ability to track flight and determine depth primarily in the y-axis	Pass
3	z-axis testing	verify system positioning/test the ability to track flight and determine depth primarily in the z-axis	Pass
4	below camera-level testing	test the ability to track and determine depth at a plane below the camera plane	Pass
5	camera-level testing	test the ability to track and determine depth at camera-level flight	Pass
6	above camera-level testing	test the ability to track and determine depth at a plane above the camera plane	Fail
7	proximity testing	test the ability to track and determine depth at close range	Pass
8	approach testing	test the ability to track approach and passing between cameras	Pass
9	range testing	test the ability to track and determine depth at distances outside Vicon range	Pass

against lights produced an increase feature count and often apparent motion thereby causing the “two-camera, PTZ system” cameras to fixate on the lights.

6.5.3 Flight Profile Examination. Different flight profiles were examined here to give the reader an understanding of the capability of this “two-camera, PTZ system” with respect to the ten Vicon camera system. Figures 6.5, 6.6, and 6.7 demonstrate the tracking of the quadcopter, using both systems, primarily in the x, y and z-axes, respectively, using the world coordinate system initially established by the “two-camera, PTZ system”. Note that the y-axis testing in Fig. 6.6 specifically shows the y-axis in both the top and bottom plots for visualizing the flight path of the quadcopter with respect to the y-axis. The fourth flight profile in Fig. 6.8 shows both systems tracking the quadcopter primarily at below camera-level flight to show another arbitrary flight profile.

All of the specific tests were performed to test the viability of the system, the accuracy of the range, and the zoom capability. Those flight paths shown in Fig.’s 6.5, 6.6, 6.7, and

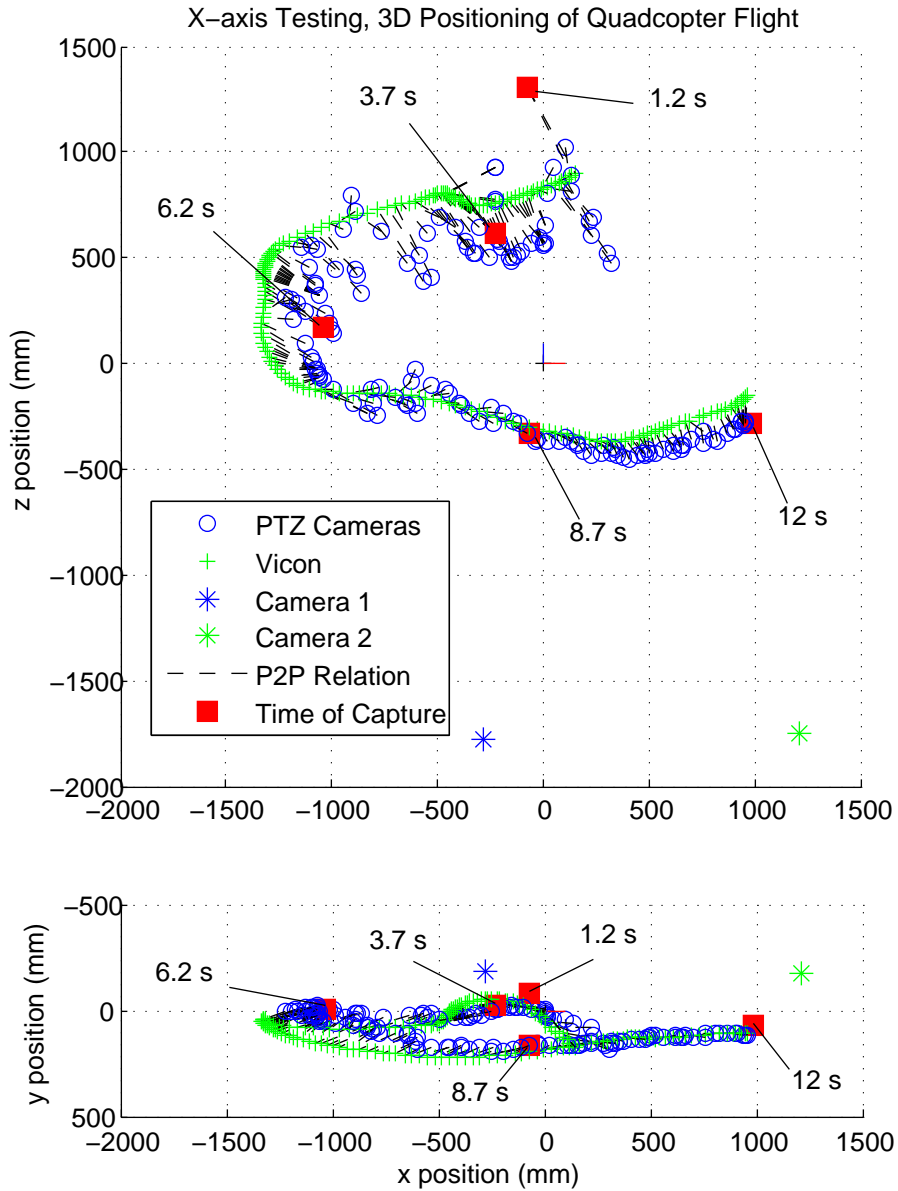


Figure 6.5: X-axis testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system” with a point-to-point (P2P) relationship. Note: The ‘Time of Capture’ relates to the image sequence shown in Fig. 6.13.

6.8 are provided to show the point-to-point (P2P) relationship between the “two-camera, PTZ system” and the Vicon system.

Table 6.2 shows the root mean square (RMS) error associated with the specific tests listed in Table 6.1. Two of the specific tests, range and above camera-level testing, are

Y-axis Testing, 3D Positioning of Quadcopter Flight

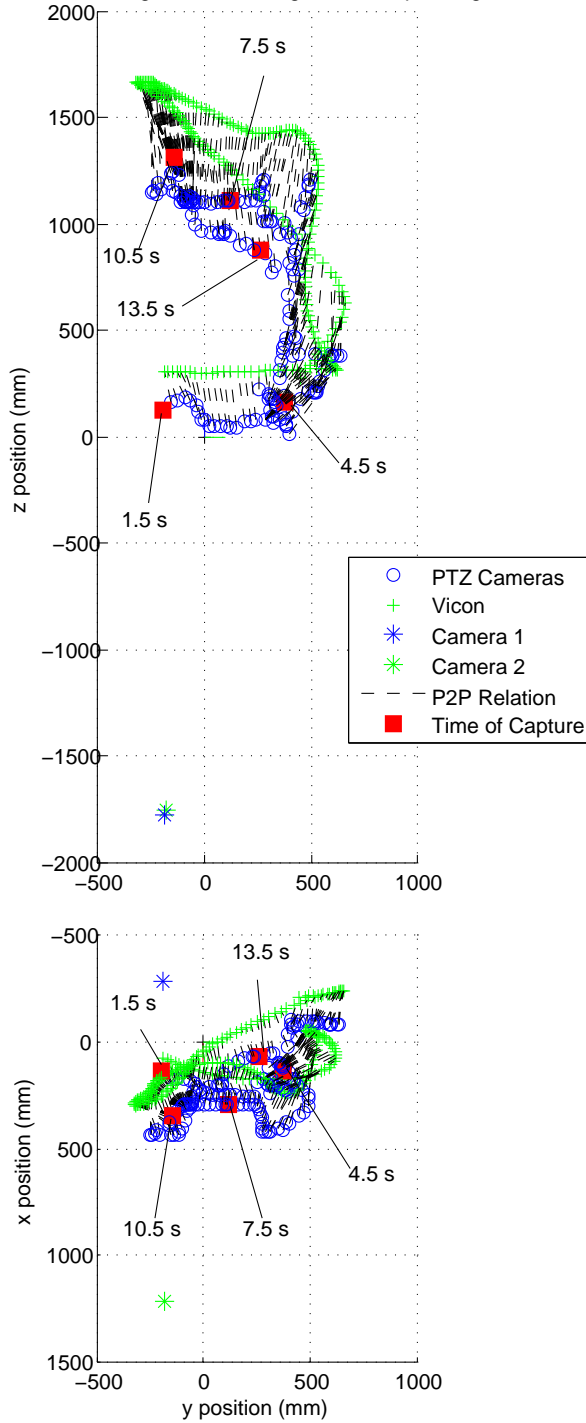


Figure 6.6: Y-axis testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system” with a point-to-point (P2P) relationship. Note: The ‘Time of Capture’ relates to the image sequence shown in Fig. 6.14.

not shown due to the lack of Vicon data for the range testing and the failure of the above camera-level testing due to the lights. The axis with the least amount of average error was

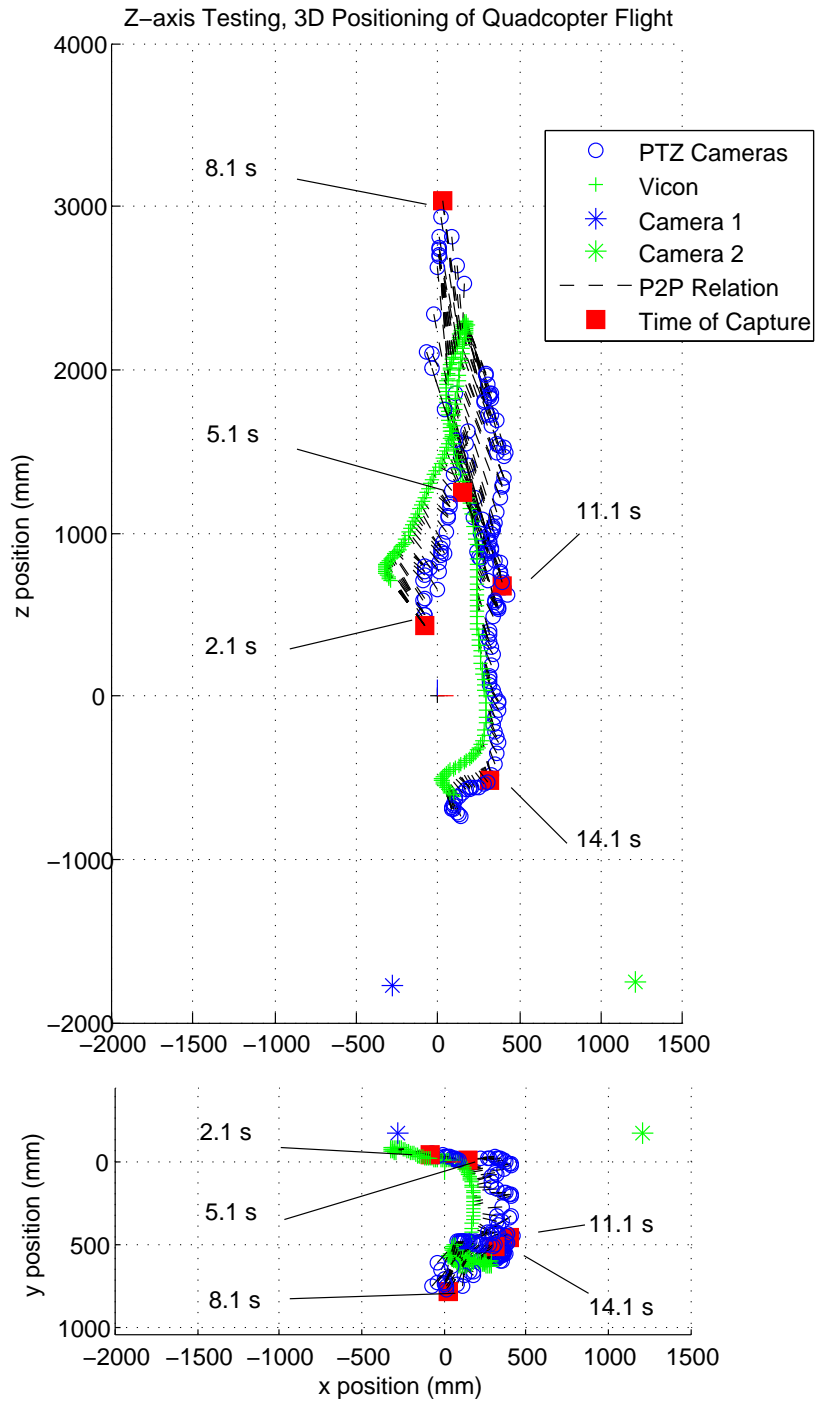


Figure 6.7: Z-axis testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system” with a point-to-point (P2P) relationship. Note: The ‘Time of Capture’ relates to the image sequence shown in Fig. 6.15.

the y-axis with $91 \text{ mm} \pm 89 \text{ mm}$ followed by the x-axis with $112 \text{ mm} \pm 34 \text{ mm}$. The x-axis had the smallest standard deviation over the course of the tests. The Z-axis (the depth

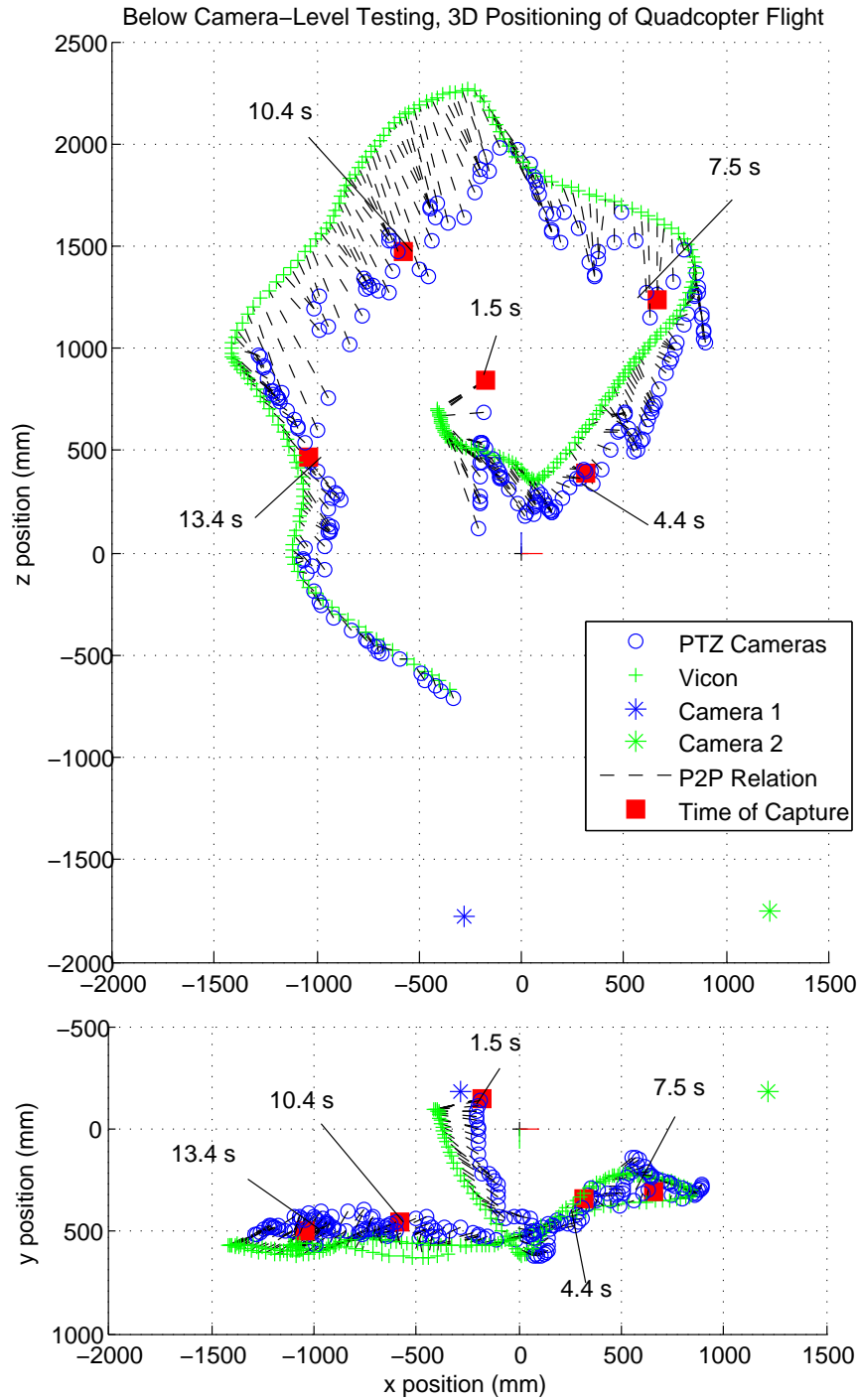


Figure 6.8: Below camera-level testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system” with a point-to-point (P2P) relationship. Note: The ‘Time of Capture’ relates to the image sequence shown in Fig. 6.16.

from the camera) had the largest error over the course of the tests with an average error of 218 mm \pm 100 mm. The total average error of 290 mm \pm 100 mm was \sim 20% less than

the maximum test subject size of 360 mm (i.e., the maximum size corresponds to possible image points that could be identified as moving points in an image).

Table 6.2: Root mean square (RMS) error associated with specified tests.

#	Test	X Error (mm)	Y Error (mm)	Z Error (mm)	Total Error (mm)
1	x-axis test, 189 frames, 11.1 seconds	139.9	40.6	119.9	201.9
2	y-axis test, 225 frames, 13.6 seconds	135.8	89.2	350.2	397.6
3	z-axis test, 211 frames, 13.2 seconds	138.6	68.8	371.2	412.9
4	below camera-level test, 234 frames, 14.9 seconds	134.4	70.8	209.7	279.1
5	camera-level test, 223 frames, 12.2 seconds	102.8	39.1	157.7	214.5
7	proximity test, 115 frames, 6.7 sec- onds	72.6	287.8	184.1	359.1
8	approach test, 152 frames, 8.8 sec- onds	58.2	41.8	132.8	165.9
	TEST AVERAGE, 1349 frames, 80.5 seconds	111.8 ± 34.4	91.2 ± 88.8	217.9 ± 102.2	290.1 ± 100.4

Using the results of Table 6.2, the average processing speed for the system, to include visualizing and recording, equates to 16.8 frames per second (fps) with a range from 15.7 - 18.2 fps. Tests without additional processes such as recording and visualizing the data showed processing times of 21 fps.

Figures 6.13, 6.14, 6.15, and 6.16 each show a sequence of five images for x-axis, y-axis, z-axis and below camera-level testing, respectively, which relate to the ‘Time of Capture’ shown in Fig.’s 6.5, 6.6, 6.7, and 6.8, respectively. On the right side of each set of images is a blue bar indicating focal length, f_0 , and a green bar indicating the commanded focal length, f_{cmd} , indicating the level of zoom accomplished (see Fig.’s 6.13, 6.14, 6.15, and 6.16).

Figure 6.9 shows the tracking error with respect to the Vicon system for all of the tests performed. Assuming the position of the camera center remains fixed with rotation causes noticeable errors with increased pan/tilt angles. Note that the Euclidean norm of the pan and tilt angles was used to highlight its effect on tracking error where tracking error noticeably increases for angles $\geq 30^\circ$ at distances less than 2000 mm (see Fig. 6.9 (top)). Figure 6.9 (bottom) shows a box plot of error versus distance where the central

mark is the median, the bottom and top lines of each box are the 25th and 75th percentiles, respectively, the whiskers are the furthest points not considered outliers, and outliers are plotted separately.

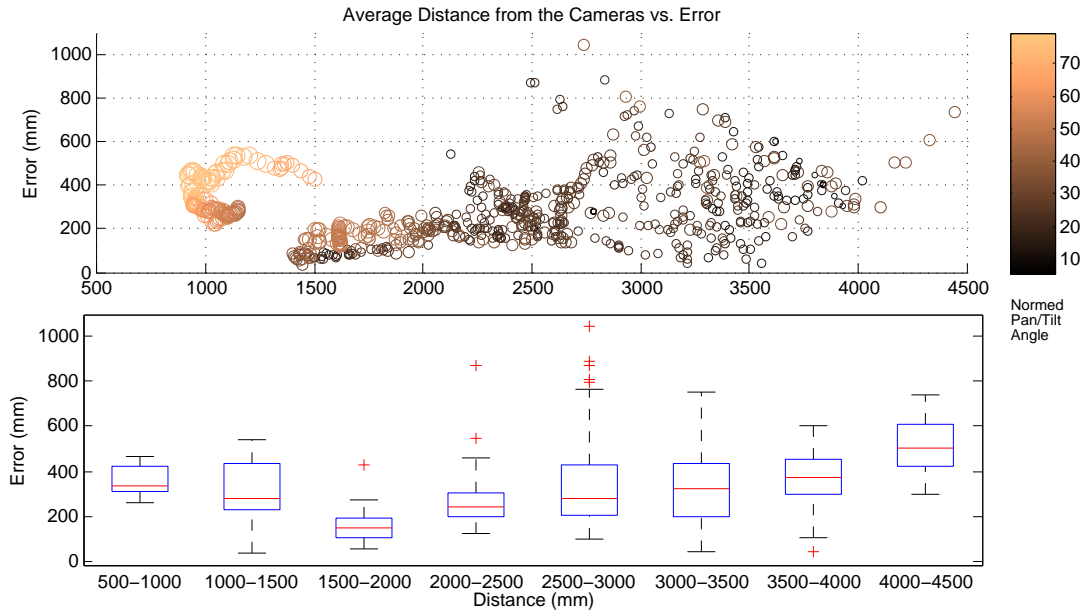


Figure 6.9: Error versus the average distance of the moving object from the cameras. Color and circle size is used to distinguish larger pan/tilt angles (top), and a box plot (bottom) is used to show the median (red line), 25th and 75th percentiles as the bottom and top edges of each box, whiskers as extreme points, and outliers plotted individually.

The following moving objects were tracked effectively (i.e., greater than 200 out of 275 frames) using two cameras:

- a remote-controlled car
- a remote-controlled helicopter
- a remote-controlled quadcopter
- a person
- a hand
- a checkerboard

6.5.4 Focal Length Estimation. The ideal focal length is estimated based upon the distance of the object from the camera (see Sec. 6.2.2). First, parameters are set for

establishing thresholds and an ideal focal length to meet the intent. The intent is to obtain higher resolution images of the moving object captured in full so that no part of the object is outside the image frame. The ideal pixel resolution is to fill 40% of the image width with the moving object. The following percentages are given for filling the screen with the moving object: 1) the lower threshold is set to no smaller than 20% of the image width, and 2) the upper threshold is set to no larger than 60% of the image width. Thus, we have the following parameters using Eq. 6.12 with varying r settings as follows:

- $r = 0.4$ for ideal focal length determination,
- $r = 0.2$ for determining lower focal length threshold,
- $r = 0.6$ for determining upper focal length threshold.

Should one camera lose the object, the other camera will continue to track. In the case of losing a moving object, the actual focal length becomes the default if the commanded focal length become less than the actual focal length.

Figures 6.10 and 6.11 show the actual, commanded focal length for each camera determined by the “two camera, PTZ system”, the ideal focal length determined by Vicon, and the upper and lower focal length thresholds using Vicon for the x-axis, y-axis, z-axis and below camera-level testing. Note that the focal lengths for cameras 1 and 2, are 1,090 pixels and 1,087 pixels, respectively. The actual focal length for all scenarios falls within the specified criteria and can be seen to slowly change over time due to setting the time constant in Eq. 6.13, k , to a value of 0.05. The figures show that automatic digital zoom may be accomplished simply by adding in the size of the moving object being tracked and specifying how much of it should fill the image frame by setting, r . Setting the zoom time constant, k , and desired image resolution ratio, r , gives the ability to maximize the object size for higher resolution measurements.

6.5.5 Additional Design Considerations. The system was designed such that each camera operates independently in addition to making it simple to add more cameras. Also, changing the test subject is easily done by stopping movement of one test subject and starting movement of another within each camera’s field of view. Remote-controlled helicopter, person, and remote-controlled car tracking show the moving object tracking

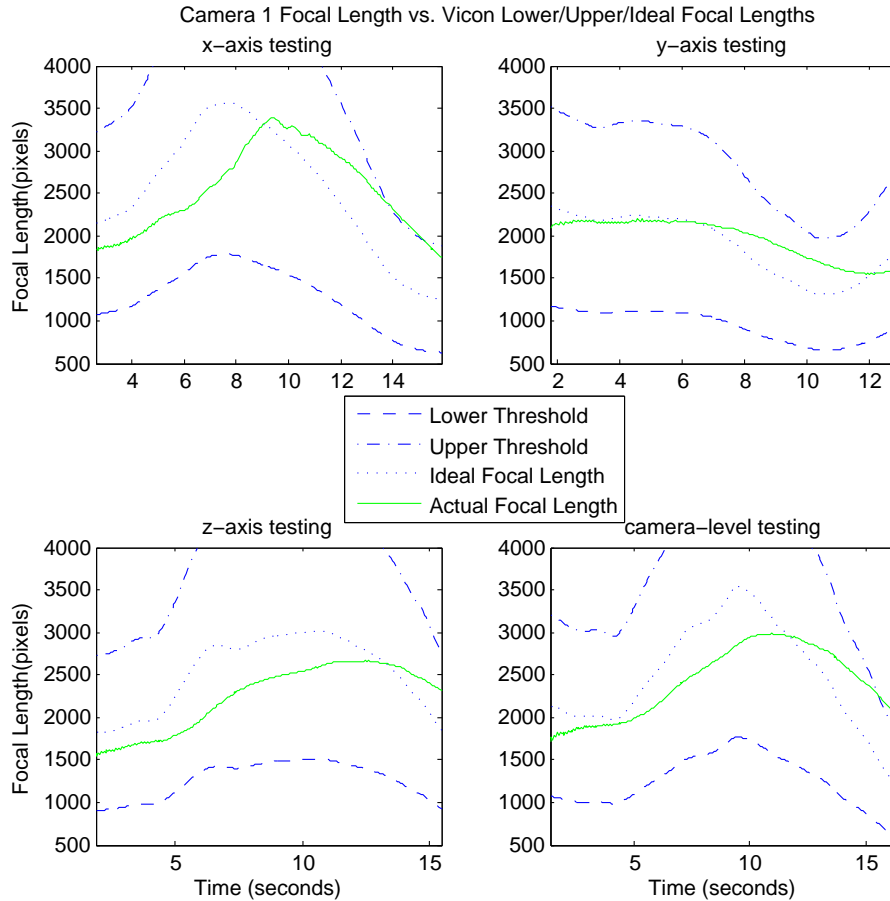


Figure 6.10: Camera 1 focal length vs. Vicon lower/upper/ideal focal length.

capability of the system developed in this chapter in Fig. 6.12. The independence of each camera allows the ability for one camera to track until the other catches the test subject within its Field of View (FOV) enabling both zoom and 3D position estimation. Tracking abilities include complex backgrounds, varying object shapes and sizes, various lighting conditions and varying speeds.

6.6 Conclusions

The system developed in this chapter requires no training or matching of features and captures faster, single moving objects better than existing general-purpose methods with the ability to zoom and estimate 3D position. The theory, parallel implementation, GPU-based functions and algorithm provide for a unique photogrammetry system capability without the need for retro-reflective markers. Modifications may be made to easily add multiple

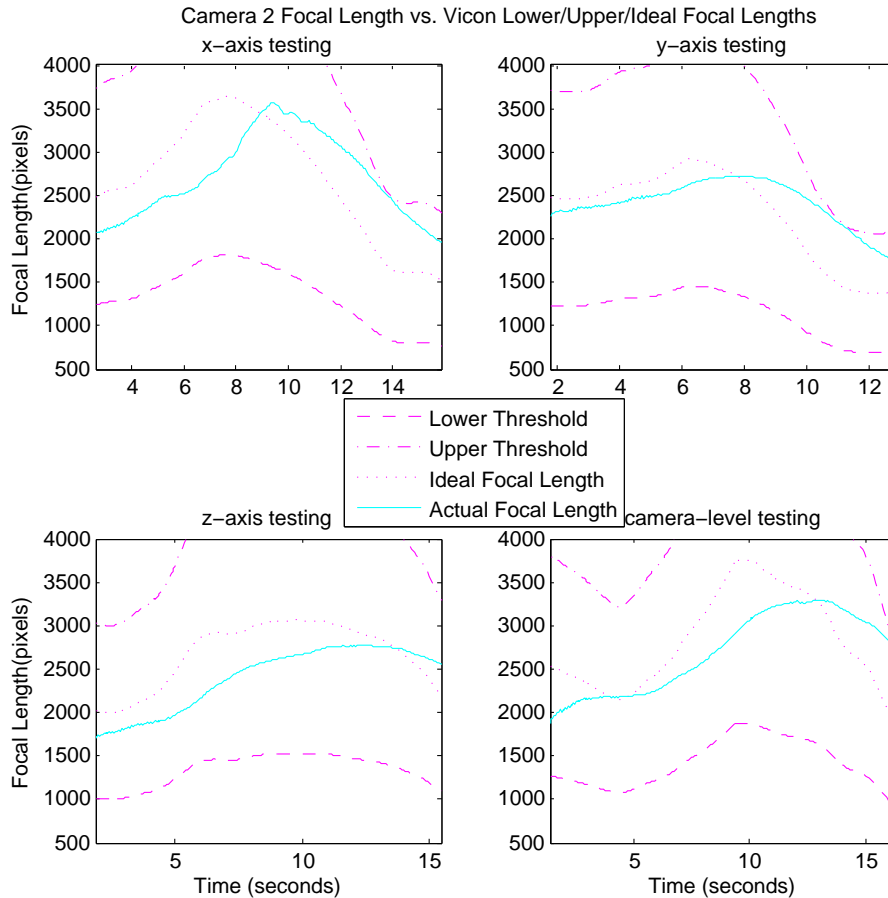


Figure 6.11: Camera 2 focal length vs. Vicon lower/upper/ideal focal length.

cameras. Zoom parameters may be adjusted for higher resolutions. The system is also very portable for various indoor or outdoor applications. The parallel implementation for CV-tracking developed in this work is versatile allowing the ability to test other CV methods with a PTZ system using known camera motion.

Several tests were performed demonstrating the ability to track with two cameras, to estimate the moving object's 3D position, and to digitally zoom according to a commanded focal length. The 3D position data was compared to data obtained by a Vicon Motion Systems, Ltd. system showing a testing average over 1349 frames, 80.5 seconds (i.e., 16.8 frames per second), of $290.1 \text{ mm} \pm 100.4 \text{ mm}$. Focal lengths for both cameras were shown to fall within set lower and upper focal length thresholds with the potential to digitally zoom up to four times the actual focal length. Different objects were tracked to show the system's generic tracking capabilities.

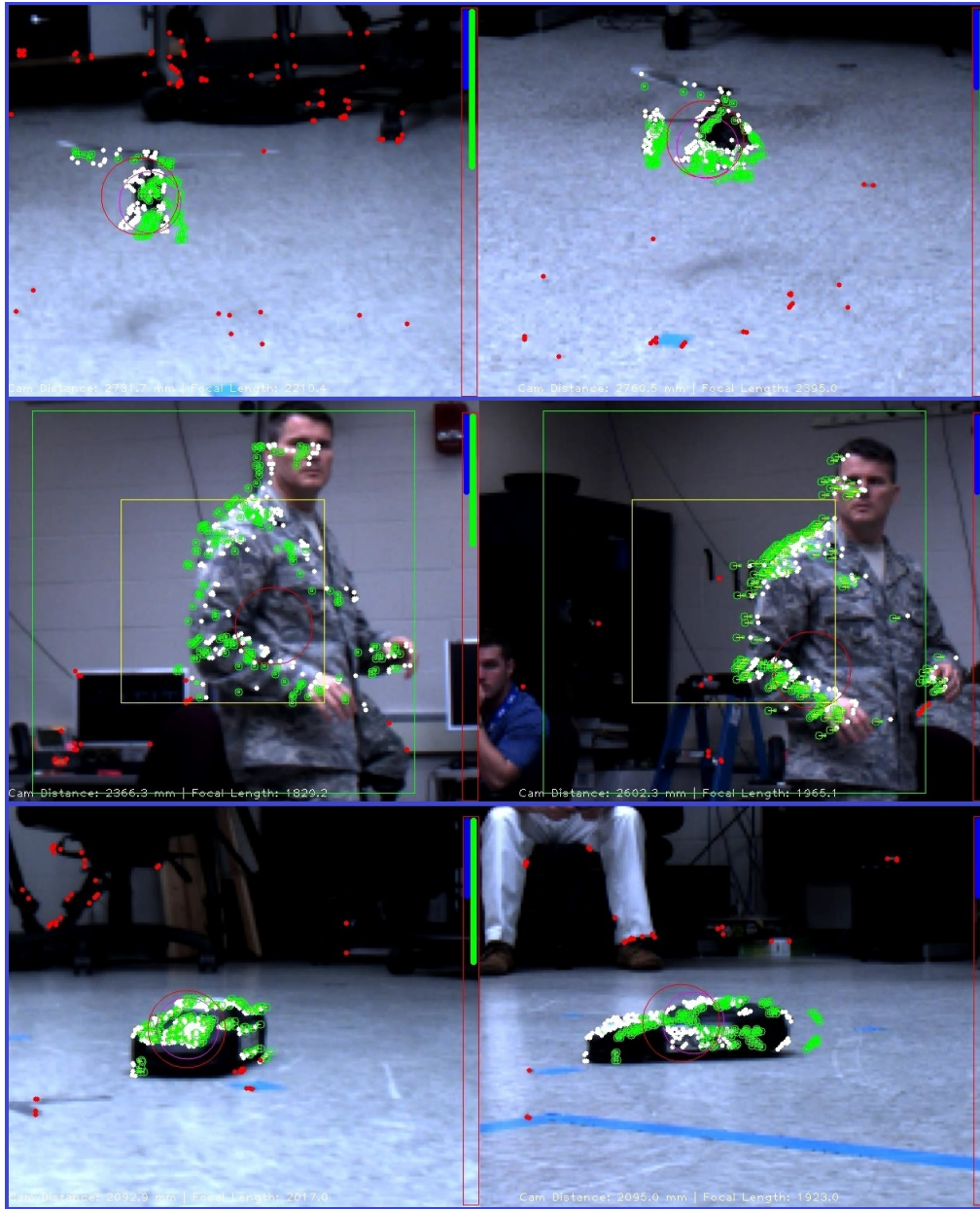


Figure 6.12: Generic moving object tracking using “two-camera, PTZ system”.



Figure 6.13: X-axis testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system”.

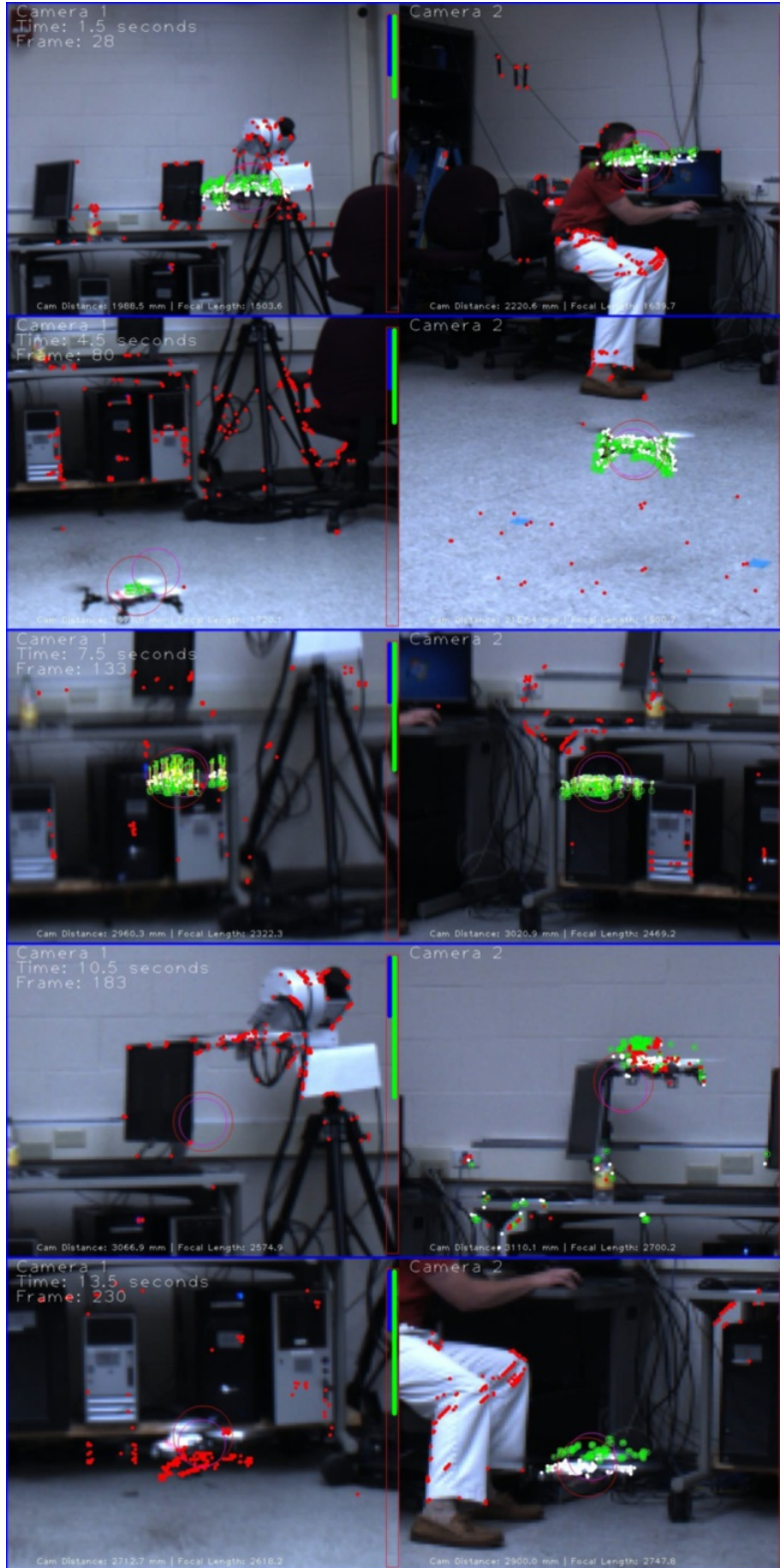


Figure 6.14: Y-axis level testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system”.

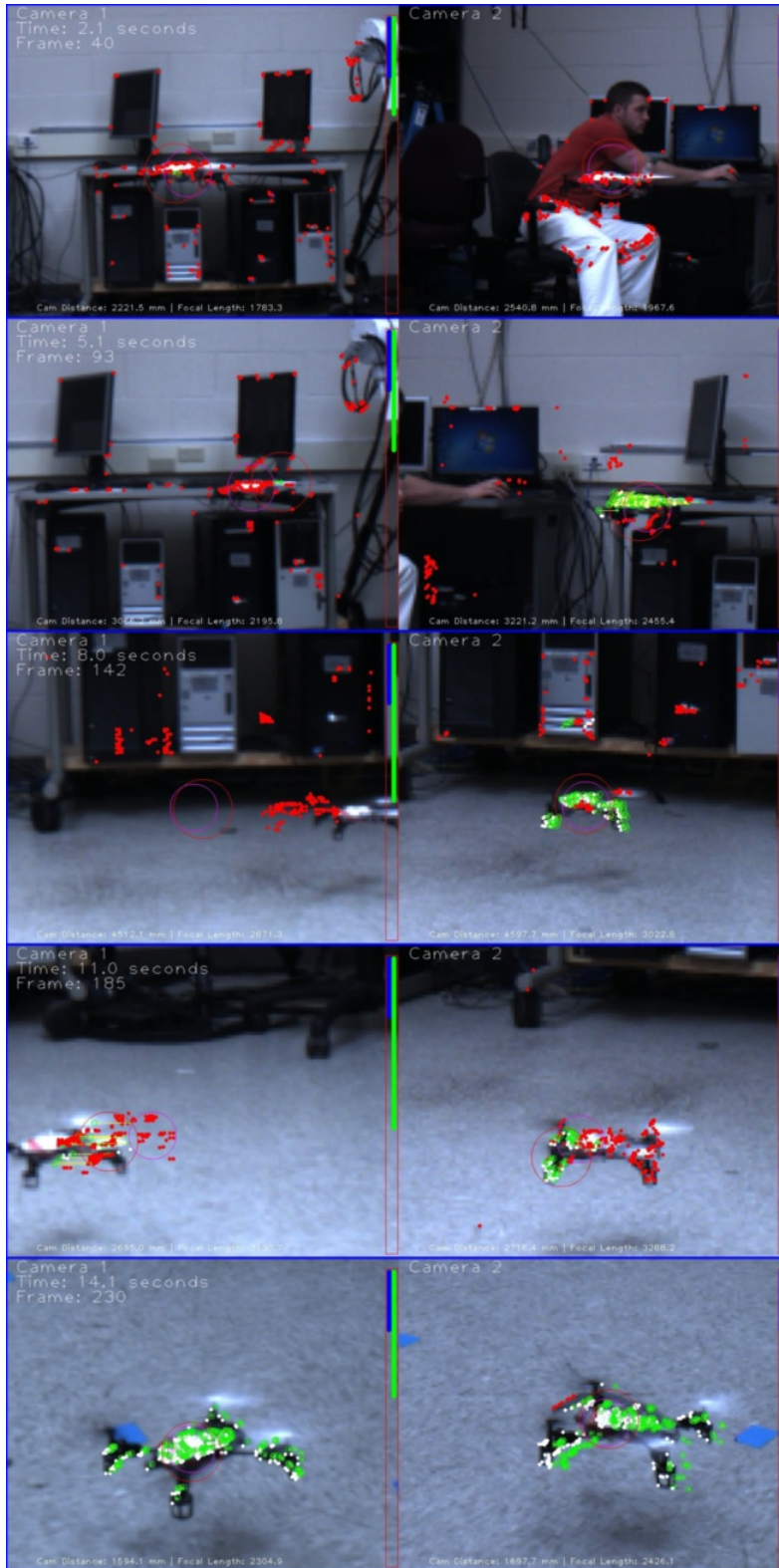


Figure 6.15: Z-axis level testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system”.

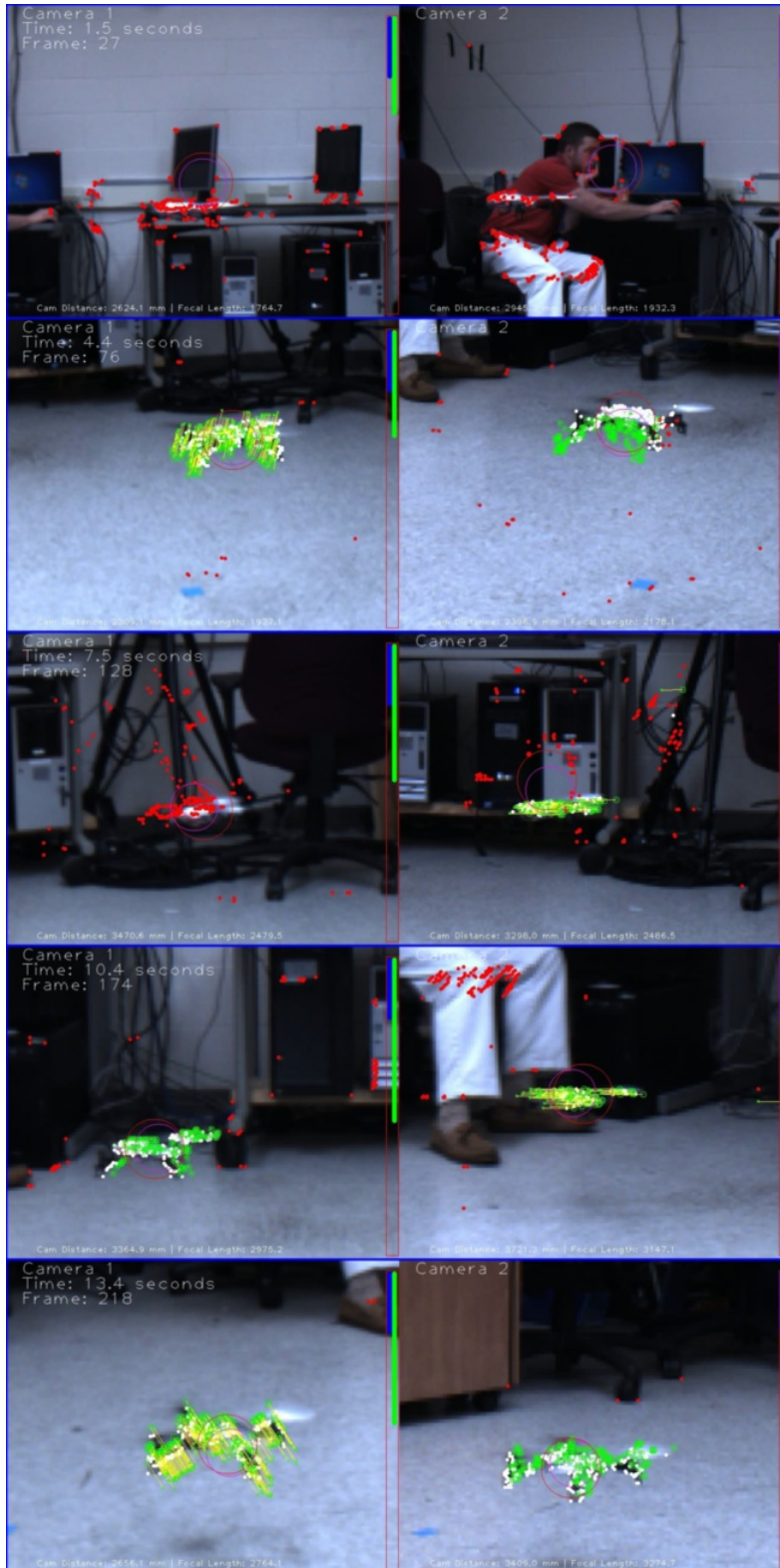


Figure 6.16: Below camera-level testing of the quadcopter’s 3D position using Vicon and “two-camera, PTZ system”.

VII. Discussion, Conclusion, and Future Work

State-of-the-art non-contact measurement systems currently track points in three-dimensional (3D) space using specialized cameras, requiring retro-reflective markers. The importance of creating a Computer Vision (CV) Pan/Tilt/Zoom (PTZ) non-contact measurement system is to advance the state-of-the-art beyond marker requirements and laser limitations. Parallelization and Graphics Processing Unit (GPU) programming enable CV algorithms and other systems to run with ease on a multi-core computer with the ability to expand to more PTZ cameras as needed. The use of CV capabilities for 3D point tracking may be extended to a multitude of points based on the use of image information and views from multiple cameras containing corresponding points. The importance of adding PTZ is to enable a dynamic system capability for varying the attainable angle and Field of View (FOV) for capturing high-resolution, high frame-rate images of the object being measured.

7.1 *Flapping Wing Research*

One particular area of research that makes this system relevant is understanding the flight dynamics of flying organisms through photogrammetric, state measurements. Many small, biological flappers go undetected in households and in our everyday surroundings. Their ease of maneuvering through complex flight paths make studying biological flying organisms an interesting area of research. Numerous military applications are feasible with the development of such small, maneuverable devices. Tracking and measuring these devices provides insight into the complex aerodynamic, aeroelastic, and wing-warping effects that make their maneuverability possible. Adding contact measurement devices to these Unmanned Aircraft System (UAS) would only hinder their normal flying operations. Therefore, non-contact methods are required towards measuring such flight characteristics. In developing a CV system, 3D measurements from stereo vision systems could be used to inform and validate the motion of a flapping UAS to provide finite element and Computational Fluid Dynamics (CFD) analyses. Target photogrammetry has been validated by laser scans and texture photogrammetry has been found to be comparable in accuracy [62]. Therefore, careful selection of proper CV algorithms could lead to comparable target and texture photogrammetry results using a collection of synchronized images from multiple cameras.

7.2 Conclusions

Visual tracking of a quadcopter was performed for ≥ 10 seconds. A specified focal length fell within the $\pm 20\%$ thresholds and is verified using truth data obtained with an industrial turn-key photogrammetry system. In addition, pan and tilt movements are verified leading to successful completion of the objectives and exit criteria listed in Table 7.1.

Table 7.1: Fulfillment of exit criteria.

Exit Criteria	Specific Criteria	Achievement
Each camera zooms out and falls within specified thresholds Pass	$0.2f_{idl} < r_{des} < 0.6f_{idl}$ ¹ for ≥ 10 seconds	The rate at which the camera zooms is sufficient to maintain a lower and upper threshold of $\pm 20\%$ over an interval ≥ 10 seconds. Up to 3.2X the focal length is achieved for each camera during the course of the experiments associated with truth data.
Each camera zooms in and falls within specified thresholds Pass	$0.2f_{idl} < r_{des} < 0.6f_{idl}$ ¹ for ≥ 10 seconds	The rate at which the camera zooms is sufficient to maintain a lower and upper threshold of $\pm 20\%$ over an interval ≥ 10 seconds.
Each camera pans and continues to track the test subject Pass	Pan $\pm 10^\circ$ movements during test	Measuring pan movement $> \pm 10^\circ$ occurred while visually tracking a test subject over an interval ≥ 10 seconds. Pan ranges for camera 1 are between -70.2° to 34.3° , and for camera 2 are between -35.8° to 35.3° .
Each camera tilts and continues to track the test subject Pass	Tilt $\pm 5^\circ$ movements during test	Measuring tilt movement $> \pm 5^\circ$ occurred while visually tracking a test subject over an interval ≥ 10 seconds. Tilt ranges for camera 1 are between -40.6° to 3.9° , and for camera 2 are between -75.7° to 1.6° .
Each camera pans, tilts, and zooms and continues to track the test subject Pass	$0.2f_{idl} < r_{des} < 0.6f_{idl}$ ¹ for ≥ 10 seconds Pan $\pm 10^\circ$ movements during test Tilt $\pm 5^\circ$ movements during test	The previous four objectives are fulfilled over an interval of ≥ 10 seconds while visually tracking the test subject.

¹Where f_{idl} is the ideal focal length using truth data.

Other objectives considered, not listed as part of the exit criteria from Chapter **I**, were also fulfilled. These objectives are listed along with specific information pertaining to their fulfillment.

- *Processing the CV algorithm should be greater than 30 frames per second (fps) in order to provide the Pan/Tilt Unit (PTU) ample time for processing and executing commands*

Processing the CV algorithm developed in this work averaged 12.7 ms with a range of 10 – 17 ms leaving ample time for the PTU to process and execute commands.

- *Digital zoom should be based on the distance of the test subject from the camera*

Digital zoom functioned independently based on the location of the test subject as shown in Chapter **VI**.

- *Incorporation of parallel-processing and GPU-based algorithms*

GPU-based algorithms are implemented starting in Chapter **V**. Parallel implementation is introduced in Chapter **VI**.

- *Development of a state-of-the-art PTZ camera system capable of tracking any UAS or moving object faster than any other generic, CV PTZ system*

The culmination of this work shown in Chapter **VI** describes a state-of-the-art PTZ camera system capable of tracking any UAS or moving object faster than any other generic, CV PTZ system.

- *Development of a versatile, parallel implementation platform, capable of examining the performance of other CV algorithms*

The system is versatile by providing a parallel implementation of CV-tracking with PTZ cameras capable of examining the performance of other CV algorithms.

- *Design of an easy setup only requiring a calibration board, enabling a versatile, portable CV PTZ system*

The system is easy to set up. The system is portable. The system only requires a 270 mm x 210 mm calibration board to establish a world coordinate system. Other calibration sizes are most likely acceptable, but have not been tested with the current system.

A real-time, CV PTZ multi-camera system capable of tracking at ~ 17 frames per second was developed using a parallel implementation and GPU-based algorithms. The system developed in this work requires no training, matching of trained or pre-defined features, or initialization/operator selection and captures faster, single-moving objects better than existing general-purpose methods with the ability to zoom and estimate 3D position. The importance behind not having to train or set pre-defined features include: reducing the setup time and the ability to easily change test subjects. The general-purpose method presented in this work enables moving objects of various shapes and sizes to be tested with only requiring a pre-defined length or width of the object size whichever is greater, a textured object, and a static background. The theory and approximations are provided to estimate *background motion* for optical flow background subtraction for pan, tilt, and zoom movements. Varying degrees of camera movement were analyzed and a classification method was developed for determining whether points are background, motion or noise. The magnitude of the camera movement was varied to show accuracy of the estimation of the background motion by comparing it with the measured optical flow. Algorithm processing times are shown for various image resolutions and number of features showing how performance can scale for various frame-rates in real time. The experiments with the static background showed minute benefit in classifying points as background further than 8 pixels from the predicted location. Actual tracking experiments showed that greater angles, greater than 4.4° between successive images, could be achieved by tracking at speeds greater than 1.2 m/s with the system operating between 6-18 fps.

The theory, parallel implementation, GPU-based functions and algorithm provide for a unique photogrammetry system capability without the need for retro-reflective markers or adding numerous lasers. Modifications may be made to easily add multiple cameras by adding parallel tasks for each camera, parallel tasks for each additional pan/tilt unit, and adjusting the inputs/outputs of the last parallel task associated with determining multiple line intersections. Zoom parameters may be adjusted for higher resolutions. The system is also very portable for various indoor or outdoor applications. The parallel implementation technique developed in this work is versatile allowing the ability to test other CV methods with a PTZ system using known camera motion.

7.3 *Limitations of the Computer Vision (CV) Pan/Tilt/Zoom (PTZ) System*

The limitations listed in this section are those related to the current system tested in this work.

The UAS must move fast enough for the CV-tracking algorithm to classify it as moving. This is based on the background threshold setting, b_{bkd} . For example, b_{bkd} was set to 8 pixels in this work requiring movement from one frame to the next greater than b_{bkd} to be seen as moving. A moving object could potentially move slow enough and not be picked up by the tracking algorithm.

The CV-tracking algorithm loses the UAS in high performance lighting. Tracking performance is based on the limitations of the optical flow tracking algorithm used. Other algorithms or methods could be explored to reduce illumination effects for specific tracking applications.

The 3D position estimation of Chapter VI, Sec. 6.2.1, was sensitive near and between the cameras. As the UAS approached the area between the cameras (within ~ 200 mm of camera vectors being in parallel), the results for 3D position estimation were asymptotic. Additional cameras could help prevent this baseline issue by employing the line intersection using two or more cameras where the UAS is not near the baseline.

Portability requires a power supply, lighting (application-specific), the system described in Chapter VI, and the calibration board. Setup requires the calibration board and knowledge of the size of the UAS.

The CV PTZ system only tracks a single UAS or moving object. Other moving objects entering a given camera's FOV will cause tracking loss or degrade tracking performance.

7.4 *Future Work*

Ideas are presented in this section as additional areas of research for incorporation into this CV PTZ system.

7.4.1 Pose Estimation. Pose estimation was briefly explored; however, due to time limitations this was not implemented in this work and is recommended for future

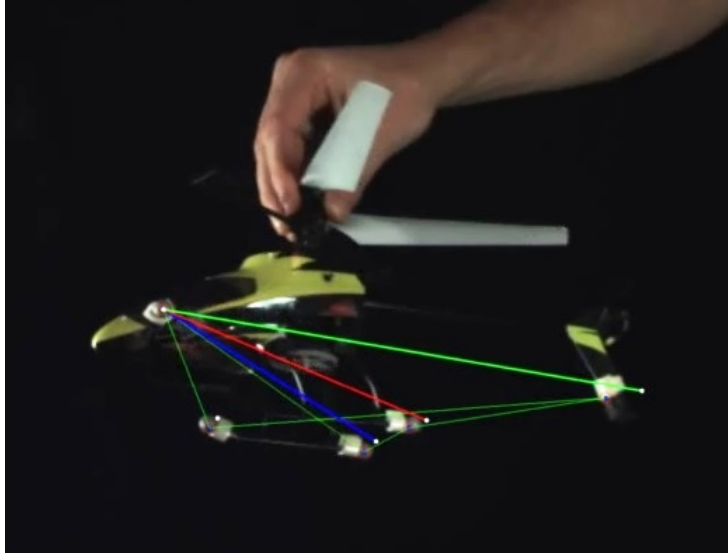


Figure 7.1: Example output using the POSIT algorithm on a helicopter with five known model coordinates.

work. The Pose from Orthography and Scaling with Iterations (POSIT) algorithm [38]¹ was implemented successfully over a sequence of images. This was done using known model coordinates and by placing optical flow points on those known locations in order to estimate pose (see Fig. 7.1).

Capturing image sequences of a moving object, with known model coordinates, with the system developed in this research and applying the POSIT algorithm would enable pose estimation.

7.4.2 Computer-Controlled Mechanical Zoom. In order to truly realize the capabilities of zoom, mechanical zoom is essential in obtaining high-resolution images. This is primarily a hardware issue where only minor modifications to the code would be required. It is expected that the features obtained using a mechanical zoom would also prove more valuable for tracking purposes since each pixel is not digitally interpreted as with digital zoom.

7.4.3 Image Registration. Many image registration techniques are available with claims of sub-pixel accuracy and matching [97, 100]. Companies continue to add CV ca-

¹OpenCV 2.4.3 Library - cvPOSIT

pabilities to their software, but automated processing often still requires operator selection/initialization. Jennings, et al. [62] state “A limitation is the often labor-intensive process of feature matching despite advances in automatic feature extraction”. Therefore, automation is sought using the studied CV algorithms for both real-time and post-processing activities.

7.4.4 Data Fusion. Fusion techniques should also be considered in post-processing to speed up operations and to ensure that matched points between frames are optimally selected using collected information to provide the best results. The Kalman Filter (KF) is used in this work to improve the individual PTZ camera tracking. There are many other areas using various Data Fusion (DF) techniques for improving the tracking process. Some areas to consider include:

- Using a KF to improve 3D point estimation
- Fusing images using image registration techniques
- Fusing multiple cameras to obtain 3D points

7.4.5 Velocity control for Pan/Tilt Unit (PTU). The CV PTZ system developed in this research tracks using x and y positions. The pan/tilt unit starts and stops abruptly for every given position command while tracking a moving object. Research should be conducted to utilize the position commands and the time required between movements for velocity control. This could help prevent the abruptness of motion currently seen with the existing system and should help to smooth out the tracking process.

7.4.6 Tracking of a Flapping-Wing Unmanned Aircraft System (UAS). Lastly, the CV PTZ system developed in this work should be used to track a flapping-wing UAS and record high frame-rate, high resolution images. These images should be reviewed to determine how many cameras may be required in order to perform photogrammetry of a free-flight, flapping-wing UAS.

7.5 Summary

The motivation of this research was to perform photogrammetry of a flapping-wing UAS. The first steps of tracking, using a non-contact method, and zooming in on a UAS were achieved. Next steps include incorporating additional PTZ cameras (with computer-controlled mechanical zoom) followed by tracking a flapping-wing UAS.

The current system utilizes cameras achieving an average frame-rate of 294 fps; the cameras are capable of capturing ~ 12 frames per flap for a flap frequency of 25 Hz. A flapping-wing UAS should be captured at this frame-rate, filling at least 20% of the image. In the beginning stages, these images could be used for matching specific points such as the nose, tail, and wingtip of the UAS.

In order to perform photogrammetry, the motivation behind this work, further study into photogrammetry techniques is required for developing a 3D model of the UAS for every synchronized capture using multiple cameras. The point in capturing high-resolution images is so that UAS features can be matched between synchronized images with greater accuracy to increase measurement precision. One recommendation is to explore the Point Cloud Library [103] for determining the 3D object representations for every synchronized capture.

This work advances the state-of-the-art in developing a parallel implementation of a non-contact, CV-tracking technique using PTZ cameras. This technique is employed with multiple cameras to determine an estimated 3D position of a moving object (e.g., a UAS) enabling the ability to zoom to a desired focal length based on the moving object's size.

PAGE INTENTIONALLY LEFT BLANK

Appendix A. Fusion Background

A.1 Introduction

This appendix provides background information regarding data fusion models. The focus will be on highlighting the revision to the Joint Director of Laboratories (JDL) data fusion model while reviewing other models that either relate or try to improve upon the JDL data fusion model. There have been several books and papers written recently regarding fusion (i.e., data fusion, multi-sensor data fusion, high-level data fusion, image fusion, mathematical techniques, etc.). Each book and paper covers models ranging from the Joint Directors of Laboratories data fusion model to the Observe, Orient, Decide and Act (OODA) model created by Col (Ret.) John Boyd, USAF, during the Korean War. These models range from being functional (i.e., illustrating the primary functions, relevant databases, and interconnectivity), architectural (i.e., specifying hardware and software components, associated data flows, and external interfaces), and mathematical (i.e., the algorithms and logical processes) [57]. multi-sensor data fusion has been broken down into six categories [81, Hackett and Shah (1990)]:

1. scene segmentation
2. representation
3. three-dimensional shape
4. sensor modeling
5. autonomous robots
6. object recognition

Differing views on fusion definitions exist and communities have tried to disassociate the term from the tracking world. One could contend that the above categories fit the tracking mold and therefore other categories likely exist. First, we start with a discussion of those differing views on terminology and then work through the JDL data fusion model and its evolution as a basis for comparison to other related fusion models. Other fusion models are briefly covered in the remaining sections of this appendix.

Although not a comprehensive listing, other surveys/reviews are listed here containing much of the information used in this appendix:

- Data Fusion Lexicon [127]
- Manufacturing automation fusion issues [82, Henderson, Allen, Mitiche, Durrant-Whyte, & Snyder, 1987]
- Spatial Reasoning [82, Kak & Chen, 1987]
- Techniques for mult-sensor integration and fusion [82, Dougherty & Giardina, 1988, pp. 273-277]
- Pattern Recognition [82, Therrien, 1989, pp. 232-242]
- An Introduction to mult-sensor Data Fusion [55]
- An Introduction to Sensor Fusion [43]
- Handbook of mult-sensor Data Fusion [56]
- JDL Level 5 Fusion Model User Refinement Issues and Applications in Group Tracking [19]
- Mathematical Techniques in mult-sensor Data Fusion [57]
- mult-sensor Data Fusion: An Introduction [89]
- High-Level Data Fusion [36]
- The International Society of Information Fusion (ISIF) provides current areas of interest as target tracking, detection, applications of information fusion, image fusion, fusion architectures and management issues, classification, learning, data mining, and Bayesian and reasoning methods [1].

A.2 Fusion

Let us start with the term fusion and expand into the worlds of data fusion, sensor fusion, and image fusion. The New Oxford American Dictionary refers to fusion as the process or result of joining two or more things together to form a single entity [87]. Let us first take a look at “things” from a mathematical point of view. If we take “things” and fuse them together, what does this mean using mathematical notation? We will call these “things” x and y and put them together to form a single entity, z . From this examination, we can infer that a process exists for producing a fused result (i.e., x “fused” $y = z$,

where “*fused*” refers to mathematical operation(s) that could be used to combine x and y (i.e., $+$, $-$, \times , \div , \oplus , \ominus , \otimes , \cup , \cap , etc.). One example is the use of an inner product, where a vector is turned into a scalar. From the scalar, we are unable to derive or separate this scalar back to its original or un-fused state without a priori knowledge. As with this example after fusion has occurred, one cannot perform a complete separation. We have covered here that the process of fusion has many forms. This complicates matters as to what form provides the best fusion and many of the aforementioned books and papers try to address findings and improvements that lead to better fusion algorithms. Fusion continues to be an highly active field of research.

So it is understandable that there can be some confusion between fusion and integration since integrate means to combine (one thing) with another so that they become a whole [87]. These definitions sound identical with the exception that fusion forms a single entity and integration forms a whole. So we can make a separation here by using another example. Say x and y are two distinct pieces of apple pie. Then x fused with y to form z might look like throwing both pieces into a blender so that both pieces are indistinguishable, but yet they are fused together. There is no separation between the two, but yet they contain all of the properties with the exception of shape. We lose the shape of each of the pieces in this example which can be linked back to the fusion process we chose which led us to a loss of information (information we could retain with a priori knowledge). They no longer look like x and y , rather, they look like z and are a single entity of apple mush. Now if we take x and integrate it with y , we have two pieces side by side or one on top of the other or pick your combination such that both pieces retain their physical properties (i.e., you can still extract x from z to get y). The point of this exercise and these examples is to show that fusion and integration are closely related, and that a fusion approach could be considered integration. If we have a priori knowledge and are able to separate the data back into its original components, then we will refer to this as integration.

A.2.1 Data Fusion. Multiple definitions have been proposed for the term data fusion, and it has often been associated with tracking and military application. Others have tried to broaden this definition such that is more encompassing. One such attempt expresses data fusion as ‘the process of utilizing one or more data sources over time to

assemble a representation of aspects of interest in an environment.’ [46, Lambert, D. A., ‘An Exegis of Data Fusion’, 2001] Another attempt to lead one away from “data fusion” as a tracking-based definition is where it is used to denote fusion of raw data. When performing a search of the term “data fusion”, it is also associated with earth image data processing and is defined as “a formal framework in which are expressed the means and tools for the alliance of data originating from different sources [2]”. Elmenreich [2001] suggests that if the earth community’s definition is used, that a prefix like “low-level” or “raw” would be adequate for making this association [43]. High-level data fusion is the study of relationships among objects and events of interest within a dynamic environment [36]. The initial data fusion lexicon as proposed by the White, et al. [1991], A process dealing with the association, correlation, and combination of data and information from single and multiple sources to achieve refined position and identity estimates, in addition to complete and timely assessments of situations and threats, as well as their significance [127]. It was later revised by Steinberg, et al. [1999] as ‘the process of combining data to refine state estimates and predictions’ [113]. In the sense that this paper is for use in developing a state-of-the-art computer vision measurement tool, the revised version of data fusion is used with the idea that we seek refined state estimates and predictions. This definition also falls nicely under the term fusion as described above. A comparative analysis of these definitions is as follows:

Table A.1: Comparison of definitions for Fusion and Data Fusion.

Fusion (New Oxford American Dictionary)	Data Fusion (Bowman, Steinman and White)
The process or result	The process
of joining two or more things together	of combining data
to form a single entity	to refine state estimates and predictions

We see that the data fusion definition differs only by expressing things as data and the single entity as a refinement of this data in terms of state (i.e., position, velocity, acceleration, attributes, etc.).

A.2.1.1 Fusion Types. Now that data fusion has been defined, we can break this down into the four types of data fusion. Mitchell [2007] categorizes data fusion types as follows [89]:

1. Property fusion - measurement of the same property to be fused (e.g., location estimates and predictions)
2. Attribute fusion - measurement of different properties to be fused (e.g., color estimates and predictions)
3. Range/domain fusion - measurement of the same attributes for different ranges or domains (e.g., specified area)
4. Time fusion - measurements of temporal information to be fused (e.g., time step k and $k+1$)

A.2.2 Sensor Fusion. Now let us move on to sensor fusion and its definition. We start by examining processes of sensor fusion that we can relate to from our human senses. The traditional five senses are sight, hearing, touch, smell and taste. Human beings use these senses to make decisions and actions based on what is being sensed. In a sense, each of us uses the senses we have been given to determine the best course of action (i.e., highest utility). Each of our senses interacts with others to tell our brains what it is we want to do. We see a baked apple pie, fresh from the oven. We smell the aroma of freshly baked apples and cinnamon and see the drizzling juices culminating in a state of salivating desire to cut a piece and enjoy. On the other hand, the smell could be that of a pungent, unpleasant stench from the burning crust. The appeal to partake leaves instantly and produces a desire to vacate the premises as soon as possible. The interaction between sight and smell are intertwined (fused) to produce desires and outcomes in this case. If the pie were delicious looking and awful smelling, the decision to eat the pie would be more difficult. If you are starving, then the utility of eating overrides the utility of evacuating or not eating. Let us look at a few more examples as we explore the idea of sensor fusion. Going to an air show is another example that will relate more directly as to where this paper will put its attention. Airplanes, people, skydivers, acrobatic aircraft, etc. are all typical performances at an air show. How is it that one can track these flying objects as they cut through sky? As an aircraft flies off into the distance, it is still known that the flying object is an aircraft. It is stored in the memory banks of the human mind, and it is tracked. As one goes to bite some of their hot dog and to drink some choice beverage, their eyes lose sight of the target, yet they are able to hear a distant rumble and look back up (perform a search) and find the spec

off in the distance that is now approaching at a rapid speed. Experience would tell one that either ear plugs are required or that covering their ears at this point might be a good idea to prevent auditory overload. Again, we can see the interaction of several sensory devices to create an enjoyable experience (i.e., seeing, hearing, tasting, smelling and touching are all present in creating this state). Total fusion provides the enjoyable experience. Fusion of seeing and hearing helps one with the search process of determining the location of the flying object. Tasting and smelling satisfies the appetite of the individual along with many other fused processes for producing a desired effect. Several papers refer to humans and animals and use of their senses to make decisions about present and future actions (see [43, 55] for these examples).

A.2.3 Mult-Sensor Data Fusion. mult-sensor data fusion can be defined as “the theory, techniques and tools which are used for combining sensor data, or data derived from sensory data, into a common representational format [89].” The aim in performing sensor fusion is to improve the quality of the information, so that it is, in a sense, *better* than would be possible if the data sources were used individually [89]. The distinction between data fusion and mult-sensor data fusion is not made clear in other texts. For example, Hall and McMullen define data fusion and mult-sensor data fusion synonymously. The terms data and information are also used interchangeably so we will use the definition that uses data and makes use of multiple sources or sensors from Hall, et al. [2004]: “combining information from multiple sources or sensors to achieve inferences *not* possible using a single sensor or source” [57].

Let us look at four different ways that data fusion may improve the performance of a system [89]:

1. Representation: The information obtained during, or at the end, of the fusion process has an abstract level, or a granularity, higher than each input data set. The new abstract level or the new granularity provides a richer semantic on the data than each initial source of information.

2. **Certainty:** If V is the sensor data before fusion and $p(V)$ is the a priori probability of the data before fusion, then the gain in certainty is the growth in $p(V)$ after fusion. If V_F denotes data after fusion, then we expect $p(V_F) > p(V)$.
3. **Accuracy:** The standard deviation on the data after the fusion process is smaller than the standard deviation provided directly by the sources. If data is noisy or erroneous, the fusion process tries to reduce or eliminate noise and errors. In general, the gain in accuracy and the gain in certainty are correlated.
4. **Completeness:** Bringing new information to the current knowledge on an environment allows a more complete view on this environment. In general, if the information is redundant and concordant, we could also have a gain in accuracy.

A.2.3.1 Sensor Configuration. We can also look at various sensor configurations that help to classify multi-sensor data fusion systems. Three basic configurations are given by Durrant-Whyte [89]:

- Complementary - sensors do not depend upon each other
- Competitive - sensors measure the same information to reduce uncertainty and erroneous information
- Cooperative - information is derived that would not otherwise be sensed using single sensor information

A.2.4 Image Fusion. Image fusion is commonly described as the task of enhancing the perception of a scene by combining information captured by different modality sensors as described by Mitianoudis and Stathaki (Image Fusion: Algorithms and Applications, p. 85). Let us provide a comparative analysis of definitions once again to image fusion by rearranging phrases:

Using this table helps us to come up with a standard definition for image fusion like that of data fusion. “The task of combining information to enhance the perception of a scene” We see that the only difference between data fusion and image fusion is the result. Image fusion is tailored specifically to refining image states. Therefore, without diluting the terms we can express image fusion as image data fusion. Extending this terminology to

Table A.2: Comparison of definitions for Fusion and Image Fusion.

Fusion (New Oxford American Dictionary)	Image Fusion (Mitianoudis and Stathaki)
The process or result	Task (captured by different modality sensors)
of joining two or more things together	by combining information
to form a single entity	to enhance the perception of a scene

other specific areas of fusion may result in audio data fusion, signal data fusion, and other percepts that provide data to be fused.

A.2.5 Other Types of Fusion. Other forms of fusion may be the following:

A.2.5.1 People Fusion. Marriage may be seen as a form of fusion. For example, Genesis 2:24 [New International Version] states 'For this reason a man will leave his father and mother and be united to his wife, and they will become one flesh.' They will become one flesh goes along with becoming a single entity with respect to fusion.

A.2.5.2 Object/Physical Fusion. A simple example of object fusion would include the combination of different colors of Playdoh (i.e., combine red and blue to get green).

A.3 Fusion Models

We have talked a little about fusion and have gone over some examples in society and nature. The desire is to develop a system that can both track and measure an object or a system of objects with human ability and computer speed using various sensing capabilities to make decisions and perform actions based on expected utilities. Now let us focus on the functional, architectural and mathematical models that represent data fusion. The intelligence cycle, the JDL model and the Boyd control models were developed in the 1980's. The Dasarthy model, the Waterfall model and the Omnibus model came about to improve upon previous fusion models in the 1990s. Revisions and additions to the JDL model continued throughout the 2000s. The following subsections are a review of these models with the JDL model as the basis for comparison.

A.3.1 Joint Directors of Laboratories (JDL) Data Fusion Model. We begin with looking at the Joint Directors of Laboratories (JDL) model that was initiated by a U.S. DoD government committee overseeing U.S. defense-technology research and development [46]. The JDL commissioned a subpanel in 1985, now called the Data-and-Information Fusion Group (DIFG), to help bring order to the community. The DIFG focused on bringing about a common language and framework that included data fusion taxonomy, lexicon and a model which has since been reviewed and revised. The evolution of some of these changes is presented in this section.

The traditional roots of the data fusion community are in sensor fusion, where the “data sources” are established sensors and the “aspects of interest in the environment” are moving objects, each typically represented by a set of state vectors [46]. This relates back to the earth community’s definition of data fusion [2], but can be easily used with our revised definition as well, ‘the process of combining data (provided by data sources/established sensors) to refine state estimates and predictions (aspects of interest in the environment/moving objects).’ The JDL functional model was intended for communication among data fusion practitioners, rather than a complete architecture detailing various processes and their interactions [36]. The JDL model was designed to be a *functional* model – a set of definitions of the functions that could comprise any data fusion system. Process models specify the interaction among functions within a system [56]. Hall and McMullen [2004] do not make this distinction of functional and process models, but relate the interconnectivity as part of the functional model. Therefore, we will be able to observe that Boyd’s Observe, Orient, Decide and Act (OODA) loop model and others can be seen as process models that can be fit to the JDL data fusion model.

A.3.1.1 Original 1992 Model. We start with the original JDL data fusion model introduced in 1992 (see Figure A.1). This model introduces sources, pre-processing, level 1: object refinement, level 2: situation refinement, level 3: threat refinement, level 4: process refinement, a database management system, and a human computer interaction element. These elements are covered thoroughly in [55].

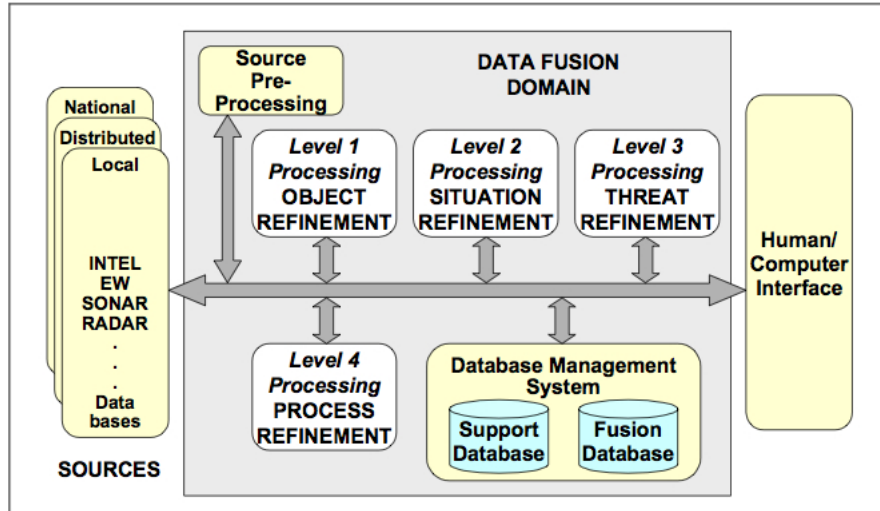


Figure A.1: JDL data fusion model (1992).

A.3.1.2 Steinberg, Bowman and White Revision, 1998. The revised JDL model from 1998 was developed under Steinberg, et al. [113] and is shown in Figure A.2. During development of the 1999 revised JDL data fusion model, it became clear to Steinberg, et al. [1999] that the initial definition for data fusion was too restrictive. So it was simplified and provided that 'data fusion is the process of combining data to refine state estimates and predictions' [113]. Furthermore, data fusion "levels" were revised in this model to account for some of the concerns of its use (i.e., that it is not a step by step process, that the focus is not just on tactical targeting, and that there are no distinguishing features for interpreting each level).

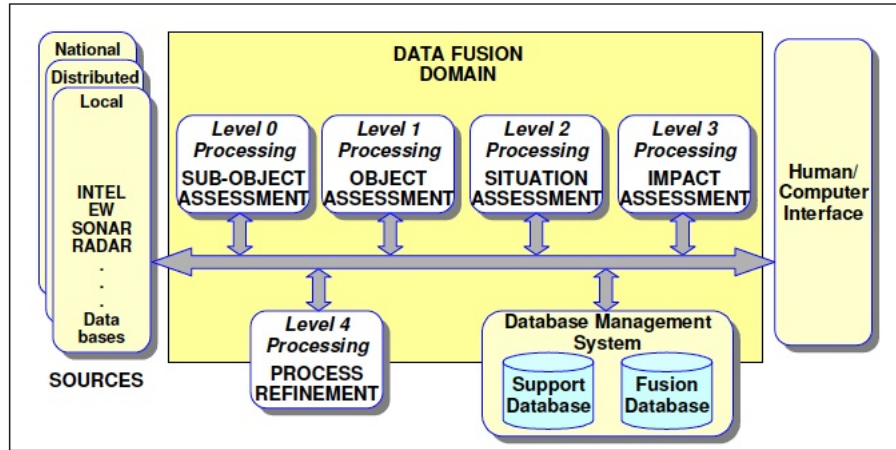


Figure A.2: Revised JDL data fusion model (1998).

The revisions are as follows: a new Level 0: sub-object assessment was added in place of the preprocessing element, object refinement was replaced by object assessment, situation refinement was replaced by situation assessment, and threat refinement was replaced by impact assessment. Each of these levels are characterized according to association process, estimation process and entity estimated to address the aforementioned concerns by distinguishing features of each level. These are described more clearly in the list below [113]:

- Level 0 - Sub-Object Data Assessment: estimation and prediction of signal/object observable states on the basis of pixel/signal level data association and characterization;
- Level 1 - Object Assessment: estimation and prediction of entity states on the basis of observation-to-track association, continuous state estimation (e.g., kinematics) and discrete state estimation (e.g., target type and ID);
- Level 2 - Situation Assessment: estimation and prediction of relations among entities, to include force structure and cross force relations, communications and perceptual influences, physical context, etc.;
- Level 3 - Impact Assessment: estimation and prediction of effects on situations of planned or estimated/predicted actions by the participants; to include interactions between action plans of multiple players (e.g., assessing susceptibilities and vulnerabilities to estimated/predicted threat actions given one's own planned actions);

- Level 4 - Process Refinement: (an element of Resource Management) adaptive data acquisition and processing to support mission objectives.

A.3.1.3 Steinberg and Bowman Revision, 2004. The 2004 model proposes extending JDL levels of data fusion to their dual levels in resource management. This approach provides a framework for less mature technology by enabling a parallel or dual framework called the Dual Node Network (DNN) technical architecture for performing performance assessment and design management functions [112]. This extension provides for affordability and reuse and can be further explored in [112].

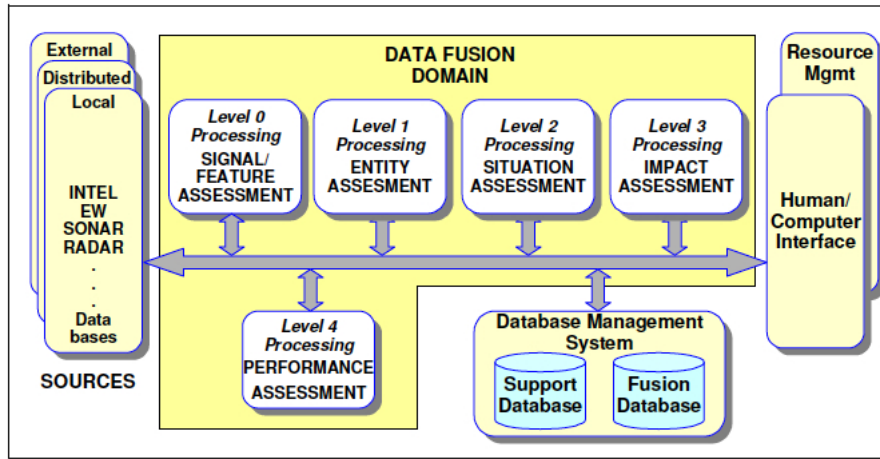


Figure A.3: Revised JDL data fusion model (2004).

Changes to the levels are as follows: Level 0: sub-object assessment was changed to signal/feature assessment, level 1: object assessment was changed to entity assessment, and resource management was added as another element next to the human computer interaction (HCI) element.

Further refinements were made regarding an understanding the internal processing of a fusion node, extending the JDL model on issues of quality control, reliability, and consistency in data fusion processing, asserting the need for co-processing of abductive/inductive and deductive inferencing processes, asserting the need for an ontologically-based approach to data fusion process design, and extending the account for the case of Distributed Data Fusion (DDF) [79]. Further information on these refinements can be found in [79].

Let us review these in more detail before proceeding onto more recent developments. Level 0 seeks to determine the presence of a signal and its current state (e.g., feature extraction in image data, signal data, etc.). In this level we are trying to figure out what is interesting about what is being sensed. Level 1 then proceeds with what has been detected (i.e., extracted features) from level 0 and associates reports. The general idea is that a set of reports contain features that can be associated across sensors or an individual sensor. Thus, we have a set of reports that represent an entity or entities based on the information detected in level 0. After associating these features with respect to a set of reports, level 2 (Situation Assessment), involves aggregating the entity or entities to provide any variety of relations - physical, organizational, information, perceptual - as appropriate to the given system's mission. Level 3 assignment is then used as a prediction function based on the developed situation assessment. Here the estimated or predicted utility or cost of an estimated world state (i.e., the location of a moving object of interest) can be evaluated and meet user-specified needs (i.e., capturing high-resolution images capable of being used for system identification or inferring a system's intent whether it is a friend or foe). The last level so far is Level 4 (Process Refinement). Estimation has already been accomplished at this point, so now there is a need for planning and controlling the process. A good question to ask at this point is, 'How can the system be improved'? This begins the process of refining the data fusion system being implemented.

A.3.1.4 Blasch and Plano JDL-User Data Fusion Model Introduction, 2002.

The last couple of models attached to the JDL data fusion model incorporate the user at all levels. As seen in Figure A.4, a level 5: user refinement is introduced at all levels to delineate human processing from machine processing of the data fusion architecture.

JDL-User Model

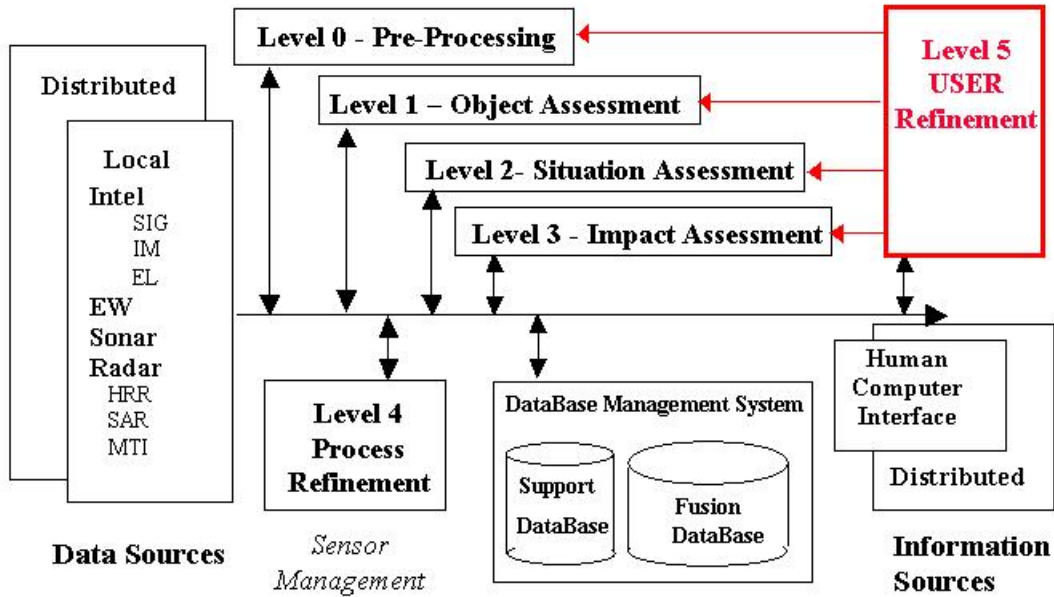


Figure A.4: JDL user data fusion model 2002 [19].

Level 5 focuses on the “human in the loop” (HIL) as related to the machine fusion process. For HIL there exists a need to address issues such as situation awareness, attention, workload, and trust. These issues and this model are explained in more detail in [19].

A.3.1.5 Blasch and Plano JDL-User Data Fusion Model Defense, 2005.

Defense of the Blasch and Plano JDL user data fusion model was proposed to highlight the need of the user-fusion model. Figure A.5 seeks to highlight the

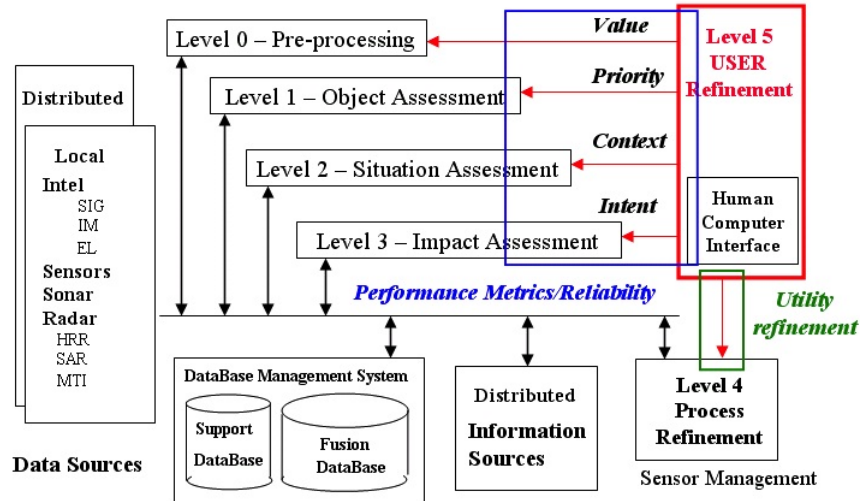


Figure A.5: JDL user data fusion model 2005 [20].

Table A.3: Review of levels and their respective roles regarding the JDL-user model [22].

Level	Role
0	Determines what and how much data value to collect
1	Determines the target priority and where to look
2	Understands scenario context and user role
3	Defines what is a threat and adversarial intent
4	Determines which sensors to deploy and activate Assesses the utility of information
5	Designs user interface controls

Blasch and Plano [2011] provide us with a framework for user refinement (UR) with the interface actions allowing coordination between machine and user. At every point in the machine process a human component is intertwined in understanding and facilitating the interfaces required for HIL programming. This model and [22] present fusion from tracking point of view and posit that as the number of targets and speed increases, a user will get overloaded requiring assistance from the machine fusion system to make routine calculations, etc. for satisfying mission requirements. It is further argued that this provides critical user information needs for knowledge representation and cognitive reasoning in order to support effective and efficient, proactive decision making [20].

A.3.1.6 *Information Fusion Situation Assessment Reference Model, 2011.*

The Information Fusion (IF) Situation Assessment reference model in Figure A.6 provides a glimpse of interactions between the other JDL levels with level 2 Situation Assessment as the hub. A need to develop SA models and metrics for interfaces, user control actions, and methods of situational analysis is proposed in [22].

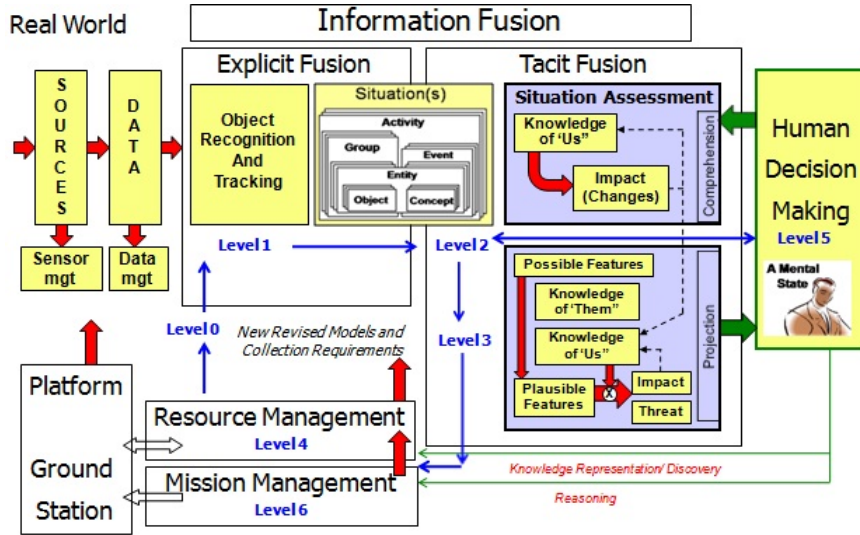


Figure A.6: Information fusion situation assessment reference model [22].

A.3.2 *Observe, Orient, Decide and Act (OODA) Loop.* The Observe, Orient, Decide and Act (OODA) loop is a common use model in the military and represents the architecture developed by Col (Ret) John Boyd, USAF. Figure A.7 shows each step as its interaction with the other steps of this loop. Col Boyd used this architecture during the Korean War while referring to the ability possessed by fighter pilots that allowed them to succeed in combat [36]. Observation refers to gathering information as in level 0, pre-processing, of the JDL model. Orientation can then be applied to JDL level 1 in trying to interpret the data and to determine how to use it. Decide then coincides with both levels 2 and 3 of the JDL model such that options are available and decisions are based on an understanding of what the possible outcomes may produce. Action is the step in which the decision is carried out.

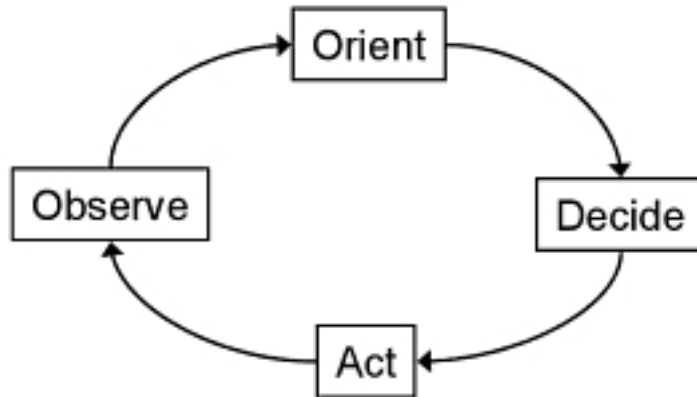


Figure A.7: Observe, Orient, Decide and Act (OODA) Loop.

Now does level 5 exist in this process? Where does user refinement exist in the OODA loop? Recall that this model was designed specifically for fighter pilots, so we can generalize that each level also coincides with user refinement (UR). The observe step is where the user provides value to the machine interface as to what observations are more important than others. The user then prioritizes important observations in the orient step (i.e., where and what to focus on). Then in the decide step, the user determines if action is required based upon the context and intent of the situation or if more observations are required, etc. This demonstrates that the fusion process, in general, provides functions and processes for dealing with information in an as needed basis as decisions are made (i.e., these fusion models are not sequential in nature). The action step of the OODA loop is not necessarily modeled according to the JDL model, but it is inferred in UR by the user providing adjustments to the utility of the situation based upon actions or what the user is provided via a user interface.

A.3.3 Data, Information, Knowledge, Wisdom (DIKW) Pyramid. The pyramid presented in Figure A.8, provides a representation of data layered from lowest to highest in understanding. The hierarchy bears some resemblance to the JDL data fusion model in the sense that both start from raw transactional data to yield knowledge at an increasing level of abstraction.

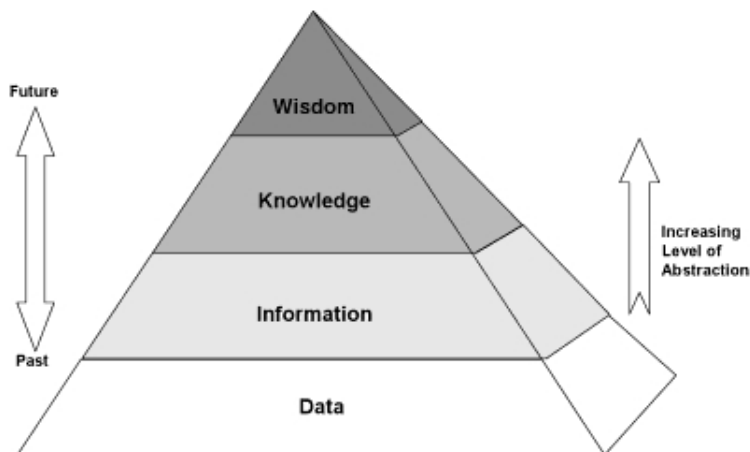


Figure A.8: Data, Information, Knowledge and Wisdom (DIKW) Pyramid.

- **Data Layer** Data is defined as the facts and statistics collected together for reference or analysis [87]. The data layer is the lowest level of the data layers and provides no utility in understanding it's purpose. For example, we have a set of numbers such as 1, 10, 100, 1000, and 10000, but we are not sure exactly what they represent. These set of numbers are just data presented to us for which we have no information, knowledge or wisdom in handling them (i.e., no utility). As we continue to climb the pryamid, we shall see an increase in understanding as Figure A.8 represents. This can be depicted as the data received from the data sources for pre-processing as in level 0 of the JDL model.
- **Information Layer** Information is defined as the facts provided or learned about something or someone [87]. The information layer is one level up from the data layer which represents some increased understanding of the data provided. For example, the number 1 corresponds to camera 1, 10 to camera 2, 100 to camera 3, 1000 to camera 4, and 10000 represents noise. This information allows us to interpret how to prioritize the data received and whether there is any utility. Note that there is still little understanding involved. This relates to levels 1 and 2 of the JDL model where features are provided by the user to determine object and situation assessment. In our example, we have four cameras (object assessment) and some noise that is of no value (situation assessment).

- **Knowledge Layer** Knowledge is defined as the facts, information, and skills acquired by a person through experience or education; the theoretical or practical understanding of a subject [87]. Since we have described the information layer as providing some utility, the knowledge layer provides a general awareness or possession of information, facts, ideas, truths, or principles [36]. Continuing with our example, we now know that we have four cameras available associated with a particular number. The user must provide intent at this point in how to use the cameras (impact assessment) and decisions on what to discard.
- **Wisdom Layer** Wisdom is defined as the quality of having experience, knowledge, and good judgment [87]. Wisdom is at the highest level of the data layers and provides us with the knowledge of what is true and right (i.e., making the right decisions and judgments in action). Although the action is not explicit in the JDL-user model, it can be inferred that based upon actions the user provides utility refinements for updating the various functions in the model.

Now that we have gone through the various layers of the pyramid. Let us take a loop at another simple example to try and make things clearer. Data - we have an orange, we do not know what to do with it or what can be done with it; Information - the orange is edible and can be processed; Knowledge - the orange is healthy to our bodies and should be eaten for sustenance; Wisdom - we first peel the orange and then eat it because of our experience in knowing that it is right to do so.

A.3.4 Rasmussen Information Processing Hierarchy. The Rasmussen information processing hierarchy provides us with three-tiers that model human information processing (Rasmussen, 1983, 1986). This tiered approach is shown in Figure A.9, and steps through the flow of information through human decision-making. Again, we can look at the HIL and the JDL-user model for interpreting this particular hierarchy.

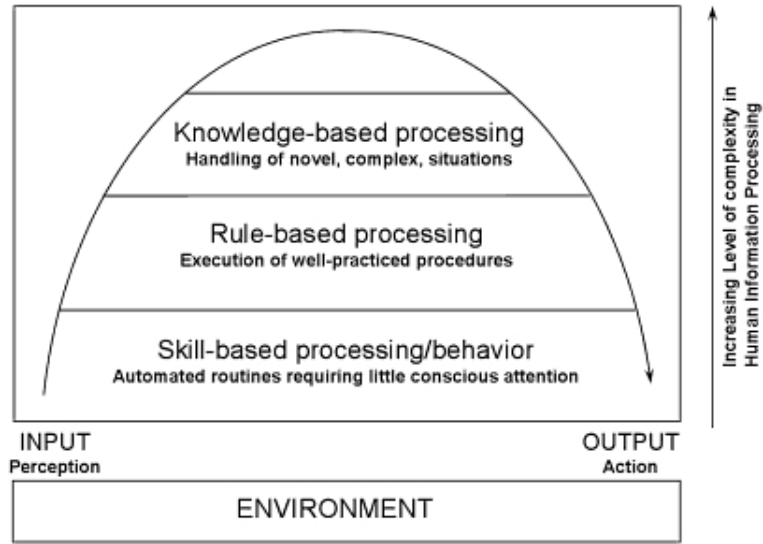


Figure A.9: Rasmussen information processing hierarchy [36].

Let us examine the three categories more closely:

- **Skill-Based Processing** The lowest level corresponds closely with the JDL level 0 of feature extraction. Example: identifying a moth in a set of images
- **Rule-Based Processing** The next level lays out execution of well-practiced procedures such as maintaining visual contact of a moth within a room as it is flying around.
- **Knowledge-Based Processing** Of course if it is too dark in the room, then utilizing night vision goggles to maintain visual contact of the moth would require novel, complex handling of the situation.

We can relate the various JDL levels in the following manner [36]:

Table A.4: Rasmussen Information Processing Hierarchy and the JDL Data Fusion Model.

Rasmussen	JDL Level
Skill/Rule-based processing	1
Rule/Knowledge-based processing	2
Rule/Knowledge-based processing	3

A.3.5 The Intelligence Cycle. Another approach to model a fusion application is to line out its cyclic character. Representatives of such an approach are the intelligence cycle [9] and the Boyd control loop [10]. The Intelligence Cycle [9] comprises the following five stages: Planning and Direction: This stage determines the intelligence requirements. Collection: Gathering of appropriate information, e. g., through sensors. Collation: Here the collected information is lined up. Evaluation: The actual fusion is done and the information gets analyzed. Dissemination: Dissemination distributes the fused intelligence [45].

A.3.6 Waterfall Model. The UK Government Techology Foresight Data Fusion Working Party [Markin et al, 1997] endorsed the Waterfall model that was proposed in 1994 [Bedworth, M.] [16]. Corresponding JDL levels are shown regarding the Waterfall model elements in Figure A.10. Feedback is not implied in this model and from observation it seems as though it is sequential in nature thus leaving room for improvement.

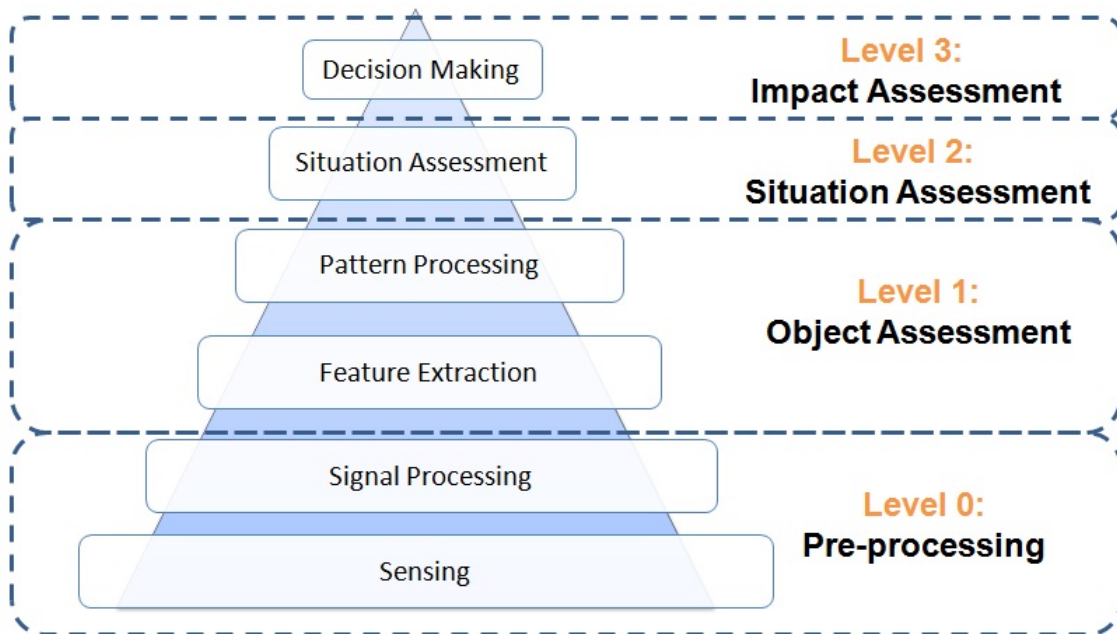


Figure A.10: Waterfall fusion process model [45].

A.3.7 Omnibus Model. The Omnibus model is an attempt to improve upon the known disadvantages of the intelligence cycle, the 1990s JDL model, the Dasarthy model and the Waterfall model [16]. Figure A.11 shows the diagram of the cyclic nature of the

intelligence cycle and the Boyd control loop, but it uses terminology more closely related to the Waterfall model [16].

Omnibus Model

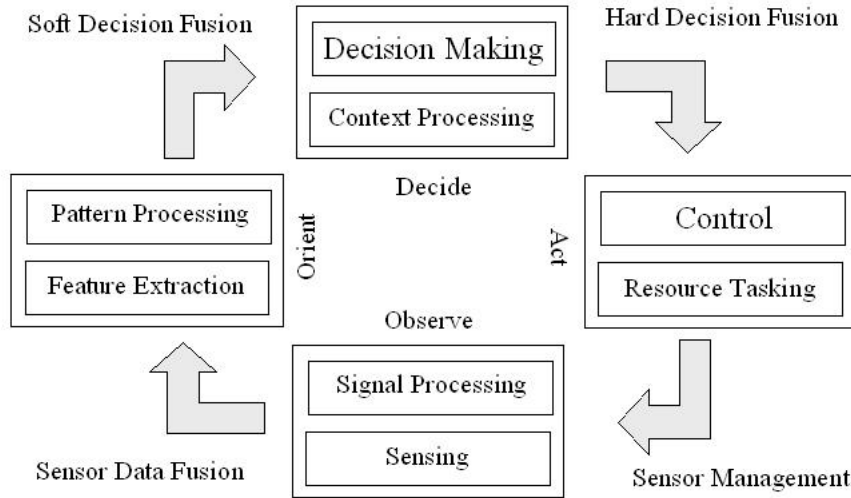


Figure A.11: Omnibus model (Bedworth, O'Brien).

It is said that disadvantages originate within the various models due to their origin (i.e., the Joint Directors of Laboratories come from military backgrounds). The Omnibus model is said to move away from a military perspective and that it is more loosely based for broader use [16]. With the updated JDL-user model from [20], we can observe that each level can be associated rather easily. Sources and level 0 of the JDL model relate to the signal processing/sensing element of the Omnibus model. Level 1 relates to the pattern processing/feature extraction element. Levels 2 and 3 relate to the decision making/context processing, and levels 4 and 5 relate to the control/resource tasking element of the Omnibus model. In Figure A.11, it can be seen that the Omnibus model is an extension of the OODA loop. Blasch and Plano refer to this model as being machine-centric related to the human reasoning strategy [19].

A.3.8 Situation Awareness Reference Model. The Situation Awareness (SAW) model looks to capitalize on the data fusion process from a human perspective. The levels of the JDL model can be shown throughout the awareness and cognitive processes of the human operator/analyst. Situation awareness has been defined as “the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their state in the future” (Endsley, 1988). Figure A.12 shows the levels as defined by Endsley as perception, comprehension, and projection.

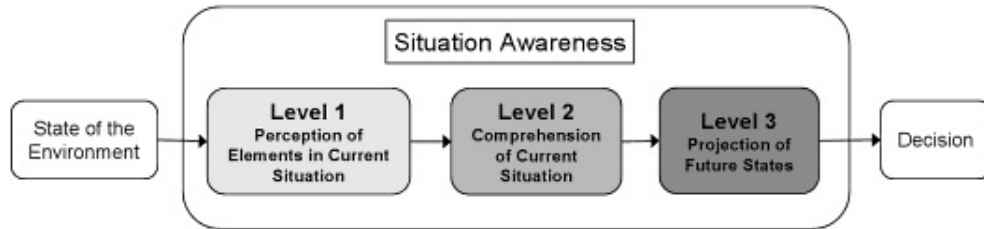


Figure A.12: Model of situation awareness [47].

Level 1: Perception of Elements in Current Situation This level takes into account the human operators ability to perceive events such as current status, attributes, and dynamics of relevant elements in the environment. This includes current and historical events. Information overload may occur in a high-tempo, dynamic environment. This can be related to the JDL model’s sources, level 0 and level 1 elements in that perception is a result of sensing data, pre-processing and determining if an object has been assessed.

Level 2: Comprehension of Current Situation At this level, the human operator takes into account all of those elements perceived in order to comprehend relationships and courses of actions based on those relationships. Comprehension may vary depending on the human operators experiences and training. This can be related to level 2 of the JDL model where the current situation is being assessed/comprehended.

Level 3: Projection of Future States The last level, interprets the perceived and comprehended relationships of elements in order to predict possible outcomes. Projection applies

experience and training of comprehended elements based on an understanding of the current dynamics. Decisions can then be made to best mitigate risk or to deal with threatening situations before or as they occur. This can be related to level 3 of the JDL model where impact assessment takes into account experience and training and determines the impact for decision-making.

Note that other levels of the JDL model are not realized in the situation awareness reference model (i.e., level 4: process refinement and level 5: user refinement).

A.3.9 Visual Data Fusion Model. The Visual Data Fusion (VDF) model is an extension of the JDL data fusion model [46, Karakowski, 1998]. It emphasizes the human as the central participant, much like that of the JDL-User model such that the human is given a visual interface for properly interpreting the problem and possible courses of action from which to select a solution. The human is involved throughout the entire process and provides relevant information requests and support to the process based on learning and experiential analysis. The purpose of the VDF model is to integrate the human participant into the JDL data fusion model.

A.3.10 Unified Data Fusion (λ JDL) Model. The Unified Data Fusion model is the integration of situation awareness (SAW) and common operating picture (COP) with respect to data fusion [46, Lambert, D. A., 2003]. Bossé, et al. [46] describe Lambert's conceptualization as the intersection between SAW and COP, essentially leading to a SAW model with levels of fusion which is described by the COP. The COP is composed of technology, psychology, and integration. Technology represents the means by which something is sensed and analyzed. Psychology represents the observations and interpretations and integration is the composition of the two. For example, (technology) an unmanned aerial vehicle (UAV) is sensed and location data is provided on a screen, (psychology) it is identified as unfriendly, (integration) location and unfriendly visualization is provided to the user. Comprehension and Projection levels of SAW would warrant further action from the

user to possibly disrupt, intercept or deceive the UAV given the ability of the fusion system with regard to technology, psychology and integration.

The mental data fusion model is similar to the SAW model and is similar to the union of the first three levels of the JDL data fusion model.

A.3.11 Perceptual Reasoning Machine (PRM). The perceptual reasoning model (PRM) [66] consists of a feedback planning/resource control system whose interacting elements are: “assess”, “anticipate”, and “predict” [21]. Figure A.13 shows the flow of queries and beliefs and hypotheses as an interface to the decision maker as well as cycle for updating those beliefs and hypotheses.

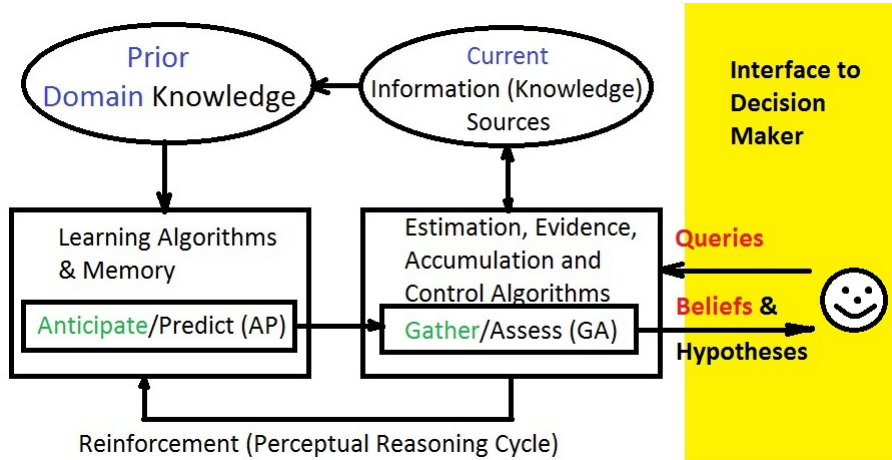


Figure A.13: Perceptual reasoning machine [21, Kadar, SPIE02].

A.3.12 Dasarathy Model. The Dasarathy Model contains an input/output interpretation of various functions. Linking these elements to the JDL model, we have sources as the data/input element which gives an output element in the Dasarathy model of ‘Data’. The functions are then linked to the arrows of the Dasarathy model such that the JDL model’s level 1: object assessment is performed for the next element of features which gives an output of ‘Features’. Level 2 of the JDL model can then be associated with the decision element which leads to an output of ‘Output/Decision’.

DDF Model

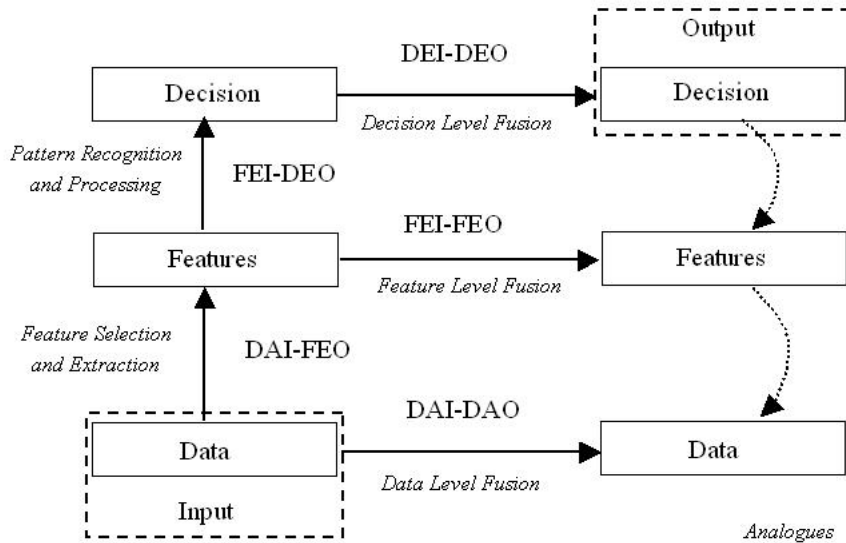


Figure A.14: Data feature decision model (Dasarathy, B., 1994).

Examining the Dasarathy model, researchers identified three main levels [16]:

1. Decisions - symbols or belief value
2. Features - or intermediate-level information
3. Data - or more specifically sensor data

Fusion can occur both within or between these levels as can be seen from our discussion above. Some of the JDL levels can be associated with those between transformations.

A.3.13 Distributed Network of Autonomous Modules [89, Henrich and Kausch, 2004].

Each module represents a separate function in the data fusion system, and it is represented by a physical, an informative, and a cognitive domain. The physical domain contains those sensors that retrieve sensory data within its environment for fusion. This domain may also contain those actuators that may modify the environment in which it resides. The informative domain processes the sensory data retrieved/shaped by the physical domain and seeks/implements decisions based upon the cognitive domain. Therefore, the cognitive

domain contains the human interface/decision-making block for interpreting the courses of actions provided by the informative domain.

A.3.14 Other Fusion Models. Other fusion models are listed in this section but are left to the reader as to whether further investigation is warranted.

A.3.14.1 Neurophysiological Information Fusion (NIF) Model [19].

A.3.14.2 Luo and Kay's Architecture (Springer).

A.3.14.3 Pau's Sensor Data Fusion Process (Springer).

A.3.14.4 Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) Architecture [43].

- Logical robot level
- Functional level
- Execution control level
- Decision level

A.3.14.5 Transformation of Requirement for the Information Process (TRIP) Model [57].

Bibliography

1. “Intl. Society of Information Fusion”. URL <http://isif.org/>.
2. “www.data-fusion.org (Information on Earth Observation Data Fusion)”. URL <http://www.data-fusion.org/>.
3. *USAF Unmanned Aircraft Systems Flight Plan 2009-2047*. Technical report, USAF, 2009.
4. Akhloufi, M. A. “Pan and tilt real-time target tracking,”. 76680K-1-10. SPIE, 2010.
5. Al Haj, M., A. Bagdanov, J. Gonzalez, and F. Roca. “Reactive Object Tracking with a Single PTZ Camera,” *Pattern Recognition (ICPR), 2010 20th Intl. Conference on*. 1690 –1693. aug. 2010.
6. Anderson, M. L. *Design and Control of Flapping Wing Micro Air Vehicles*. Ph.D. thesis, Air Force Institute of Technology, 2011.
7. Anderson, M. L. and R. G. Cobb. “Toward Flapping Wing Control of Micro Air Vehicles,” *AIAA Journal of Guidance, Control, and Dynamics*, 35: 296–308 (Jan 2012).
8. Arbelaez, P., M. Maire, C. Fowlkes, and J. Malik. *Contour Detection and Hierarchical Image Segmentation*. Technical Report UCB/EECS-2010-17, EECS Department, University of California, Berkeley, Feb 2010. URL <http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-17.html>.
9. Asanovic, K., R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick. *The Landscape of Parallel Computing Research: A View from Berkeley*. Technical report, University of California at Berkeley.
10. Azzari, P., L. Di Stefano, and A. Bevilacqua. “An effective real-time mosaicing algorithm apt to detect motion through background subtraction using a PTZ camera,” *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on*. 511 – 516. Sept. 2005.
11. Bae, J. S., S. H. Lee, Y. Kim, and Y. S. Jung. “An imaging target tracking software for a precision guided missile application,” *Information Fusion (FUSION), 2010 13th Conference on*. 1–7. July 2010.
12. Baker, S., D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. “A Database and Evaluation Methodology for Optical Flow,” *Proc. IEEE 11th Int. Conf. Computer Vision ICCV 2007*. 1–8. 2007.
13. Baker, S., D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. “A Database and Evaluation Methodology for Optical Flow,” *Int. J. Comput. Vision*, 92: 1–31 (March 2011).
14. Bauer, N., P. Pathirana, and P. Hodgson. “Robust Optical Flow with Combined Lucas-Kanade/Horn-Schunck and Automatic Neighborhood Selection,” *Proc. Int. Conf. Information and Automation ICIA 2006*. 378–383. 2006.

15. Bay, H., A. Ess, T. Tuytelaars, and L. Vangool. "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, 110: 346–359 (2008).
16. Bedworth, M. and J. O'Brien. "The Omnibus Model: A New Model of Data Fusion?," *Fusion 1999 Conference*. 1999.
17. Bevilacqua, A. and P. Azzari. "High-Quality Real Time Motion Detection Using PTZ Cameras," *Video and Signal Based Surveillance, 2006. AVSS '06. IEEE Intl. Conference on*. 23. Nov. 2006.
18. Bhaskar, H., L. Mihaylova, and A. Achim. "Automatic object detection based on adaptive background subtraction using symmetric alpha stable distribution," *Target Tracking and Data Fusion: Algorithms and Applications, 2008 IET Seminar on*. 195. April 2008.
19. Blasch, E. and S. Plano. "JDL Level 5 Fusion Model User Refinement Issues and Applications in Group Tracking," *Aerosense*. 270–279. 2002.
20. Blasch, E. and S. Plano. "DFIG Level 5 (User Refinement) issues supporting Situational Assessment Reasoning," *Fusion 05*. July 2005.
21. Blasch E., K. I. S. J. K. M. M. D. S. P. G. M. C. D. D. and E. H. Ruspini. "Issues and Challenges in Situation Assessment (Level 2 Fusion)," *Journal of Advances in Information Fusion*, 1: 122–139 (December 2006).
22. Blasch E. P., S. J. J. and G. Tadda. "Measuring the Worthiness of Situation Assessment," *IEEE National Aerospace & Electronics Conference*. July 2011.
23. Boggan, S. and D. M. Pressel. *GPUs: An Emerging Platform for General-Purpose Computation*. Technical report, U.S. Army Research Laboratory.
24. Bondzulich, B. and V. Petrovic. "mult-sensor Background Extraction and Updating for Moving Target Detection," *Fusion 2008 Conference Proceedings*. 1844–1851. International Society of Information Fusion, 2008.
25. Bouguet, J.-Y. "Pyramidal Implementation of the Lucas-Kanade Feature Tracker", 2000.
26. Bradski, G. *The OpenCV Library*. <http://opencv.org>, 2000.
27. Broaddus, C., T. Germano, N. Vandervalk, A. Divakaran, S. Wu, and H. Sawhney. "ACT-Vision: Active Collaborative Tracking for Multiple PTZ Cameras," *mult-sensor, Multisource Information Fusion: Architectures, Algorithms, and Applications*. 2009.
28. Camplani, M. and L. Salgado. "Adaptive multi-camera system for real time object detection," *Consumer Electronics (ICCE), 2011 IEEE International Conference on*. 797–798. 2011.
29. Cao, J., X. Xie, J. Liang, and D. Li. "GPU Accelerated Target Tracking Method," in *Advances in Multimedia, Software Engineering and Computing Vol.1*. (Eds.) D. Jin and S. Lin. Springer Berlin Heidelberg, 2012.
30. Chang, Y.-Z., J.-F. Hou, Y. H. Tsao, and S.-T. Lee. "Calibration of a dual-PTZ camera system for stereo vision", 2010. URL <http://dx.doi.org/10.1117/12.860331>.

31. Chen, Q., N. D. Georganas, and E. M. Petriu. "Real-time Vision-based Hand Gesture Recognition Using Haar-like Features," *Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE*. 1–6. May 2007.
32. Clady, X., F. Collange, F. Jurie, and P. Martinet. "Object tracking with a pan-tilt-zoom camera: application to car driving assistance," *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE Intl. Conference on*. 1653 – 1658 vol.2. 2001.
33. Clady, X., F. Collange, F. Jurie, and P. Martinet. "Tracking with a pan-tilt-zoom camera for an ACC system," *12th Scandinavian Conference on Image Analysis (SCIA '01)*. 561–566. 2001.
34. Comaniciu, D. and P. Meer. "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24: 603–619 (May 2002).
35. Comaniciu, D., V. Ramesh, and P. Meer. "Kernel-Based Object Tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, 25: 564–575 (May 2003).
36. Das, S. *High-Level Data Fusion*. Artech House, 2008.
37. Deluca, A. M., M. F. Reeder, and R. G. Cobb. "An Experimental Investigation into the Effect of Flap Angles for a Piezo-Driven Wing," *International Journal of Micro Air Vehicles*, 5: 55–92 (March 2013).
38. DeMenthon, D. and L. S. Davis. "Model-Based Object Pose in 25 Lines of Code," *Intl. Journal of Computer Vision*, 15: 123–141 (1995).
39. Dinh, T., Q. Yu, and G. Medioni. "Real time tracking using an active pan-tilt-zoom network camera," *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems IROS 2009*. 3786–3793. 2009.
40. Doman, D. B., M. W. Oppenheimer, and D. O. Sigthorsson. "Dynamics and Control of a Biomimetic Vehicle Using Biased Wingbeat Forcing Functions: Part II - Controller," *48th AIAA Aerospace Sciences Meeting*. January 2010.
41. Doyle, D. D., A. L. Jennings, and J. T. Black. "Optical Flow Background Subtraction for Real-Time PTZ Camera Object Tracking," *IEEE International Instrumentation and Measurement Technology Conference*. May 2013.
42. El-Bakry, H. M. "An efficient algorithm for pattern detection using combined classifiers and data fusion," *Information Fusion*, 11: 133–148 (2010).
43. Elmenreich, W. *An Introduction to Sensor Fusion*. Research Report 47/2001, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 2001.
44. Elmenreich, W. *Sensor Fusion in Time-Triggered Systems*. Ph.D. thesis, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 3/3/182-1, 1040 Vienna, Austria, 2002.
45. Elmenreich, W. "A Review on System Architectures for Sensor Fusion Applications," *International Federation for Information Processing (IFIP)*, 4761: 547–559 (2007).

46. Eloi Bosse Jean Roy and S. Wark. *Concepts, Models, and Tools for Information Fusion*. Artech House, 2007.
47. Endsley, M. R. “Toward a Theory of Situation Awareness in Dynamic Systems,” *Human Factors*, 37: 32–64 (1995).
48. Everts, I., N. Sebe, and G. Jones. “Cooperative Object Tracking with Multiple PTZ Cameras,” *Proc. 14th Int. Conf. Image Analysis and Processing ICIAP 2007*. 323–330. 2007.
49. Fukunaga, K. and L. Hostetler. “The estimation of the gradient of a density function, with applications in pattern recognition,” *Information Theory, IEEE Transactions on*, 21: 32–40 (1975).
50. Fulkerson, B., A. Vedaldi, and S. Soatto. “Class Segmentation and Object Localization with Superpixel Neighborhoods,” *Proceedings of the Intl. Conference on Computer Vision (ICCV)*. 2009.
51. Gadsden, S. A., S. R. Habibi, and T. Kirubarajan. “A novel interacting multiple model method for nonlinear target tracking,” *Information Fusion (FUSION), 2010 13th Conference on*. 1–8. July 2010.
52. Goshtasby, A. A. and S. Nikolov. “Image fusion: Advances in the state of the art,” *Information Fusion*, 8: 114–118 (2007).
53. Grinias, I. and G. Tziritas. “Robust pan, tilt and zoom estimation,” *Proc. 14th Int. Conf. Digital Signal Processing DSP 2002*. 679–682. 2002.
54. Guha, P., D. Palai, D. Goswami, and A. Mukerjee. “DynaTracker: Target tracking in active video surveillance systems,” *Advanced Robotics, 2005. ICAR '05. Proceedings., 12th Intl. Conference on*. 621–627. July 2005.
55. Hall, D. L. and J. Llinas. “An introduction to mult-sensor data fusion,” *Proceedings of the {IEEE}*, 85: 6–23 (January 1997).
56. Hall, D. L. and J. Llinas. *Handbook of mult-sensor Data Fusion*. CRC Press LLC, 2001.
57. Hall, D. L. and S. A. H. McMullen. *Mathematical Techniques in mult-sensor Data Fusion: Second Edition*. Artech House, 2004.
58. Hastie T., R. Tibshirani and J. Friedman. *The Elements of Statistical Learning; Data Mining, Inference, and Prediction* (Second Edition). Springer-Verlag, New York, New York, USA, 2009.
59. Hoedl, T., D. Brandt, U. Soergel, and M. Wiggenhagen. “Real-time orientation of a PTZ-camera based on pedestrian detection in video data of wide and complex scenes,” . 663–668. 2008.
60. Horn, B. K. P. and B. G. Schunck. “Determining Optical Flow,” *ARTIFICIAL INTELLIGENCE*, 17: 185–203 (1981).
61. Huang, G., Y. Tian, Y. Wang, Y. Yang, and X. Tai. “Self-Recalibration of PTZ Cameras,” *Machine Vision and Human-Machine Interface (MVHI), 2010 International Conference on*. 292–295. April 2010.

62. Jennings, A. and J. Black. "Texture-Based Photogrammetry Accuracy on Curved Surfaces," *AIAA Journal*, 50: 1060–1071 (May 2012).
63. Jin-Jie, H., J. Yong-Fu, C. Guang-ming, and S. Guo-Bing. "An improved approach to calculation of the optical flow field for color image sequences," *Proc. Int Strategic Technology (IFOST) Forum*. 337–342. 2010.
64. Junejo, I. and H. Foroosh. "Refining PTZ camera calibration," *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. 1–4. 2008.
65. Junejo, I. N. and H. Foroosh. "Practical PTZ camera calibration using Givens rotations," *Proc. 15th IEEE Int. Conf. Image Processing ICIP 2008*. 1936–1939. 2008.
66. Kadar, I. "Data fusion by perceptual reasoning and prediction," *Proceedings, 1987 Tri-Service Data Fusion Symposium, John Hopkins University Applied Physics Laboratory*. June 1987.
67. Kalal, Z., J. Matas, and K. Mikolajczyk. "P-N learning: Bootstrapping binary classifiers by structural constraints,". 49–56. June 2010.
68. Kanatani, K. "Camera Rotation Invariance of Image Characteristics," *Computer Vision, Graphics, and Image Processing*. 328–354. Sept. 1987.
69. Kang, S., J.-K. Paik, A. Koschan, B. R. Abidi, and M. A. Abidi. "Real-time video tracking using PTZ cameras," *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*. 103–111. April 2003.
70. Ke, Y. and R. Sukthankar. "PCA-SIFT: A more distinctive representation for local image descriptors," *Computer Vision and Pattern Recognition (CVPR)*. 8. 2004.
71. Kirchmaier, U., S. Hawe, and K. Diepold. "Dynamical information fusion of heterogeneous sensors for 3D tracking using particle swarm optimization," *Information Fusion*, 12: 275–283 (2011).
72. Kumar, P., A. Dick, and T. S. Sheng. "Real Time Target Tracking with Pan Tilt Zoom Camera," *Proc. DICTA '09. Digital Image Computing: Techniques and Applications*. 492–497. 2009.
73. Kumar, S., C. Micheloni, and C. Piciarelli. "Stereo Localization Using Dual PTZ Cameras," *Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns*. 1061–1069. Berlin, Heidelberg: Springer-Verlag, 2009.
74. Kyrkou, C. and T. Theodorides. "A Flexible Parallel Hardware Architecture for AdaBoost-Based Real-Time Object Detection," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 19: 1034–1047 (June 2011).
75. Lee, J.-S. and J.-H. Han. "Experimental study on the flight dynamics of a bio-inspired ornithopter: free flight testing and wind tunnel testing," *Smart Materials and Structures*, 21: 094023 ().
76. Li, Y., Q., Xie and G. Yu. "Real-time Tracking System Combining Mixture of Gaussian Model and Continuously Adaptive Mean Shift (CAMShift) Algorithm," *Intl. Conference on Image Processing and Pattern Recognition in Industrial Engineering*. 2010.

77. Lindholm, G. and R. Cobb. "Closed-Loop Control of a Constrained, Resonant-Flapping Micro Air Vehicle," *AIAA Guidance, Navigation, and Control Conference*. August 2012.
78. Liu, C. *Beyond pixels: exploring new representations and applications for motion analysis*. Ph.D. thesis, Massachusetts Institute of Technology, 2009.
79. Llinas J., B. C. L. R. G. S. A. W. E. and F. E. White. "Revisiting the {JDL} data fusion model II," *NSSDF Conference Proceedings, JHAPL*. 1218–1230. Proceedings of the Seventh International Conference on Information Fusion, {FUSION} 2004, 2004.
80. Lowe, D. G. "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, 60: 91–110 (2004).
81. Luo, R. C., Y. C. Chou, and O. Chen. "mult-sensor Fusion and Integration: Algorithms, Applications, and Future Research Directions," *Proc. Int. Conf. Mechatronics and Automation ICMA 2007*. 1986–1991. 2007.
82. Luo, R. C. and M. G. Kay. "Introduction," in *mult-sensor Integration and Fusion for Intelligent Machines and Systems*. (Eds.) R. C. Luo and N. C. S. U. Michael G. Kay. Ablex Publishing Corporation, 1995.
83. Ma, J., C. Han, and Y. Yang. "Visual tracking based on adaptive multi-cue integration," *Information Fusion, 2009. FUSION '09. 12th International Conference on*. 1737–1742. July 2009.
84. Ma, Y., S. Soatto, J. Kosecka, and S. S. Sastry. *An Invitation to 3D Vision*. Springer, Nov. 2003.
85. Magree, D. *A Photogrammetry-based Hybrid System for Dynamic Tracking and Measurement*. Master's thesis, Air Force Institute of Technology, 2010.
86. Manap, N. A., G. Di Caterina, J. Soraghan, V. Sidharth, and H. Yao. "Smart surveillance system based on stereo matching algorithms with IP and PTZ cameras," *Proc. 3DTV-Conf.: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*. 1–4. 2010.
87. McKean, E. *New Oxford American Dictionary, 2d Edition*. Oxford University Press, 2005.
88. Mikolajczyk, K. and C. Schmid. "A Performance Evaluation of Local Descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*. October 2005.
89. Mitchell, H. B. *Multi-Sensor Data Fusion: An Introduction*. Springer Science + Business Media, 2007.
90. Mohammadi, G., F. Dufaux, T. H. Minh, and T. Ebrahimi. "Multi-view video segmentation and tracking for video surveillance," *Mobile Multimedia/Image Processing, Security, and Applications 2009*. 735104. SPIE, 2009.
91. Muja, M. and D. G. Lowe. "Fast approximate nearest neighbors with automatic algorithm configuration," *Science*, 340: 331–340 (2009).
92. Murray, D. and A. Basu. "Motion Tracking with an Active Camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 449–459. May 1994.

93. N., D. and B. Triggs. “Histograms of Oriented Gradients for Human Detection,” *Computer Vision and Pattern Recognition (CVPR)*. 8. 2005.
94. Nelson, J. K. and H. Roufarshbaf. “A Tree Search Approach to Target Tracking in Clutter,” *Information Fusion (FUSION), 2009 12th Intl. Conference on*. 834–841. July 2009.
95. Pei, L., P. Zhang, and R. Wang. “Multifeature fusion tracking in a particle filter framework,” *MIPPR 2009: Automatic Target Recognition and Image Analysis*. 74954T. SPIE, 2009.
96. Piccardi, M. “Background subtraction techniques: a review,” *Systems, Man and Cybernetics, 2004 IEEE Intl. Conference on*. 3099 – 3104 vol.4. 2004.
97. Pribula, O. and J. Fischer. “Real time precise position measurement based on low-cost CMOS image sensor: DSP implementation and sub-pixel measurement precision verification,” *Proc. 18th Int Systems, Signals and Image Processing (IWSSIP) Conf.* 1–4. 2011.
98. Qigui, Z. and L. Bo. “Search on automatic target tracking based on PTZ system,” *Image Analysis and Signal Processing (IASP), 2011 Intl. Conference on*. 192–195. oct. 2011.
99. Ratti, J. and G. Vachtsevanos. “A Biologically-Inspired Micro Aerial Vehicle,” *Journal of Intelligent and Robotic Systems*, 60 (2010).
100. Riha, L., J. Fischer, R. Smid, and A. Docekal. “New interpolation methods for image-based sub-pixel displacement measurement based on correlation,” *Proc. IEEE Instrumentation and Measurement Technology IMTC 2007*. 1–5. 2007.
101. Martinez-del Rincon, J., E. Herrero-Jaraba, J. R. Gomez, C. Orrite-Urunuela, C. Medrano, and M. A. Montanes-Laborda. “multi-camera sport player tracking with Bayesian estimation of measurements,” *Optical Engineering*, 48: 47201 (2009).
102. Russell, S. J. and Norvig, P. *Artificial Intelligence: A Modern Approach, 3d Edition*. Prentice Hall, Upper Saddle River, NJ, USA, 2010.
103. Rusu, R. B. and S. Cousins. “3D is here: Point Cloud Library (PCL),” *IEEE International Conference on Robotics and Automation (ICRA)*. May 9-13 2011.
104. Sahai, R., K. Galloway, and R. Wood. “Elastic Element Integration for Improved Flapping-Wing Micro Air Vehicle Performance,” *Robotics, IEEE Trans. on*, 29: 32–41 (2013).
105. Scalzo, M., G. Horvath, E. Jones, A. Bubalo, M. Alford, R. Niu, and P. K. Varshney. “Adaptive filtering for single target tracking,” *Signal Processing, Sensor Fusion, and Target Recognition XVIII*. 73360C. SPIE, 2009.
106. Shi, J. and C. Tomasi. “Good Features to Track,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 593–600. June 1994.
107. Shih, F. Y. *Image Processing and Pattern Recognition: Fundamentals and Techniques*. John Wiley & Sons, Inc., 2010.

108. Shirmohammadi, B. and C. J. Taylor. "Distributed Target Tracking using Self Localizing Smart Camera Networks," *ACM Intl. Conf. on Distributed Smart Cameras*. 16–24. Sept. 2010.
109. Smith, M., A. Boxerbaum, G. Peterson, and R. Quinn. "Electronic image stabilization using optical flow with inertial fusion," *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ Intl. Conf. on*. 1146 –1153. oct. 2010.
110. Srkk, S., A. Vehtari, and J. Lampinen. "Rao-Blackwellized particle filter for multiple target tracking," *Information Fusion*, 8: 2–15 (2007).
111. Starzacher, A. and B. Rinner. "Embedded Realtime Feature Fusion based on ANN, SVM and NBC," *Information Fusion (FUSION), 2009 12th Intl. Conference on*. 482–489. July 2009.
112. Steinberg, A. and C. L. Bowman. "Rethinking the JDL Data Fusion Levels," *NSSDF Conference Proceedings*. June 2004.
113. Steinberg, A. N., C. L. Bowman, and F. E. White. "Revisions to the JDL data fusion model," *Sensor Fusion: Architectures, Algorithms, and Applications III*. 430–441. SPIE, 1999.
114. Suhr, J. K., H. G. Jung, G. Li, S.-I. Noh, and J. Kim. "Background Compensation for Pan-Tilt-Zoom Cameras Using 1-D Feature Matching and Outlier Rejection," 21: 371–377 (2011).
115. Sun, D., S. Roth, and M. J. Black. "Secrets of optical flow estimation and their principles," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*. 2432–2439. 2010.
116. Szeliski, R. *Computer Vision: Algorithms and Applications*. Springer, 2010.
117. Tamersoy, B. and J. Aggarwal. "Exploiting Geometric Restrictions in a PTZ Camera for Finding Point-correspondences Between Configurations," *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*. 488. 29 2010-sept. 1 2010.
118. Tola, L. V., E. and P. Fua. "DAISY: An efficient dense descriptor applied to wide baseline stereo," . 31. 2010.
119. Tordoff, B. and D. Murray. "Reactive control of zoom while fixating using perspective and affine cameras," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26: 98 –112 (jan. 2004).
120. Tuzel, P. F., O. and P. Meer. "Learning on Lie Groups for Invariant Detection and Tracking," *Computer Vision and Pattern Recognition (CVPR)*. 10. 2006.
121. Tuzel, P. F., O. and P. Meer. "Region covariance: A fast descriptor for detection and classification," *European Conference on Computer Vision (ECCV)*. 1–12. 2006.
122. Tyagi, A., J. W. Davis, and G. Potamianos. "Steepest Descent For Efficient Covariance Tracking," *2008 IEEE Workshop on Motion and video Computing*. 1–6. IEEE, January 2008.

123. Vijverberg, J. A., M. J. H. Loomans, C. J. Koeleman, and P. H. N. de With. “Two novel motion-based algorithms for surveillance video analysis on embedded platforms,” *Real-Time Image and Video Processing 2010*. 77240I. 2010.
124. Viola, P. and M. Jones. “Robust real-time object detection,” *Intl. Journal of Computer Vision* (2002).
125. Wan, D. and J. Zhou. “Multiresolution and Wide-Scope Depth Estimation Using a Dual-PTZ-Camera System,” *Image Processing, IEEE Transactions on*, 18: 677–682 (march 2009).
126. Weisstein, E. W. “Line-Line Intersection”. From MathWorld—A Wolfram Web Resource. URL <http://mathworld.wolfram.com/Line-LineIntersection.html>. (retrieved: 7/27/2013).
127. White, F. E. *Data Fusion Lexicon*. Technical Report 20100621258, Data Fusion Panel, Joint Directors of Laboratories Technical Panel for C3, Naval Ocean Systems Center, San Diego, CA, October 1991.
128. Wijnhoven, R. and P. H. N. de With. “Fast Training of Object Detection using Stochastic Gradient Descent,” *Proc. IEEE Intl. Conference on Pattern Recognition (ICPR)*. 424–427. 2010.
129. Wu, Z. and R. J. Radke. “Keeping a Pan-Tilt-Zoom Camera Calibrated,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35: 1994–2007 (2013).
130. Xue, K., G. Ogunmakin, Y. Liu, P. Vela, and Y. Wang. “PTZ camera-based adaptive panoramic and multi-layered background model,” *Image Processing (ICIP), 2011 18th IEEE International Conference on*. 2949–2952. sept. 2011.
131. Yang, C.-S., R.-H. Chen, C.-Y. Lee, and S.-J. Lin. “PTZ camera based position tracking in IP-surveillance system,” *Sensing Technology, 2008. ICST 2008. 3rd Intl. Conference on*. 142–146. 30 2008-dec. 3 2008.
132. Yilmaz, A. “Kernel-based object tracking using asymmetric kernels with adaptive scale and orientation selection,” *Mach. Vision Appl.*, 22: 255–268 (March 2011).
133. Yilmaz, A., O. Javed, and M. Shah. “Object tracking: A survey,” *ACM Comput. Surv.*, 38 (Dec 2006).
134. You, L., S. Li, and W. Jia. “Automatic Weak Calibration of Master-Slave Surveillance System Based on Mosaic Image,” *Pattern Recognition (ICPR), 2010 20th International Conference on*. 1824–1827. aug. 2010.
135. Yousf, W., O. Elmowafy, and I. Abdl-Dayem. “C18. Modified CAMShift algorithm for adaptive window tracking,” *Radio Science Conference (NRSC), 2012 29th National*. 301–308. april 2012.
136. Zhang, J., Y. Wang, J. Chen, and K. Xue. “A framework of surveillance system using a PTZ camera,” *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*. 658–662. 2010.
137. Zhang, L., K. Xu, S. Yu, R. Fu, and Y. Xu. “An effective approach for active tracking with a PTZ camera,” *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*. 1768–1773. dec. 2010.

138. Zhang, H., J. Hong, W. Lin, and L. Li. “Design of Tracking System Based on Mean-shift and Kalman Filter,” *MIPPR 2009: Automatic Target Recognition and Image Analysis*. 2009.
139. Zivkovic, Z. “Improved adaptive Gaussian mixture model for background subtraction,” *Proceedings of the 17th Intl. Conference on Pattern Recognition 2004 ICPR 2004*, 2: 28–31 (2004).
140. Zivkovic, Z., A. T. Cemgil, and B. Krse. “Approximate Bayesian methods for kernel-based object tracking,” *Computer Vision and Image Understanding*, 113: 743 – 749 (2009).

REPORT DOCUMENTATION PAGE

*Form Approved
OMB No. 0704-0188*

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to the Department of Defense, Executive Service Directorate (0704-0188). Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE (DD-MM-YYYY) 18-10-2013		2. REPORT TYPE Dissertation		3. DATES COVERED (From - To) 08-2010 - 18-10-2013	
4. TITLE AND SUBTITLE Real-Time, Multiple Pan/Tilt/Zoom Computer Vision Tracking and 3D Positioning System for Unmann Aerial System Metrology				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Doyle, Daniel D., Lt Col, USAF				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way Wright-Patterson AFB OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT-ENY-DS-13-D-08	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research 875 North Randolph Street, Suite 325, Room 3112, Arlington, VA., 22203-1768 Email: info@afosr.af.mil				10. SPONSOR/MONITOR'S ACRONYM(S) AFOSR	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT The study of structural characteristics of Unmanned Aerial Systems (UASs) continues to be an important field of research for developing state of the art nano/micro systems. Development of a metrology system using computer vision (CV) tracking and 3D point extraction would provide an avenue for making these theoretical developments. This work provides a portable, scalable system capable of real-time tracking, zooming, and 3D position estimation of a UAS using multiple cameras. Current state-of-the-art photogrammetry systems use retro-reflective markers or single point lasers to obtain object poses and/or positions over time. Using a CV pan/tilt/zoom (PTZ) system has the potential to circumvent their limitations. The system developed in this paper exploits parallel-processing and the GPU for CV-tracking, using optical flow and known camera motion, in order to capture a moving object using two PTU cameras. The parallel-processing technique developed in this work is versatile, allowing the ability to test other CV methods with a PTZ system using known camera motion. Utilizing known camera poses, the object's 3D position is estimated and focal lengths are estimated for filling the image to a desired amount. This system is tested against truth data obtained using an industrial system.					
15. SUBJECT TERMS object tracking, pan/tilt/zoom, optical flow, real-time, computer vision, photogrammetry					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 178	19a. NAME OF RESPONSIBLE PERSON Dr. Jonathan T. Black, AFIT/ENY
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code) (937) 255-3636 x4578 jonathan.black@afit.edu

Reset