

ABSTRACT

Title of Dissertation: **DEVELOPMENT AND VALIDATION OF
FEATURE-BASED VISION ALGORITHMS
FOR AUTONOMOUS SHIP-DECK LANDING**

Victoria Britcher
Doctor of Philosophy, 2025

Dissertation Directed by: **Professor Anubhav Datta**
Department of Aerospace Engineering

This dissertation investigates feature-based computer vision algorithms for ship-deck landing. These algorithms incorporate 2D, 3D, and stereo computer vision techniques. The vision algorithms were validated using camera and computer hardware integrated into a quadrotor aircraft. Each of the evaluated vision algorithms is capable of estimating ship-deck pose or ship motion in real-time using only on-board vision hardware.

First, a previously developed 2D feature-based algorithm was refined and experimentally characterized. The algorithm utilizes feature matching to estimate a ship-deck pose using a stored 2D reference image. Performance was compared for a realistic helipad reference image and a detailed, customized image. Experimental validation was performed for Sea-state 6 ship-deck motion simulated using a Stewart platform. Additional experiments were performed for visually-degraded conditions, including occlusion, glare, and low illumination, and free flight. The results indicated that while the algorithm can utilize realistic landing pad designs, a feature-rich, custom

landing pad image is needed for good performance. With this customized reference image, the algorithm was found to accurately estimate the ship-deck pose across all test scenarios.

A 3D feature-based vision algorithm was developed. The algorithm is based on the structure-from-motion technique used in 3D computer vision, and is capable of estimating the pose of a known 3D object. A simplified 3D object was created to represent a ship-deck for experimental validation. The accuracy and computational speed of the algorithm were evaluated for different feature detectors. Experimental tests were carried out for Sea-state 6 motion. The results indicated good pose estimation performance, with comparable accuracy to 2D vision methods. Additional tests validated performance for visually-degraded conditions, specifically glare, occlusion, and low illumination, as well as free hover. The algorithm was found to be robust to occlusion and low illumination, but performance was reduced in severe glare.

A feature-based stereo vision algorithm was also developed. The algorithm can estimating the motion of arbitrary ship structures without prior knowledge of their visual appearance or geometry. Experimental results showed that the stereo vision algorithm is capable of accurately estimating the motion of a simulated ship. As the algorithm estimates motion between frames, an initial pose is necessary, and pose estimates will drift over time. A Kalman filter was used to fuse results from the stereo vision algorithm with the previous 2D feature-based algorithm, which provides full estimates of ship pose and does not suffer from drift. This fused algorithm was validated using a realistic simulated ship incorporating 3D structures and 2D flight deck markings representative of a DDG-51 type ship. The fused algorithm was evaluated for Sea-state 6 motion, large-angle motion, and free flight. Results showed very accurate pose estimation results for Sea-state 6 motion, but accuracy was reduced for the large-angle case. In the free-flight test, the fused algorithm relied heavily on the stereo vision algorithm due to failure of the

2D algorithm to detect the flight deck, but still provided reasonable pose estimates.

DEVELOPMENT AND VALIDATION OF FEATURE-BASED VISION
ALGORITHMS FOR AUTONOMOUS SHIP-DECK LANDING

by

Victoria Britcher

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2025

Advisory Committee:

Professor Anubhav Datta, Chair/Advisor
Professor Inderjit Chopra
Professor Robert Sanner
Professor James Baeder
Professor Maria Cameron

© Copyright by
Victoria Britcher
2025

Acknowledgments

I would like to acknowledge my advisor Dr. Anubhav Datta for all of his help and guidance throughout this research. Additionally, I am thankful to my committee members for their valuable advice. I would also like to thank my family, friends, and fellow students for their support.

This work was supported under the NAVAIR Grant N0042121S0001 with technical monitoring from James Bumbaugh, Daniel Shafer, and Peter Arslanian, and partly under the Army/Navy/NASA Vertical Lift Research Center of Excellence (VLRCE) number W911W61120012 with technical monitoring from Dr. Mahendra Bhagwat. I would like to express my gratitude for this support.

Table of Contents

Acknowledgements	ii
Table of Contents	iii
List of Tables	vi
List of Figures	vii
List of Abbreviations	xii
Chapter 1: Introduction	1
1.1 Ship-Deck Landing	1
1.1.1 Characterization of ship-deck motion	2
1.1.2 State-of-the-art piloted landing procedure	4
1.1.3 Typical operating limits	5
1.1.4 Modern shipboard landing systems	5
1.2 Motivation	8
1.3 Computer Vision	10
1.3.1 Computer vision in aviation	10
1.4 Vision-Based Landing	11
1.4.1 Vision-based ship landing	11
1.5 Objective of this Research	15
1.6 Contribution and Technical Approach	16
1.7 Organization of the Dissertation	17
Chapter 2: 2D Feature-Based Algorithm	19
2.1 Fiducial-Based Algorithm	19
2.2 Quadrotor Development	20
2.2.1 Flight computer	21
2.2.2 Structure and specifications	21
2.2.3 Propulsion and power	24
2.2.4 Avionics and control	24
2.2.5 Vision hardware	25
2.3 Vision Algorithm	25
2.3.1 Feature extraction and matching	26
2.3.2 SIFT algorithm	27
2.3.3 Camera pose estimation	28

2.3.4	Ship-deck detection algorithm	30
2.3.5	Algorithm implementation	33
2.4	Algorithm Validation	35
2.5	Initial Algorithm Validation and Refinement	36
2.5.1	Time delays and update frequency	37
2.5.2	Baseline conditions	38
2.5.3	Reference image selection	38
2.5.4	Feature match and inlier thresholds	42
2.5.5	Kalman filters	43
2.6	Hand-Held Tests	45
2.6.1	Tracking ship-deck motion	46
2.6.2	Visually degraded conditions	52
2.6.3	Increasing camera distance	62
2.6.4	Feature match and inlier summary	65
2.7	Free-Flight Hover Tests	66
2.7.1	Baseline conditions	66
2.7.2	Visually degraded conditions	69
2.7.3	Ship-deck motion	76
2.8	Performance Comparison with Fiducial-Based Approaches	78
2.9	Summary	79
Chapter 3:	3D Feature-Based Algorithm	81
3.1	3D Vision Algorithm	81
3.1.1	Feature extraction and matching	82
3.1.2	Feature detectors	82
3.1.3	3D triangulation	83
3.1.4	Ship-deck detection algorithm	87
3.1.5	Initial algorithm validation	92
3.2	Vision System Integration	94
3.2.1	Code integration	94
3.3	Algorithm Validation with Simulated Ship-Deck	95
3.3.1	Ship-deck	95
3.3.2	Algorithm testing	96
3.4	Comparison of Feature Detectors	97
3.4.1	Feature extraction results	98
3.4.2	Algorithm performance comparison	98
3.5	Inlier Thresholding	103
3.6	Stationary Camera and Landing Pad	103
3.7	Ship-Deck Motion	106
3.8	Visually Degraded Conditions	113
3.8.1	Feature match and inlier counts	121
3.9	Free-Flight Test	122
3.10	Performance Comparison with 2D Vision Methods	122
3.11	Increasing Algorithm Speed	125
3.11.1	Filtering visible features	126

3.11.2	Process for filtering visible features	127
3.11.3	Performance comparison	129
3.12	Summary	130
Chapter 4:	Stereo Vision Algorithm	132
4.1	Motivation for New Stereo Vision Approach	132
4.2	Hardware and Test Setup	133
4.2.1	Quadrotor details	133
4.2.2	Cameras	134
4.2.3	Flight computer	135
4.3	Feature-Based Stereo Vision Algorithm	136
4.3.1	Stereo vision	136
4.3.2	Feature matching	138
4.3.3	Estimating rigid body transformations	139
4.3.4	Algorithm overview	142
4.4	Ship-Deck Pose Estimation	145
4.4.1	Kalman filter for fusing vision algorithm results	145
4.5	Full Algorithm Implementation and Architecture	149
4.6	Experimental Validation	150
4.7	Initial Validation and Refinement	152
4.7.1	Baseline performance test	152
4.7.2	Computational speed	159
4.7.3	Feature match, inlier, and error thresholds	160
4.7.4	Isolating moving ship structures from background	163
4.8	Ship-Deck Motion	166
4.8.1	Ship motion estimation with stereo algorithm only	171
4.9	Free-Flight Test	176
4.10	Summary	179
Chapter 5:	Autonomous Landing Algorithm	181
5.1	Autonomous Ship-Deck Landing Algorithm	181
5.1.1	Implementation and architecture	181
5.2	Summary	187
Chapter 6:	Summary and Conclusions	188
6.1	Summary	188
6.2	Conclusions	189
6.3	Recommendations for Future Work	192
6.3.1	Autonomous landing	192
6.3.2	Additional vision algorithm development	195
	Bibliography	197

List of Tables

1.1	Sea-states defined by Douglas sea scale.	3
2.1	Quadrotor characteristics.	22
2.2	RMSE for different reference images.	41
2.3	RMSE for ship-deck motion.	52
2.4	RMSE for visually degraded test cases.	53
2.5	Summary of mean feature match and inlier counts for different test cases.	65
2.6	RMSE for free flight in visually degraded conditions.	69
3.1	Mean computational time for different feature detectors and image resolutions. . .	102
3.2	Mean pitch estimation error for different feature detectors and image resolutions. .	102
3.3	Mean position estimation error for different feature detectors and image resolutions. .	102
3.4	RMSE for ship-deck motion cases.	113
3.5	RMSE for visually degraded conditions.	121
3.6	Summary of mean feature matches and inliers for each test case.	121
3.7	Mean computational time for visible feature filtering with different rotation thresholds. .	129
3.8	RMSE for visible feature filtering with different rotation thresholds.	130
4.1	RMSE for the ideal baseline test case.	156
4.2	Update speed and power consumption for the two individual vision algorithms. . .	160
4.3	Update speed and power consumption for the two vision algorithms running in parallel.	160
4.4	RMSE with and without thresholding applied.	162
4.5	Mean update frequencies for different settings.	165
4.6	RMSE for ship-deck motion cases.	170
4.7	RMSE of estimated rigid body motion from stereo vision algorithm.	176
4.8	RMSE for simple free flight test.	179

List of Figures

1.1	Illustration of ship motions along reference axes.	2
1.2	Example time history showing Sea-state 5 roll and pitch motions.	3
1.3	Fourier transform of roll acceleration. Colors represent different time histories. . .	4
1.4	On May 25, 1967, Sea King helicopter CH-SS2 is hauled down by a Bear Trap to a landing on the deck of Her Majesty's Canadian Ship Assiniboine. (Courtesy Department of National Defence / PCN-67-415)	6
1.5	Shipboard landing with ASIST system [1]	7
1.6	UCARS system components [2]	8
1.7	Airbus DeckFinder system [3]	8
1.8	Examples of vision-based landing techniques.	12
1.9	Various techniques in prior work for visual detection of a moving ship-deck. . . .	13
1.10	Tracking moving platform using fiducial marker [4].	14
1.11	Detecting landing pad using feature matching [5].	14
2.1	Visually degraded conditions resulting in detection failure of an ArUco marker [5].	20
2.2	CAD diagram of new quadrotor showing major components and axes system. Note that the z-axis is positive down.	21
2.3	Photo of the new larger quadrotor.	23
2.4	Estimating camera pose using 3D-2D point correspondences.	29
2.5	Flowchart illustrating the steps in the vision algorithm.	34
2.6	Vicon system and netted enclosure.	36
2.7	Example of vision algorithm result for large roll motions with uncorrected 400 millisecond time delay.	37
2.8	Estimation of aircraft attitude angles relative to a stationary landing marker from on-board camera.	39
2.9	Estimation of aircraft position relative to a stationary landing marker from on-board camera.	40
2.10	Reference images evaluated using vision algorithm.	40
2.11	Estimation of aircraft attitude angles relative to a stationary landing marker from on-board camera. Results are compared with and without the application of feature match and inlier thresholding.	43
2.12	Estimation of aircraft position relative to a stationary landing marker from on-board camera. Results are compared with and without the application of feature match and inlier thresholding.	44

2.13	Small Stewart platform with attached 24×17 cm landing pad image.	47
2.14	Estimation of aircraft attitude angles relative to a moving Stewart platform from on-board camera, with aircraft held stationary.	48
2.15	Estimation of aircraft position relative to a moving Stewart platform from on-board camera, with aircraft held stationary.	49
2.16	Estimation of aircraft attitude angles relative to a moving Stewart platform from on-board camera, with aircraft also moving.	50
2.17	Estimation of aircraft position relative to a moving Stewart platform from on-board camera, with aircraft also moving.	51
2.18	Estimation of aircraft attitude angles relative to a landing pad undergoing ship-deck motion under low (10 lux) illumination.	54
2.19	Estimation of aircraft attitude angles relative to a landing pad undergoing ship-deck motion under low (10 lux) illumination.	55
2.20	Landing pad image with significant glare created by a 900 lumen light; change in glare pattern due to platform motion during a typical test is illustrated.	55
2.21	Estimation of aircraft attitude angles relative to a landing pad undergoing ship-deck motion from on-board camera, with a 900 lumen light used to simulate glare.	56
2.22	Estimation of aircraft position relative to a landing pad undergoing ship-deck motion from on-board camera, with a 900 lumen light used to simulate glare.	57
2.23	Occluded landing pad images used to evaluate algorithm performance.	58
2.24	Estimation of aircraft attitude angles relative to an occluded landing pad undergoing ship-deck motion. The occlusions uniformly cover 57% of the marker patterns on the landing pad image.	58
2.25	Estimation of aircraft position relative to an occluded landing pad undergoing ship-deck motion. The occlusions uniformly cover 57% of the marker patterns on the landing pad image.	59
2.26	Estimation of aircraft attitude angles relative to an occluded landing pad undergoing ship-deck motion. Irregular occlusions partially or fully cover all marker patterns on the landing pad image.	60
2.27	Estimation of aircraft position relative to an occluded landing pad undergoing ship-deck motion. Irregular occlusions partially or fully cover all marker patterns on the landing pad image.	61
2.28	Estimation of aircraft attitude angles relative to a landing pad undergoing ship-deck motion, with Z-axis (vertical) distance increased from previous tests.	63
2.29	Estimation of aircraft position relative to a landing pad undergoing ship-deck motion, with Z-axis (vertical) distance increased from previous tests.	64
2.30	Estimation of hovering aircraft attitude angles relative to a stationary landing marker in indoor illumination (300 lux).	67
2.31	Estimation of hovering aircraft position relative to a stationary landing marker in indoor illumination (300 lux).	68
2.32	Estimation of hovering aircraft attitude angles relative to a stationary landing marker in low illumination (10 lux).	70
2.33	Estimation of hovering aircraft position relative to a stationary landing marker in low illumination (10 lux).	71

2.34	Estimation of hovering aircraft attitude angles relative to a stationary, 57% occluded landing marker in indoor illumination (300 lux).	72
2.35	Estimation of hovering aircraft position relative to a stationary, 57% occluded landing marker in indoor illumination (300 lux).	73
2.36	Estimation of hovering aircraft attitude angles relative to a stationary, 57% occluded landing marker in low illumination (10 lux).	74
2.37	Estimation of hovering aircraft position relative to a stationary, 57% occluded landing marker in low illumination (10 lux).	75
2.38	Estimation of hovering aircraft attitude angles relative to a landing pad undergoing ship-deck motion.	76
2.39	Estimation of hovering aircraft position relative to a landing pad undergoing ship-deck motion.	77
3.1	During ship landing, the pilot visually tracks the moving 3D ship structures. . . .	82
3.2	Triangulation of a 3D point from two images.	86
3.3	Images illustrating the steps in the algorithm.	88
3.4	Estimated object pose compared to ground truth across a sequence of new images. . . .	93
3.5	Example photographs of the ship-deck used for feature and 3D point extraction, collected at each of the three camera positions.	96
3.6	Example of vision algorithm result for roll motion with uncorrected 500 ms delay. . . .	97
3.7	Feature extraction results with different feature detectors.	98
3.8	Setup used for collecting images of the ship deck.	99
3.9	Pose estimation error at varying distance from the object for 50% resolution images. . . .	101
3.10	Estimation of relative attitude angles. Raw algorithm output is plotted alongside results with a minimum inlier threshold applied.	104
3.11	Estimation of relative position. Raw algorithm output is plotted alongside results with a minimum inlier threshold applied.	105
3.12	Estimation of relative attitude angles, with the quadrotor held stationary above the ship-deck.	106
3.13	Estimation of relative position, with the quadrotor held stationary above the ship-deck.	107
3.14	Stewart platform with 3D object attached. The Stewart platform is used to simulate ship-deck motion.	108
3.15	Estimation of relative attitude angles, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform.	109
3.16	Estimation of relative position, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform.	110
3.17	Estimation of relative attitude angles, with the quadrotor moved in attitude and position above the ship-deck attached to moving Stewart platform.	111
3.18	Estimation of relative position, with the quadrotor moved in attitude and position above the ship-deck attached to moving Stewart platform.	112
3.19	The 3D object simulating the ship-deck with added occlusions.	114
3.20	Estimation of relative attitude angles of a partially occluded ship-deck, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform. . . .	114

3.21	Estimation of relative position of a partially occluded ship-deck, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform.	115
3.22	Estimation of relative attitude angles of a ship-deck in low illumination, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform.	116
3.23	Estimation of relative position of a ship-deck in low illumination, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform.	117
3.24	The 3D ship-deck with glare and shadows produced by a 900 lumen flashlight.	118
3.25	Estimation of relative attitude angles of a ship-deck with glare, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform.	119
3.26	Estimation of relative position of a ship-deck with glare, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform.	120
3.27	Estimation of relative attitude angles, with the quadrotor hovering above the ship-deck.	123
3.28	Estimation of relative position, with the quadrotor hovering above the ship-deck.	124
3.29	Images illustrating the steps in filtering visible features.	127
4.1	The quadrotor UAV incorporating on-board computer and vision hardware.	134
4.2	Triangulation of a point in 3D space using a stereo camera with two parallel image sensors.	137
4.3	Illustration of steps in the stereo vision algorithm.	143
4.4	The reference image used for the 2D feature-based algorithm. The image is similar to the flight deck markings on a DDG-51 ship.	151
4.5	The simulated ship used to experimentally validate the new vision algorithm. The simulated ship incorporates the 2D reference image as well as 3D ship structures.	153
4.6	Estimation of aircraft attitude angles relative to a stationary simulated ship. Results are compared for the 2D feature-based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.	154
4.7	Estimation of aircraft position relative to a stationary simulated ship. Results are compared for the 2D feature-based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.	155
4.8	Estimation of aircraft attitude angles relative to a stationary simulated ship using only stereo vision algorithm results. The initial pose is obtained from Vicon data.	157
4.9	Estimation of aircraft position relative to a stationary simulated ship using only stereo vision algorithm results. The initial pose is obtained from Vicon data.	158
4.10	The error in estimated translational motion compared with feature match count and final least squared error in rigid body transformation. Horizontal lines illustrate selected thresholds for minimum feature matches and maximum error.	162
4.11	Using depth map to remove image background and isolate moving ship structures.	164
4.12	Comparison of mean corresponding 3D point pairs and inliers for each setting.	166
4.13	Estimation of attitude angles relative to a simulated ship undergoing Sea-state 6 motion, with the quadrotor held stationary. Results are compared for the 2D feature-based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.	167

4.14	Estimation of position relative to a simulated ship undergoing Sea-state 6 motion, with the quadrotor held stationary. Results are compared for the 2D feature-based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.	168
4.15	Estimation of attitude angles relative to a simulated ship undergoing Sea-state 6 motion, with the quadrotor undergoing large attitude and linear motions. Results are compared for the 2D feature-based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.	169
4.16	Estimation of position relative to a simulated ship undergoing Sea-state 6 motion, with the quadrotor undergoing large attitude and linear motions. Results are compared for the 2D feature-based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.	170
4.17	Estimation of aircraft attitude angles relative to a stationary simulated ship using only stereo vision algorithm results. The initial pose is obtained from Vicon data.	172
4.18	Estimation of aircraft position relative to a stationary simulated ship using only stereo vision algorithm results. The initial pose is obtained from Vicon data.	173
4.19	Estimation of aircraft attitude angles relative to a stationary simulated ship using only stereo vision algorithm results. The initial pose is obtained from Vicon data.	174
4.20	Estimation of aircraft position relative to a stationary simulated ship using only stereo vision algorithm results. The initial pose is obtained from Vicon data.	175
4.21	Estimation of aircraft attitude angles relative to a stationary simulated ship, with the quadrotor in manually piloted free-flight. Results are compared for the 2D feature based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.	177
4.22	Estimation of aircraft position relative to a stationary simulated ship, with the quadrotor in manually piloted free-flight. Results are compared for the 2D feature based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.	178
5.1	Diagram illustrating basic autonomy architecture for vision-based landing.	183
6.1	Large Stewart platform with attached 6×6 foot fiberglass surface to be used for vision-based autonomous landing. The platform can recreate severe sea states up to Sea-State 7.	194

List of Abbreviations

CPU	Central processing unit
CUDA	Compute Unified Device Architecture
DLT	Direct linear transformation
EKF	Extended Kalman filter
ESC	Electronic speed controller
FFT	Fast Fourier transform
GNSS	Global navigation satellite system
GPS	Global Positioning System
GPU	Graphics processing unit
HHRSD	Helicopter hauldown and rapid securing device
IMU	Inertial measurement unit
MAVLink	Micro Air Vehicle Link
ORB	Oriented FAST and rotated BRIEF
PID	Proportional-integral-derivative
RF	Radio frequency
RMSE	Root mean square error
ROS	Robot Operating System
SVD	Singular value decomposition
SIFT	Scale invariant feature transform
UART	Universal asynchronous receiver and transmitter
UAS	Unmanned aerial system
UAV	Unmanned aerial vehicle
UMD	University of Maryland
USB	Universal serial bus

Chapter 1: Introduction

This chapter introduces the topic of this dissertation. It defines the research topic, covers the background and motivation behind the research topic, surveys the literature, and provides an overview of prior work at UMD. It then states the overall contributions of this work and provides an outline of the dissertation.

1.1 Ship-Deck Landing

Ship-decks present a challenging and highly dynamic environment, with stochastically varying linear and large rotational motions. Landing a helicopter or any other vertical lift aircraft on a moving ship-deck therefore remains one of the most challenging and dangerous tasks for a human pilot.

The objective ship for this dissertation is a small ship, such as a DDG-51 type ship or similar. Larger ships, including carriers, have large inertia and very small motions, removing the challenges of dynamic deck motion. This work assumes that the target ship has no added instrumentation on-board to aid in landing; additionally, on the aircraft itself, GPS is denied and lidar is unavailable. Landing must therefore be achieved using only computer vision and on-board hardware. This is the challenging context in which we seek to solve the problem.

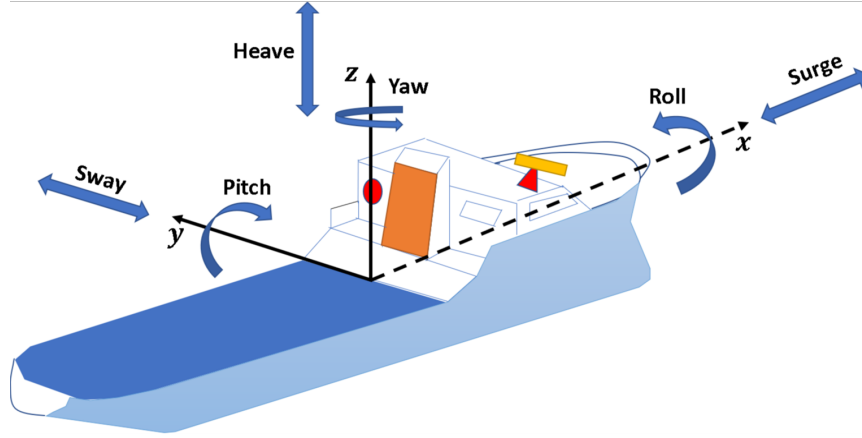


Figure 1.1: Illustration of ship motions along reference axes.

1.1.1 Characterization of ship-deck motion

Ships undergo motion along six degrees of freedom, as shown in Figure 1.1. The three rotational motions are described using Euler angles: roll about the X-axis, pitch about the Y-axis, and yaw about the Z-axis. The three translational motions are known as surge, sway, and heave, describing linear motion along the X-, Y-, and Z-axes respectively.

The Douglas sea scale is used to characterize the roughness of sea conditions. It defines various sea-states ranging from calm to extremely rough sea, as summarized in Table 1.1. Higher sea-states correspond to more severe ship motions.

Ship-deck motion is stochastic, with random variations in amplitude and frequency. It will typically include alternating periods of larger motion and quiescent (calmer) motion. However, its statistical properties can be evaluated using time histories of ship-deck motion, obtained either through real-world measurement or simulations. The Navy SCONE dataset provides multiple motion time histories obtained through time-domain simulation [6]. These time histories are representative of a DDG-51 type ship and include results for a range of sea-states. An example time history is shown in Figure 1.2; note the quiescent periods with smaller roll amplitudes at

Table 1.1: Sea-states defined by Douglas sea scale.

Sea-state	Wave height (m)	Description
0	0	Calm (glassy)
1	0-0.1	Calm (rippled)
2	0.1-0.5	Smooth
3	0.5-1.25	Slight
4	1.25-2.5	Moderate
5	2.5-4.0	Rough
6	4.0-6.0	Very rough
7	6.0-9.0	High
8	9.0-14.0	Very high
9	>14.0	Phenomenal

approximately 100 s and 250 s.

Fourier transforms of the time histories can be used to approximate the typical frequencies of ship-deck motion. An FFT (Fast Fourier Transform) plot of roll acceleration is shown in Figure 1.3. For all motions, the predominant frequencies fall between 0.05 and 0.3 Hz.

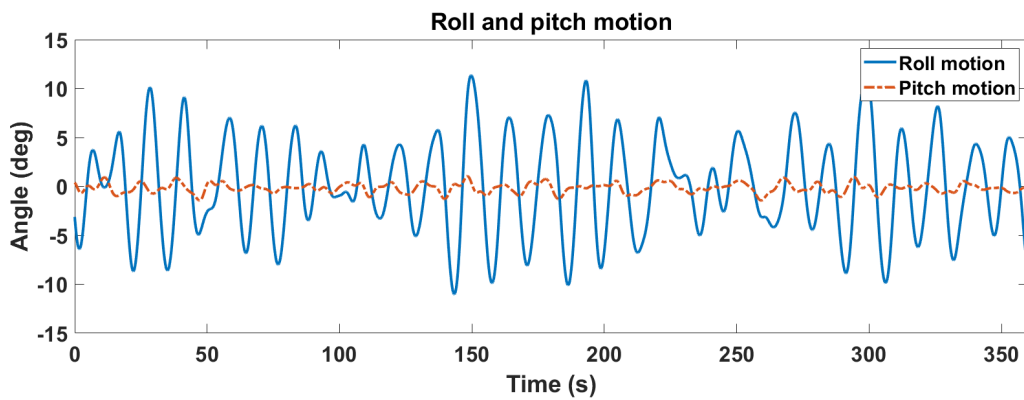


Figure 1.2: Example time history showing Sea-state 5 roll and pitch motions.

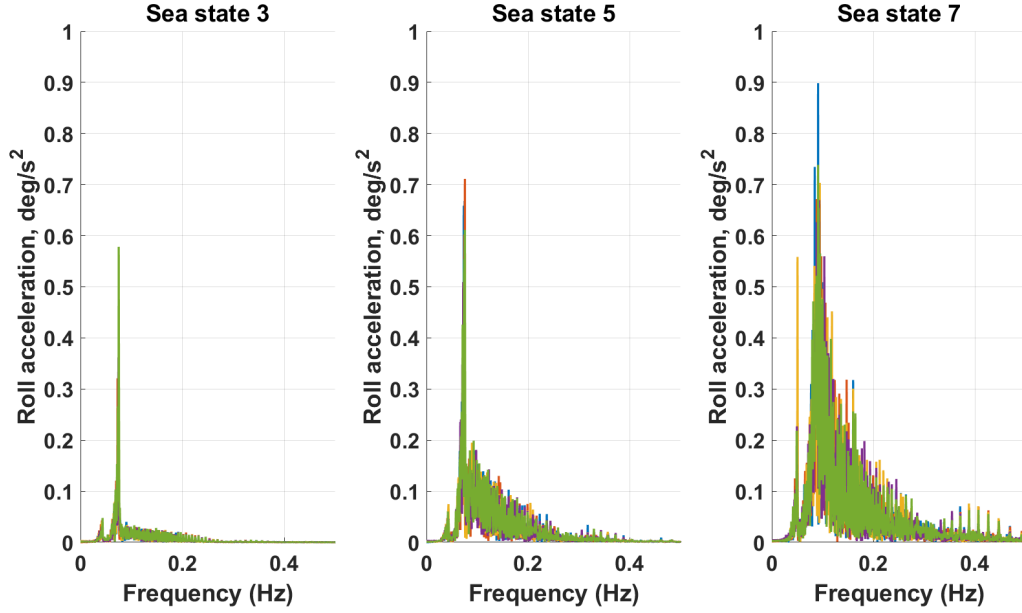


Figure 1.3: Fourier transform of roll acceleration. Colors represent different time histories.

1.1.2 State-of-the-art piloted landing procedure

For helicopter landings today, pilots wait for a quiescent (calm) period of ship-deck motion to safely descend and land. While waiting for a safe landing opportunity, the pilot hovers above the flight deck, tracking its position. This process is very challenging as the pilot must simultaneously monitor the ship's motion and accurately maintain the aircraft position above the small landing area.

Ship landings commonly utilize a helicopter hauldown and rapid securing device (HHRSD), also known as a beartrap. Developed by the Royal Canadian Navy in the late 1950s, the beartrap enables landing at higher sea states with large deck motions and improves the overall safety of landing [7]. During landing, a cable is lowered from the helicopter hovering above the ship deck. Crew members on the deck then secure the cable to the beartrap. A landing safety officer controls a winch below the deck to maintain tension in the cable, aligning the motion of the helicopter and

ship. This is shown in Figure 1.4 [8]. During descent, the helicopter drops power and is slowly pulled down by the cable and winch. When the helicopter lands, the beartrap is closed to secure it to the deck [9], [10]. This ensures that the aircraft will not slide, tip over, or fall overboard during handling, even in higher sea-states. Once secured, the beartrap device can be moved along a track to transport the aircraft to the onboard hangar.

1.1.3 Typical operating limits

Operational limits define conditions in which a shipboard landing can be carried out safely. These limits are typically specific to each aircraft and/or ship, and incorporate factors such as wind conditions, flight deck motion, and aircraft structural limitations [11]. The defined limits may also vary across different operational phases, including launch, landing, and deck handling. During the final landing stage, deck rotation and motion criteria are used to determine quiescent periods of motion for safe descent and landing. For full-size helicopters, an example of typical limits are a maximum roll angle of 8 degrees and a maximum pitch angle of 3 degrees [12]. In comparison, the ship deck roll and pitch motions may be up to 30 and 10 degrees respectively.

1.1.4 Modern shipboard landing systems

A number of commercial systems have been developed for shipboard landing. Some systems assist with piloted landings, while others can be utilized for autonomous landing.

One example of the former is the Aircraft Ship Integrated Secure and Traverse (ASIST) system by Indal Technologies (now Curtiss-Wright). Optical sensors track a helicopter equipped with a laser beacon, determining the position of the helicopter relative to the flight deck. This



Figure 1.4: On May 25, 1967, Sea King helicopter CH-SS2 is hauled down by a Bear Trap to a landing on the deck of Her Majesty's Canadian Ship Assiniboine. (Courtesy Department of National Defence / PCN-67-415)



Figure 1.5: Shipboard landing with ASIST system [1]

data is provided to the pilot to aid in landing, and is also used to align a securing device on the deck below the helicopter [1]. Figure 1.5 shows a landing carried out using this system.

An example system for autonomous UAV landing is the UAV Common Automatic Recovery System (UCARS) by Sierra Nevada Corporation. A ground tracker installed on the flight deck localizes and tracks a transponder on the aircraft, shown in Figure 1.6. Position data from the system is used to perform precise autonomous landings [2].

DeckFinder by Airbus is another system used for autonomous landings of both manned and unmanned aircraft. Six ground units on the ship-deck emit RF ranging signals. These signals are received and processed by a range measurement unit onboard the aircraft, which precisely calculates the aircraft relative position [3]. The system is illustrated in Figure 1.7.



(a) Ground tracker installed on flight deck.



(b) Airborne transponder.

Figure 1.6: UCARS system components [2]



(a) Landing demonstration.



(b) Position localization using RF ranging signals.

Figure 1.7: Airbus DeckFinder system [3]

For all of these systems, heavy equipment and instrumentation is needed on the ship deck; this, they are not the type of autonomy envisioned in this dissertation.

1.2 Motivation

Even with more recent advancements in aircraft stability and control systems, ship-deck landing remains an extremely difficult and hazardous task. There is a significant risk of accidents due to pilot error, particularly at high sea states. Landing using the beartrap system requires crew to be present on the ship-deck, placing them at risk of accidents or falling overboard.

Additionally, the modern ship landing systems possess significant limitations. All of these systems require specialized equipment on both the ship and the aircraft, adding significant expense and limiting their use to ships and aircraft that can accommodate this added equipment. Additionally, these systems provide only positioning information, rather than both deck relative position and orientation. This is a major drawback at high sea states, where orientation information is needed to determine safe landing periods.

An autonomous landing system, one which is reliant only on on-board sensors, is needed to mitigate the limitations of existing ship landing systems. Vision-based systems, which estimate the ship pose using on-board vision hardware, could potentially fill this need. Vision systems mimic the perception of the human eye, but with the precision of a computer, to calculate what the human brain cannot - the pose of the deck - quickly and accurately and even under degraded visual conditions. Using only cameras on the aircraft, a vision system could estimate the pose of a moving ship-deck with high accuracy. An autonomous landing controller can process this information and respond with greater speed and precision than a human pilot. This would remove the risk of pilot error during landing, and enable safer landing at higher sea states.

With the growing use of smaller-scale unmanned aerial systems (UAS), there is also a need to implement safe ship landing procedures for these aircraft, ideally without the need for manual pilot intervention. Due to their low weight and power requirements, vision systems can be scaled down for smaller UAS in addition to larger, full-size rotorcraft.

1.3 Computer Vision

Computer vision encompasses theory and techniques for processing and acquiring information from images. The field began to emerge in the 1960s from both the early development of digital image processing and research into the biological vision system. Many foundational computer vision techniques, including edge detection and optical flow, were later developed throughout the 1970s and 1980s. Preliminary work in neural networks for computer vision was also carried out during this time [13]. Significant advances would later be made in the area of 3D computer vision, including 3D reconstruction. Many feature detection and matching algorithms would also be developed throughout the late 1990s, 2000s, and 2010s [14, 15]. With advancements in computing technology and availability of large datasets, machine learning techniques gained significant attention in the early 2010s, and remain a very active research area to this day. This has led to major advancements in common computer vision tasks, such as image identification, object detection, and 3D vision [16, 17].

A wide range of applications exist for computer vision, including facial recognition, medical image analysis, industrial inspection, text recognition, and terrain mapping. Computer vision is widely used for autonomous and semi-autonomous ground vehicles for tasks such as navigation and obstacle detection.

1.3.1 Computer vision in aviation

Computer vision has also seen widespread use in aviation, encompassing a wide range of research, commercial, and military applications. On full-size aircraft, applications of computer vision include navigation in GPS-denied environments, surveillance, and pilot assistance systems.

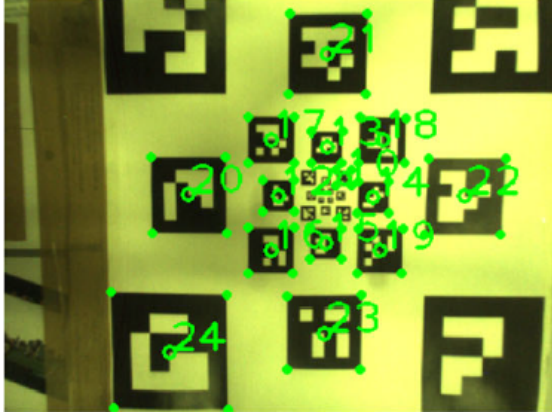
The low weight and power requirements of onboard cameras has enabled widespread use in small UAVs, including multirotor aircraft. Common computer vision uses include obstacle avoidance and object tracking. Vision tasks are processed using either small onboard computers or offboard ground stations.

1.4 Vision-Based Landing

A large number of studies have addressed the problem of vision-based rotorcraft landing, with many focusing on multirotor UAVs landing on stationary or linearly moving targets. Various approaches have been investigated for detecting the landing target using one or more onboard cameras. Some examples are illustrated in Figure 1.8. One common approach is the use of fiducial markers or tags, such as ArUco markers [18–20]. Other work has utilized simple custom-designed tags [21, 22] or arrangements of lights or markings on the landing pad [23]. Feature detection and matching techniques have also been explored to detect known areas of ground terrain near a landing location [24, 25]. More recently, machine learning approaches have been investigated for landing target detection, particularly in visually challenging scenarios [26, 27].

1.4.1 Vision-based ship landing

Some prior work has investigated the specific problem of vision-based landing on moving ships. Ship-decks produce large, stochastic oscillating motions, which introduces significant difficulty to both visual detection and safe landing. Similarly to the more general work on rotorcraft landing, previous work on ship-deck landing has mainly utilized simple visual references to detect ship-deck motion. This includes fiducial markers [28, 29], H-shaped landing targets



(a) Detection of AprilTag fiducial markers [18].

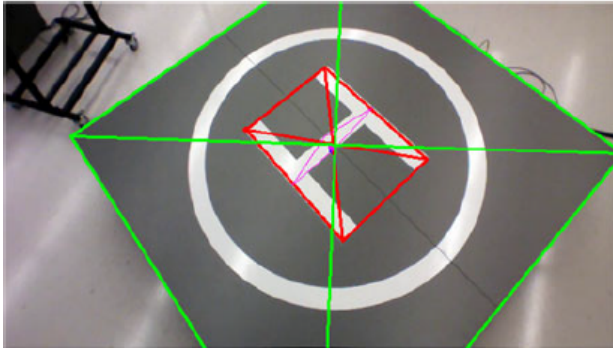


(b) Detection of landing marker using neural network [27].

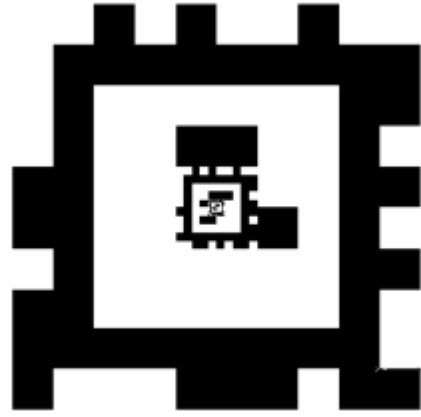
Figure 1.8: Examples of vision-based landing techniques.

[30, 31], or arrangements of simple visual markers [32–34]. Detecting ship structures using machine learning techniques has been explored in more recent work [35, 36]. Examples of vision-based landing approaches are illustrated in Figure 1.9.

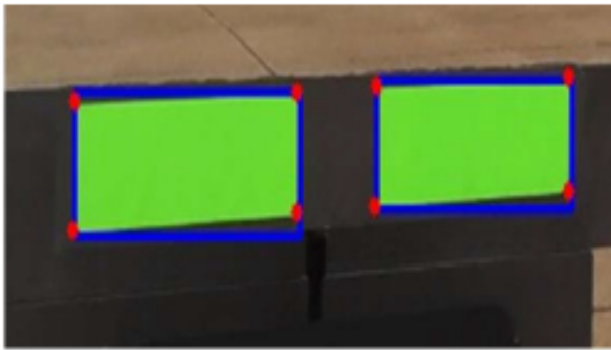
Previous work at UMD has investigated the use of vision for ship-deck landing using fiducial markers, particularly for tracking stochastic ship-deck motion in high sea states, where severe oscillations are present. It was validated that a ship-deck could be detected and tracked using only vision with a single camera [4]. A quadrotor was developed in-house to demonstrate tracking and landing on a stochastically moving ship-deck. A fiducial-based algorithm, which detects the pose (position and orientation) of an AprilTag marker on the ship-deck, was used and executed on the quadrotor using an on-board flight computer. In experimental tests, the algorithm was able to detect and track the ship-deck at various oscillation frequencies as well as for stochastic motion, as shown in Figure 1.10. However, the fiducial-based algorithm fails in visually-degraded conditions, including for any type of occlusion on the marker, since the entire marker must be clearly visible for detection and pose estimation.



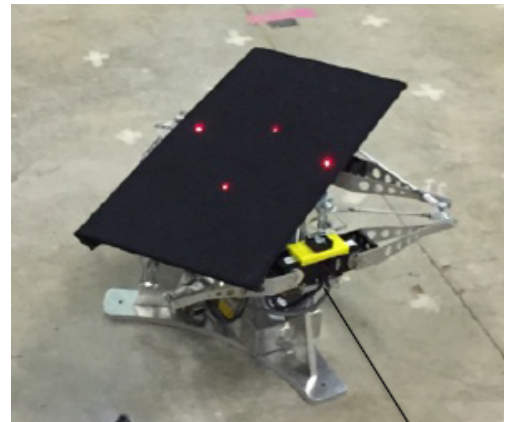
(a) Detection of H-shaped landing marker [30].



(b) Recursive AprilTag fiducial marker [29].



(c) Detection of colored rectangular markers on ship [34].



(d) Arrangement of lights on moving deck [32].

Figure 1.9: Various techniques in prior work for visual detection of a moving ship-deck.

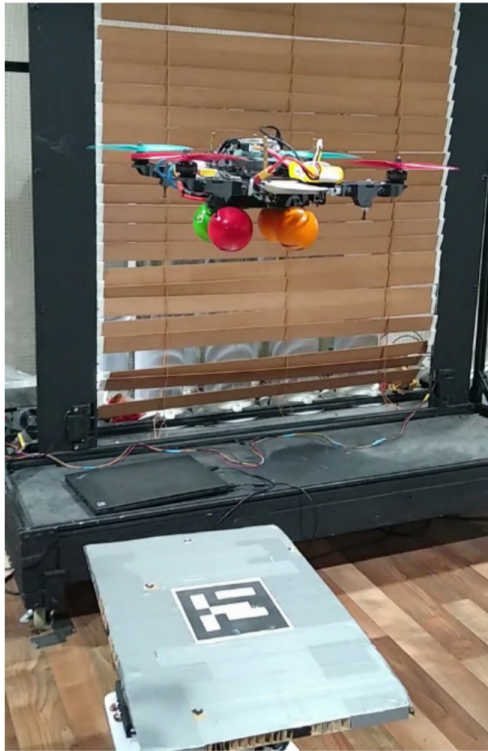


Figure 1.10: Tracking moving platform using fiducial marker [4].

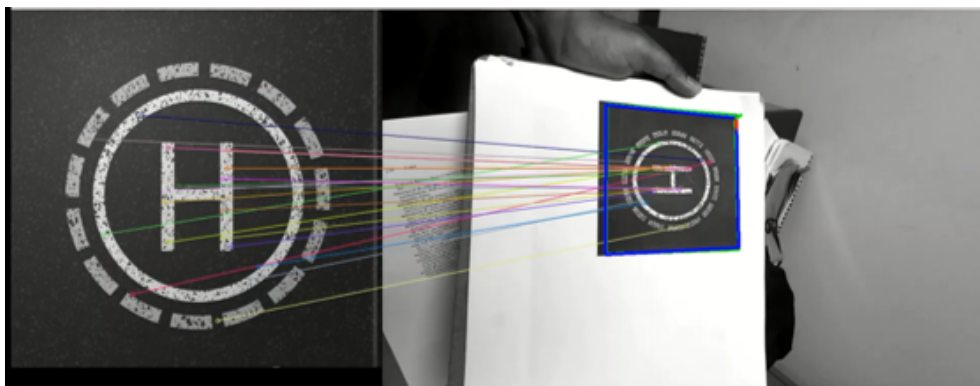


Figure 1.11: Detecting landing pad using feature matching [5].

To rectify this deficiency, a new feature-based approach was investigated [5]. The new technique, which utilizes feature detection and matching to estimate the pose of a landing pad, is shown in Figure 1.11. While feature detection has been investigated for rotorcraft landing [24], this technique has not been previously used to track the pose of a moving landing platform. This new algorithm is intended to accurately track the pose of a ship-deck in both ideal and visually-degraded conditions. It was verified to be robust to occlusions and illumination differences under simple bench-top test conditions. This algorithm is also capable of working with generic landing pad patterns, unlike the fiducial-based algorithm, which requires a pre-designed tag. However, the algorithm was computationally intensive, requiring a flight computer with significantly increased power requirements and weight. This computer could not be integrated into the previous quadrotor aircraft.

1.5 Objective of this Research

Prior efforts on vision-based ship landing have primarily focused on low sea-states, where ship motions, in particular rotational motions, are small. The majority of prior experimental work also does not address visually degraded conditions such as occlusion or glare, despite such conditions being commonly experienced in real ship-deck landing scenarios. Vision approaches utilizing simple visual references, including fiducial markers, can potentially fail in such conditions.

A vision approach based on feature detection and matching can mitigate these concerns. Feature-based approaches offer robustness to visually degraded conditions; in addition, these techniques can be used for 2D landing pad detection, or can be combined with 3D computer vision methods to track 3D ship structures.

The main objective of this research is to develop and experimentally evaluate feature-based vision approaches for autonomous ship-deck landing, with a particular focus on high sea-states and visually degraded conditions.

1.6 Contribution and Technical Approach

The unique contribution of this research is the development of feature-based vision algorithms for ship-deck landing which can eliminate the need for special tags, can handle degraded visual conditions, and rely only on computer vision hardware and software onboard the aircraft, removing the need for instruments on the ship. These combinations constitute a unique set of requirements that have not been addressed in the past. The key contributions were the following:

1. Development of a 2.75 kg quadrotor UAV, incorporating the hardware necessary for vision-based landing, without the need for additional off-board sensors or computer hardware.
2. Refinement of the 2D feature-based algorithm introduced in [5] to mitigate individual poor pose estimates.
3. Extensive experimental characterization of the 2D feature-based algorithm for sea-state 6 motion, visually-degraded conditions, and free flight.
4. Development of a new 3D feature-based algorithm based on the structure-from-motion computer vision technique; the algorithm is able to accurately estimate the pose of a known 3D object.
5. Experimental validation of the 3D feature-based algorithm for sea-state 6 motion, visually-degraded conditions, and free hover.

6. Development of a feature-based stereo vision algorithm, capable of estimating the motion of arbitrary ship structures without advance knowledge of the ship's visual appearance.
7. Creation of a new vision algorithm combining output from both the 2D and stereo vision algorithms; the combined algorithm can estimate the pose of realistic ships without the need for added markers or tags.
8. Validation of the stereo and combined algorithms using a realistic ship representative of a DDG-51 ship; experiments were performed for sea-state 6 motion and free flight.
9. Initial development of an autonomous landing algorithm to demonstrate vision-based landing.

1.7 Organization of the Dissertation

The dissertation is organized into six chapters. Chapter 2 describes the implementation of the 2D feature-based algorithm, including the development of a quadrotor UAV incorporating the required on-board vision hardware. This chapter also describes refinements made to the algorithm and provides extensive experimental characterization for high sea-state motion, visually degraded conditions, and free flight. Chapter 3 details the development of a new 3D feature-based algorithm, which estimates the pose of a 3D object rather than a 2D reference image. The algorithm is described in detail, and experimental validation is performed for ship-deck motion, visually degraded conditions, and free hover. Chapter 4 describes a new feature-based stereo vision algorithm to estimate the motion of arbitrary ship structures. The algorithm is also combined with the previous 2D feature-based algorithm to enable full estimates of ship-deck pose without drift. The stereo vision and combined algorithms are validated for ship-deck motion and free

flight using a realistic simulated ship. Chapter 5 details efforts to develop an autonomous landing algorithm using pose estimation data from the feature-based vision algorithms. Chapter 6 lists the key conclusions and recommendations for future work.

Chapter 2: 2D Feature-Based Algorithm

This chapter documents a 2D feature-based algorithm for ship-deck pose estimation and its experimental validation. First, the development of a new quadrotor UAV platform incorporating all required vision hardware is described. Next, the results of controlled algorithm testing for simulated ship-deck motion and under visually degraded conditions are provided. Finally, the algorithm performance is evaluated for free-flight, first in visually ideal and then in visually degraded conditions. The material presented in this chapter was previously published in the Journal of the American Helicopter Society [37].

2.1 Fiducial-Based Algorithm

Fiducial markers, such as AprilTags or ArUco markers, are commonly used for vision-based UAV landing. A fiducial-based approach was used in previous work at UMD, with vision-based landing successfully demonstrated in ideal visual conditions [4]. However, this approach fails under visually degraded scenarios, such as those illustrated in Figure 2.1 as the entire marker must be clearly visible for successful detection and pose estimation. A feature-based vision approach is robust to visually-degraded conditions such as glare, occlusion, and distortion; this motivated the development of a 2D feature-based algorithm to detect and estimate the pose of a ship-deck [5]. However, this algorithm could not previously be demonstrated due to hardware

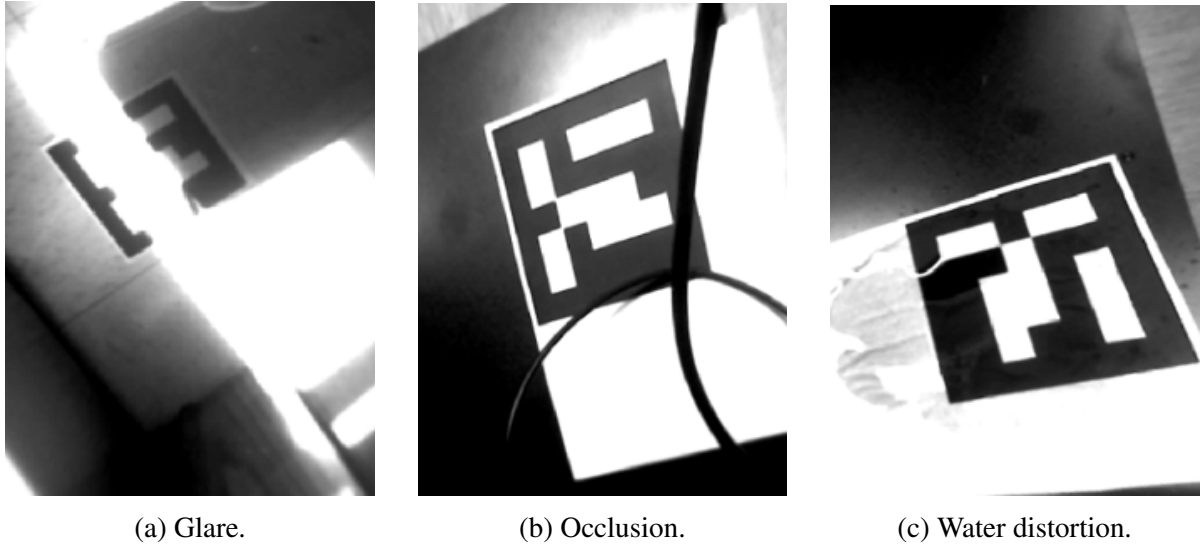
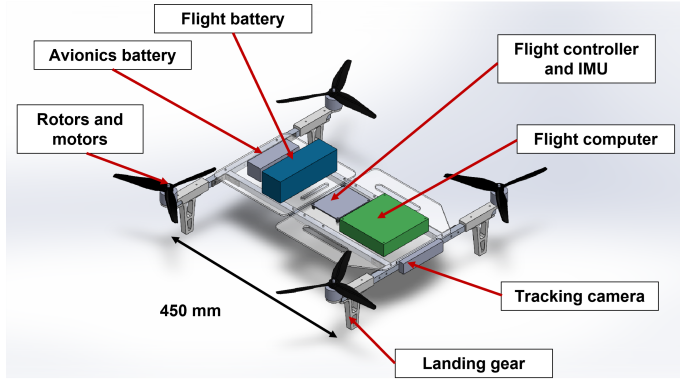


Figure 2.1: Visually degraded conditions resulting in detection failure of an ArUco marker [5].

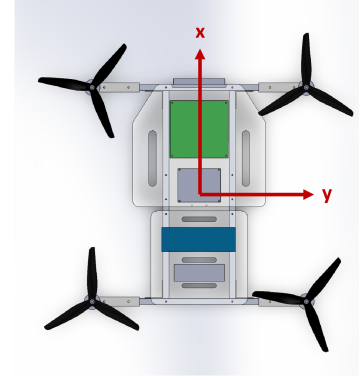
and computational limitations. A refined implementation of the algorithm on new hardware are discussed in this chapter.

2.2 Quadrotor Development

The intent of our vision algorithm is that it must function using only on-board computer and vision hardware, as ship-deck landings may occur in isolated locations or poor weather, where external communication is unreliable or compromised. The feature-based vision algorithm is computationally intensive, requiring a powerful on-board computer to operate in real-time. Therefore, there were physical size and weight constraints, because of which it was not possible to readily integrate the required computer into an existing aircraft. There was also a need to customize the onboard vision and computer hardware based on evolving research requirements. Additionally, it was also an objective to establish the absolute minimum of hardware and software requirements for vision, with no unnecessary weight overhead. For these reasons, developing a custom quadrotor in-house was deemed desirable. A 2.75 kg gross weight quadrotor UAV was



(a) Major components.



(b) Axes system.

Figure 2.2: CAD diagram of new quadrotor showing major components and axes system. Note that the z-axis is positive down.

sized, designed, and fabricated around this new computer and other ancillary hardware.

2.2.1 Flight computer

While the feature-based algorithm is more capable than our previous fiducial-based system [4], it is also much more computationally intensive. An UP Xtreme embedded computer, featuring an Intel i5 processor, was selected for the large quadrotor UAV.

This computer is able to update the vision algorithm at a maximum rate of 12 Hz in benchtop tests. In contrast, the computer used in previous experiments is only capable of updating the feature-based algorithm at a rate of 0.7 Hz, far too slow for real-time operation [5].

2.2.2 Structure and specifications

The new, larger quadrotor was developed and fabricated in-house using a combination of in-house custom-built and off-the-shelf components. The overall specifications of the final aircraft are listed in Table 2.1. A CAD model of the quadrotor was first developed as shown in Figure 2.2, with the overall layout and major components. The mass and moments of inertia were estimated

from CAD software. In designing and choosing components for this UAV, we kept in mind future modifications and possible hardware additions. Therefore, the quadrotor was designed to be larger and to carry a greater payload than the current test hardware. This allows components to be added in the future with fewer weight or geometric constraints.

Table 2.1: Quadrotor characteristics.

Parameters	Value
Gross take off mass	2.75 kg
Rotor spacing (x and y)	0.45 m
Rotor diameter	0.229 m
Rotor RPM at hover	7300 RPM
Tip Mach	0.26
Center of mass (x)	-0.020 m
Center of mass (y)	0.0 m
I_{xx}	0.0350 kgm ²
I_{yy}	0.0699 kgm ²
I_{zz}	0.1018 kgm ²
Battery mass	1.0 kg
Hover endurance	12.8 minutes

The new quadrotor has a frame size of 450 mm $\times$ 450 mm (measured from the centers of the motors) and a gross take-off weight of 2.75 kg. This is much larger than the previous quadrotor, which had a frame size of 270 mm $\times$ 270 mm. The frame is constructed of 0.5 inch (12.7 mm) 6061-T6 rectangular aluminum tubes, with 1/16 inch (1.59 mm) wall thickness. The 6061 aluminum alloy offers greater strength compared to the 6063 aluminum alloy used in the previous quadrotor. The tubes are arranged in an H-shape and attached with steel L-shaped corner brackets, similar to the configuration used in the previous quadrotor. Aluminum tubes

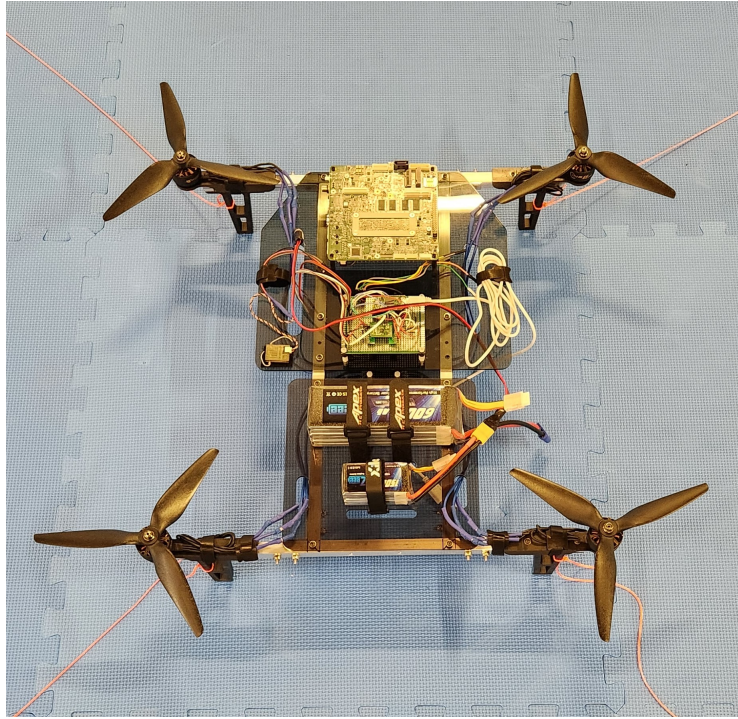


Figure 2.3: Photo of the new larger quadrotor.

were chosen because they provide structural strength and stiffness while being easy to machine.

The key components, namely the flight computer, flight controller, batteries, and camera are mounted to laser-cut acrylic plates mounted to the frame. If any components need to be changed or rearranged, or if these acrylic plates are damaged in a crash, these items can be quickly re-cut with in-house laser cutters and replaced.

The motor mounts and landing gear were also custom-designed and produced using in-house additive manufacturing facilities. They were printed using a Markforged Mark 2 3D printer using Onyx, a micro carbon fiber filled nylon material. Additional strength was provided through printed carbon-fiber reinforcement layers.

A photograph of the assembled quadrotor is shown in Figure 2.3.

2.2.3 Propulsion and power

The vehicle features four 9-inch diameter, 3-bladed rotors, driven by T-Motor Cine66 925KV 2812 brushless electric motors. Using manufacturer-provided test stand data for this motor/rotor combination, we calculated a maximum hover endurance of 12.8 minutes.

The four motors are driven by a Holybro Tekko32 4-in-1 ESC. This ESC supports a maximum current of 65 A for each motor. While the motor current is typically well below this level, in the case of sudden maneuvers or transients, this provides a factor of safety above the maximum motor current provided in test data.

Separate lithium polymer batteries are used to power the motors and avionics. A 1500 mAh, 4 cell (14.8 V), 120C battery powers the avionics, including the flight computer. Another 6000 mAh, 6 cell (25.2 V), 50C battery is used to power the motors via the ESC. The weight of the batteries are 0.18 kg and 0.83 kg, respectively.

2.2.4 Avionics and control

The flight control loop used for attitude control and stabilization is executed on a Teensy 4.0 microcontroller with an ARM Cortex-M7 processor, which is capable of updating the attitude control loop at 1000 Hz. A ST LSM6DSOX IMU, incorporating a gyroscope and accelerometer, is used to measure angular velocity and linear acceleration along 3 axes, respectively. The Teensy communicates with the IMU through the I2C communication protocol, and collects updated measurements on each control loop update. The Teensy then sends commands for each motor to the ESC using the digital DShot protocol.

The flight controller receives commands from a 2.4 GHz radio receiver using UART, a serial

communication method, utilizing an open-source library to read each message. This receiver is paired with a transmitter, enabling an operator to manually control the quadrotor.

The flight controller stabilizes attitude with nested PID controllers. The inner control loop takes in desired angular rates and outputs desired moments. These moments are converted to motor commands using a control mixer. Gyroscope measurements are used to calculate the current angular rates. A Kalman filter is applied to the raw gyroscope measurements before sending to the controller in order to filter noise and produce more reliable angular rate estimates. The outer control loop takes in desired attitude angles and outputs desired angular rates, which are sent to the inner loop controller. A Kalman filter fuses measurements from the accelerometer and gyroscope to produce estimated attitude angles.

2.2.5 Vision hardware

Images of the ship-deck are collected using a See3CAM USB camera, manufactured by e-con Systems. This is the same camera previously used for initial benchtop testing of the feature-based algorithm [5]. The camera is mounted on the underside of the quadrotor, facing downward. This camera features a built-in auto-exposure algorithm, which automatically varies the camera shutter speed for varying light intensity received by the camera. This allows the camera to compensate for a wide range of scene illumination, though at the cost of increased motion blur for low illumination levels.

2.3 Vision Algorithm

The 2D feature-based algorithm we developed earlier in [5] is described in this section.

For this work, functions provided in OpenCV, an open-source computer vision library, were used for various image processing tasks, such as image reading and scaling. Additionally, we utilized feature detection and matching functions provided in OpenCV. The new contribution is in calculating the pose of a moving ship-deck using these techniques and additional algorithms.

2.3.1 Feature extraction and matching

Features, in this context, can be described as small patches of interest in an image. Features should preferably be invariant to transformations in scale, translation, in- and out-of-plane rotation, and illumination.

Locations of interest, or keypoints, in an image are typically found by detecting strong corners or blobs, which have significant intensity variation in both directions. Multiple techniques exist for detecting keypoints in images. One common method is the Harris corner detector [38]. For a small window or patch of pixels w , the detector evaluates the change resulting from a small displacement (u, v) . For corners, displacement in any direction will result in significant change. Given a grayscale image I , this is expressed as

$$E(u, v) = \sum_{(x,y) \in w} [I(x + u, y + v) - I(x, y)]^2 \quad (2.1)$$

From Taylor expansion, an approximation can be obtained:

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \quad (2.2)$$

The structure tensor M is defined as

$$M = \sum_{(x,y) \in w} \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \quad (2.3)$$

where I_x and I_y are the image gradients in the x and y directions, respectively. The eigenvalues of M are found from $\det(M) = \lambda_1 \lambda_2$. If both λ_1 and λ_2 are large, the window contains a corner. If one eigenvalue is much larger than the other, the patch is an edge (significant change only in one direction). If both are small, the region is "flat" with little variation.

Image patches at locations of interest are extracted and converted to feature descriptors. A descriptor is a numerical vector representation of the image patch. Extracted features can be matched to find corresponding points across multiple images. Good matches are found by comparing the distance between two descriptors (vectors), finding pairs with the smallest distance between them.

2.3.2 SIFT algorithm

The Scale Invariant Feature Transform (SIFT) algorithm, developed by Lowe [14], is a commonly used algorithm for feature detection and extraction. The algorithm generates vectors of length 128 containing integer values as feature descriptors. The algorithm is invariant to changes in scale and translation, and is somewhat resilient to changes of in- and out-of-plane rotation and illumination. For feature matching, the distance between two features is found using the L2 norm or Euclidean distance, computed using the formula

$$d(p, q) = ||p - q|| = \sqrt{\sum_{k=1}^n (p_k - q_k)^2} \quad (2.4)$$

where p and q are vectors of length n . In this case, these are two feature descriptors of length 128.

SIFT keypoints are detected using a technique known as the Difference of Gaussians method, which detects image edges. A technique similar to the Harris corner detector is then used to identify corners and blobs, eliminating edge points. This technique is applied at different image scales to achieve scale invariance.

Good SIFT features should be distinctive, meaning that these features can be reliably matched across multiple images with few incorrect matches. Good features typically consist of corner- or blob-like regions with high contrast. Low contrast or non-distinctive features are undesirable for feature matching tasks.

For this work, an implementation of SIFT provided in OpenCV was used.

2.3.3 Camera pose estimation

The vision algorithm must estimate the pose of the ship-deck relative to the camera (and aircraft). To do this, it is necessary to obtain correspondences between known 3D points in the scene and 2D points projected onto the camera image. An optimal rotation and translation can then be obtained that minimizes the error across all points, as illustrated in Figure [2.4](#).

For a single point correspondence, a model can be used to describe the mapping from 3D world coordinates to 2D pixel coordinates. This model is given below:

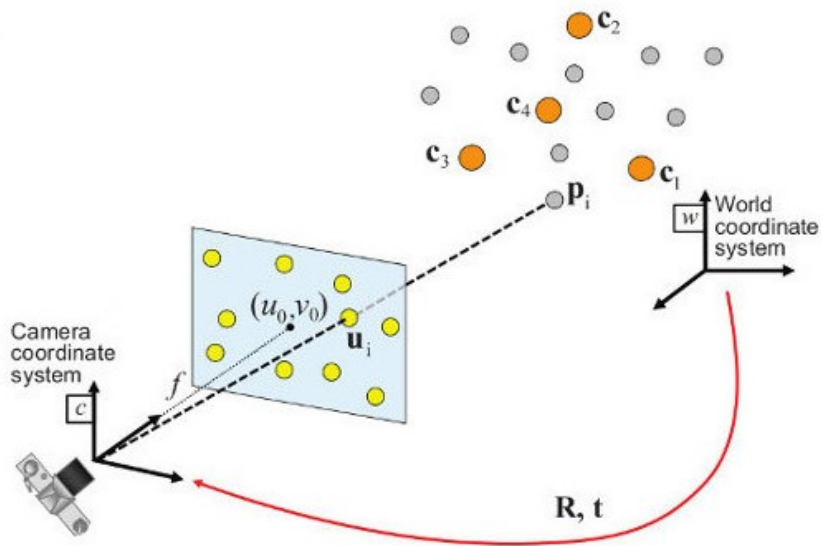


Figure 2.4: Estimating camera pose using 3D-2D point correspondences.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R_{3 \times 3} & T_{3 \times 1} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (2.5)$$

This can alternatively be written as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (2.6)$$

where P is a 3×4 camera matrix containing both the intrinsic and extrinsic parameters.

The 2D point location in pixel coordinates is (u, v) , while the 3D point in world coordinates

is (x, y, z) . Note that these coordinates are represented using homogeneous coordinates, with an additional component equal to 1, for mathematical convenience. K is the 3×3 intrinsic matrix, which transforms points from the 3D camera coordinate system to the 2D image coordinate system. This matrix is obtained through camera calibration. The extrinsic parameters R (a 3×3 rotation matrix) and T (a 3×1 translation vector) describe the transformation from 3D world coordinates to 3D camera coordinates, or the camera pose.

The Perspective-n-Point (PnP) problem involves finding the pose of the camera (the R and T terms in Equation 2.5) using a set of corresponding 3D world points and 2D image points. Various methods have been developed to solve the PnP problem; implementations of these methods provided by OpenCV are used for this work.

2.3.4 Ship-deck detection algorithm

The present algorithm detects and extracts SIFT features in images of the moving ship-deck. These features were matched to features extracted from a 2D reference image of the landing pad.

The following procedure was used to detect and estimate the pose of the ship deck:

1. First, extract SIFT features from a reference image of the pattern located on the ship-deck.

The extraction recovers the feature descriptors and their coordinates in the image. Any image may be used with the algorithm, but an image with many good SIFT features is preferable.

2. Load the most recent frame from the quadrotor camera, and extract SIFT features from the image.

3. For each feature in the reference image, find the two closest matches (best and second best match) in the features extracted from the camera image. The distance between features is found by calculating the L2-norm of the difference between the feature descriptors as given in Equation 2.4.
4. Filter out poor matches using a series of tests:

- (a) Lowe's ratio test: the ratio of the calculated distance for the best and second best match must be below a threshold, meeting the condition

$$d_1 < threshold \times d_2 \quad (2.7)$$

where d_1 is the distance for the best match and d_2 is the distance for the second-best match. This algorithm sets $threshold = 0.8$, as this value was found by Lowe to reject most incorrect matches while retaining the majority of correct matches [14].

- (b) Symmetry test: the best feature match from the reference image to camera frame should also be the best match from the camera frame to the reference image.
- (c) Distance test: the distance to a matched feature should be less than the minimum distance across all matches multiplied by a constant, 3 in this case. This is checked using the condition

$$d < 3 \times d_{min} \quad (2.8)$$

A match that fails any of these tests will be removed as a bad match.

5. Find the transformation from the reference image to the ship-deck in the camera image.

A homography can be used to transform points between two planar surfaces up to a scale factor. As both the reference image and ship-deck are planar, the homography can be used to describe the transformation from a 2D point location in the reference image to a 2D point location in the camera image:

$$s \begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix} = H \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} \quad (2.9)$$

where H is a 3×3 homography matrix, s is a scale factor, (x_r, y_r) is the coordinate of a point in the reference image, and (x_c, y_c) is the coordinate of the corresponding point in the camera image. The homography matrix is unknown and must be calculated; a minimum of 4 feature matches are required to do so. Therefore, if at least 4 feature matches are found after filtering poor matches, proceed with the following steps. Otherwise, report that the deck was not detected.

6. Use the Random Sample Consensus (RANSAC) algorithm to find a good homography. This algorithm works by repeatedly taking random samples of the matched points, which are used to calculate a homography. The remaining matched points are checked for consistency with the estimated homography. This procedure produces a result that is robust to outliers [39].
7. Use the homography to transform the corners of the deck image, obtaining the 2D coordinates of the deck corners within the camera image. Verify that the transformed points are geometrically valid by checking that the transformed deck corners form a convex quadrilateral.

If this is the case, the transformed corners can then be used to determine the relative camera pose using PnP. For this procedure, the center of the landing pad image is defined as the origin of the world coordinate system. As the aircraft axis (Z-down) convention is used, the coordinates of the image corners are $(\pm w/2, \pm h/2, 0)$ where w and h are the known width and height, respectively. The known 3D corner locations and transformed 2D corner points are then used to compute the pose of the ship-deck, or its position and orientation in 3D relative to the camera. For this algorithm, the PnP method described by Collins for coplanar 3D points is used [40].

2.3.5 Algorithm implementation

Figure 2.5 illustrates the processing steps in the vision algorithm. The filtering and refinement steps, applied to the initial pose estimate obtained using the procedure discussed above, are discussed in the next section.

The feature-based vision algorithm was written in Python and implemented on the quadrotor for real-time operation. As discussed earlier, the OpenCV library was used for basic computer vision functions. It used ROS (Robot Operating System) 2, a set of robotics software libraries, for asynchronous messaging between individual processes, or nodes. The vision algorithm and logging runs on a single node. A separate node continuously retrieves frames from the camera and publishes them on a topic, or a bus allowing communication between nodes. The vision node subscribes to this topic, collecting updated camera frames as needed. Using ROS, the vision node can then publish the pose estimates, making this data available to other nodes such as a ship-deck tracking controller.

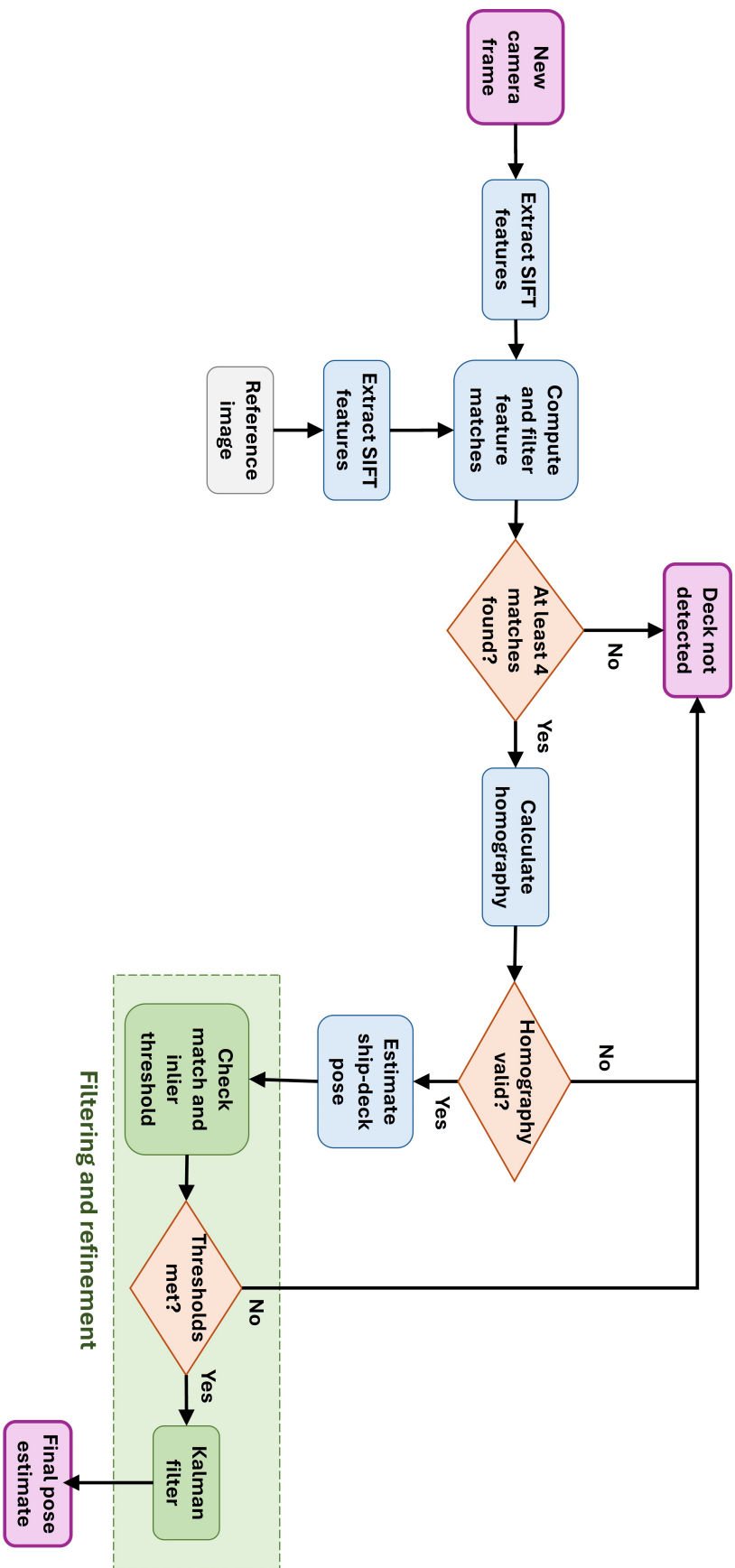


Figure 2.5: Flowchart illustrating the steps in the vision algorithm.

2.4 Algorithm Validation

Before this vision algorithm can be used for autonomous tracking and landing, it is necessary to validate the algorithm’s performance and reliability under varying conditions. For validation, hand-held and free-flight hover tests of the vision algorithm were conducted.

A primary motivation for the feature-based algorithm is its robustness to visually difficult conditions, compared to fiducial-based approaches, where even minor visual degradation will result in detection failure. The algorithm should provide usable estimates of the pose of the landing target under visually degraded conditions; because of this, it is necessary to investigate conditions where the performance of the feature-based algorithm may deteriorate. Data was collected for visually ideal conditions — in this context, typical indoor lighting and no added occlusions or lighting limiting visibility of the landing image — as well as under various types of visual degradation.

First, several large-angle controlled tests, in which the quadrotor was held by hand, were carried out for ideal and visually degraded conditions. Next, tests were conducted in free-flight hover.

Ground truth data was provided through a Vicon motion-capture system with 8 Vicon cameras, shown in Figure 2.6. The setup was calibrated before each experimental session, with a final measurement error of less than 0.5 mm reported by Vicon software. The absolute poses of the object and camera recorded by the Vicon were converted to relative poses, in order to be consistent with the vision algorithm output.

For all tests, quadrotor yaw was held constant. Only minimal yaw motion was present in the ship-deck motion time histories. Due to the lack of yaw motion, and the greater importance

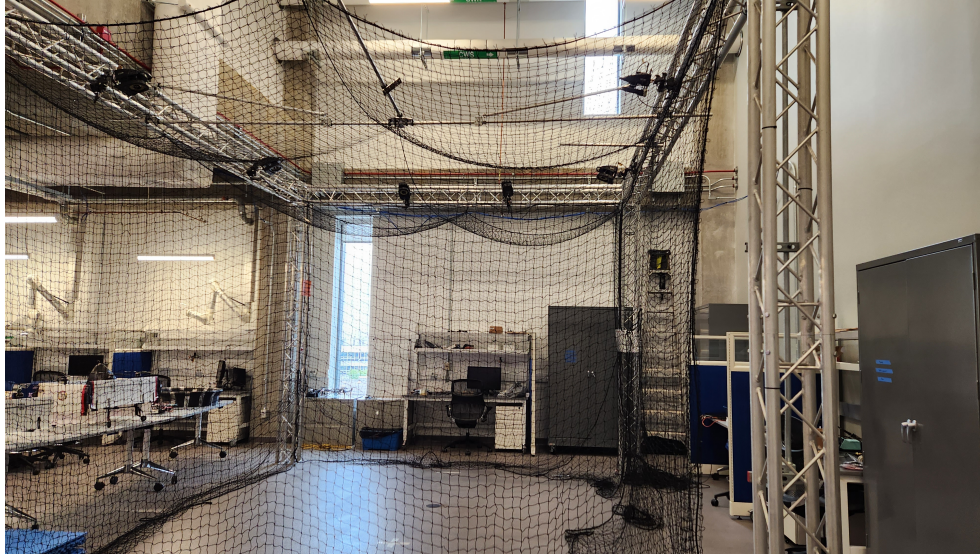


Figure 2.6: Vicon system and netted enclosure.

of linear, roll, and pitch motions in landing, yaw results are omitted from result plots.

In all tests, images from the camera were scaled down by one-half from their original resolution of 1920×1080 pixels, and were processed by the vision algorithm at a resolution of 960×540 pixels. This was needed because feature extraction is much more computationally expensive at higher resolutions, causing the algorithm to update too slowly for real-time operation.

It is important to note that the vision algorithm calculates the pose of the landing pad image relative to the quadrotor camera. This means that the relative pose is affected by both the movement of the quadrotor and the landing pad. Vicon ground truth data is also provided here as a relative pose, which is extracted from the true poses of the quadrotor and landing target in the world frame.

2.5 Initial Algorithm Validation and Refinement

Before evaluating more complex scenarios, simple initial hand-held tests were completed to provide baseline performance and uncover basic limitations in the algorithm, such as time

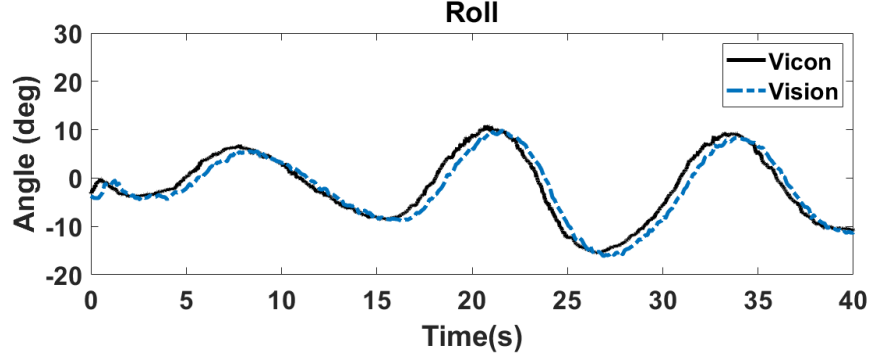


Figure 2.7: Example of vision algorithm result for large roll motions with uncorrected 400 millisecond time delay.

delays. Though the feature-based algorithm can function in visually-degraded scenarios, such conditions may potentially cause errors in the pose estimates from the algorithm. Therefore, this section also discusses techniques to refine the algorithm output.

2.5.1 Time delays and update frequency

From benchtop testing, a time delay of approximately 400 milliseconds was observed in the timestamps of logged pose estimates compared to ground truth. This was further verified by examining logged data from hand-held testing. The delay is caused by a combination of two factors: 1) the computational time needed to compute a pose estimate, and 2) latency in retrieving and publishing frames from the camera. The length of the delay may be reduced with improved hardware or through software optimizations, but cannot be entirely eliminated, and therefore must be accounted for during real-time tracking and landing.

As the time delay is consistent, the plotted results in this paper are shown with the pose estimate times shifted to remove the delay. This allows for better visual comparison in plots. Additionally, this allows us to quantitatively measure error using methods such as RMSE (root mean square error). Figure 2.7 shows an example of vision data with the delay present.

The vision algorithm is computationally expensive, resulting in long delays between updates. In these tests, the algorithm demonstrated an update speed of approximately 9-11 Hz with the deck in frame. This was found to be adequately fast to estimate ship-deck oscillation, which displayed a maximum frequency of 0.34 Hz in the Navy SCONE dataset [6]. However, the low frequency of algorithm updates make poor individual pose estimates problematic without additional filtering.

2.5.2 Baseline conditions

The algorithm's performance was evaluated under ideal, static conditions. This verified that the algorithm was producing the expected results and provides a baseline level of performance.

The test was conducted under normal indoor lighting (300 lux), and with the landing pad secured to a fixed, horizontal position on the floor. The quadrotor was held stationary above the landing pad, with the image fully in view of the camera, for several seconds.

The attitude and position estimation results are shown in Figures 2.8 and 2.9. The estimates of both linear position and attitude had minimal error. The final RMSE of linear position is below 1.3 cm on all axes. Roll, pitch, and yaw have an RMSE of 0.8, 1.4, and 1.5 degrees, respectively.

2.5.3 Reference image selection

The feature-based algorithm can be used with any generic reference image, as long as a sufficient number of features can be reliably detected and matched. This means that the existing markings on a ship-deck could potentially be used as a reference image. However, these markings have relatively few good features compared to a custom, detailed reference image.

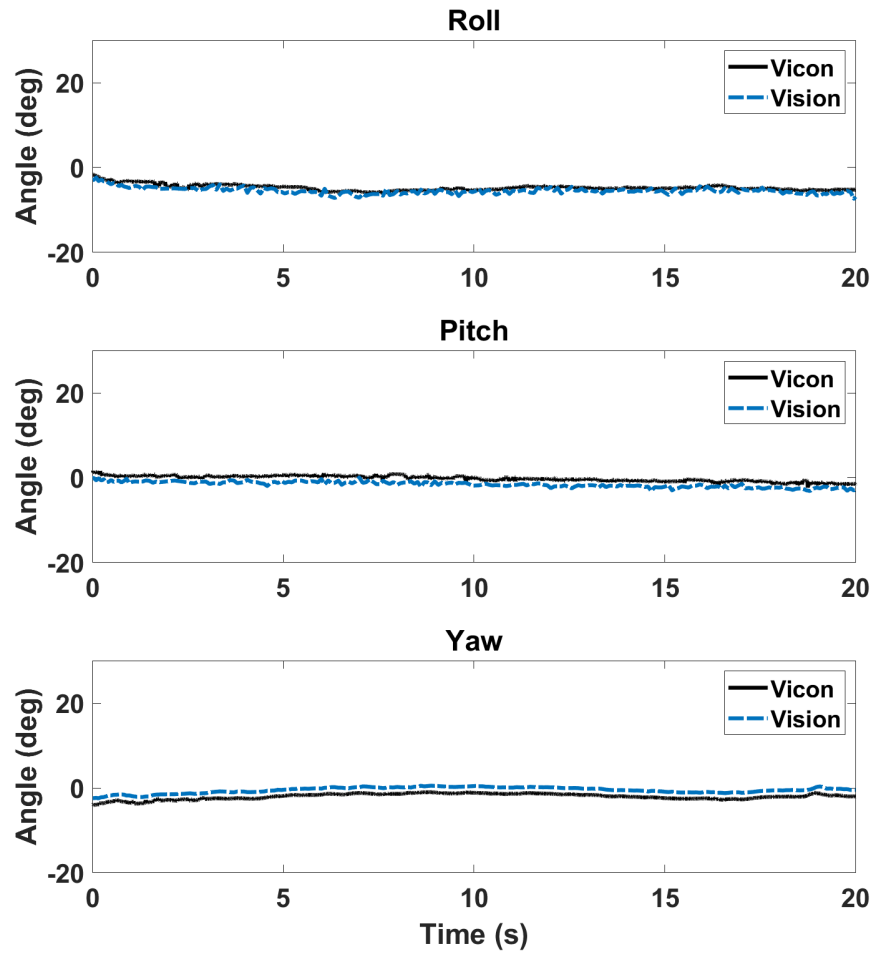


Figure 2.8: Estimation of aircraft attitude angles relative to a stationary landing marker from on-board camera.

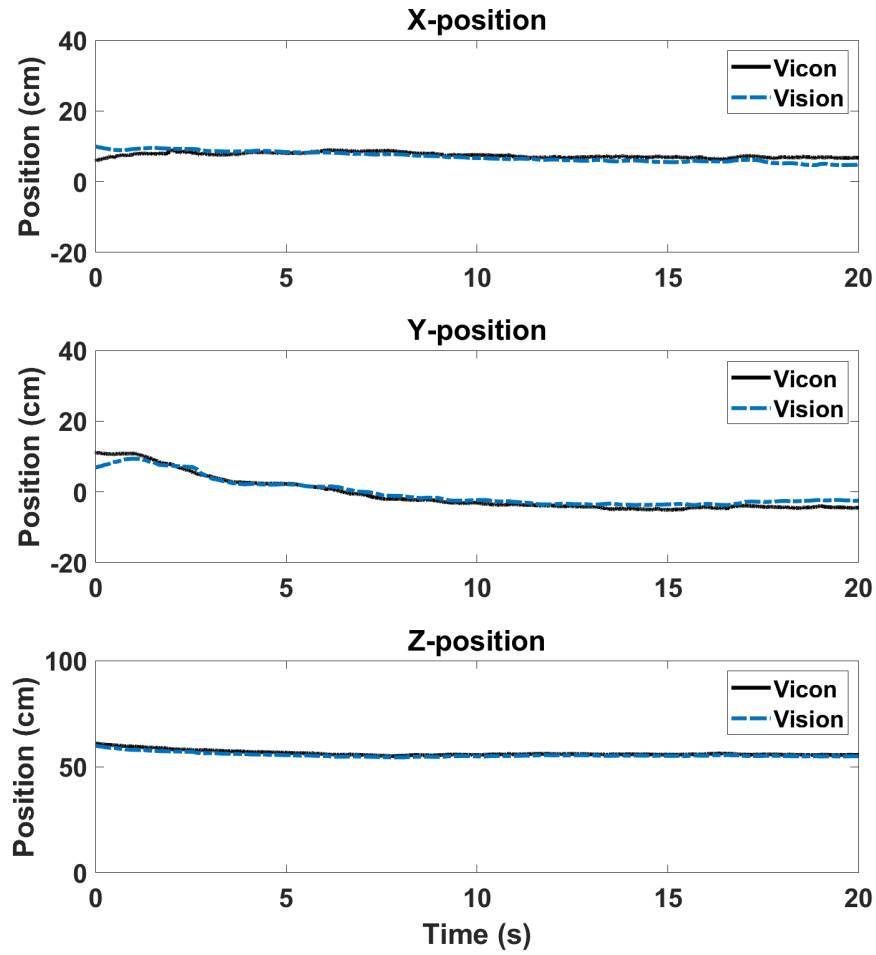
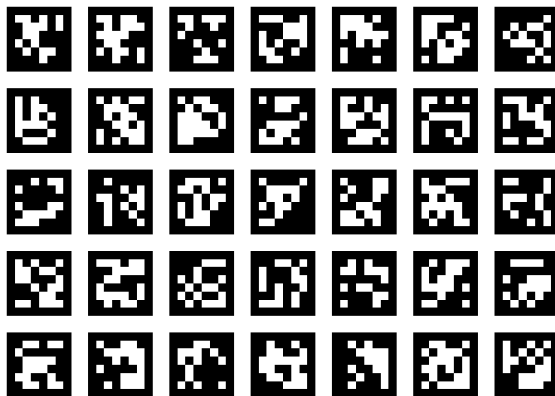
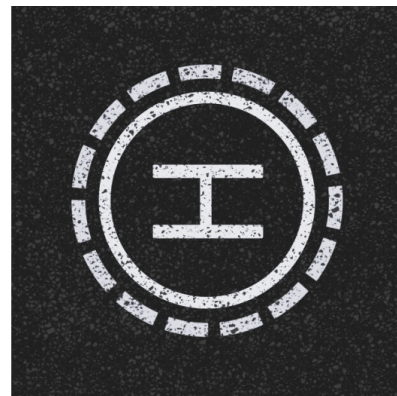


Figure 2.9: Estimation of aircraft position relative to a stationary landing marker from on-board camera.



(a) Detailed reference image composed of an AprilTag marker grid.



(b) Image representative of typical ship-deck landing pads.

Figure 2.10: Reference images evaluated using vision algorithm.

Algorithm performance was compared using two different reference images. The first, shown in Figure 2.10a, consists of a customized grid of AprilTag markers. This pattern was selected as it is rich in distinct, high-contrast features. The second image, shown in Figure 2.10b, is representative of typical helicopter landing pad (helipad) markings.

The test was conducted under normal indoor lighting (300 lux). The quadrotor was held stationary above a landing pad with linear and attitude motion, keeping the image fully within the camera frame. The same motion trajectory was used for both images for consistency.

Table 2.2: RMSE for different reference images.

Test case	X (cm)	Y (cm)	Z (cm)	Roll (deg)	Pitch (deg)
Customized grid	0.4	0.5	1.2	1.6	1.6
Helipad	1.6	2.0	2.0	2.2	5.0

The RMSE results, given in Table 2.2, indicate a significant increase in pose estimation accuracy with a detailed, customized landing pad image compared to a typical helipad marking. This difference can be explained with the greater number of strong, distinctive features in the detailed image compared to the typical landing pad. This results in a greater number of feature matches between the reference and camera image. With the customized image, an average of 369 feature matches were obtained in each update; however, for the helipad markings, only 40 matches were obtained on average.

These results show that, while it is feasible to use a feature matching algorithm with existing ship-deck markings, a customized image with many good features is ideal for good algorithm performance. The feature-rich image evaluated here is therefore used in all subsequent tests in this paper.

2.5.4 Feature match and inlier thresholds

Calculating a homography requires a minimum of 4 feature matches. However, with a small number of feature matches, incorrect matches can cause the calculated homography to be incorrect. When executing RANSAC, the number of inliers, or feature matches consistent with the homography, is calculated. The number of inliers is also useful to determine the quality of a pose estimate; a small number of inliers may indicate a poor result. The algorithm output could therefore be improved by implementing minimum thresholds for both the number of good feature matches and the number of inlier matches, and rejecting images that fall below these thresholds.

A vision algorithm test was completed to determine good match and inlier thresholds. Algorithm output was collected with the quadrotor held by hand above a stationary landing pad. The quadrotor was moved in quick, random motions in attitude and position, with the landing pad either fully or partially in view of the camera. For each pose estimate, the number of feature matches and inliers were logged. This was intended to generate pose estimates with varying numbers of matches and inliers.

The attitude and position results are shown in Figures [2.11](#) and [2.12](#). The raw pose estimates with no thresholds (Raw data) correspond to the output of the baseline vision algorithm [5]. These results are plotted alongside data which is then filtered to remove pose estimates falling below match and inlier thresholds (Thresholding). The baseline algorithm produces a number of inaccurate and noisy pose estimates, particularly in roll and pitch; this inaccuracy was also observed during experimental testing in visually degraded conditions [41]. However, these poor estimates are removed with the added thresholding. The ideal minimum threshold for feature matches was found to be 25, while the best threshold for inliers was found to be 17. These values

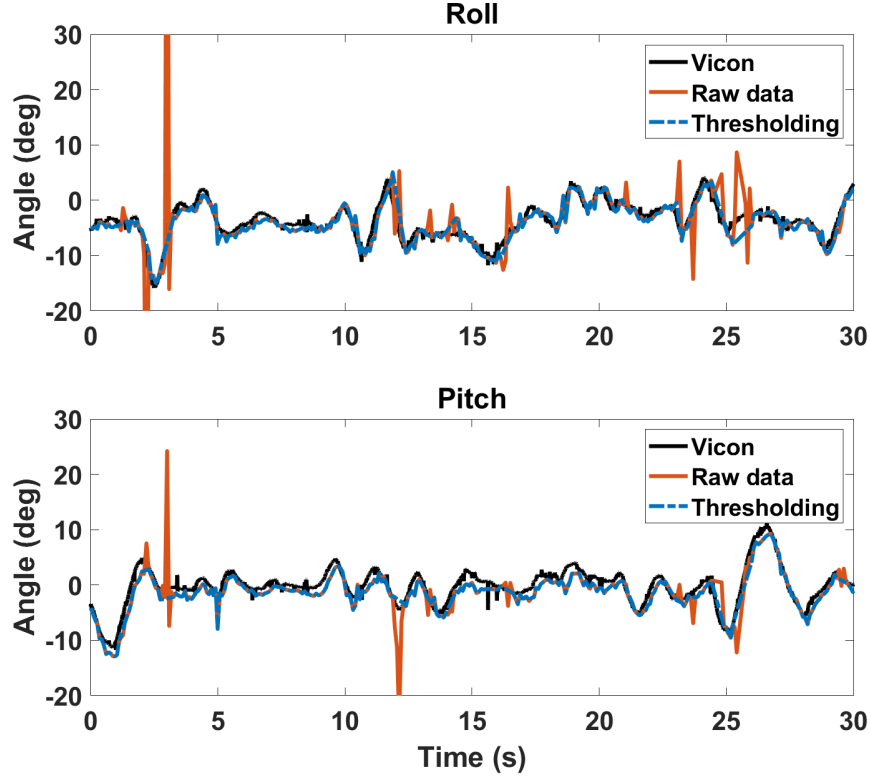


Figure 2.11: Estimation of aircraft attitude angles relative to a stationary landing marker from on-board camera. Results are compared with and without the application of feature match and inlier thresholding.

were chosen to reject a majority of inaccurate pose estimates while minimizing false rejections.

As shown in these results, applying feature match and inlier thresholds improves the pose estimates significantly, removing almost all of the poor estimates. These thresholds were incorporated into the vision algorithm, and were applied to all tests later in this paper.

2.5.5 Kalman filters

A Kalman filter was also integrated into the vision algorithm, providing filtered estimates of the ship-deck pose alongside the unfiltered estimates. The filter can mitigate potential noise and errors in the algorithm output with minimal computational cost.

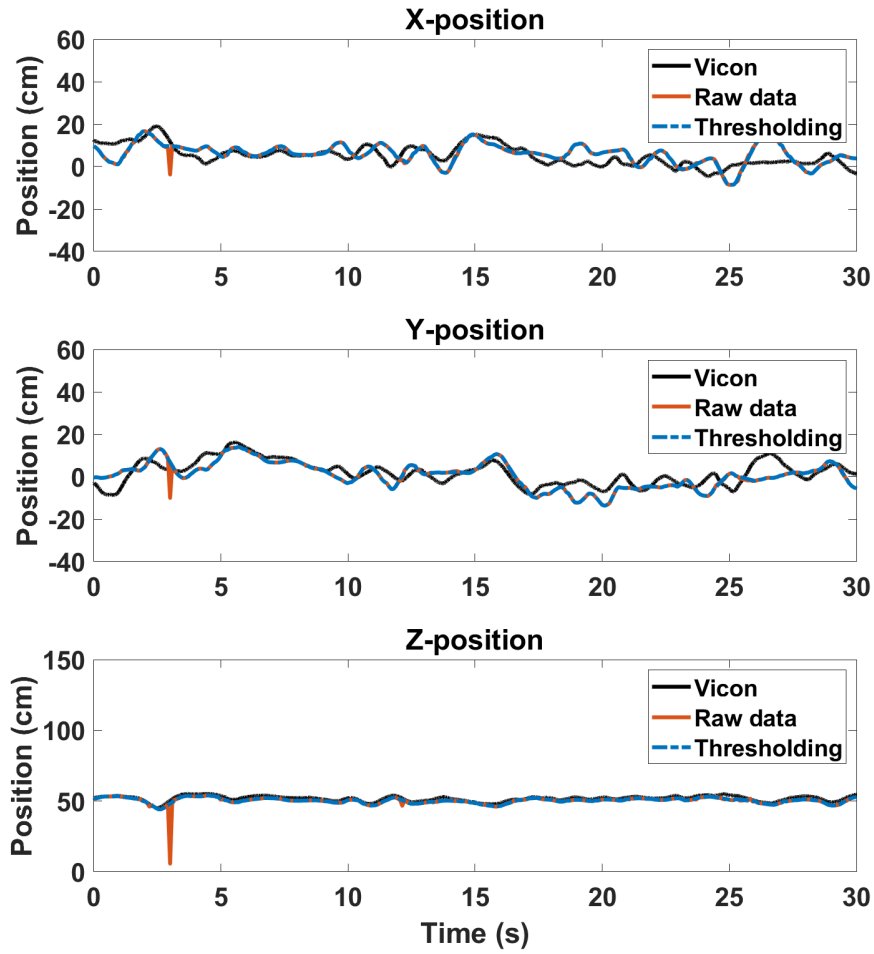


Figure 2.12: Estimation of aircraft position relative to a stationary landing marker from on-board camera. Results are compared with and without the application of feature match and inlier thresholding.

The logged data for linear position (X, Y, and Z position) was processed using a three-dimensional constant-velocity Kalman filter, shown below.

$$\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \\ y_{k+1} \\ \dot{y}_{k+1} \\ z_{k+1} \\ \dot{z}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \\ y_k \\ \dot{y}_k \\ z_k \\ \dot{z}_k \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 & 0 \\ \Delta t & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & \Delta t \end{bmatrix} w_k \quad (2.10)$$

where x_k , y_k , and z_k are the linear position at timestep k , and w_k is the process disturbance.

A separate Kalman filter, using the same constant-velocity model, was also used to process the estimated Euler angles (roll, pitch, and yaw).

The Kalman filter was integrated into the vision algorithm directly, filtering raw position and attitude estimates produced from ship-deck tracking. The code logs both raw and filtered pose estimates, allowing both outputs to be evaluated.

2.6 Hand-Held Tests

A series of tests were conducted to evaluate the feature-based algorithm's performance in various scenarios in a controlled manner. Two cases were investigated and are discussed below:

1) tracking ship-deck motion in ideal visual conditions; 2) tracking under visually degraded conditions. The quadrotor was held by hand above the landing pad. This setup removes the

effects of vibration and motion present in free flight, allowing evaluation of different scenarios without introducing additional difficulties. It also allows for exaggerated pitch and roll motions of the UAV to stress the robustness of the algorithm. Such motions would be impractical to recreate in free flight.

For all tests, the landing pad image was attached to a small Stewart platform (Acrome Stewart Pro) capable of generating linear and rotational motions, shown in Figure 2.13. The platform follows a motion trajectory representing Sea-state 6, obtained from ship-deck motion histories provided in the Navy SCONE database (Ref. [6]). The motion is stochastic and predominantly features large roll motions of up to 15 degrees, with smaller variations in pitch and linear position.

For clarity, only the vision algorithm output after match and inlier thresholding is shown in this section, labeled as 'Vision'. The Kalman filtered output is omitted.

2.6.1 Tracking ship-deck motion

In this section, the algorithm is tested for tracking motion in ideal visual conditions.

First, the quadrotor is held stationary above the moving Stewart platform, simulating a stable hover above the ship-deck. In this scenario, the algorithm can correctly follow the movement of the platform, estimating both attitude and position with minimal error, as shown in Figures 2.14 and 2.15.

Next, the quadrotor is moved with an arbitrarily varying pitch, roll, and linear motions above the moving platform. As the quadrotor was held by hand, it was possible to produce pitch motions of up to 15 degrees and roll motions exceeding 20 degrees. As shown in the results in Figure 2.16, the algorithm is able to accurately follow complex variations of pitch and roll. The

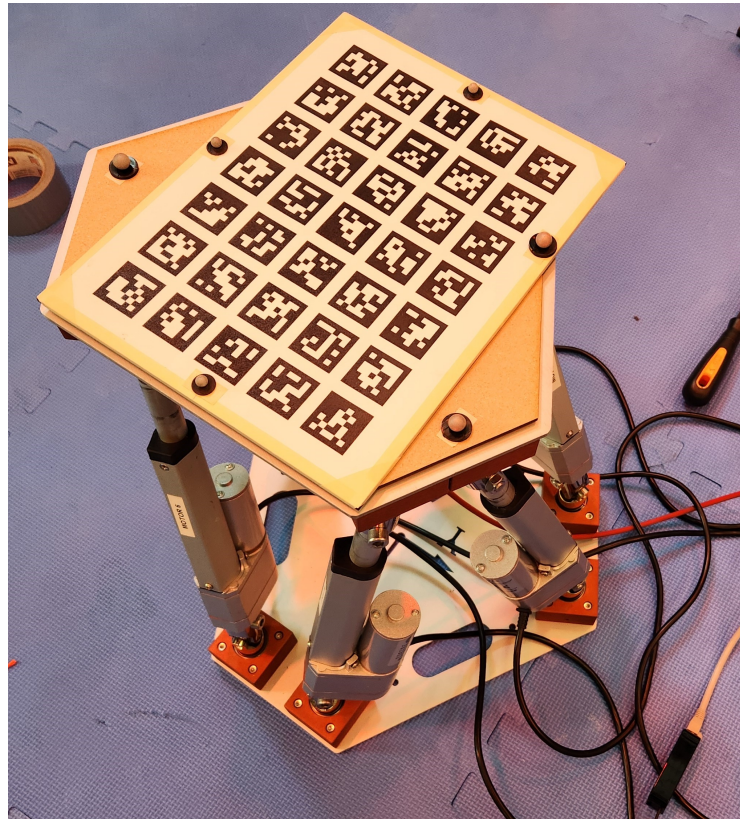


Figure 2.13: Small Stewart platform with attached 24×17 cm landing pad image.

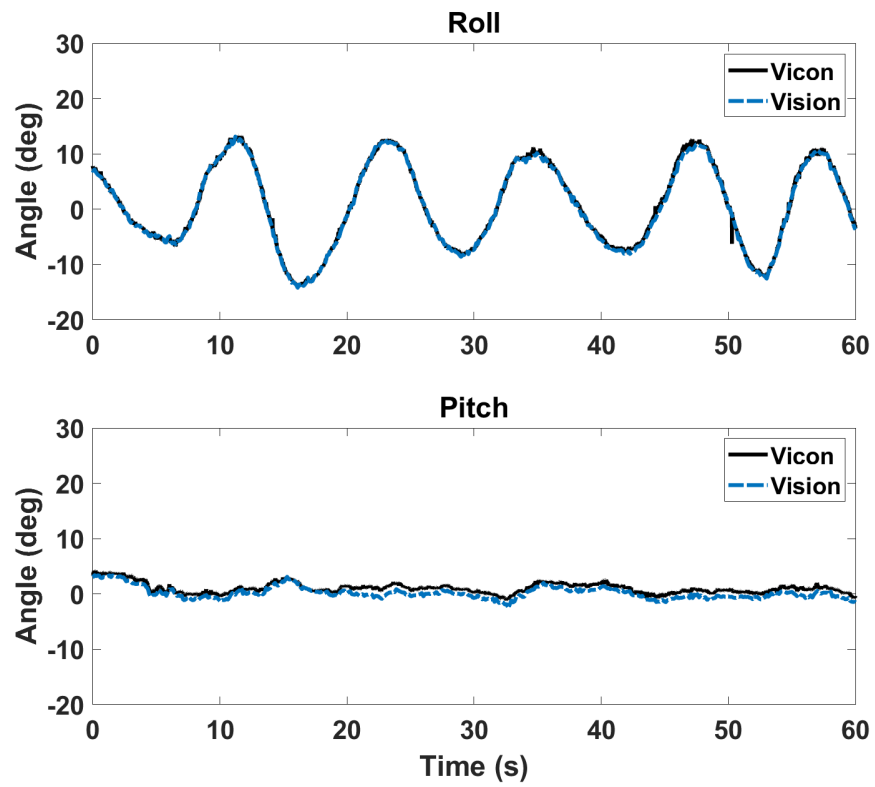


Figure 2.14: Estimation of aircraft attitude angles relative to a moving Stewart platform from on-board camera, with aircraft held stationary.

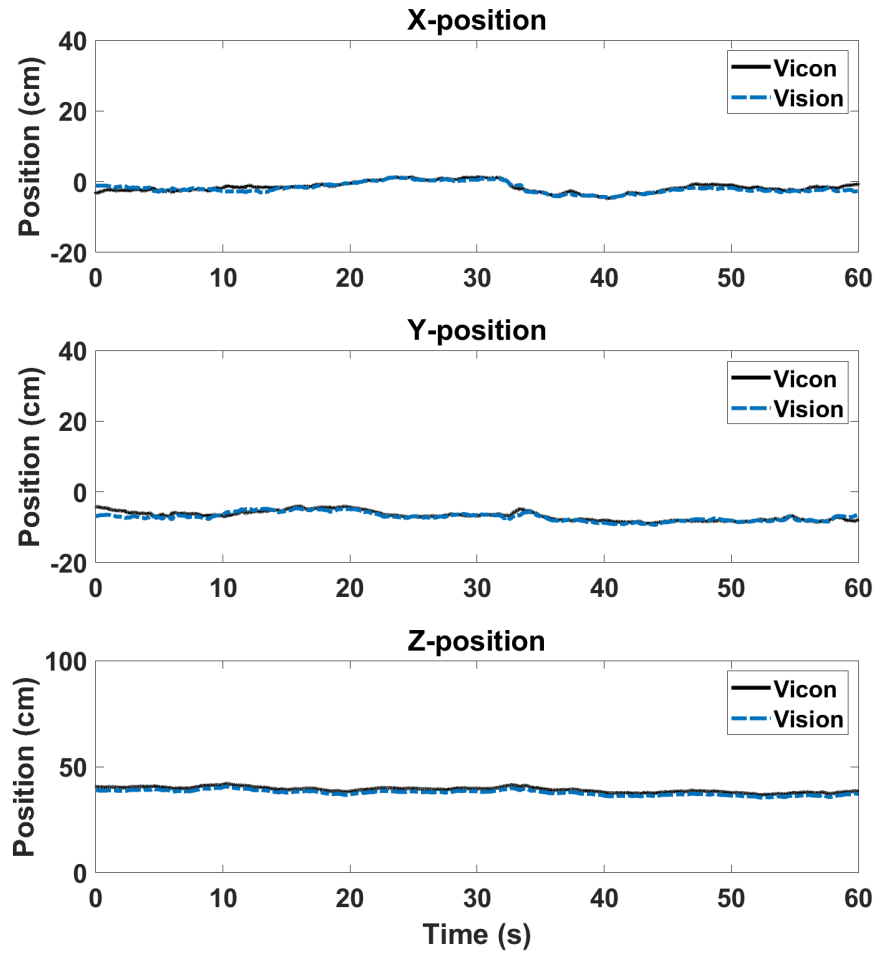


Figure 2.15: Estimation of aircraft position relative to a moving Stewart platform from on-board camera, with aircraft held stationary.

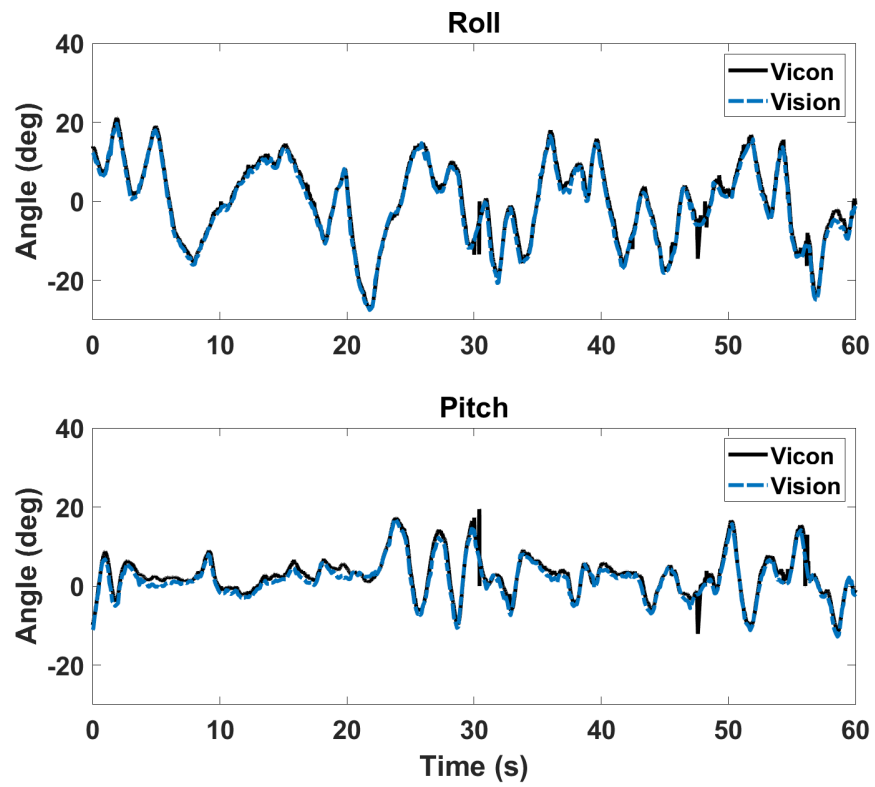


Figure 2.16: Estimation of aircraft attitude angles relative to a moving Stewart platform from on-board camera, with aircraft also moving.

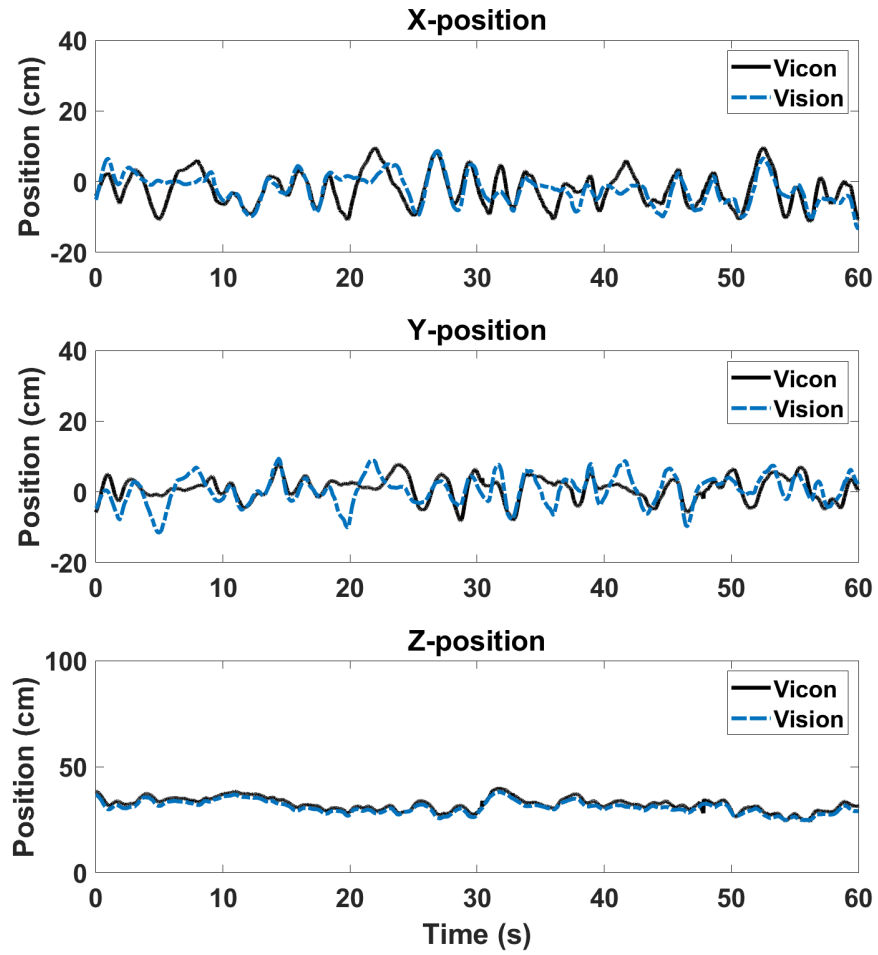


Figure 2.17: Estimation of aircraft position relative to a moving Stewart platform from on-board camera, with aircraft also moving.

results for Z-position are again very precise, as indicated in the results in Figure 2.17. However, some error appears in the estimates of X- and Y-position.

The RMSE values, given in Table 2.3, provide a quantitative measure of algorithm performance in these scenarios. For ship-deck motion alone, the errors in pose estimation are very low. However, these errors increase with significant motion of both the ship-deck and quadrotor. During an actual landing, the quadrotor waits for a quiescent period when the deck's orientation and rate of change are small. As observed from the results, the algorithm is able to estimate orientation satisfactorily. In contrast, the estimate of linear position is used in the quadrotor's control loop to track the deck, in order to remain above the target for landing. The position errors from the logged data are small compared to the quadrotor size; additionally the quadrotor will land on a much larger deck than the one tested in these experiments. As a result, these errors are unlikely to interfere significantly with landing.

Table 2.3: RMSE for ship-deck motion.

Test case	X (cm)	Y (cm)	Z (cm)	Roll (deg)	Pitch (deg)	Yaw (deg)
Quadrotor stationary	0.6	0.5	1.4	0.5	1.0	1.3
Quadrotor moving	3.5	3.5	1.5	1.3	1.3	1.5

2.6.2 Visually degraded conditions

A critical advantage of the feature-based algorithm is that it offers robustness to occlusion or varying lighting conditions, where the fiducial algorithm will fail outright. A series of tests were completed to analyze performance under visually challenging conditions. In all tests, the quadrotor was held stationary above the moving Stewart platform. Three specific cases were

evaluated: 1) low illumination; 2) glare; and 3) occlusion. The errors for each case are given in Table 2.4; each case is described in more detail below.

Table 2.4: RMSE for visually degraded test cases.

Test case	X (cm)	Y (cm)	Z (cm)	Roll (deg)	Pitch (deg)	Yaw (deg)
Low illumination	1.5	1.2	1.4	0.8	1.5	1.3
Glare	1.5	1.2	1.1	0.6	1.4	1.3
Occlusion (regular)	1.4	1.4	1.3	1.1	1.2	1.3
Occlusion (irregular)	1.5	1.9	1.7	1.7	2.0	1.7
Increased camera distance	1.4	1.2	1.2	1.9	1.9	1.4

The first test evaluates the algorithm in low illumination conditions. Dark indoor lighting, approximately 10 lux, was used for this test case. The results of this test are provided in Figures 2.18 and 2.19. The pose estimates are not significantly degraded by the low illumination condition, as shown by both the quantitative error calculations and the close match in the plots between the estimated and ground truth poses.

The second test evaluates the algorithm under glare. A 900 lumen flashlight was mounted above the landing pad, creating significant glare on the image, as shown in Figure 2.20. This setup is intended to simulate glare, such as when bright lights or reflections are present on a real ship-deck. The test was also conducted under low illumination (10 lux) to increase the effect of this glare. The resulting attitude and position estimates are provided in Figure 2.21 and 2.22. The estimates of attitude and position again closely follow the ground truth, indicating that the algorithm can perform acceptably in this condition. The RMSE results show comparable error to the low illumination case.

Finally, tests were completed to observe the performance of the algorithm for a landing

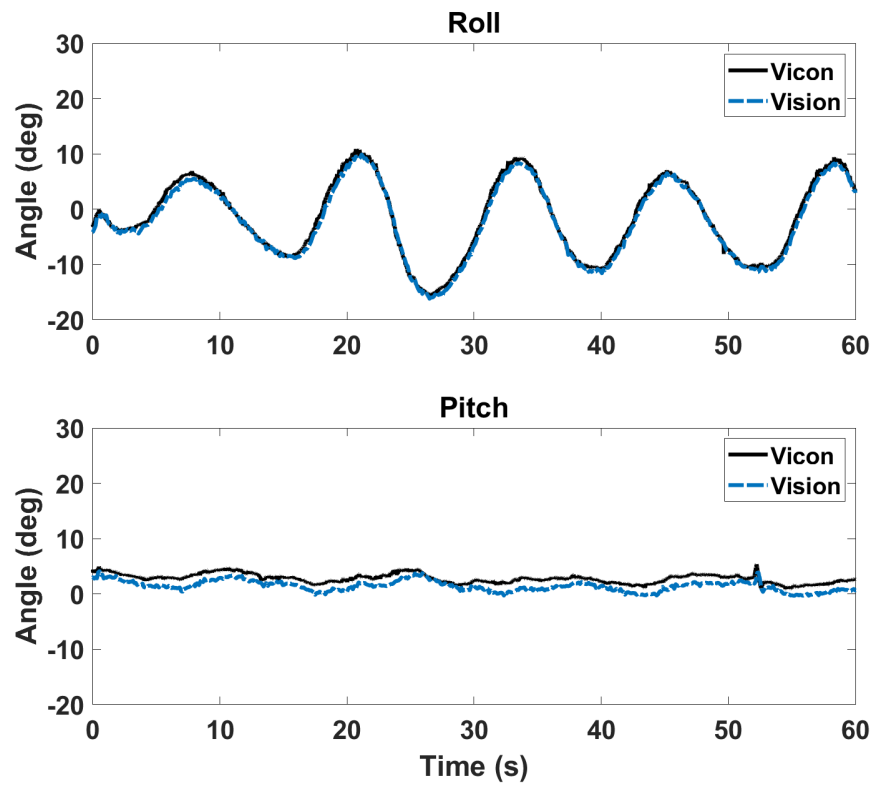


Figure 2.18: Estimation of aircraft attitude angles relative to a landing pad undergoing ship-deck motion under low (10 lux) illumination.

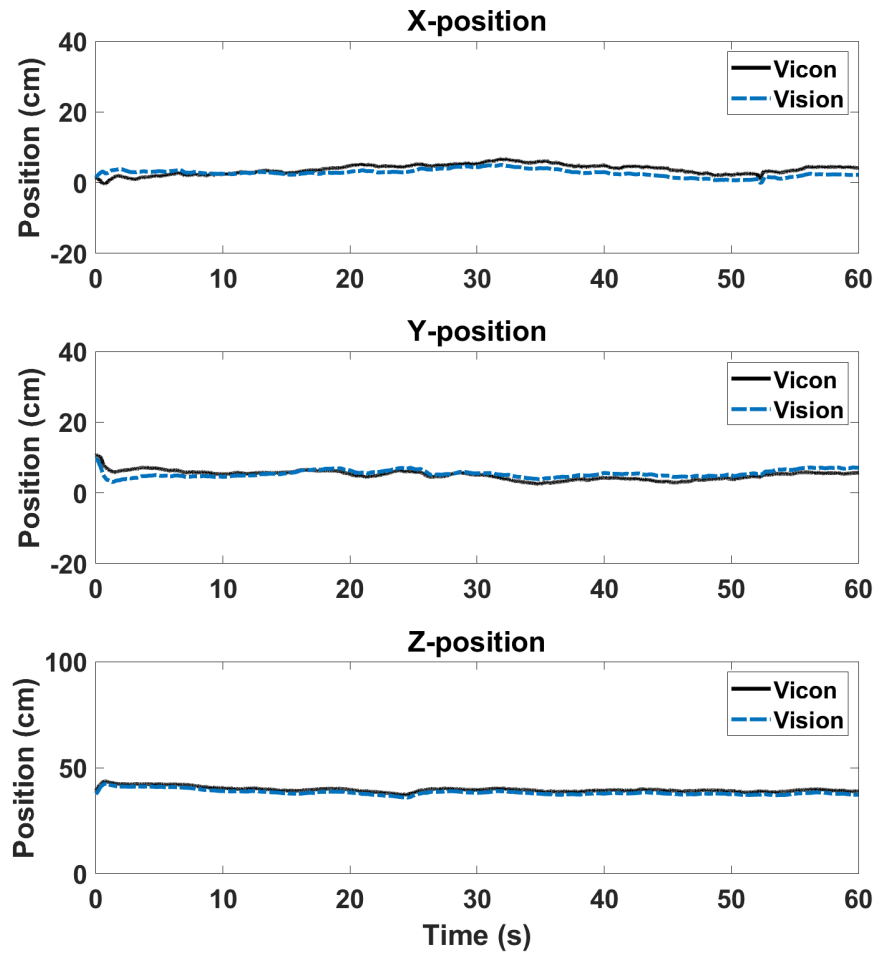


Figure 2.19: Estimation of aircraft attitude angles relative to a landing pad undergoing ship-deck motion under low (10 lux) illumination.

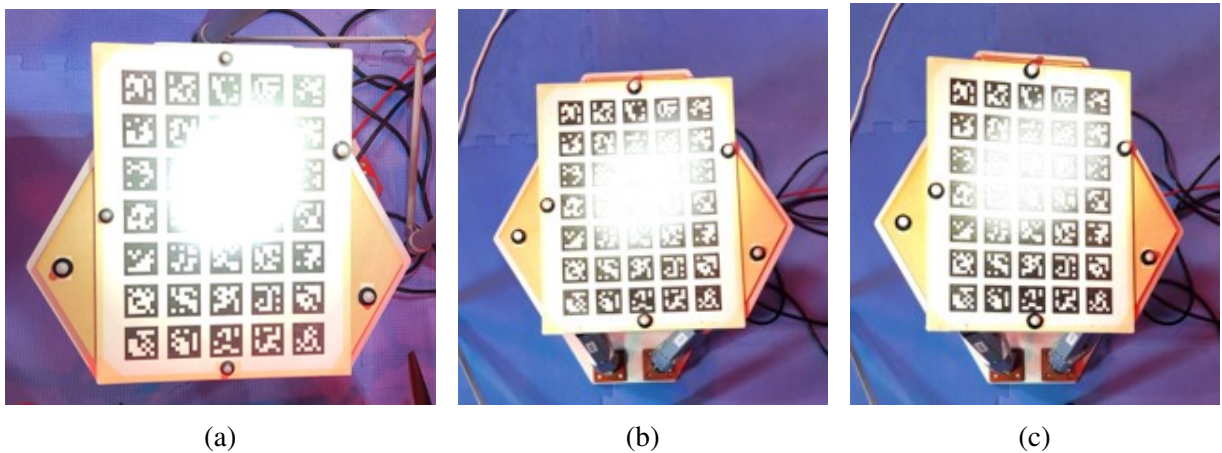


Figure 2.20: Landing pad image with significant glare created by a 900 lumen light; change in glare pattern due to platform motion during a typical test is illustrated.

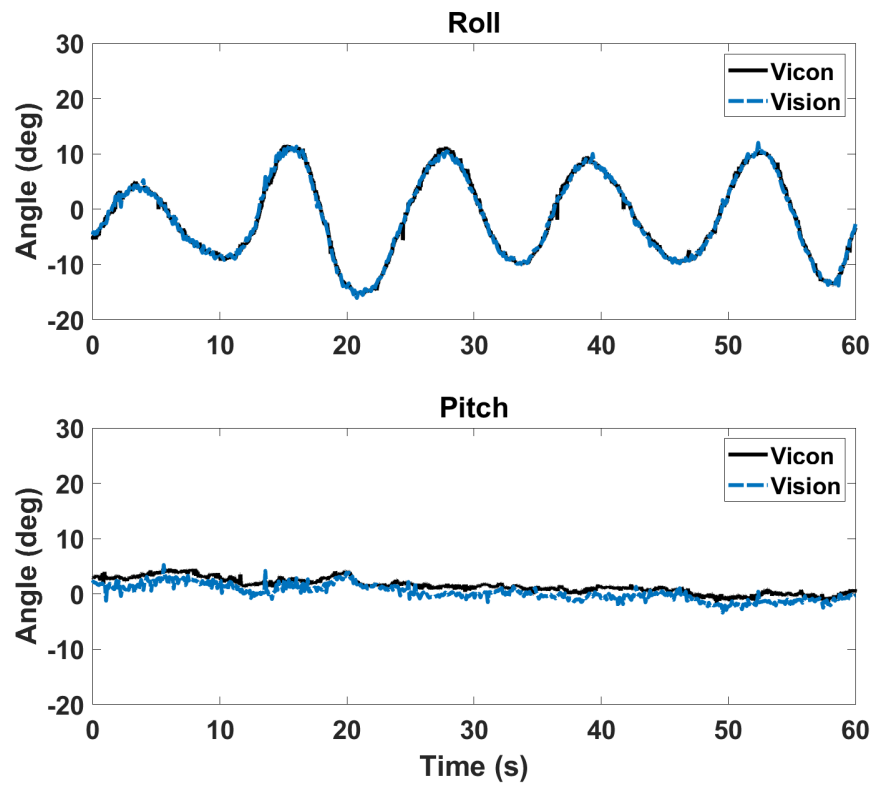


Figure 2.21: Estimation of aircraft attitude angles relative to a landing pad undergoing ship-deck motion from on-board camera, with a 900 lumen light used to simulate glare.

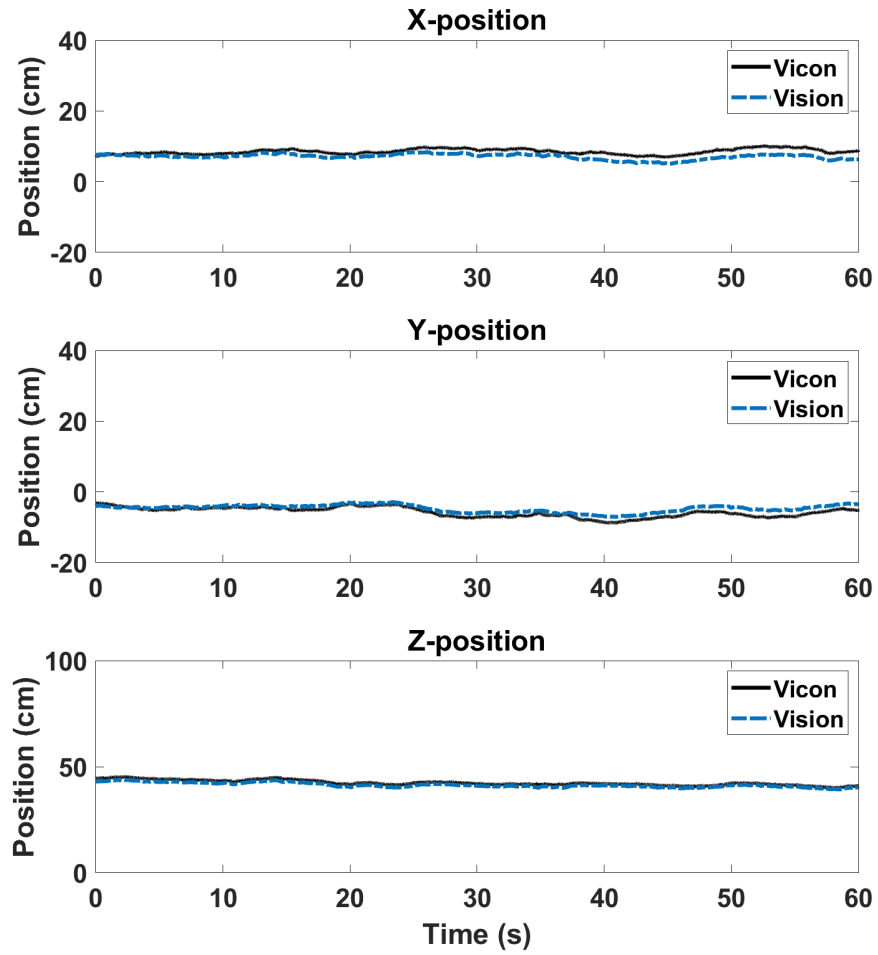
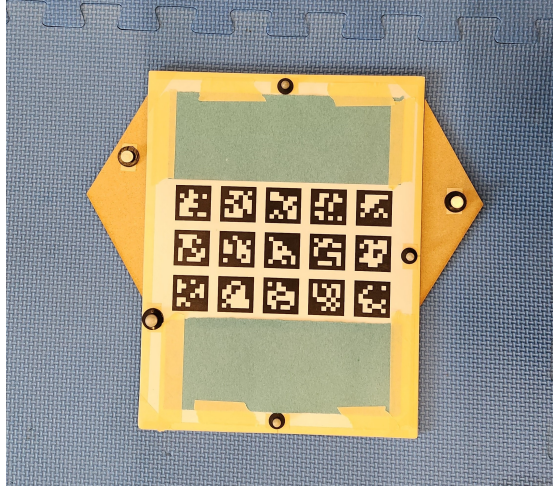
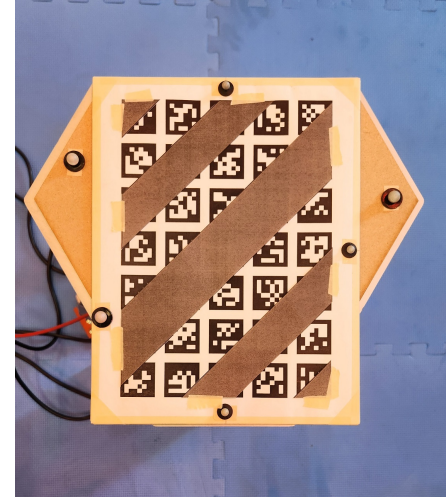


Figure 2.22: Estimation of aircraft position relative to a landing pad undergoing ship-deck motion from on-board camera, with a 900 lumen light used to simulate glare.



(a) Occlusions uniformly covering 20 of the 35 tags, or 57% of the landing pad image.



(b) Irregular occlusions partially or fully covering all tags. These occlusions cover 49% of the image.

Figure 2.23: Occluded landing pad images used to evaluate algorithm performance.

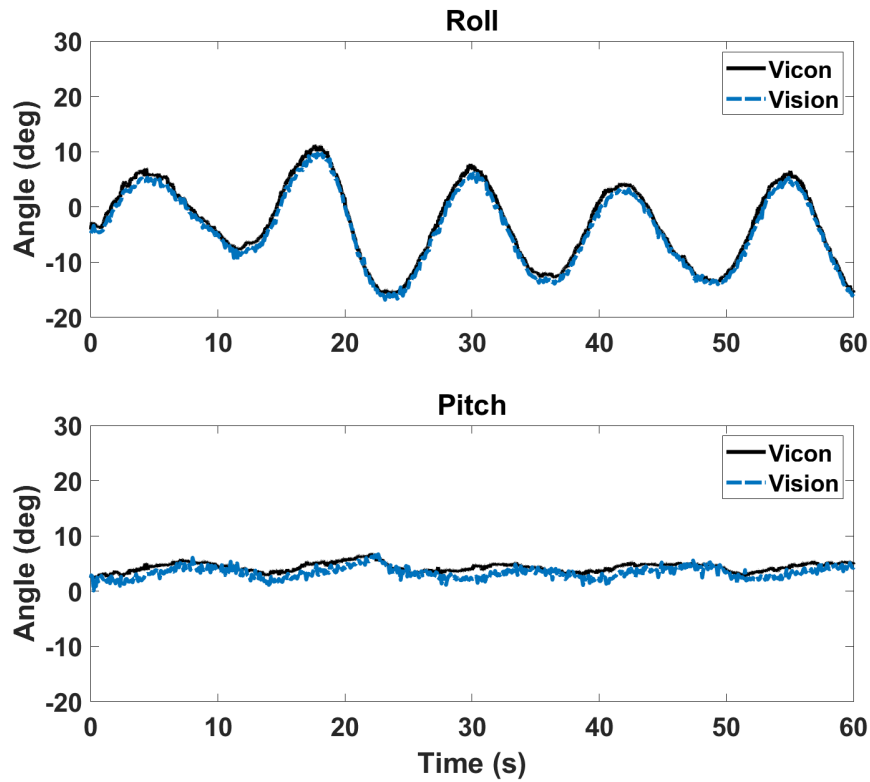


Figure 2.24: Estimation of aircraft attitude angles relative to an occluded landing pad undergoing ship-deck motion. The occlusions uniformly cover 57% of the marker patterns on the landing pad image.

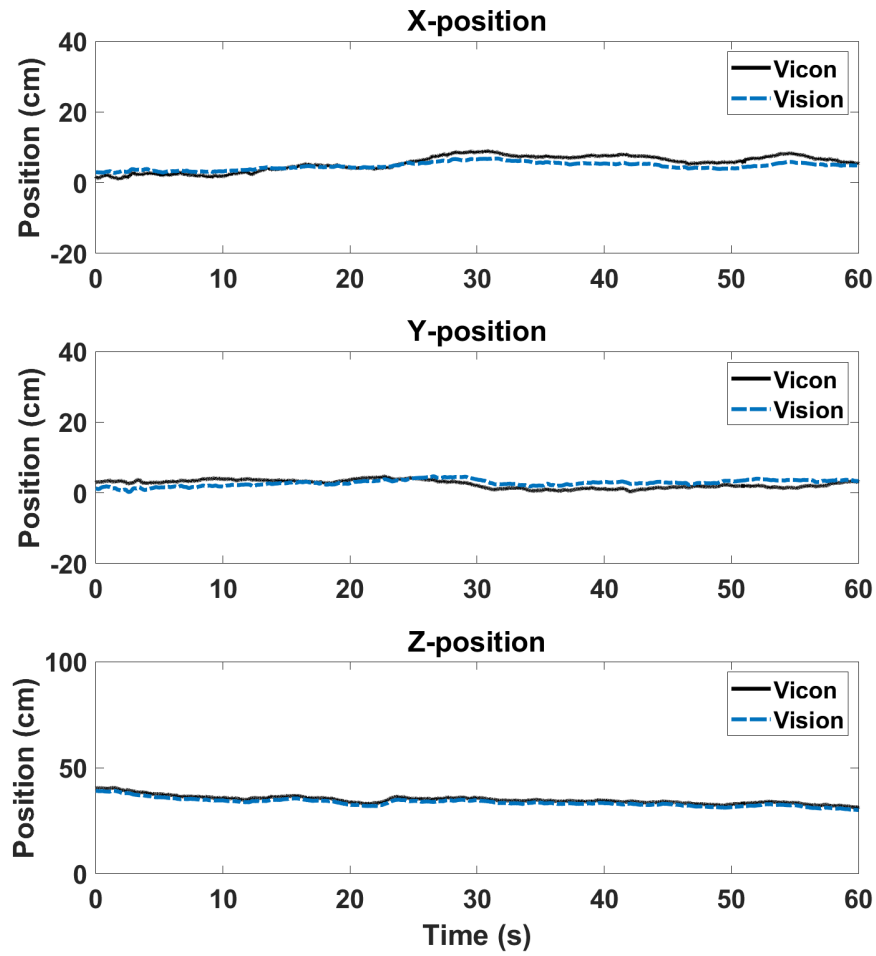


Figure 2.25: Estimation of aircraft position relative to an occluded landing pad undergoing ship-deck motion. The occlusions uniformly cover 57% of the marker patterns on the landing pad image.

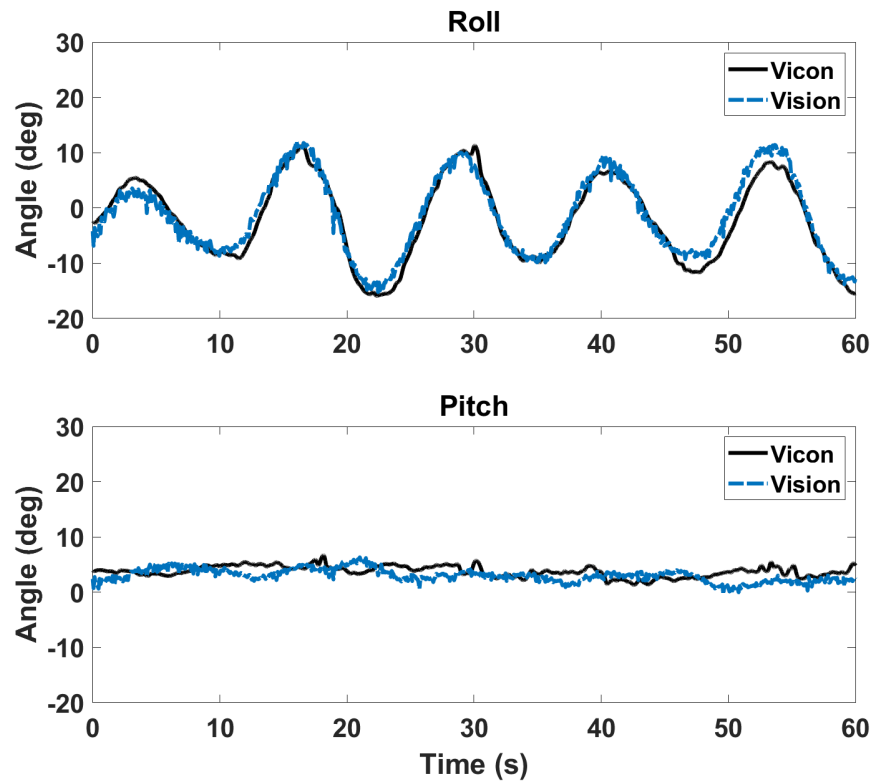


Figure 2.26: Estimation of aircraft attitude angles relative to an occluded landing pad undergoing ship-deck motion. Irregular occlusions partially or fully cover all marker patterns on the landing pad image.

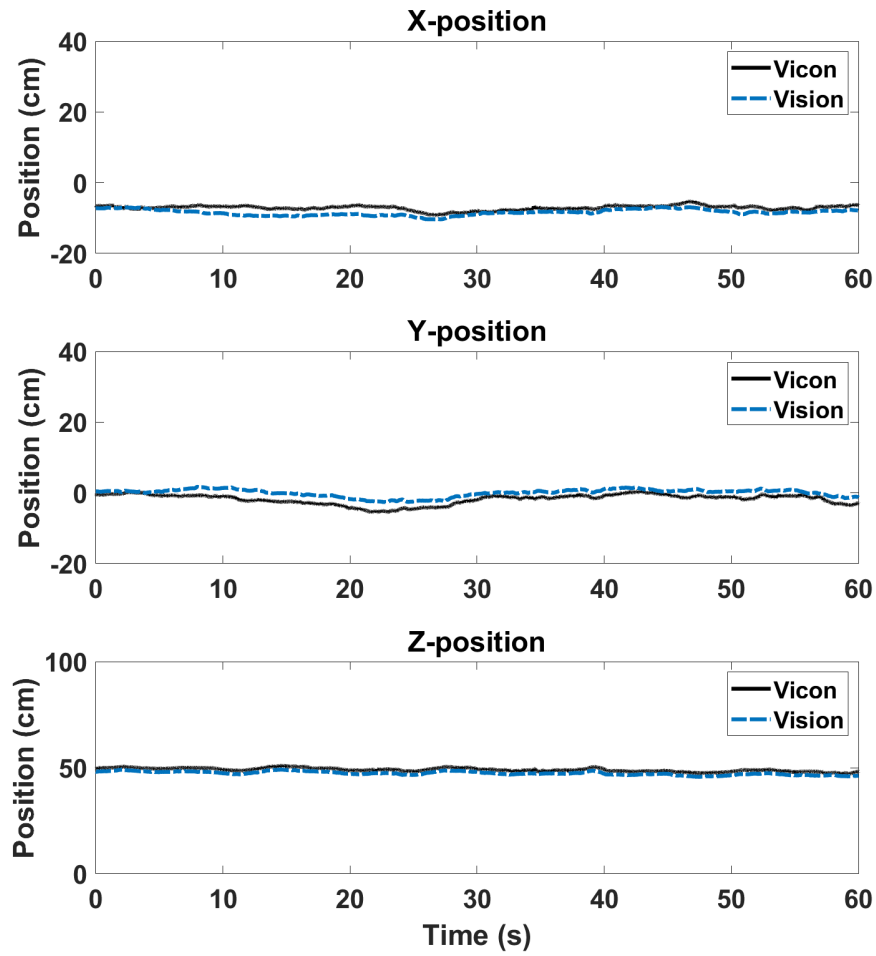


Figure 2.27: Estimation of aircraft position relative to an occluded landing pad undergoing ship-deck motion. Irregular occlusions partially or fully cover all marker patterns on the landing pad image.

pad under significant occlusion. Paper coverings are placed to occlude a portion of the image, as shown in Figure 2.23. Two test cases were evaluated: 1) regular occlusion, and 2) irregular occlusion. The tests were conducted in normal indoor lighting (300 lux), to avoid confusing the effects of occlusion and low illumination.

For the first case, shown in Figure 2.23a, 20 of the 35 tags (57% of the image) are uniformly covered. In the second case shown in Figure 2.23b, a number of shapes are arranged in a diagonal pattern, partially or fully occluding all of the tags on the image. These shapes cover approximately 49% of the image. Note that, due to the partial obstruction of all tags, a fiducial algorithm will fail outright under this condition.

The results of the first case are shown in Figures 2.24 and 2.25. As in the previous two cases, the results accurately follow the ground truth data, and the RMSE suggests similar levels of error to the previous cases. The results for the second case are given in Figures 2.26 and 2.27. The estimates of roll and pitch are noisier than in the previous case, and, while still accurate overall, display increased error compared to ground truth. The RMSE results also show increased error for both attitude and position estimates compared to the previous case.

2.6.3 Increasing camera distance

For the previous hand-held tests, the camera was held at a vertical distance of approximately 40 cm from the landing pad. This condition simulates the terminal landing phase, in which the aircraft is hovering closely above the ship-deck. However, the effect of increasing this distance is also of interest, as the feature-matching approach may also be used during earlier landing phases in which the camera is further from the landing pad. Therefore, a test was completed to

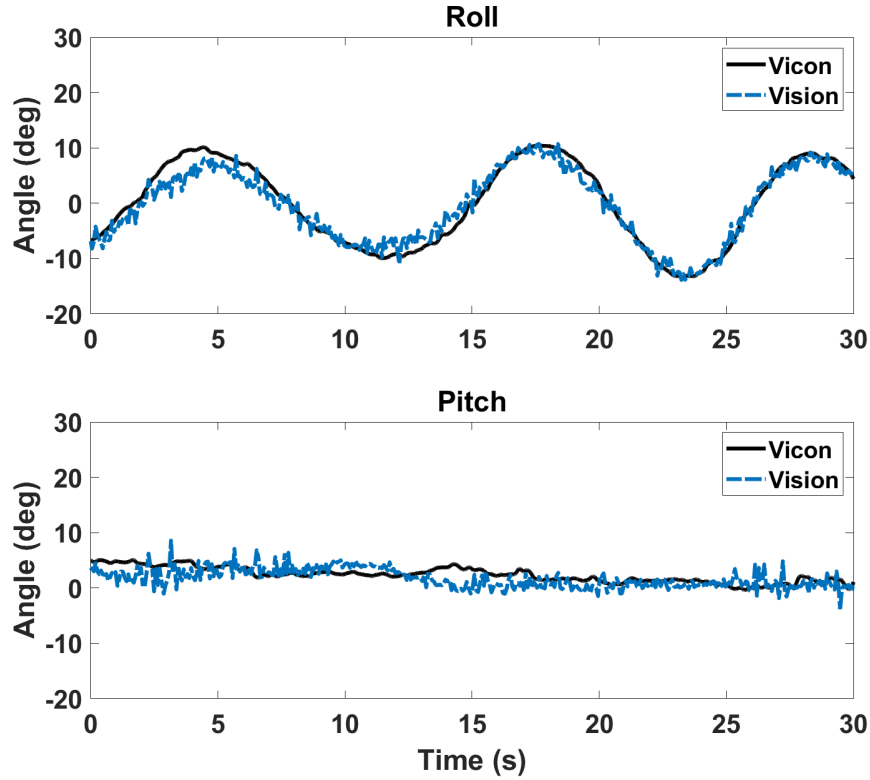


Figure 2.28: Estimation of aircraft attitude angles relative to a landing pad undergoing ship-deck motion, with Z-axis (vertical) distance increased from previous tests.

demonstrate the effect of increasing the camera distance from the landing pad image. For this test, the vertical (Z-axis) distance was increased to approximately 85 cm, double the distance for the previous tests.

The results of this test are provided in Figures 2.28 and 2.29. The attitude estimation results are accurate overall, but show increased noise compared to previous tests. The position results follow the ground truth closely. The RMSE results further confirm these observations; the position estimation error remains low, but the error in attitude estimation is similar to that of the irregular occlusion case.

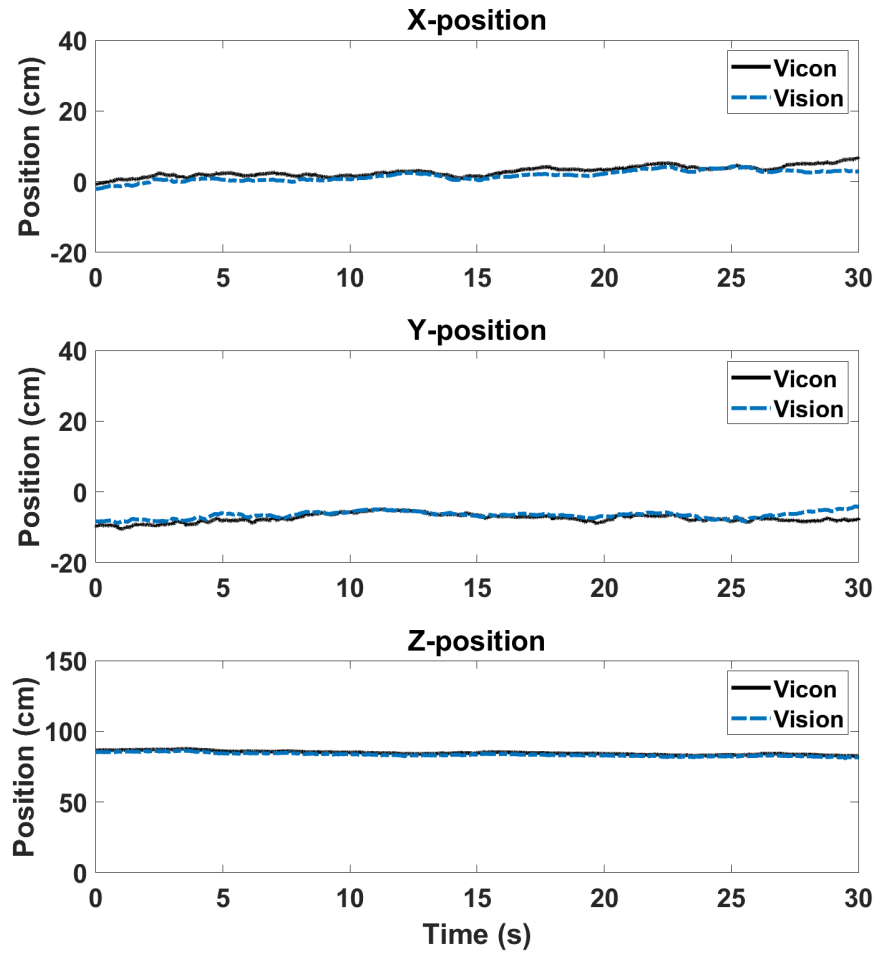


Figure 2.29: Estimation of aircraft position relative to a landing pad undergoing ship-deck motion, with Z-axis (vertical) distance increased from previous tests.

2.6.4 Feature match and inlier summary

As the count of feature matches and inliers has a significant effect on the quality of pose estimates, analyzing this data for different test cases can provide additional insight. Mean counts of feature matches and inliers are provided in Table 2.5.

Table 2.5: Summary of mean feature match and inlier counts for different test cases.

Test case	Mean feature matches	Mean inliers
Quadrotor stationary	346	302
Quadrotor moving	252	197
Low illumination	360	317
Glare	154	127
Occlusion (regular)	199	161
Occlusion (irregular)	65	50
Increased camera distance	44	37

From this data, a general correlation can be observed between attitude estimation error and feature match and inlier counts. In cases with low numbers of feature matches, specifically the irregular occlusion and increased distance cases, the RMSE for roll and pitch is notably larger than the other cases.

However, a similar correlation is not apparent for position estimates. The RMSE for linear position is comparable for all visually degraded test cases, excluding the irregular occlusion case. This is despite the significant variation in mean number of feature matches. The error in linear position is much greater for the quadrotor moving case. For this case, the large-angle motions introduced factors such as motion blur, large out-of-plane rotations, and the landing image partially moving outside the camera field of view. This indicates that these conditions

have a major effect on the accuracy of position estimates. The error also increases modestly in the irregular occlusion case. This suggests that occlusions interfering with larger patterns on the landing image, and by extension larger features, may have a moderate effect on position estimation error.

2.7 Free-Flight Hover Tests

The algorithm was next validated during hovering flight. The quadrotor was manually piloted through takeoff, hover above the image, and landing. Algorithm output was logged during the entire flight for each test. Only the hover segment is presented here since the goal of these experiments is to evaluate performance in hover.

2.7.1 Baseline conditions

Results were initially collected for ideal visual conditions, indoor lighting (300 lux) and no occlusion of the pad. For this test, the landing image was secured to the ground so no deck motions were present. The results of this test are shown in Figures [2.30](#) and [2.31](#). The estimated pose, both attitude and position, closely follow the ground truth. Quantitatively, this data shows an RMSE of 2 cm in each linear axis, and an RMSE of 2.3 degrees and 1.3 degrees in pitch and roll, respectively. While some minor noise is present in the raw estimates for pitch and roll due to quadrotor vibration as well as the increase in Z-position compared to the hand-held test condition, this noise is mostly mitigated by the Kalman filter. This indicates that, under ideal visual conditions, vibration and small motions from actual quadrotor hover have a minimal effect on the quality of algorithm output.

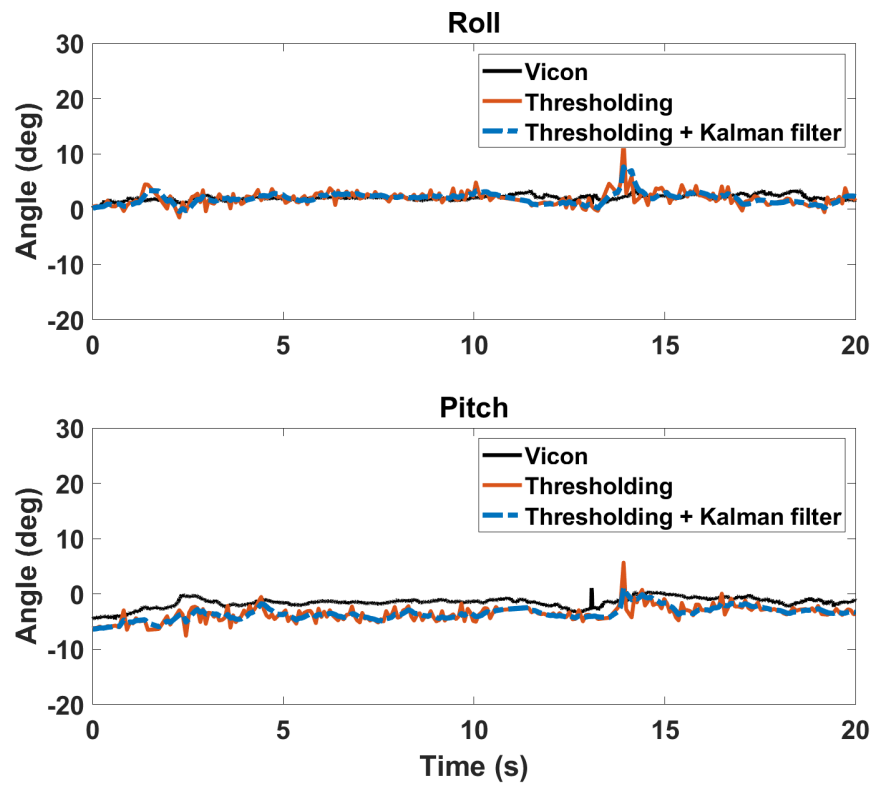


Figure 2.30: Estimation of hovering aircraft attitude angles relative to a stationary landing marker in indoor illumination (300 lux).

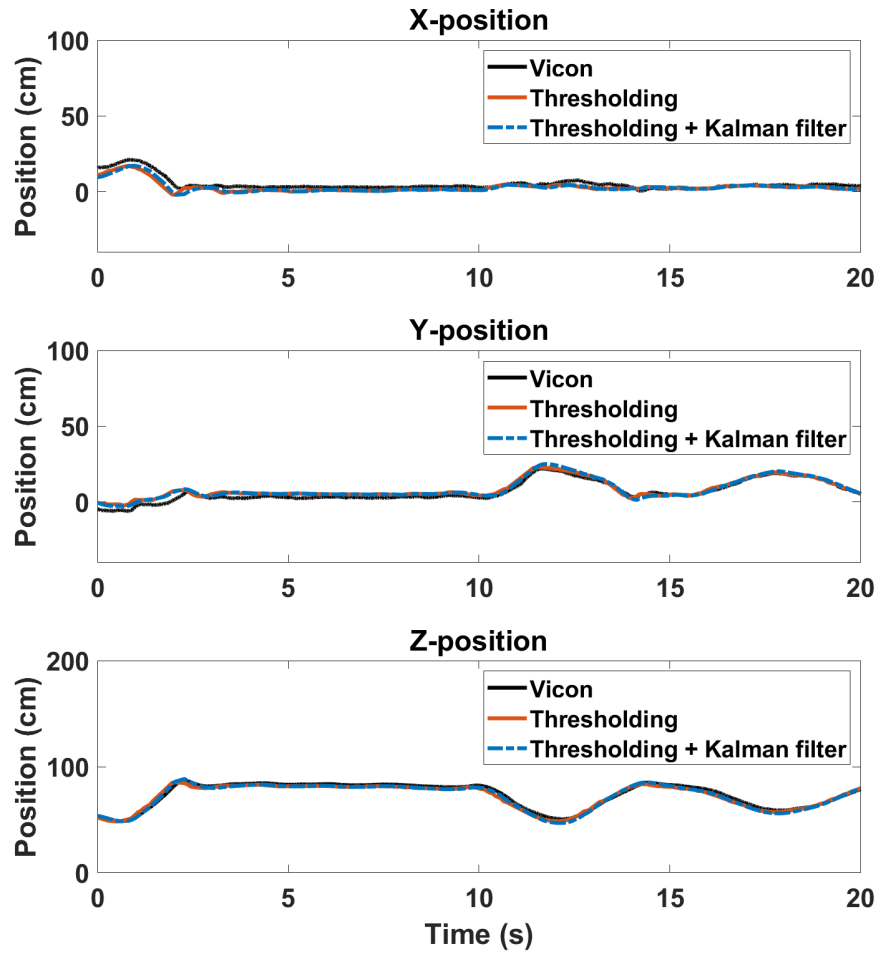


Figure 2.31: Estimation of hovering aircraft position relative to a stationary landing marker in indoor illumination (300 lux).

2.7.2 Visually degraded conditions

Next, visually degraded conditions were introduced. Occlusion and reduced illumination were considered, as these conditions were the most challenging in the controlled hand-held test cases earlier. As in the previous test, the landing pad was fixed to the ground. The quantitative RMSE results for all cases are given in Table 2.6.

Table 2.6: RMSE for free flight in visually degraded conditions.

Test case	X (cm)	Y (cm)	Z (cm)	Roll (deg)	Pitch (deg)
Low illumination only	3.0	3.0	1.7	2.0	1.2
Occlusion only	0.9	1.0	2.1	1.1	1.2
Low illumination and occlusion	1.3	1.3	3.0	2.2	2.0

A test was first completed in dark indoor illumination (10 lux) for a non-occluded landing pad. These results are shown in Figures 2.32 and 2.33. Additional tests were completed for an occluded landing pad in indoor illumination (300 lux), shown in Figures 2.34 and 2.35, and dark indoor illumination (10 lux), shown in Figures 2.36 and 2.37. The landing pad was occluded by covering 57% of the tags on the image.

For all cases, the estimated position closely follows the ground truth position, showing no significant change in performance with visually degraded conditions.

For occlusion in indoor lighting, some noise is present in the raw estimates of roll and pitch, though this noise is mostly mitigated by the Kalman filter. With a combination of low illumination and occlusion, an RMSE of 2.2 degrees in pitch and 2.0 degrees in roll is observed, higher than with occlusion or low illumination alone. However, some individual attitude estimates show error exceeding 10 degrees. Errors such as this could cause the controller to falsely reject a period of

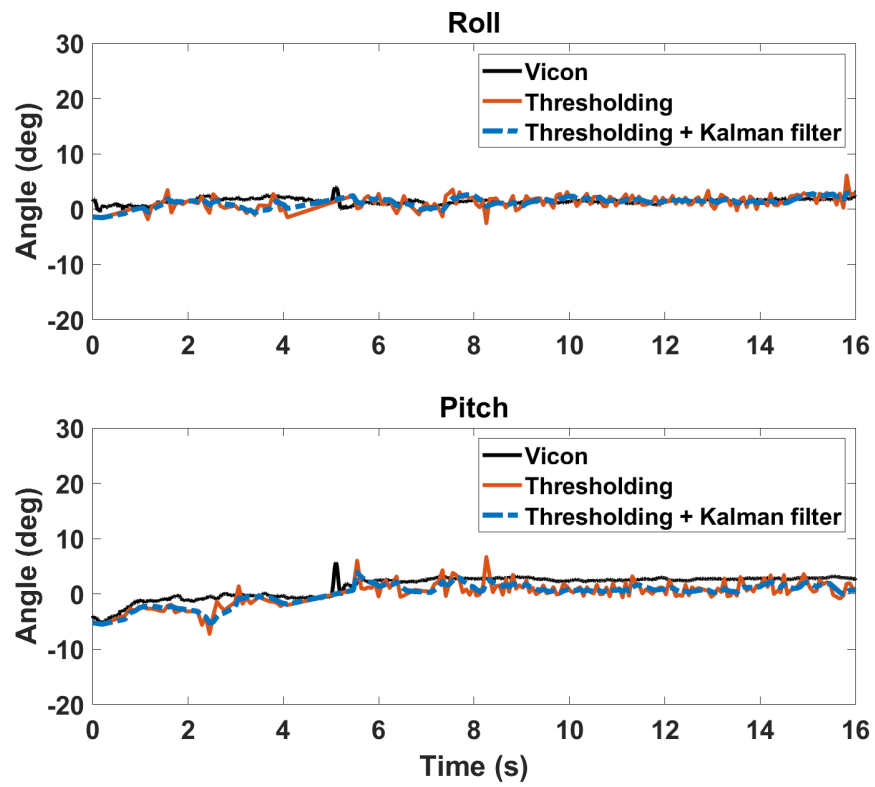


Figure 2.32: Estimation of hovering aircraft attitude angles relative to a stationary landing marker in low illumination (10 lux).

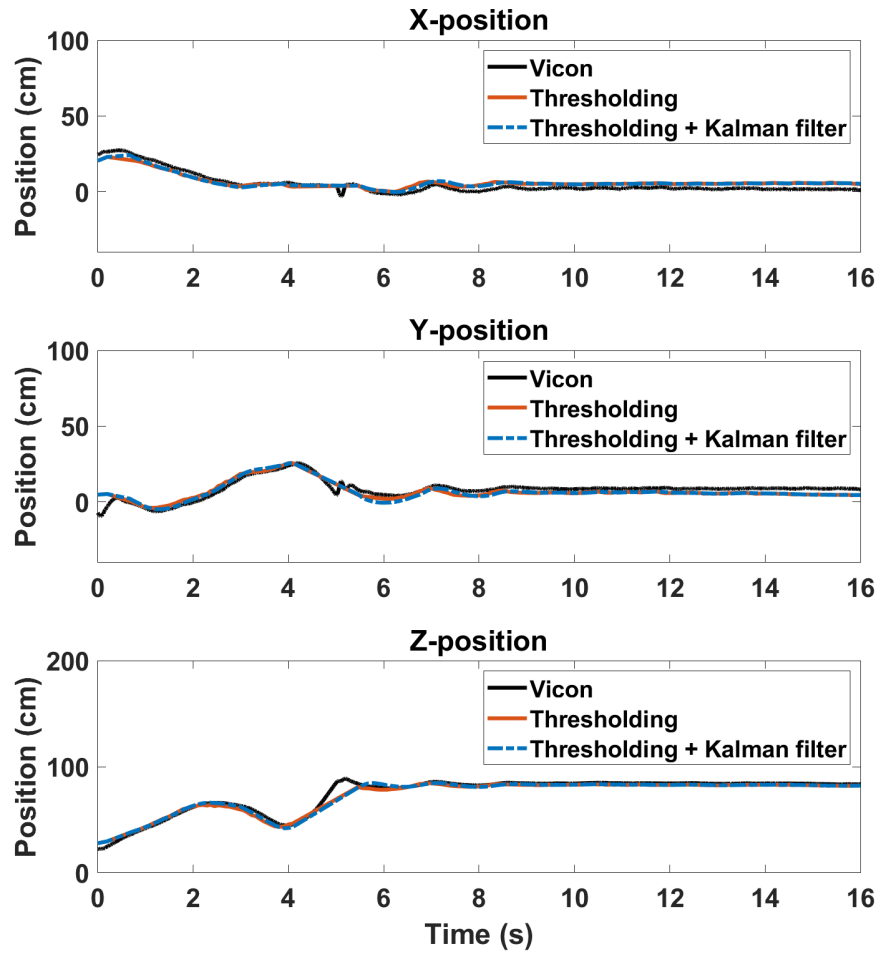


Figure 2.33: Estimation of hovering aircraft position relative to a stationary landing marker in low illumination (10 lux).

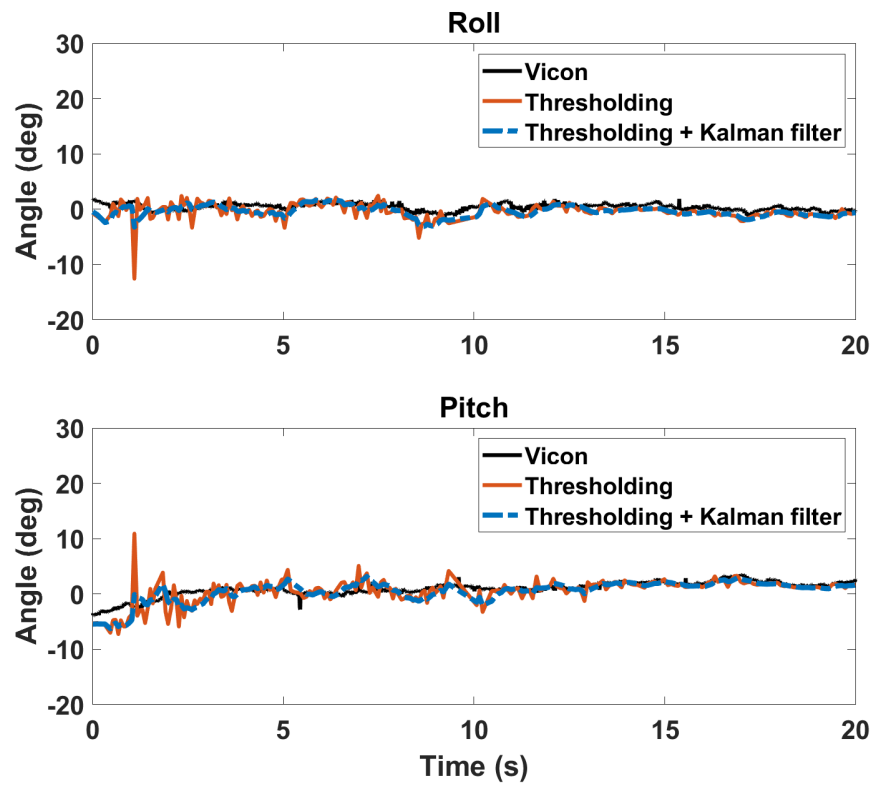


Figure 2.34: Estimation of hovering aircraft attitude angles relative to a stationary, 57% occluded landing marker in indoor illumination (300 lux).

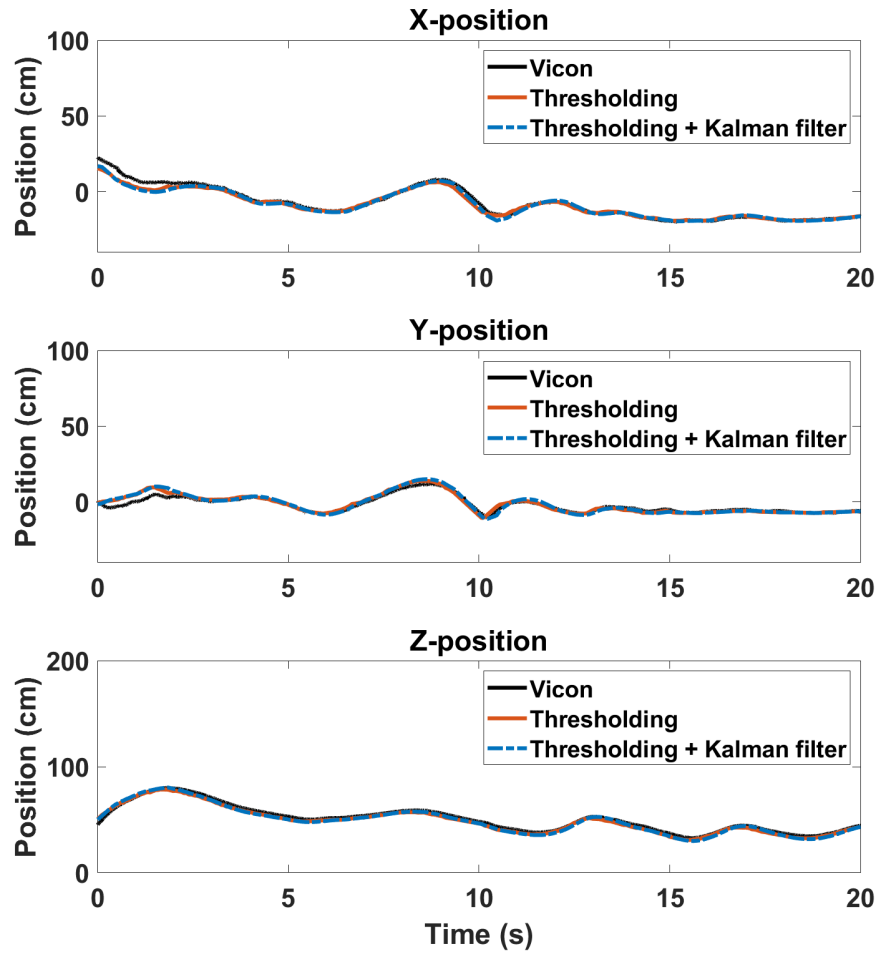


Figure 2.35: Estimation of hovering aircraft position relative to a stationary, 57% occluded landing marker in indoor illumination (300 lux).

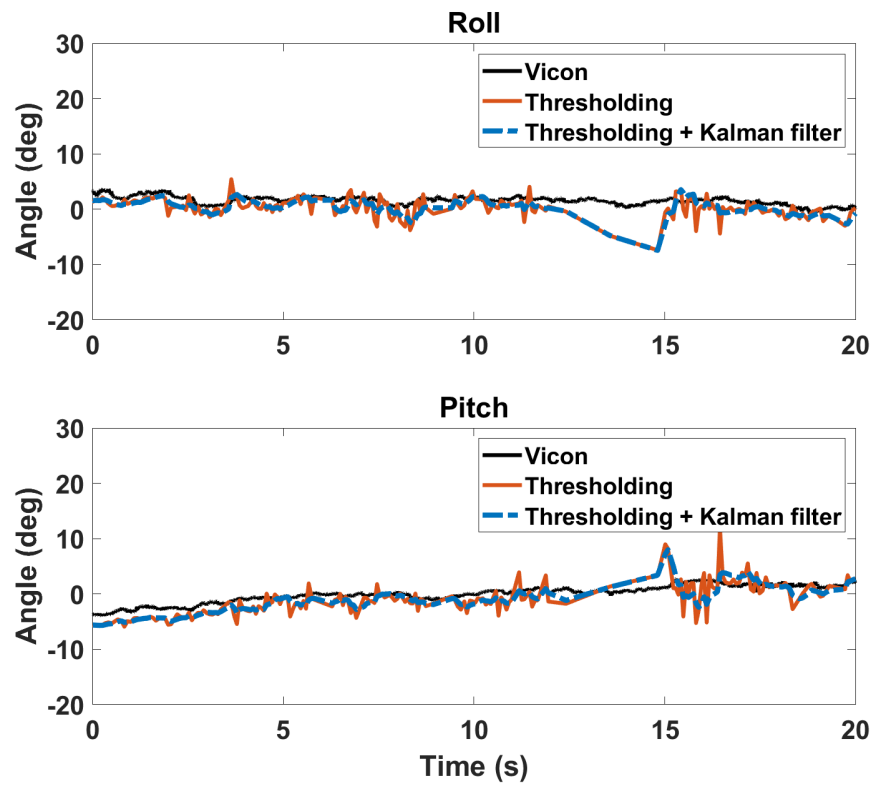


Figure 2.36: Estimation of hovering aircraft attitude angles relative to a stationary, 57% occluded landing marker in low illumination (10 lux).

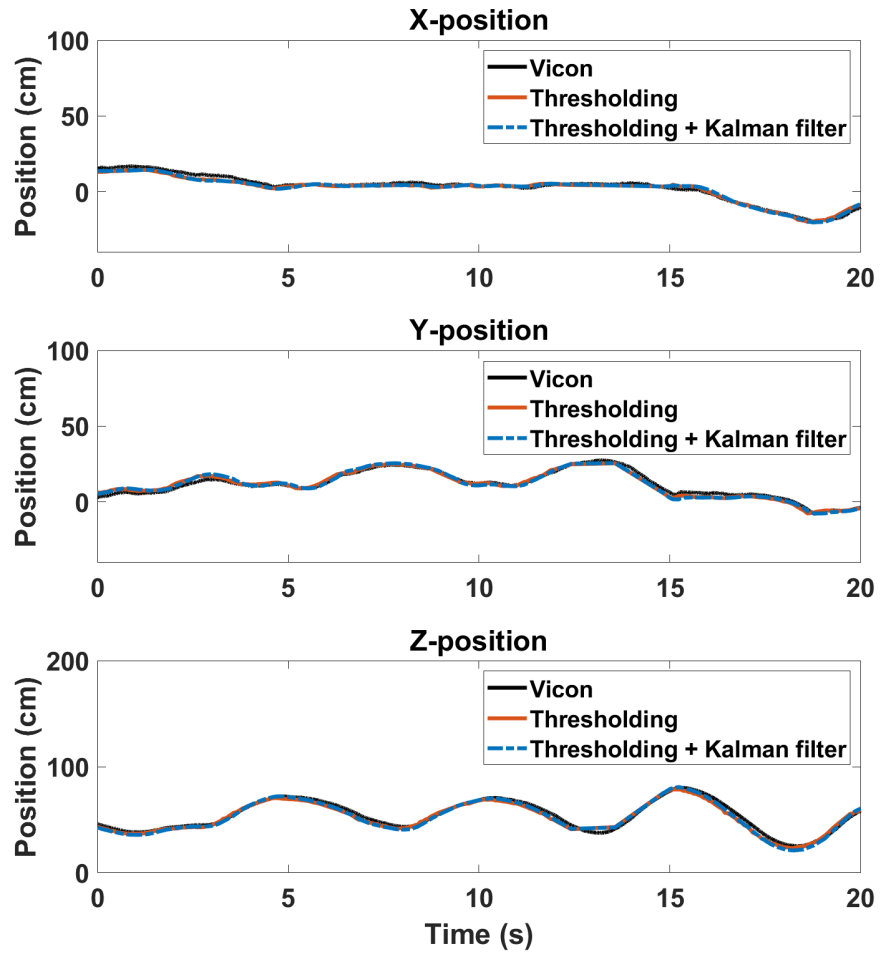


Figure 2.37: Estimation of hovering aircraft position relative to a stationary, 57% occluded landing marker in low illumination (10 lux).

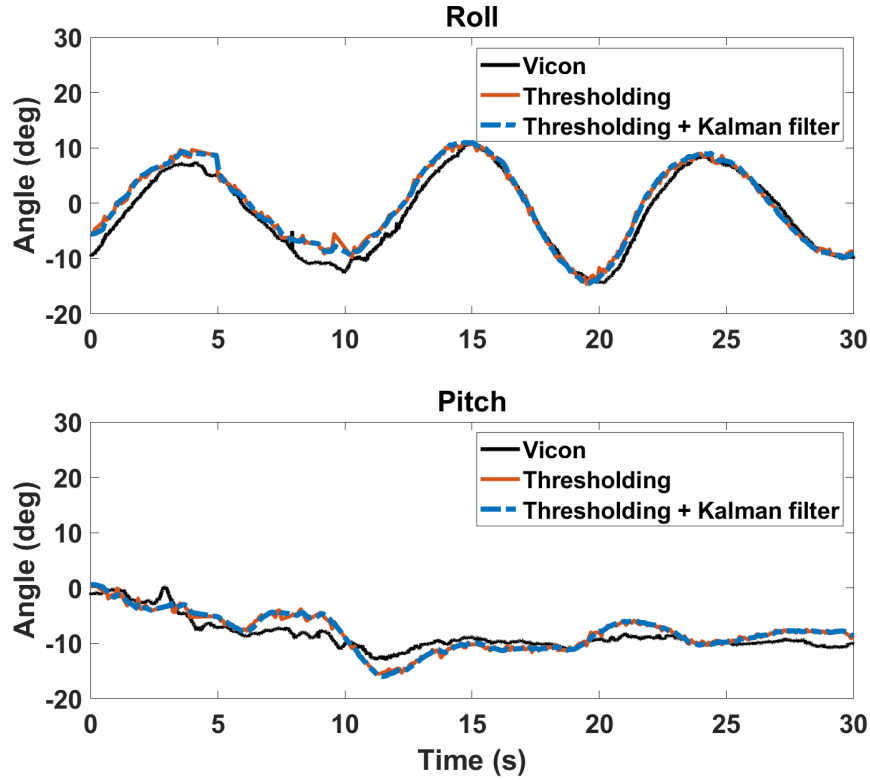


Figure 2.38: Estimation of hovering aircraft attitude angles relative to a landing pad undergoing ship-deck motion.

calm deck motion for landing. While this in itself will not result in a crash, this could make it more difficult to autonomously carry out a landing.

Compared to the hand-held tests, the quality of pose estimation decreases somewhat in free flight. Quantitatively comparing the RMSE results, while the pose estimation errors are still low overall, the errors do increase due to vibrations and aircraft motions in free flight.

2.7.3 Ship-deck motion

Finally, a test was completed to evaluate algorithm performance in hover for ship-deck motion. For this test, the landing pad was attached to a Stewart platform following the same Sea-state 6 motion history used for the hand-held tests. The test was conducted under ideal visual

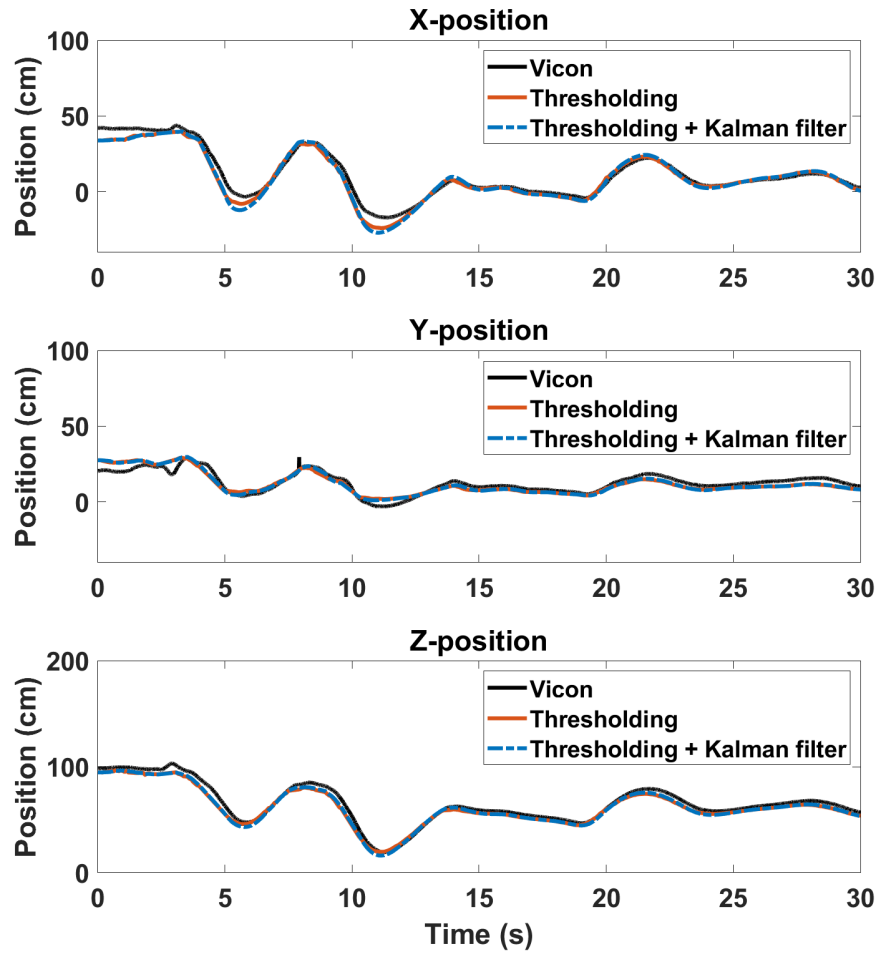


Figure 2.39: Estimation of hovering aircraft position relative to a landing pad undergoing ship-deck motion.

conditions, indoor lighting (300 lux) and with no occlusion of the landing image. The results of this test are given in Figures 2.39 and 2.38. As in the previous tests, the results follow the ground truth accurately. For the position estimates, this data shows an RMSE of 3.1 cm in the X-direction, 2.9 cm in the Y-direction, and 3.7 cm in the Z-direction. For attitude estimates, an RMSE of 1.8 degrees in pitch and 1.9 degrees in roll is observed. These errors show only a modest increase from the baseline case. The feature-based algorithm is therefore capable of accurately estimating the pose of a moving ship-deck in free hover.

2.8 Performance Comparison with Fiducial-Based Approaches

Previous work on vision-based landing using fiducial markers can provide a useful benchmark for the performance of the feature-based algorithm. The feature-based algorithm offers advantages over fiducial-based techniques, namely robustness to visual degradation; additionally, the chosen landing pad image resembles typical fiducial marker designs. Therefore, it is important to benchmark if the feature-based technique offers comparable pose estimation accuracy to fiducial-based approaches.

Our previous fiducial-based algorithm, evaluated under oscillating motion in visually ideal conditions, produced pose estimates with error of approximately 1-2 cm along each linear axis, 1-2 degrees in pitch, and 5 degrees in roll [4]. Araar et al. utilized a landing pattern consisting of multiple AprilTag markers for landing on a linearly-moving (non-rotating) platform, reporting a pose estimation error of approximately 1 cm for each linear axis and 0.5-7 degrees for rotation [18]. Nicholson et al. estimated the pose of a ship-deck at varying sea-states using a recursive AprilTag fiducial marker array; vision-based estimates were fused with inertial data for improved estimation quality. The results found an average error of 8-23 cm for position estimates and

1.8-3.1 degrees for orientation [29].

For the feature-based algorithm, error across various test scenarios was found to be less than 4 cm along each linear axis and 2.5 degrees for pitch and roll. Based on these results, the feature-based algorithm demonstrates comparable pose estimation accuracy to approaches using fiducial markers under ideal visibility conditions where fiducial markers can be used.

2.9 Summary

This chapter covered the refinement and experimental characterization of a 2D feature-based vision algorithm for ship-deck landing under degraded visual conditions. Systematic tests were carried out on a quadrotor UAS developed in-house which incorporates a computationally powerful flight computer. All computations were on-board. A series of real-time tests, with the quadrotor moved by hand for large angles, were completed to first evaluate the algorithm's performance. Free-flight tests were then carried out. From these results, the following key conclusions are drawn:

1. A feature-based algorithm is capable of accurately estimating the pose of a stochastically-moving platform at Sea-state 6 in both controlled and free-flight conditions.
2. A reference image with many distinct, high-contrast features is ideal for good algorithm performance. While the algorithm can function with typical ship-deck or helipad markings, performance degrades as these patterns have few good features. Therefore, a custom image is desired.
3. In ideal visual conditions, the vision algorithm could successfully estimate large variations in attitude in excess of 20 degrees.

4. The performance of the algorithm is resilient to visually-challenging conditions. In hand-held tests, where roll motions of up to 15 degrees were present, the ship-deck pose could be accurately estimated with low illumination, glare, and occlusion of the landing pad, as well as with increased camera distance. The deck pose could also be accurately estimated in free-flight hover with occlusion and low illumination.
5. Vibrations and small motions introduced in free hover have a minor effect on algorithm performance. Noise in algorithm output can be mitigated through Kalman filtering.
6. A time delay of approximately 400 milliseconds was present in the vision algorithm output. This was caused by a combination of camera latency and algorithm computation time. However, this delay is not expected to significantly affect tracking and landing.
7. With on-board computer and vision hardware, the algorithm updates at a speed of 9-11 Hz, which is adequate for capturing ship-deck motion.
8. The pose estimation results can be significantly improved by incorporating thresholds for feature matches and inliers; such thresholding allows poor pose estimates to be automatically rejected.

Chapter 3: 3D Feature-Based Algorithm

This chapter describes the development and experimental validation of a 3D feature-based algorithm. It describes the algorithm based on the structure-from-motion computer vision technique, vision system integration into a quadrotor UAV, comparison of feature detectors, inlier thresholding, speed enhancement, and the results in ideal and visually degraded conditions. Material presented in this chapter was previously published in the AIAA Journal of Aircraft [\[42\]](#).

3.1 3D Vision Algorithm

A new three-dimensional feature-based vision algorithm is developed to estimate the pose of a moving ship-deck. This new approach can potentially address many of the deficiencies of 2D feature-based approaches. It estimates the pose of a 3D object rather than a 2D reference image; this is more representative of visual estimation performed by human pilots as illustrated in Figure [3.1](#), and removes the need for a 2D image on the landing location. Additionally, this approach can tolerate large rotational motions which could cause the 2D feature-based algorithm to fail [\[41\]](#). The algorithm incorporates multiple fundamental techniques used in 3D computer vision.



Figure 3.1: During ship landing, the pilot visually tracks the moving 3D ship structures.

3.1.1 Feature extraction and matching

The concept of features remain the same for this new algorithm. They are small patches of interest, such as regions of high contrast or rich texture in a 2D image, which are robust to transformations in scale, translation, illumination, and in- and out-of-plane rotation. Locations of good features are typically found by detecting strong corners or blobs (locations with significant change in pixel intensity in two directions). Image patches at these locations are extracted and converted to feature descriptors, which are a numerical vector representation of the image patch.

Features can be matched by comparing these extracted feature descriptors, which can be used to find corresponding points across different images. The same technique was used in the 2D vision algorithm described in Chapter 2.

3.1.2 Feature detectors

Multiple algorithms have been developed in the past by other researchers to detect and extract image features. For this work, implementations of these algorithms provided in the

OpenCV library were used.

The Scale Invariant Feature Transform (SIFT) algorithm, developed by Lowe [14], is a commonly used feature detector. It generates vectors of length 128 as feature descriptors. Features can be compared by calculating the L2-norm (Euclidean distance) between two descriptors. The algorithm is invariant to changes in scale and translation, and is partially robust to changes in in- and out-of-plane rotation and illumination. While the algorithm provides generally good feature matching results, it is more computationally intensive than many other detectors [43].

The Oriented FAST and rotated BRIEF (ORB) algorithm [15] is another common feature detector. The algorithm uses BRIEF binary descriptors; each of the 256 elements has the value 1 or 0. Features can be compared using the Hamming distance, which describes how many elements differ between two descriptors. For binary descriptors, this distance is computed using the formula

$$d(p, q) = \sum_{k=1}^n |(p_k - q_k)| \quad (3.1)$$

where p and q are two descriptors of length n (256 in this case). While ORB is less robust, it is significantly faster computationally than SIFT [43].

3.1.3 3D triangulation

Using two or more images, it is possible to determine the location of a point in 3D space. This process, triangulation, is illustrated in Figure 3.2. A point in 3D space is projected onto images from two cameras with known parameters and poses. Using the two corresponding 2D image points, the coordinates of the 3D point can be recovered.

An ideal triangulation case is shown in Figure 3.2a. In this case, the 2D image points are known exactly without noise or error. The 3D point, labeled X in the illustration, is projected onto the two 2D image planes along two straight lines. These lines pass through the focal points of each camera, labeled O_1 and O_2 , and intersect at X .

As previously explained in Chapter 2, a model to map 3D world coordinates to 2D pixel coordinates is given as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (3.2)$$

where P is a 3×4 camera matrix containing both the intrinsic and extrinsic parameters, (u, v) is the 2D point location in pixel coordinates, and (x, y, z) is the corresponding 3D point in world coordinates. This notation can be simplified to

$$x = PX \quad (3.3)$$

where x is the 2D point coordinate and X is the corresponding 3D point coordinate. If the correspondences and camera matrices are known exactly, the exact 3D point coordinate X will satisfy two equations:

$$x = PX \quad (3.4)$$

$$x' = P'X \quad (3.5)$$

where P and P' are the known camera matrices of the two cameras and x and x' are the two known 2D points corresponding to the same visible 3D point.

In practice, the 2D point coordinates are not known exactly. Techniques for finding these 2D points in images are generally subject to noise and small error. This is illustrated in Figure 3.2b. The straight lines passing through the 2D point and the camera focal point will not pass through the 3D point X , and may not intersect with each other at all. Since it is not possible to find X exactly, it is necessary to find an optimal estimate that minimizes error.

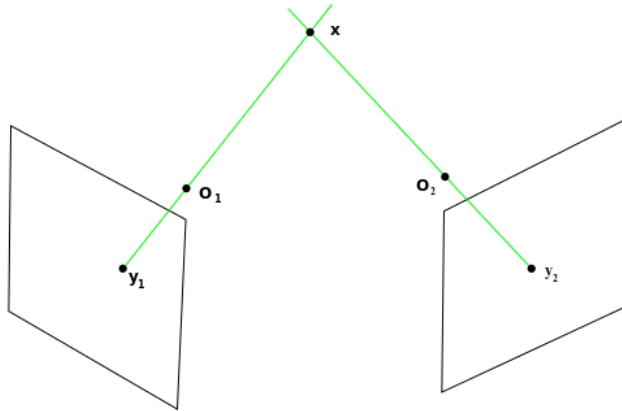
Multiple techniques exist to solve this problem. One common technique is the direct linear transformation (DLT) method [44]. First, rewrite Equations 3.4 and 3.5 as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \end{bmatrix} X \quad (3.6)$$

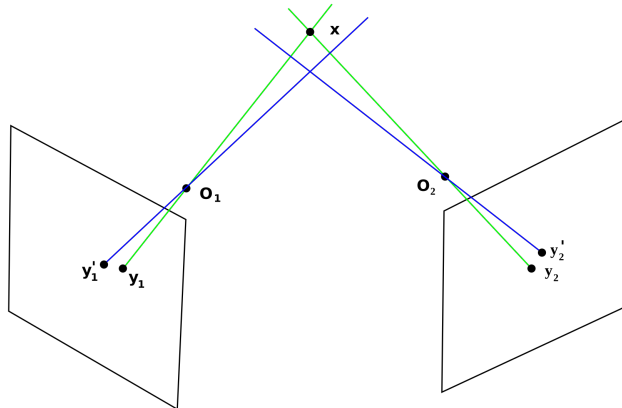
$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} p_1'^T \\ p_2'^T \\ p_3'^T \end{bmatrix} X \quad (3.7)$$

A linear system is then derived:

$$\begin{bmatrix} vp_3^T - p_2^T \\ p_1^T - up_3^T \\ v'p_3'^T - p_2'^T \\ p_1'^T - u'p_3'^T \end{bmatrix} X = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.8)$$



(a) Triangulation with exact point correspondences. In this scenario, recovering the 3D point location is straightforward.



(b) Triangulation with imperfect point correspondences, as is typical in real-world scenarios.

Figure 3.2: Triangulation of a 3D point from two images.

This is a homogeneous linear system of the form $AX = 0$, which can be solved to find an optimal estimate of X . The DLT triangulation technique is implemented in OpenCV; this implementation is used for this work.

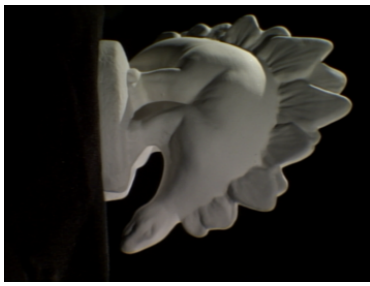
The triangulation process requires the poses, or positions and orientations, of the camera(s) corresponding to each image. The camera poses can be measured directly or recovered using a visual reference of known size and pose, such as a fiducial marker. Alternatively, the relative camera poses can be estimated using multiple corresponding or matched 2D image points. The disadvantage of this approach is that it is ambiguous in scale; an additional scale reference must be used to recover this information.

While only two images are necessary for triangulation, additional images can be used to recover additional 3D points for which correspondences could not be found in the previous images. If point correspondences are found across more than two images, the accuracy of the recovered 3D points can be improved further through numerical optimization techniques. This is described in more detail in the following section.

3.1.4 Ship-deck detection algorithm

A new ship-deck detection algorithm was developed as part of this research. The algorithm is separated into two stages: 1) extraction of features and their 3D positions from a set of images of a target object; and 2) estimating the pose of the object using this extracted data. The assumption is that the target object is known in advance.

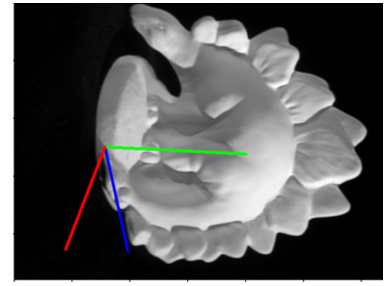
The feature extraction stage is developed from the structure-from-motion technique, which is used to recover the 3D structure of an object from a sequence of 2D images of it [45]. In this



(a) An example dataset image.



(b) A 3D point cloud showing the positions of features detected on the object. This result is produced using several dataset images.



(c) The estimated pose of the object in a new image, found using the data illustrated in (b).

Figure 3.3: Images illustrating the steps in the algorithm.

application, the 3D structure consists of a 3D point cloud (a collection of points in 3D space), and each point is associated with the 3D position of an extracted feature. An overview of this process is given below:

1. Obtain a sequence of images of the target object along with initial estimated camera poses.

These images can be collected from one or more cameras, and should include a sequence of varying object orientations relative to the camera(s), covering a range of object poses similar to those that may be observed during the pose estimation stage. Each image is assigned a numerical ID value, which is used in later processing steps. The estimated camera poses (defined relative to the origin of the target object) can be obtained through physical measurement of the camera position or by use of a known visual reference, such as an attached fiducial marker.

2. Begin the process of recovering features and their 3D locations using the following steps:

- (a) Extract and match 2D features in the first two images in the sequence. For each match, there will be a pair of corresponding 2D feature locations (in pixel coordinates) in

each image. Implementations from OpenCV, an open-source computer vision library, are used for feature extraction and matching.

- (b) Use the corresponding points and camera poses to triangulate the 3D positions of these corresponding points in physical units.
- (c) Store the matched features and their 3D positions.
- (d) Assign a unique numerical ID to each stored feature. For each 2D point recovered in Step 1, store the 2D point coordinates. In addition, store the numerical ID values of both the image and feature corresponding to each 2D point. This data is necessary for later steps.

3. Continue this process for each subsequent image:

- (a) Extract features from the next image in the sequence.
- (b) First, check for any matches with previously stored features. This allows individual features to be tracked across several images.
- (c) Repeat steps 2(a) and 2(b), matching features between the current and previous image and triangulating their 3D locations. Remove any matched features that were already matched to stored features in the previous step, and only retain the 3D point locations of new features. Store these new features and their 3D locations.
- (d) Repeat step 2(d). For each new feature recovered in the previous step, assign a new, unique numerical ID.

4. Repeat this process for all images in the sequence. The end result will be a list of extracted features and their triangulated 3D point positions. Additionally, there will be a list of stored

2D point coordinates for the features extracted in each image, as well as image and feature ID values corresponding to each 2D point.

5. For each image in the sequence, calculate the reprojection error for each of the saved 2D image points. This is done by projecting the 3D points detected in that image back onto the 2D image plane (using the estimated camera pose), and calculating the distances between the projected points and those obtained during feature matching. Remove any stored 2D points for which the reprojection error exceeds a specified threshold. This ensures that outlier points with large reprojection errors will not affect the optimization performed in the next step. If no 2D image points corresponding to a particular 3D point remain after this step, remove the 3D point and its corresponding feature.
6. Refine the estimated 3D point positions and camera poses with numerical optimization, also known as bundle adjustment. Specifically, a nonlinear least squares method is used to simultaneously optimize the camera poses and 3D points. An implementation of this method from SciPy, a Python scientific computing library, was used for this step. As each camera pose is represented with 6 numerical values (3 rotation and 3 translation), this requires optimizing $n = 6n_{poses} + 3n_{points}$ parameters. Residuals are found by computing the reprojection errors using the previously stored 2D points, using the same method described in step 5. The optimization refines the parameters through multiple iterations, minimizing the sum of squares of the residual. The result is that the estimated camera poses and 3D points are refined to minimize the reprojection errors across all points, improving the accuracy of the 3D point locations.
7. Remove any 3D points, and their corresponding features, that are not located on the object

itself. These points may be features located in the background or may be outliers resulting from incorrect matching. In this paper, these points are removed by generating a 3D bounding box (an imaginary 3D box which fully and tightly encloses the object geometry). Points within the box are assumed to fall on the object and are retained, while points that fall outside it are removed. This procedure removes background features and incorrect 3D points, preventing unwanted matches to these features during pose estimation.

8. Finally, save the refined 3D points and associated 2D feature descriptors.

The next stage is used for actual pose estimation. This is the portion that will be run on-board a rotorcraft during landing. This stage utilizes a monocular (single) camera, as 2D features from a single image are matched to stored 3D features. The process is described in detail below:

1. Load the saved features and their corresponding 3D positions.
2. Obtain a new 2D image of the target object, such as a frame from an on-board camera. Extract features from this image, and find matches with the features loaded in the previous step. This provides a list of 2D points in the image and their corresponding points in the stored 3D point cloud.
3. Solve the Perspective-n-Point (PnP) problem, which estimates the pose of the target object from corresponding 3D and 2D points. An OpenCV implementation of PnP, employing the RANSAC (Random Sample Consensus) method, is used for this step. RANSAC iterates through many random samples of the corresponding points, finding pose estimates for each. The estimate with the largest number of inliers, or points consistent with the pose estimate, is selected as the final result. This makes the result more robust to outliers (poor matches).

4. The final output will be the pose (position and orientation) of the object in the 2D image.

This process is repeated independently for any new images inputted.

5. If additional filtering of the raw data is necessary, the results are fed into a Kalman filter to mitigate any noise or small errors in the pose estimation results.

The features in a camera frame will be extracted and matched to the saved features from the first step. The matches between the 3D object points and the 2D points in the camera frame are used to recover the pose of the 3D object (the ship-deck). Because this process extracts features from many viewpoints of the object, instead of a single 2D image, the pose estimation is expected to be far more robust to rotation of the ship-deck.

3.1.5 Initial algorithm validation

The algorithm was initially validated using a publicly-available dataset, consisting of photos of an object taken from many viewpoints [46]. This dataset is useful for basic validation because it contains both pre-determined camera poses and accurate ground truth data. We inputted a subset of the images in the dataset, consisting of one full revolution about the object by the camera, into the feature extraction stage. Figure 3.3 (a) provides an example of a dataset image used for feature extraction. A total of 962 features and their corresponding 3D positions were successfully extracted, as shown in Figure 3.3 (b).

For pose estimation, another subset of images was used, consisting of viewpoints not used to extract features. These images form a sequence in which the camera again revolves around the object, though with the camera rotated 15 degrees from the original images to ensure these viewpoints are not the same as those used earlier to extract features. The sequence covers a 315

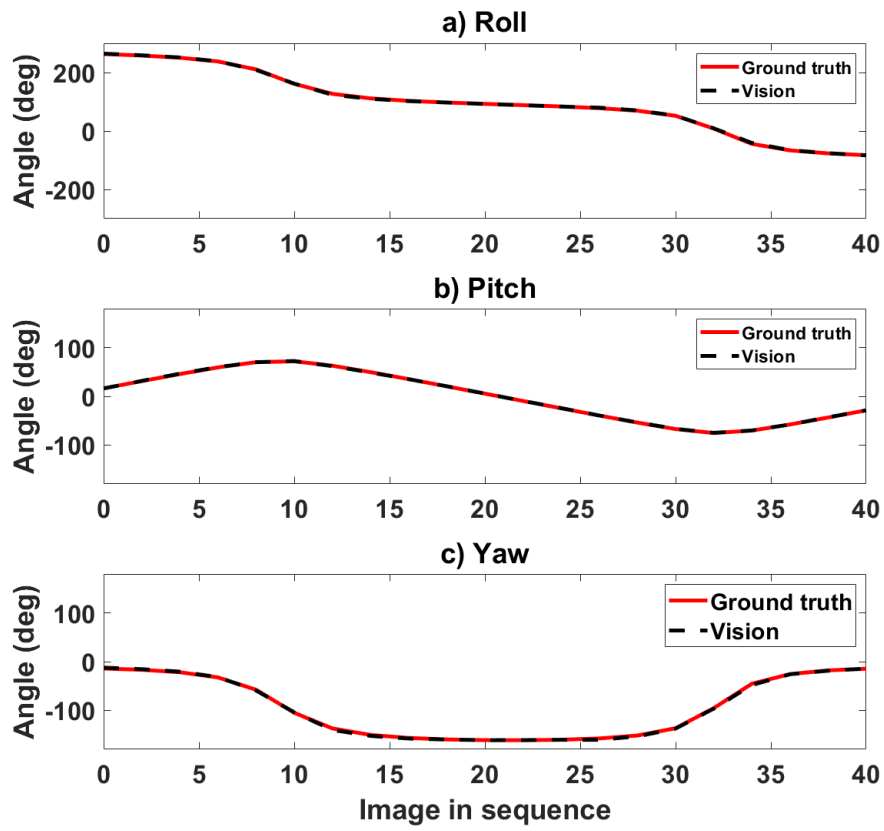


Figure 3.4: Estimated object pose compared to ground truth across a sequence of new images.

degree view around the object. One of these estimated object poses is illustrated in Figure 3.3 (c). The estimated poses for the sequence were then compared with ground truth, shown in Figure 3.4. Note that images in the sequence are numerically labeled in order, starting from image 0 and ending with image 40. For each image, there is a corresponding object pose. The plot shows that these poses can be recovered precisely.

3.2 Vision System Integration

The vision algorithm is integrated into a quadrotor UAV to demonstrate the feasibility of vision-based ship-deck landing. Since external communication may be unreliable during real ship-deck landings, the vision algorithm must run using only on-board hardware. The hardware is the same as was used for testing the 2D vision algorithm, specifically the camera, flight computer, and the quadrotor. Therefore, no change in hardware is required for migrating from 2D to 3D.

3.2.1 Code integration

The new pose estimation algorithm was implemented into the same on-board flight computer. The code uses ROS 2 (Robot Operating System 2), a set of robotics libraries, for asynchronous communication between individual processes, or nodes. The pose estimation algorithm and logging runs on a single node. An additional node is dedicated to continuously retrieving and publishing frames from the camera, which are then received as needed by the vision algorithm.

3.3 Algorithm Validation with Simulated Ship-Deck

The performance of the new algorithm must be evaluated with real-world objects before it can be incorporated into an aircraft for ship-deck landing.

3.3.1 Ship-deck

The vision algorithm was validated with a 3D object simulating a ship-deck. The deck measures $266.7 \times 266.7 \times 104.8$ mm ($10.5 \times 10.5 \times 4.125$ inch) and incorporates several simple 3D shapes, as shown in Figure 3.5. ArUco markers of varying sizes have been affixed to the surfaces of the ship-deck to provide many good features for detection. ArUco markers were specifically chosen to provide some consistency with the ArUco marker grid used to validate the 2D feature-based algorithm [41].

To extract features and 3D points from the object, we created a set of images of the object in a controlled manner. A camera was placed at three different fixed positions, and the object was rotated on a base in 10 degree increments to collect images around the entire object. With this process, a total of 111 photographs were collected from different camera poses. Example images from each camera position are provided in Figure 3.5. For improved image quality, these photographs were taken using an illuminated photography tent with a monochrome background. This reduces illumination differences and shadows and minimizes unwanted detections of background features. One of the ArUco fiducial markers on the object, which was fully visible in all views, was used to obtain initial estimates of the camera poses relative to the ship-deck during feature extraction. OpenCV functions were used to detect the fiducial marker in each image and compute the relative pose of the camera.



Figure 3.5: Example photographs of the ship-deck used for feature and 3D point extraction, collected at each of the three camera positions.

3.3.2 Algorithm testing

The performance of the algorithm was evaluated using the simulated ship-deck in various scenarios. For all tests, the algorithm was run in real-time using the previously described flight computer and camera. Unless otherwise specified, the camera was mounted to the underside of the quadrotor UAV, with the vehicle held and moved by hand to allow large angles and for more controlled results.

An important consideration is that the vision algorithm outputs the pose of the ship-deck relative to the camera. Therefore, all results presented here are relative poses. Note that these poses are affected by both the motion of the camera and the motion of the ship-deck.

Ground truth data was collected using a Vicon motion-capture system, which tracked the true pose of the quadrotor and target object during tests. The raw algorithm output, rather than the Kalman filtered data, is presented to more accurately evaluate algorithm performance.

With the 3D vision algorithm, a time delay of approximately 500 milliseconds was observed in the timestamps of logged pose estimates compared to ground truth. This delay is due to a combination of computational time and latency in receiving camera frames. To allow for better visual comparison in plots, the delay is corrected for in all real-time results presented in the paper. The root-mean-square error (RMSE) was used to quantitatively evaluate error in real-time

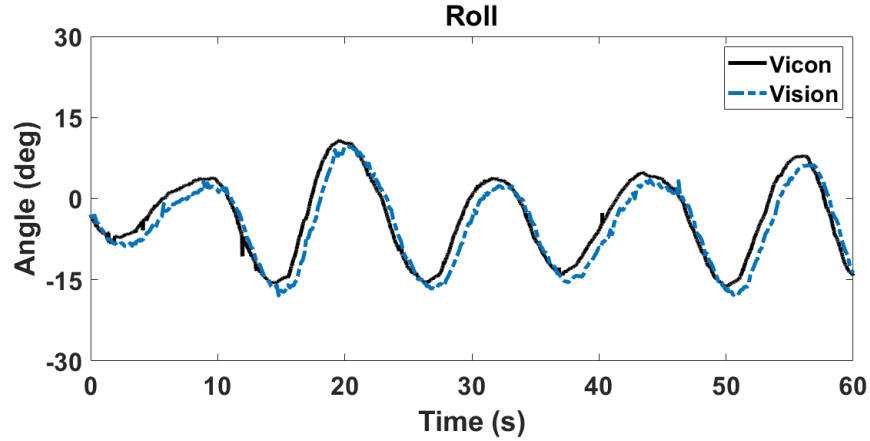


Figure 3.6: Example of vision algorithm result for roll motion with uncorrected 500 ms delay.

test results after incorporating the time delay correction. Figure 3.6 provides an example of vision algorithm output with the time delay present.

3.4 Comparison of Feature Detectors

Different feature detectors present critical trade-offs for algorithm performance. These include the number of extracted features, matching accuracy, robustness to scale and rotation, and computational cost [43]. Therefore, it is necessary to quantitatively compare the performance of the algorithm using different feature detectors in controlled conditions. The two feature detectors SIFT and ORB were compared using these tests.

With the OpenCV implementation of ORB, it is also possible to specify the maximum number of features retained from each image (nFeatures). After features are extracted from an image, they are ranked according to what is known as the Harris score. This score is a measure of intensity change in all directions across the image patch. It will be higher for strong corners, which constitute good features, and lower for more edge-like patches or flat patches with little variation [38]. The highest-scoring features, up to a maximum of nFeatures, will be retained.

A smaller value of $nFeatures$ will result in fewer features extracted and therefore fewer possible feature matches, but will increase computational speed.

3.4.1 Feature extraction results

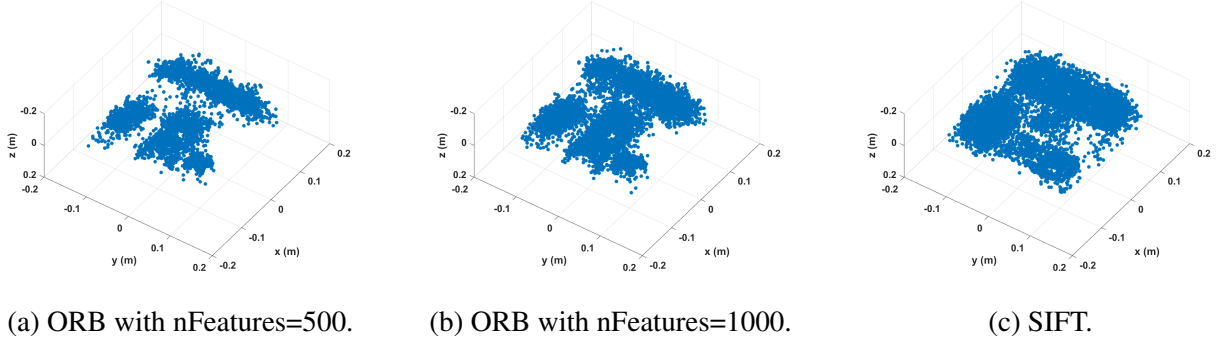


Figure 3.7: Feature extraction results with different feature detectors.

Features and their corresponding 3D points were extracted from the ship-deck images using both SIFT and ORB. For ORB, this was performed with $nFeatures=1000$ and $nFeatures=500$. The resulting 3D point clouds are shown in Figure 3.7. ORB with $nFeatures=500$ and $nFeatures=1000$ results in 5131 and 10602 unique features, respectively, while SIFT extracted 11317 unique features.

3.4.2 Algorithm performance comparison

The object simulating the ship-deck was placed horizontally on the floor in front of the camera, as seen in Figure 3.8. For image collection, the camera was mounted to a miniature tripod, allowing the camera position and pitch to be adjusted manually. The camera was set to three different linear positions and pitch angles, ensuring that the deck is fully visible in the image for each position. Three photos were collected at each camera position to account for random

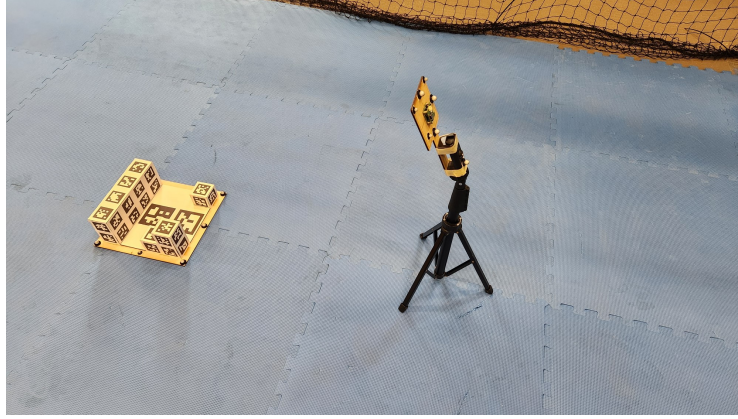


Figure 3.8: Setup used for collecting images of the ship deck.

image noise and error. For each test case, results were calculated for each image and averaged for each camera position.

The resolution of the camera image is also an important consideration for the vision algorithm. Extracting features from a lower resolution image is less computationally expensive, but can result in failure to detect smaller features in the image, especially at larger distances from the object [5]. The algorithm was therefore tested on images at the original camera resolution (1920×1080), as well as images scaled down by 50% (960×540).

The mean computational times required for the algorithm, averaged across all evaluated camera locations, are given in Table 3.1. Note that this is the time for an entire pose estimation update, which includes the total time to 1) extract features from the image of the ship-deck; 2) match features between the camera image and stored data; and 3) estimate the pose of the deck.

For full resolution images, SIFT is unacceptably slow, taking a mean of 0.569 seconds to process a single image. Reducing the resolution reduces this time dramatically to 0.164 seconds, though this is still longer than the ORB detector. In contrast, for ORB, the mean computational time is not meaningfully reduced by decreased image resolution. Varying nFeatures does show

a significant impact, as the computational time needed for $nFeatures=1000$ is almost double the time for $nFeatures=500$.

The mean errors in pitch and position estimate, again averaged across all three camera locations, are given in Tables 3.2 and 3.3, respectively. For 100% resolution images, SIFT demonstrates the highest accuracy, with an error of 0.21 degrees in pitch estimation. For ORB, an error of approximately 1 degree is observed for both values of $nFeatures$. For 50% resolution images, SIFT displays very similar results to the 100% resolution case. However, the performance of ORB degrades with reduced resolution, with errors for both estimated pitch and position increasing significantly. The error is especially severe for estimated pitch with $nFeatures=1000$.

Figure 3.9 shows the variation of error with distance from the camera to the object for the 50% resolution images. With SIFT, the pose estimation error is similar for all distances tested. ORB shows similar error to SIFT when the camera is closer to the object. However, the error increases severely when the camera moves further from the object, indicating ORB fails at longer distances. In particular, ORB with $nFeatures=1000$ displays a dramatic increase in error. Inspection of the matched features on the image indicates that many of the features on the 3D object were incorrectly matched to background details in the image. With the value of $nFeatures=1000$, additional, lower-quality features are extracted compared to $nFeatures=500$. This results in a higher number of erroneous matches. This failure is not acceptable for simulated landing, since for the experimental algorithm tests, the camera is located a distance of 50-90 cm from the ship-deck origin.

From these results, ORB with $nFeatures=500$ and 100% resolution camera frames was selected for testing real-time pose estimation. This combination was chosen as it provides both relatively low computational time and acceptable performance.

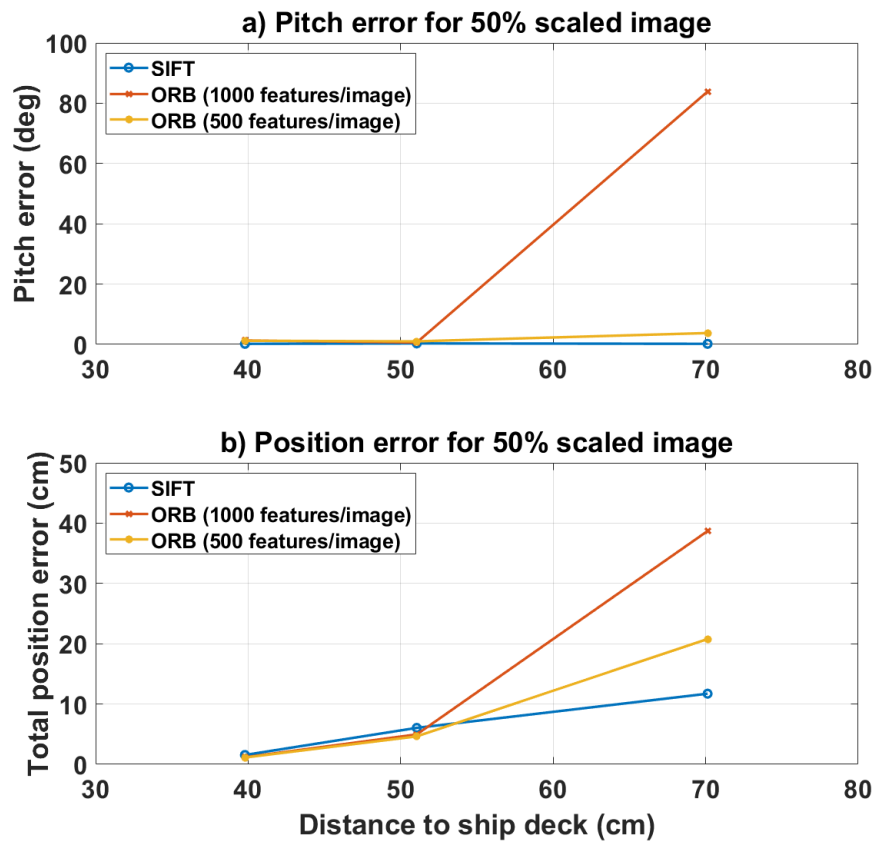


Figure 3.9: Pose estimation error at varying distance from the object for 50% resolution images.

Table 3.1: Mean computational time for different feature detectors and image resolutions.

Feature detector	100% resolution	50% resolution
SIFT	0.569 s	0.164 s
ORB (nFeatures=1000)	0.156 s	0.156 s
ORB (nFeatures=500)	0.080 s	0.078 s

Table 3.2: Mean pitch estimation error for different feature detectors and image resolutions.

Feature detector	100% resolution	50% resolution
SIFT	0.21°	0.23°
ORB (nFeatures=1000)	1.00°	28.60°
ORB (nFeatures=500)	0.97°	1.99°

Table 3.3: Mean position estimation error for different feature detectors and image resolutions.

Feature detector	100% resolution	50% resolution
SIFT	6.43 cm	6.44 cm
ORB (nFeatures=1000)	5.44 cm	14.94 cm
ORB (nFeatures=500)	5.41 cm	8.85 cm

An important note is that in this scenario, pose estimation was completed on single, pre-collected images, rather than updating and logging continuously alongside other running scripts. Due to this increased computational load, the update frequencies are expected to be lower during real-time operation.

3.5 Inlier Thresholding

Estimating a pose with PnP requires a minimum of 4 point correspondences. However, with a small number of corresponding points, errors in matching can result in an incorrect pose estimate. RANSAC iterates to find an estimated pose with the highest number of inliers, or point correspondences which are consistent with the estimated pose. The number of inliers can be used to evaluate the quality of a pose estimate; a small number of inliers may mean a poor result. The quality of pose estimation could potentially be improved by implementing a minimum threshold for the number of inliers, rejecting any estimates that fall below this threshold.

A vision algorithm test was completed to determine a good threshold for inliers. Algorithm output was collected with the quadrotor held by hand above the 3D ship-deck. The quadrotor was moved in quick, random motions in attitude and position, with the ship-deck either fully or partially in view of the camera, so as to generate pose estimates with varying quantity of inliers. For each pose estimate, the quantity of inliers was logged.

The attitude and position results are shown respectively in Figures [3.10](#) and [3.11](#). The raw pose estimates (no thresholds applied) are plotted alongside data which is filtered to remove pose estimates falling below the minimum inlier threshold. It was found that a threshold of 50 inliers yielded the best improvement in these results.

3.6 Stationary Camera and Landing Pad

A test was completed to show a baseline of performance. The quadrotor, with camera attached, was held stationary above the stationary 3D object, with the object fully in view of the

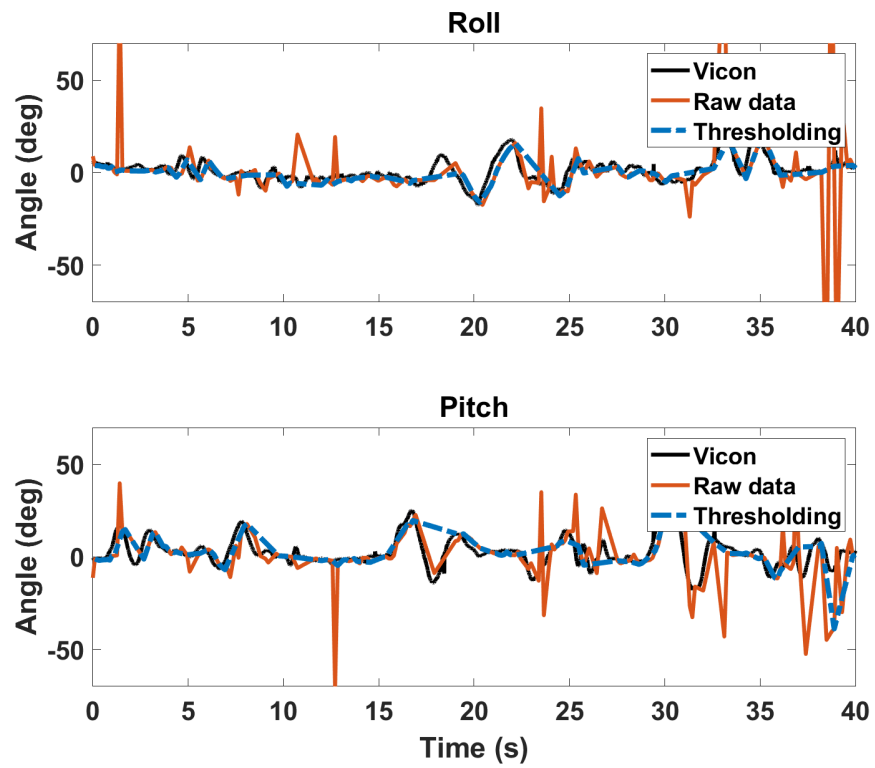


Figure 3.10: Estimation of relative attitude angles. Raw algorithm output is plotted alongside results with a minimum inlier threshold applied.

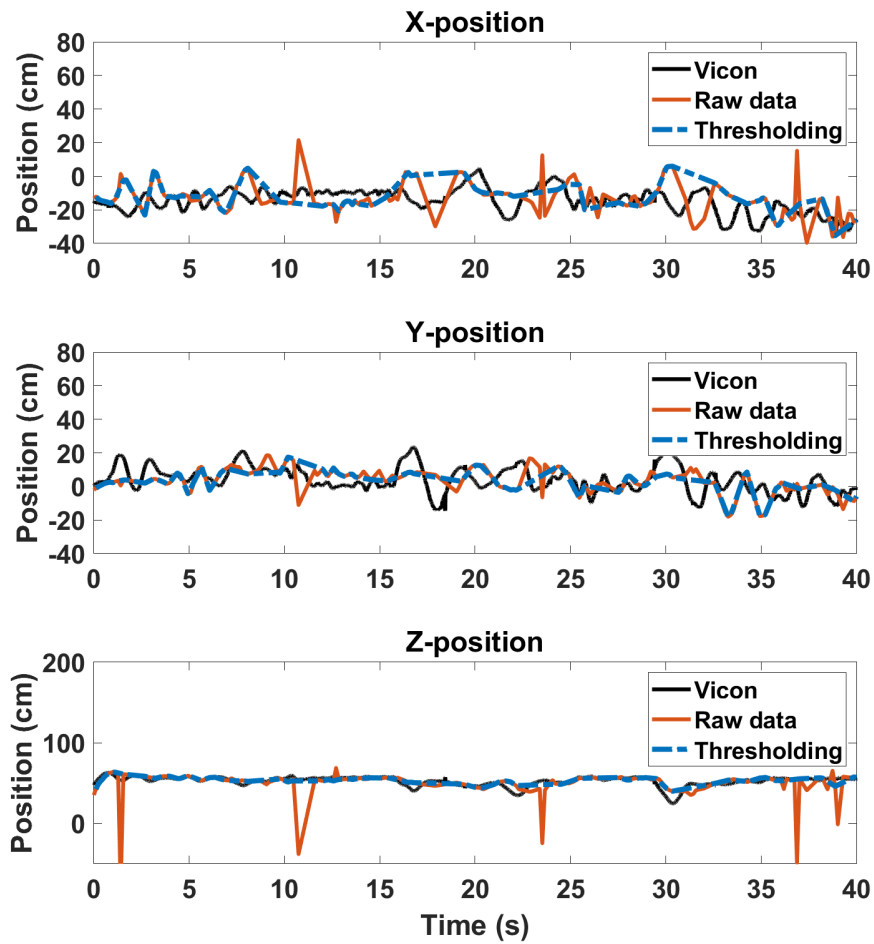


Figure 3.11: Estimation of relative position. Raw algorithm output is plotted alongside results with a minimum inlier threshold applied.

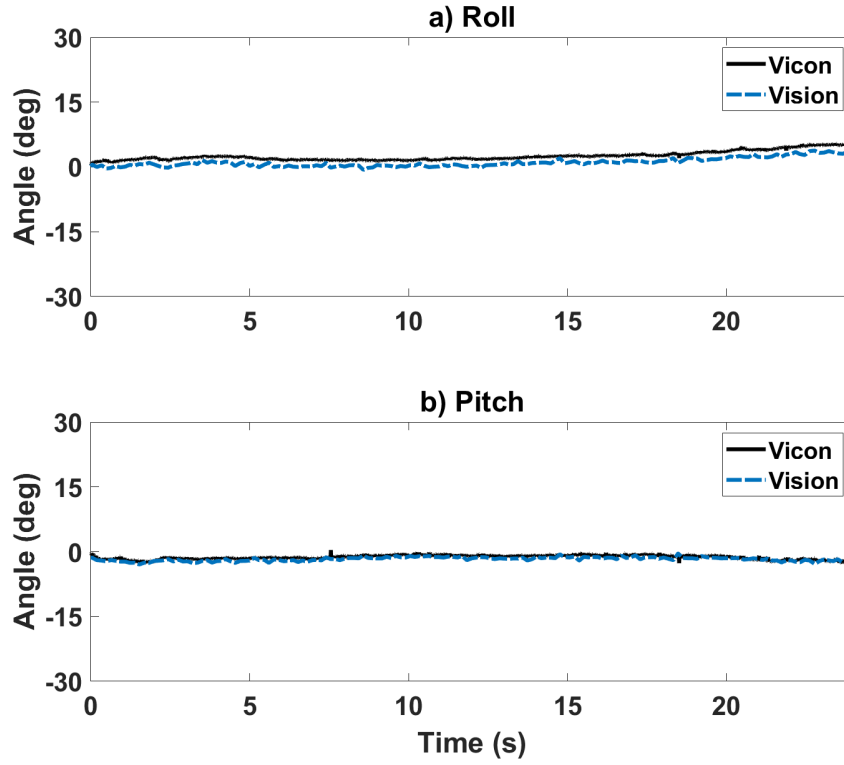


Figure 3.12: Estimation of relative attitude angles, with the quadrotor held stationary above the ship-deck.

camera.

The attitude and position results for this test are shown in Figures 3.12 and 3.13, respectively. These plots show very accurate estimation of the ship-deck pose. The RMSE for pitch and roll are less than 2 degrees, while the RMSE for linear position is less than 1.3 cm on any axis.

3.7 Ship-Deck Motion

The performance of the algorithm was then evaluated for a landing pad undergoing stochastic ship-deck motion. The ship-deck motion, representing Sea-state 6, was obtained from time histories provided in the Navy SCONE dataset, representing the motions of a full-scale DDG-

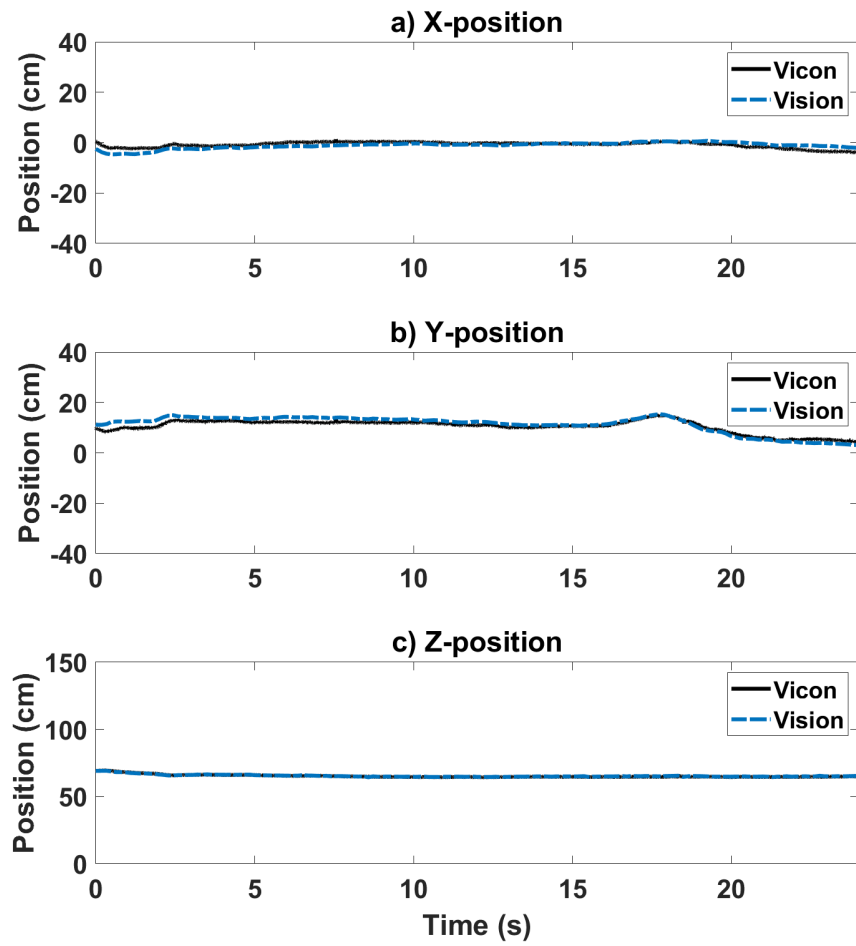


Figure 3.13: Estimation of relative position, with the quadrotor held stationary above the ship-deck.

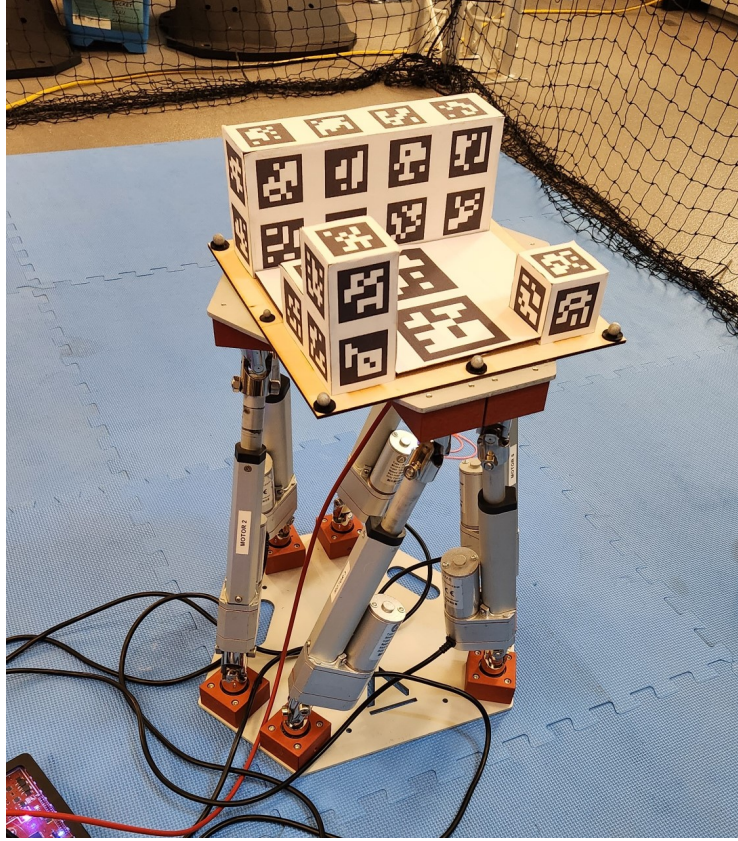


Figure 3.14: Stewart platform with 3D object attached. The Stewart platform is used to simulate ship-deck motion.

51 type ship [6]. The roll, pitch, and yaw motions are generated exactly from the time history without scaling, while the translational (surge, sway, and heave) motions are scaled to 0.5% of their original values due to the smaller scale of the experimental setup. A Stewart platform with the simulated ship-deck attached, as shown in Figure 3.14, was used to generate this ship-deck motion for vision algorithm testing. The vision algorithm is used to estimate the ship-deck pose in real-time.

First, the quadrotor was held stationary above the moving ship-deck, simulating a stable hover condition. The attitude and position results are provided in Figures 3.15 and 3.16, respectively. The results indicate that the algorithm can estimate the deck pose accurately in this case, as the

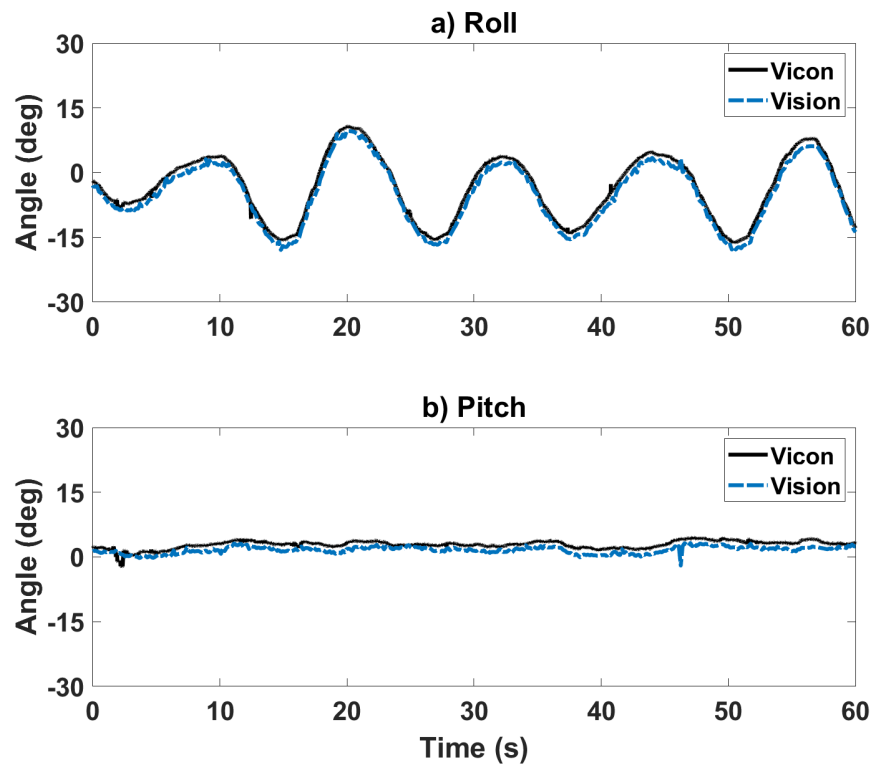


Figure 3.15: Estimation of relative attitude angles, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform.

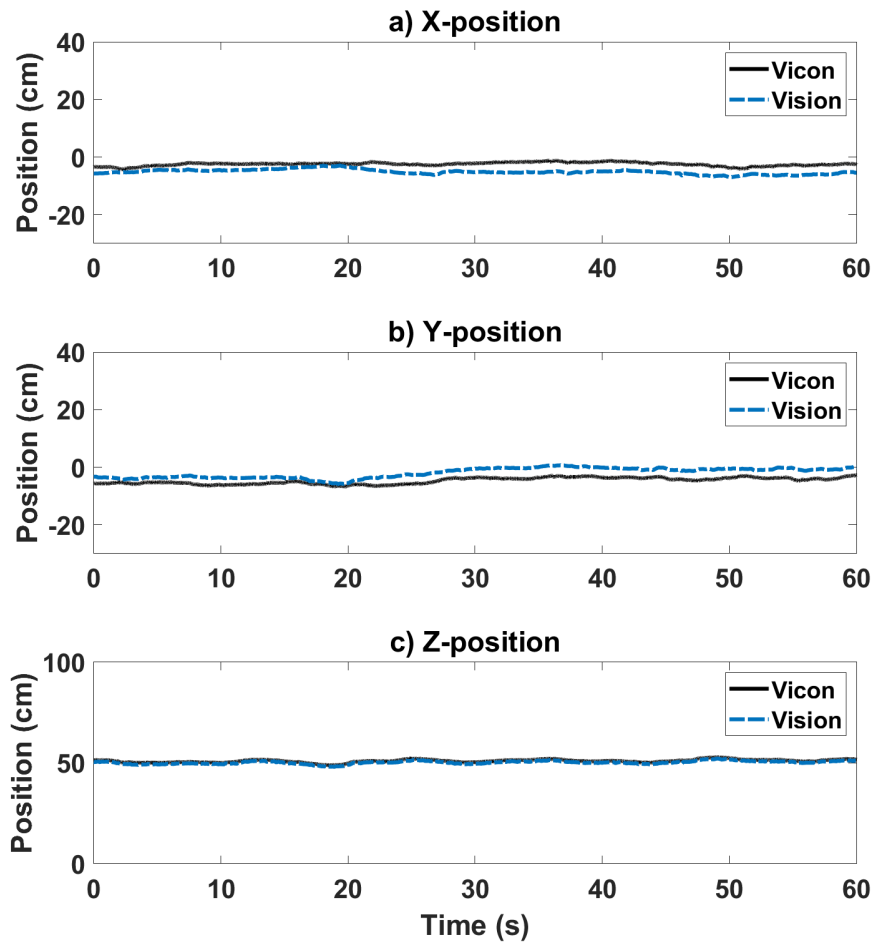


Figure 3.16: Estimation of relative position, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform.

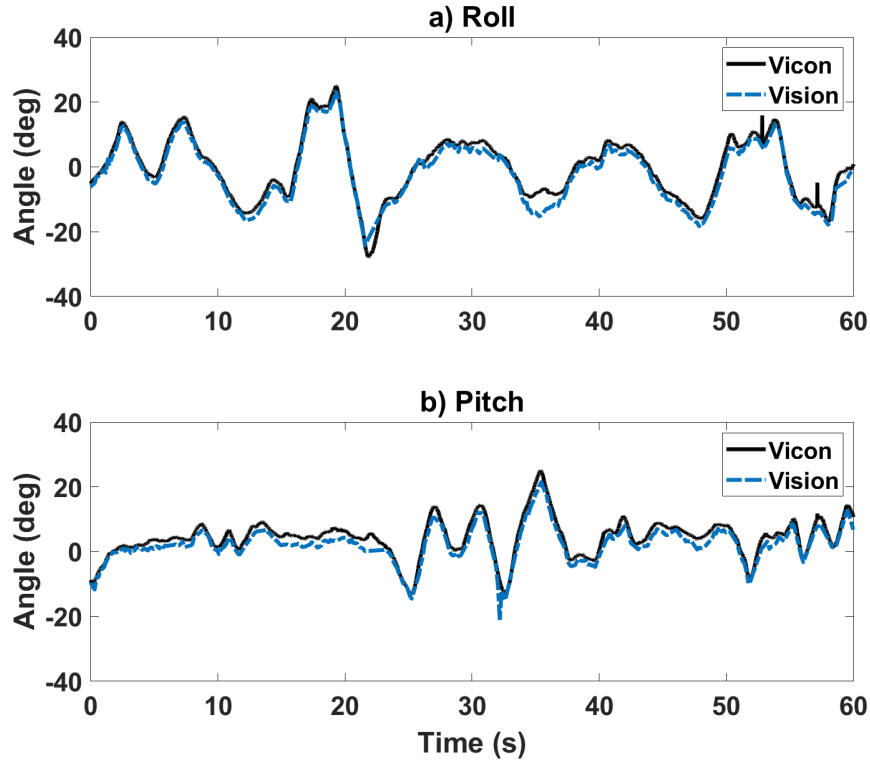


Figure 3.17: Estimation of relative attitude angles, with the quadrotor moved in attitude and position above the ship-deck attached to moving Stewart platform.

pose estimates closely follow the ground truth.

Next, the quadrotor was moved randomly with varying attitude and position. This creates large, complex relative motions between the camera and ship-deck which must be captured by the vision algorithm. The results of the test are shown in Figures 3.17 and 3.18. The attitude results closely follow the ground truth; the algorithm is able to accurately capture complex pitch and roll motions in excess of 20 degrees. Additionally, the estimate of Z-position closely matches ground truth. However, some errors are present in the estimates of X- and Y-position, as the vision algorithm estimates do not capture all of the rapid linear motions present in the ground truth.

The algorithm performance in both cases is quantitatively compared using the RMSE of

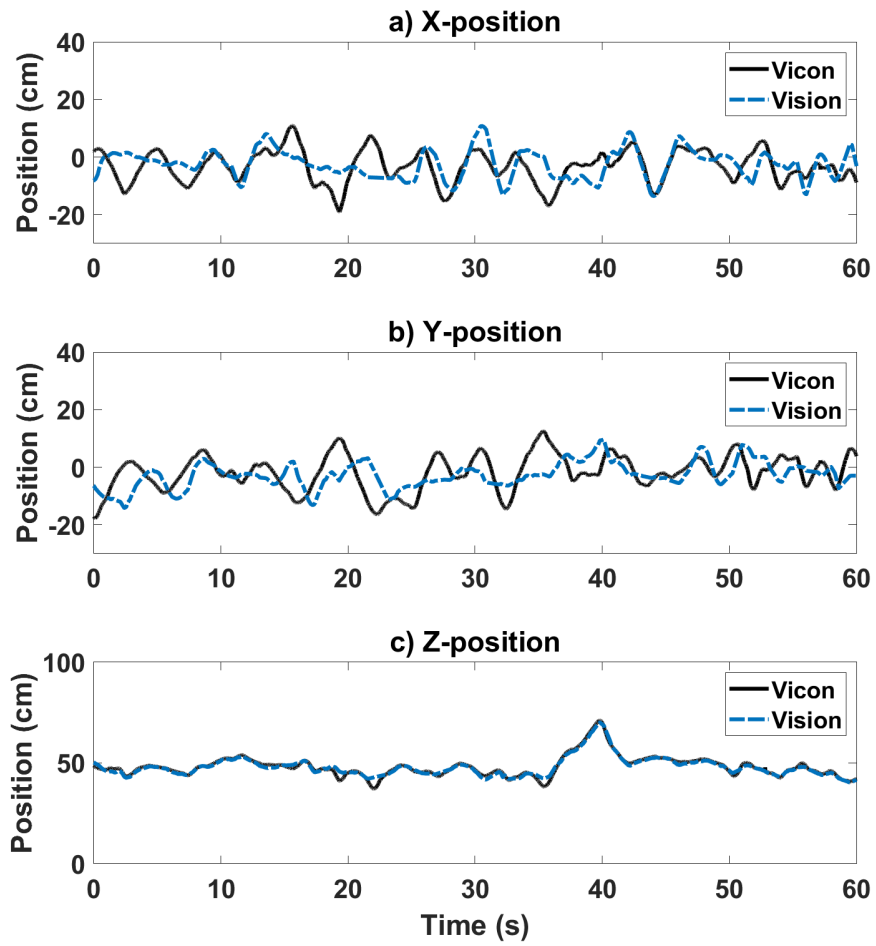


Figure 3.18: Estimation of relative position, with the quadrotor moved in attitude and position above the ship-deck attached to moving Stewart platform.

position and attitude, provided in Table 3.4. With the quadrotor stationary, the attitude results show similar error to the baseline case, though the X- and Y-position errors are increased. With significant relative motion of the quadrotor and platform, the X- and Y-position errors are significantly increased, while the attitude and Z-position errors are modestly increased. The movement of the quadrotor introduces motion blur, which contributes to these errors. Additionally, the 3D object is only partially visible in frame during portions of the test, resulting in fewer feature matches and therefore increased error.

Table 3.4: RMSE for ship-deck motion cases.

Test case	X	Y	Z	Roll	Pitch
Quadrotor stationary	2.9 cm	3.0 cm	0.8 cm	1.5°	1.4°
Quadrotor moving	5.7 cm	6.3 cm	1.1 cm	2.2°	2.4°

3.8 Visually Degraded Conditions

The algorithm’s performance was then evaluated in visually degraded conditions. Three conditions were specifically evaluated: 1) occlusion; 2) low illumination; and 3) glare. For all tests, the ship-deck was mounted to the Stewart platform simulating Sea-state 6 motions, with the quadrotor held stationary above the moving platform.

The vision algorithm was first evaluated under occlusion. The visible portions of the 3D object were partially occluded using paper coverings, as shown in Figure 3.19. The results for attitude and position are shown in Figures 3.20 and 3.21, respectively. The pose estimates in this case are accurate overall, closely matching the Vicon ground truth, with only minor noise in the pitch and roll estimates.

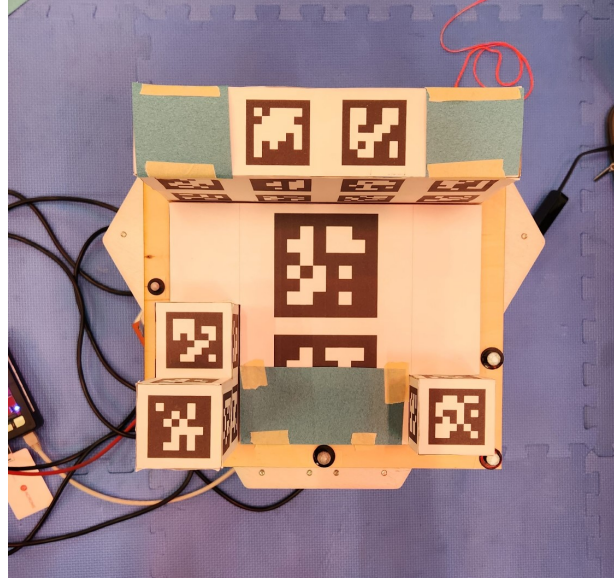


Figure 3.19: The 3D object simulating the ship-deck with added occlusions.

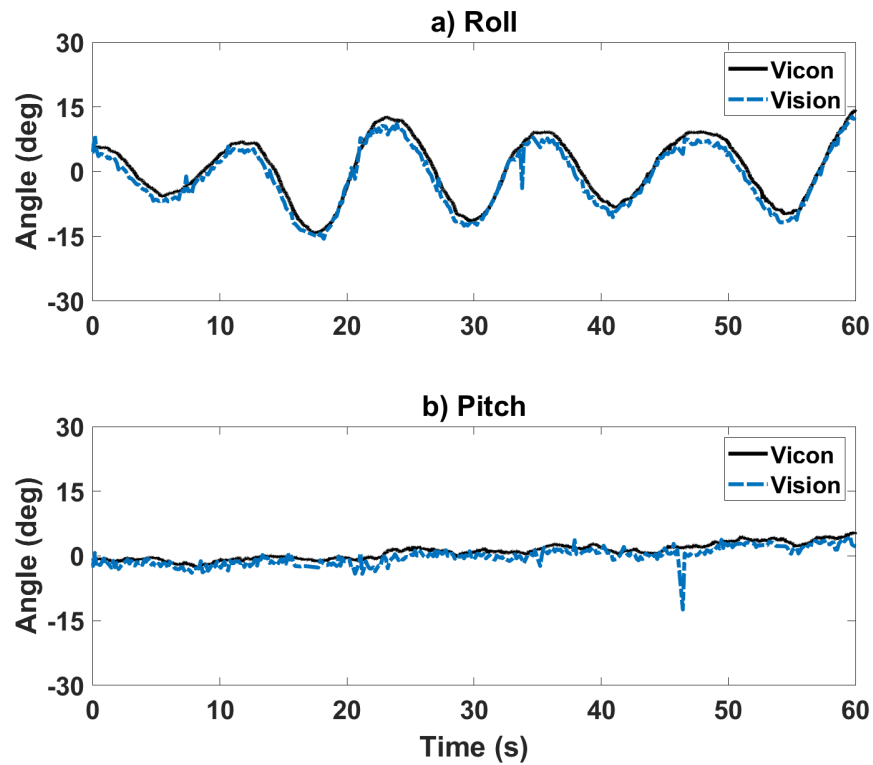


Figure 3.20: Estimation of relative attitude angles of a partially occluded ship-deck, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform.

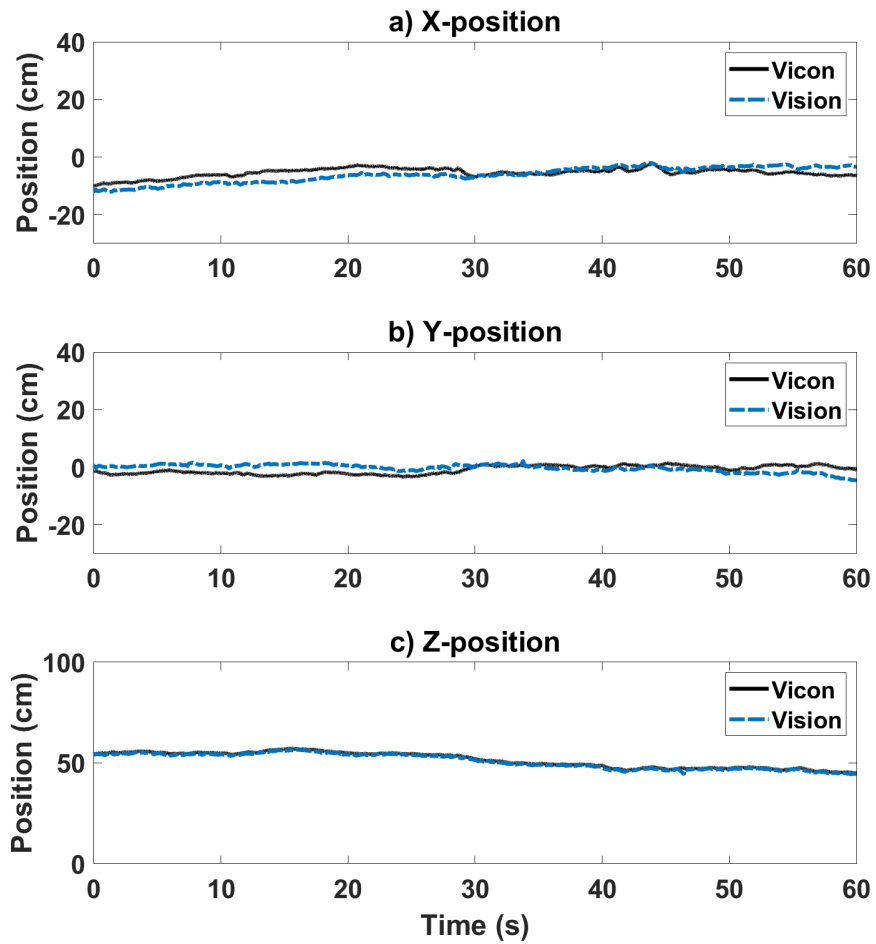


Figure 3.21: Estimation of relative position of a partially occluded ship-deck, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform.

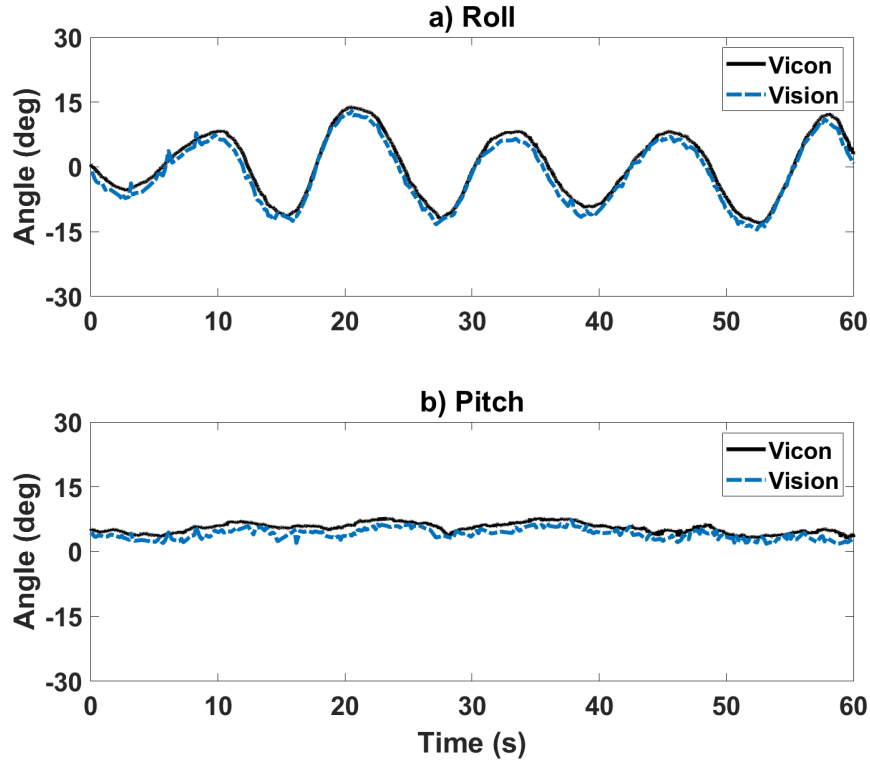


Figure 3.22: Estimation of relative attitude angles of a ship-deck in low illumination, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform.

The algorithm performance was then evaluated for low illumination conditions. Low indoor lighting was used to create an illumination of approximately 10 lux. The results of the test are shown in Figures 3.22 and 3.23. The attitude and position estimates for this case are accurate, closely matching the ground truth.

Finally, the algorithm was tested in glare conditions. A 900 lumen flashlight was used to create significant glare and shadows on the ship-deck, as shown in Figure 3.24. The attitude and position estimates are provided in Figures 3.25 and 3.26, respectively. The roll and pitch results follow the ground truth acceptably, and the position estimates match the ground truth very accurately.

Though the overall results appear accurate, there were some instances during the test in

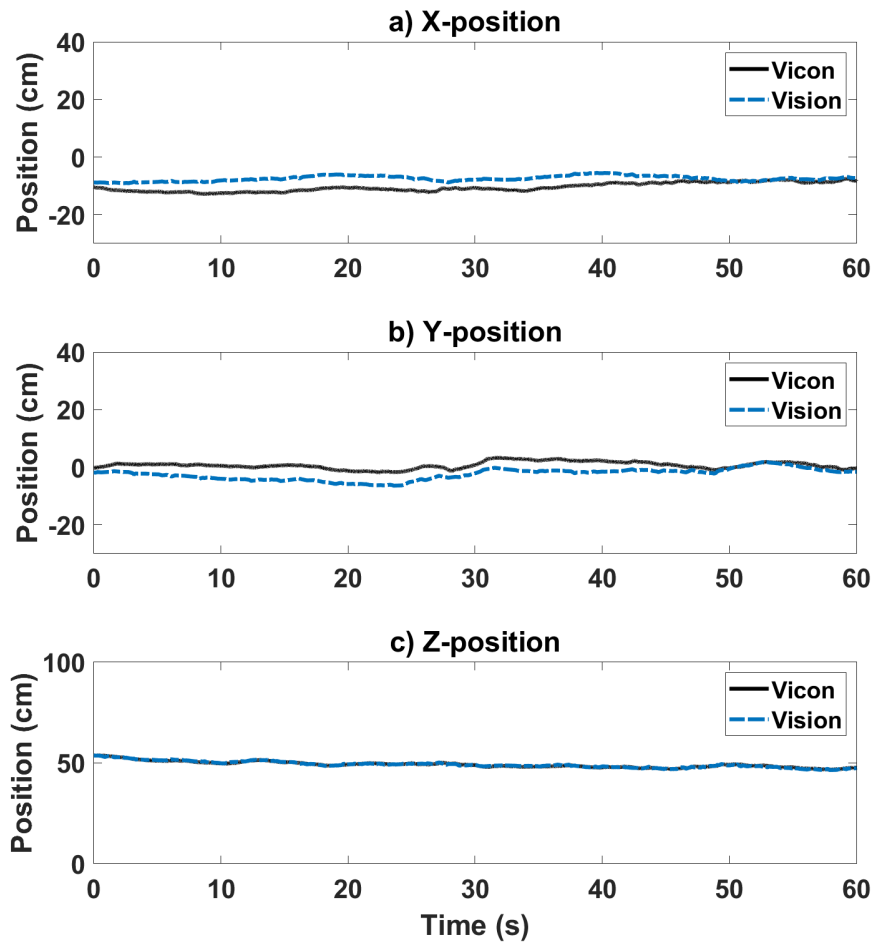


Figure 3.23: Estimation of relative position of a ship-deck in low illumination, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform.

which the vision algorithm failed to detect the ship-deck at all due to the glare obscuring a large portion of the ship-deck. In particular, the algorithm did not successfully log any pose estimates between the times of 12-16 seconds, 24-29 seconds, and 40-43 seconds during the test. Such a failure could be problematic during an actual ship-deck landing, as the controller will not receive any updated ship-deck pose estimates for several seconds.

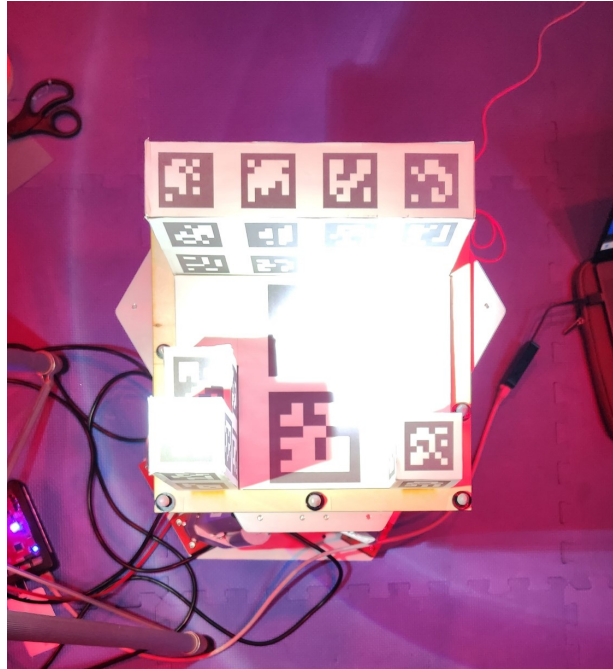


Figure 3.24: The 3D ship-deck with glare and shadows produced by a 900 lumen flashlight.

The RMSE of linear position and attitude, summarized in Table 3.5, can be used to further compare each test case. The low illumination and occlusion cases show comparable error to the ideal ship-deck motion test, indicating the algorithm is robust to these conditions. In the glare case, the errors are lower than in the occlusion or low illumination conditions. However, as the algorithm failed to detect the ship-deck several times during the test, robustness in glare needs to be further improved.

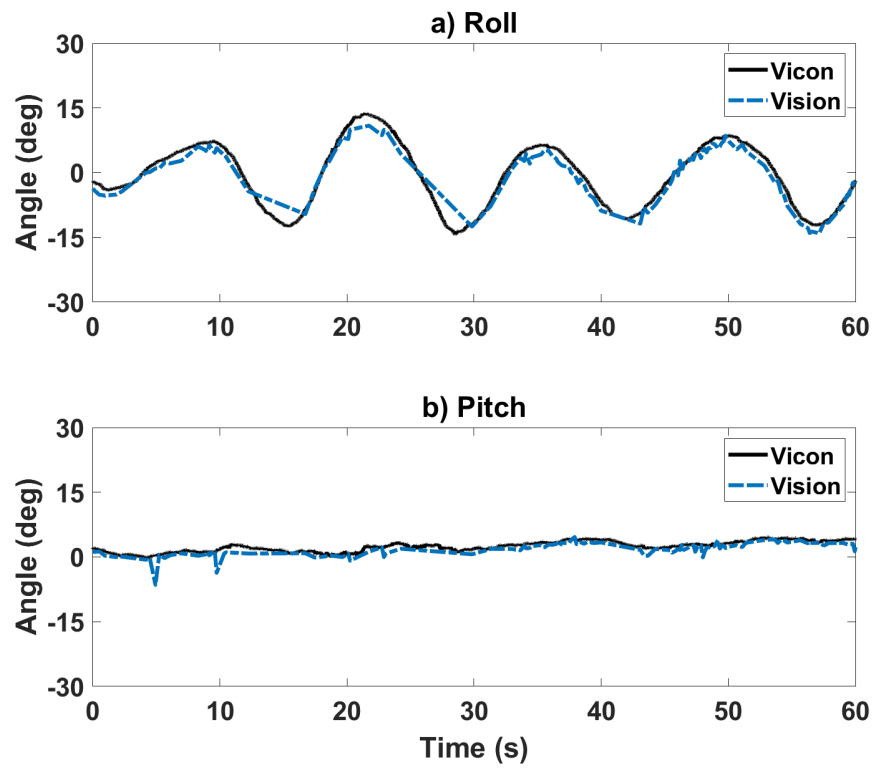


Figure 3.25: Estimation of relative attitude angles of a ship-deck with glare, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform.

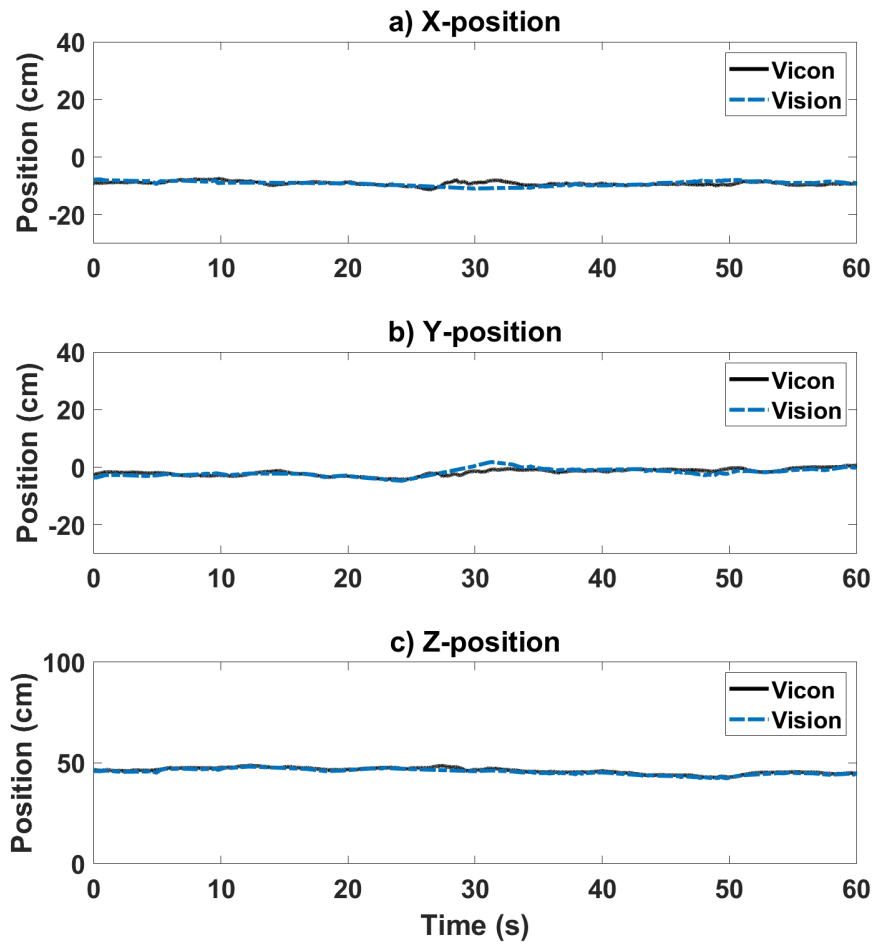


Figure 3.26: Estimation of relative position of a ship-deck with glare, with the quadrotor held stationary above the ship-deck attached to moving Stewart platform.

Table 3.5: RMSE for visually degraded conditions.

Test case	X	Y	Z	Roll	Pitch
Occlusion	2.2 cm	2.3 cm	0.6 cm	2.0°	1.6°
Low illumination	3.3 cm	3.4 cm	0.3 cm	1.5°	2.4°
Glare	0.9 cm	0.9 cm	0.5 cm	1.7°	1.0°

3.8.1 Feature match and inlier counts

As discussed earlier, the number of feature matches and inliers obtained during pose estimation can significantly affect the quality of results. Therefore, analyzing these values for various test cases can provide additional insight into algorithm performance in different scenarios. The mean number of feature match and inlier counts for each test case are summarized in Table 3.6.

Table 3.6: Summary of mean feature matches and inliers for each test case.

Test case	Mean feature matches	Mean inliers
Quadrotor stationary	391	102
Quadrotor moving	383	87
Occlusion	370	75
Low illumination	342	61
Glare	370	69

The mean numbers of feature matches and inliers decrease somewhat for visually degraded conditions compared to the visually ideal test cases. However, a corresponding decrease in accuracy is not observed in pose estimation error. This suggests that, if a sufficient number of accurate feature correspondences can be obtained, additional matches may not significantly improve accuracy. On the other hand, additional inlier matches improve robustness to detection

failure, compared to having a number of inliers very close to the minimum threshold. This is notable in the glare case, where pose estimates were not generated during multiple portions of the test due to an insufficient inlier count.

3.9 Free-Flight Test

A simple free-flight test was performed to validate the algorithm's performance in hover. In hover, the vision algorithm must tolerate vibration and small motions that are not present with a hand-held camera.

The ship-deck was secured to the floor for safety. The quadrotor was manually piloted and held in hover above the ship-deck for several seconds. The results of the test are given in Figures [3.27](#) and [3.28](#).

In hover, the algorithm is able to accurately estimate position and attitude, indicating it is viable for use in actual flight. The RMSE for pitch and roll are only 1.2 and 1.4 degrees, respectively, while an RMSE of 4 cm is present for the X-, Y-, and Z-position estimates. This indicates that the vibrations and small motions that occur in free hover do not significantly degrade algorithm performance.

3.10 Performance Comparison with 2D Vision Methods

The performance of the 3D feature-based vision algorithm is now compared to that of more typical 2D vision methods for ship-deck landing. The 3D feature-based algorithm offers significant advantages over 2D approaches, including its ability to work with 3D objects and its robustness to large rotations; it is therefore important to investigate whether it provides a

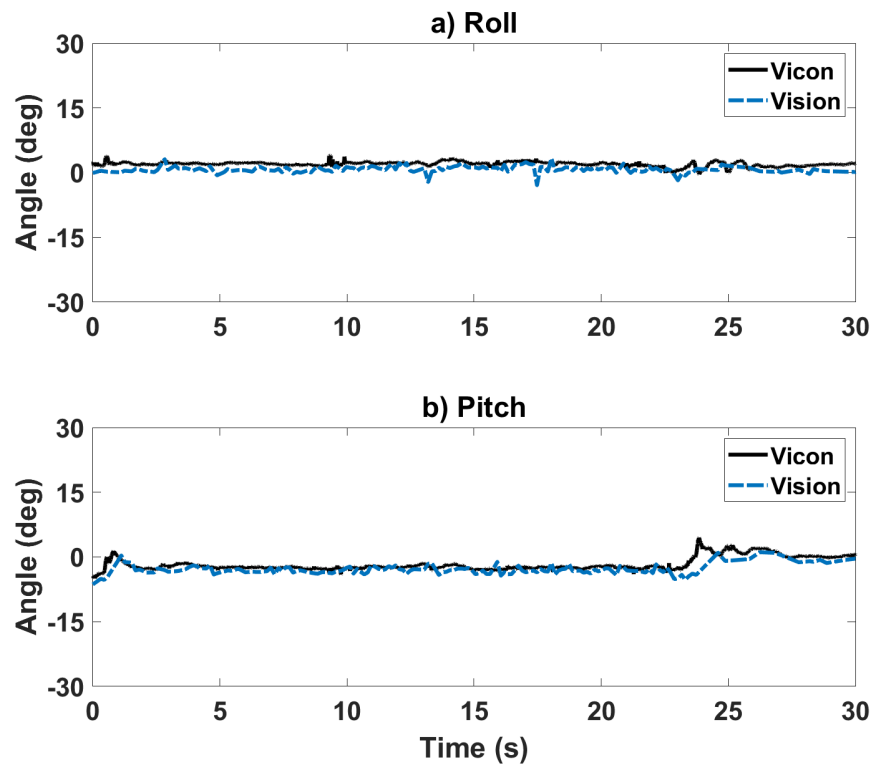


Figure 3.27: Estimation of relative attitude angles, with the quadrotor hovering above the ship-deck.

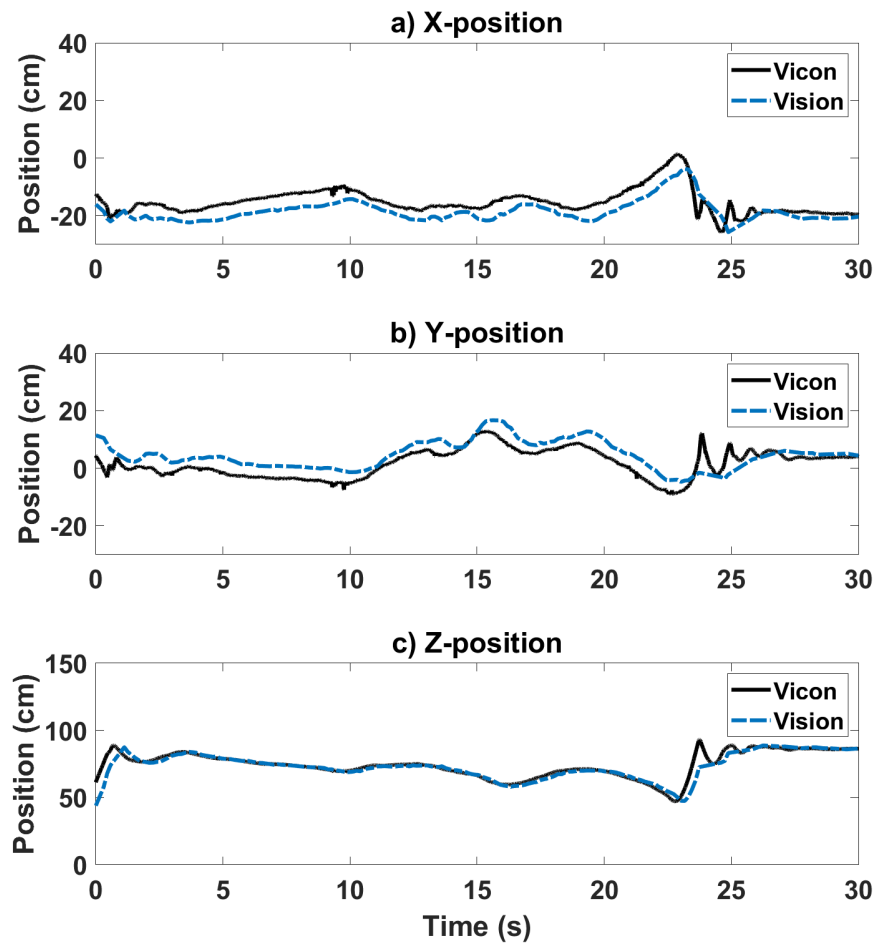


Figure 3.28: Estimation of relative position, with the quadrotor hovering above the ship-deck.

similar level of pose estimation accuracy. For consistency, the comparison will be limited to work evaluating small-scale UAVs.

When evaluating our previously developed 2D feature-based algorithm under similar test scenarios as those presented in this chapter, the pose estimation accuracy was found to be less than 4 cm along each linear axis and less than 2.5 degrees for pitch and roll [37]. Using a recursive AprilTag fiducial marker, Nicholson et. al. reported error in vision-based estimation of 8-23 cm in position and 1.8-3.1 degrees in rotation [29]. Using a typical H-shaped landing target, Lin et. al. observed position estimation error of 3-5 cm along each linear axis [31].

The test results for the 3D feature-based algorithm showed 0.5-6.3 cm of error along each linear axis and 1.2-2.4 degrees of error in pitch and roll. Its performance is therefore comparable to that of 2D vision approaches.

3.11 Increasing Algorithm Speed

The 3D feature-based algorithm is naturally far more computationally intensive than the 2D algorithm. A much larger number of features are extracted from the 3D object across many different camera positions, compared to the single image used for the 2D algorithm. Approximately 1,200 features were extracted from the 2D ship-deck image, while the 3D ship-deck image set provided 5,000-13,000 features, depending on feature detector and settings. This means that a much larger number of features must be matched in each update, increasing computational load dramatically.

For the ship-deck tests, the algorithm was able to update at a speed of 5-6 Hz. To be feasible for ship-deck pose estimation, at minimum, the algorithm must update at a sufficient frequency

to adequately capture the stochastic ship-deck motion. From analyzing Fourier transforms of the SCONE ship-deck motion histories, the maximum frequency of ship-deck motion is approximately 0.34 Hz. Therefore, this update frequency is adequately fast to track the motion in real-time. However, it is desirable to improve the update speed to improve the quality of pose estimation and to minimize delay in providing updated pose estimates to the controller. Additionally, more complex objects, such as real ships, may yield more features and corresponding points during feature extraction than the simplified object used here, increasing computational load. It is therefore important to explore techniques to potentially decrease computational time and increase update speed.

3.11.1 Filtering visible features

When updating the algorithm using a new frame, features detected in this frame are matched to features previously extracted from the 3D ship-deck. However, only a subset of the features on the 3D ship-deck will actually be visible in any single camera image. As feature matching is one of the most computationally demanding steps in the algorithm, restricting feature matching only to these visible features could potentially improve the speed of the algorithm.

The feature extraction process tracks which features are visible in each camera pose. From this, we can determine the ship-deck rotations where each feature should be visible. During real-time pose estimation, an initial pose estimate can be used to restrict feature matching in the next update to points visible in nearby rotations. The details of this process are described in the following subsections.

For this process, rotations were stored and processed as quaternions. This improves computational

efficiency and allows necessary mathematical operations to be performed; in particular, using quaternions, it is possible to calculate distances between rotations as a single angle. Additionally, methods have been developed to efficiently compute the average of multiple quaternions [47].

3.11.2 Process for filtering visible features

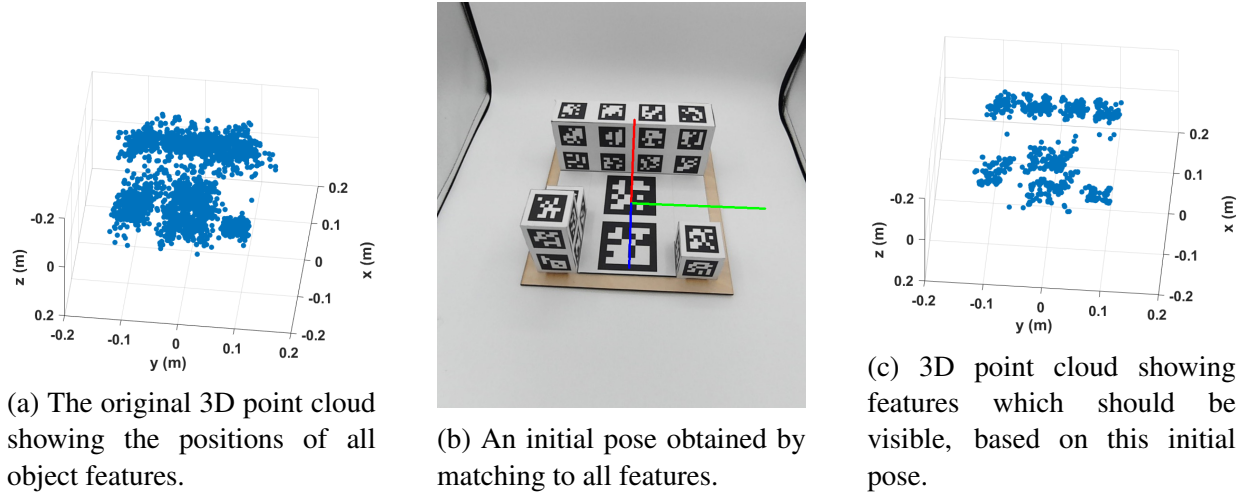


Figure 3.29: Images illustrating the steps in filtering visible features.

For each image of the ship-deck used in feature extraction, there is an associated camera pose. Its orientation is stored as a rotation vector, which is converted to a quaternion representation for the filtering process. The visible rotations of each feature were obtained using the following procedure:

1. From the feature extraction results, we can recover the camera poses where a feature on the ship-deck is visible. Each camera pose corresponds to a rotation.
2. Features may be visible in multiple camera poses. However, comparing multiple rotations for each feature is less computationally efficient than comparing a single rotation. Therefore,

an average rotation for each feature is found by calculating the average of quaternions for all rotations where the feature was detected.

3. The average rotation for each feature is then stored in an array. Once this has been repeated for all of the features, save this array with the features and 3D points for later use.

The pose estimation process is modified to include this filtering. This is described below:

1. Choose a rotation threshold for a feature to be considered visible. When calculating the distance between rotations, if a rotation is within this threshold, it will be considered visible. The threshold should be large enough to account for reasonable differences between the camera pose and feature rotations, as well as movement of the camera and platform between algorithm updates.
2. Obtain an initial ship-deck rotation. This is done by matching image features to all of the saved object features, the same as in the original algorithm.
3. For each stored feature, calculate the distance between its rotation and the initial rotation. Create a modified list of features which meet the rotation threshold.
4. Complete the feature matching step using the modified list of features. Use this to find an updated pose estimate.
5. Repeat this process while the ship-deck is in view of the camera. Each time a new pose estimate is obtained, the pose is used for the initial rotation in the next step.

Figure [3.29](#) illustrates the steps in this procedure.

3.11.3 Performance comparison

Basic validation of the computational speed and accuracy of the algorithm with this added filtering was then performed. Four cases were compared: 1) filtering with a rotation threshold of 60° ; 2) a rotation threshold of 75° ; 3) a rotation threshold of 90° ; 4) no filtering of visible features. The 111 initial images of the ship-deck were used for this test. The images were provided in sequence to the algorithm, and the pose estimates from the algorithm were compared to the ground truth poses of the ship-deck.

One goal of the visible feature filtering is to make slower feature detectors viable for real-time pose estimation. Because of this, the validation test deliberately used the feature detector ORB with $nFeatures=1000$, which required long computational times in earlier tests.

The results for computational time are provided in Table 3.7. The visible feature filtering results in a significant decrease in computational time compared to the algorithm without added filtering. This computational time decreases with smaller rotation thresholds.

Table 3.7: Mean computational time for visible feature filtering with different rotation thresholds.

Test case	Mean computational time
60° threshold	0.108 s
75° threshold	0.122 s
90° threshold	0.139 s
No filtering	0.199 s

The RMSE results for position and attitude are provided in Table 3.8. The error in estimated position is similar to the baseline case for both the 75° and 90° thresholds, but increase slightly for the 60° case. For the estimated roll and pitch, the error is modestly increased for the 60° and

75° cases. However, a larger increase in error is seen with the 90° rotation threshold.

Table 3.8: RMSE for visible feature filtering with different rotation thresholds.

Test case	X	Y	Z	Roll	Pitch
60° threshold	0.2 cm	0.6 cm	1.0 cm	3.9°	3.1°
75° threshold	0.2 cm	0.5 cm	0.8 cm	3.9°	2.6°
90° threshold	0.2 cm	0.6 cm	0.8 cm	4.0°	5.2°
No filtering	0.2 cm	0.5 cm	0.8 cm	3.7°	2.6°

From these results, visible feature filtering can be used to improve the vision algorithm’s computational speed, but a tradeoff must be made with the accuracy of the results.

3.12 Summary

This chapter covered the development of a new feature-based algorithm to estimate the pose of a 3D ship-deck. The algorithm was incorporated into on-board quadrotor vision hardware and thoroughly validated using a simulated 3D ship-deck with a Stewart platform. Algorithm performance with different feature detectors, specifically ORB and SIFT, was quantitatively evaluated. The algorithm was demonstrated to be capable of estimating ship-deck pose in real time under ideal conditions, Sea-State 6 ship-deck motion, and visually degraded conditions. Logged pose data was compared to Vicon ground-truth data. From these results, the following conclusions are drawn:

1. It is possible to use 3D computer vision to extract the pose of the ship-deck using matched features.
2. The algorithm demonstrates robustness to low illumination and occlusion, and was able to

estimate the pose of the moving-ship deck accurately in these conditions. However, the algorithm performance degraded in the glare condition, as it failed to detect the ship-deck for several seconds during the test.

3. In ideal visual conditions, the 3D algorithm is capable of estimating the pose of a stationary ship-deck, as well as a ship-deck undergoing Sea-State 6 motion. The algorithm showed comparable accuracy to 2D vision methods.
4. Different feature detectors present trade-offs for real-time performance. SIFT is accurate even at longer distances from the ship-deck, but is very computationally slow; ORB performs poorly at longer distances, but is much faster, making it viable for real-time use on the quadrotor computer hardware.
5. The 3D feature extraction process produces a large number of features, which has a significant penalty to algorithm speed due to the large number of features that must be matched. By determining which features are visible based on an initial orientation, and matching only to these features, it is possible to improve algorithm update speed significantly. However, this comes at the cost of increased error in pose estimation.
6. Filtering pose estimates based on the number of inlier points can improve the quality of algorithm output by removing poor quality results.

Chapter 4: Stereo Vision Algorithm

This chapter describes a new feature-based stereo vision algorithm capable of estimating the motion of arbitrary ship structures, eliminating some of the limitations of the algorithms described in Chapters 2 and 3. The new vision algorithm and its implementation are described in detail. Experimental validation results for a realistic simulated ship are presented.

4.1 Motivation for New Stereo Vision Approach

While the vision algorithms described in Chapters 2 and 3 can accurately estimate the ship-deck pose in challenging conditions, some important limitations were observed. The 2D feature-based algorithm demonstrated very robust and accurate performance when a customized, feature-rich landing pad image was used. Though the algorithm can be used with realistic landing pad images, the low number of features resulted in significantly reduced accuracy as well as frequent failures to detect the ship-deck due to insufficient feature matches. The 3D feature-based algorithm does not need a customized landing pad image, but it still required images of the ship and significant pre-processing burden to extract features and feature locations. The need for pre-collected images meant advance knowledge of the visual appearance of the ship was required, limiting the ability to land on ships which are not known in advance.

It is therefore desirable to develop a new vision approach which does not rely on customized

images or tags, and which requires minimal pre-processing steps and stored data. This chapter describes a new feature-based algorithm utilizing a stereo camera to meet these requirements. The algorithm can estimate the motion of arbitrary ship structures, removing the need for advance knowledge of the ship's appearance.

4.2 Hardware and Test Setup

As in the previous chapters, our vision algorithm for ship-deck landing continues to utilize only on-board computer and vision hardware, as external communication may be unreliable or unavailable during real ship landings.

To allow for more efficient implementation and testing with the stereo vision hardware, a new quadrotor utilizing primarily off-the-shelf hardware was used, rather than the custom quadrotor UAV described in Chapter 2. The quadrotor incorporates all of the necessary hardware for on-board processing and autonomous operation. The details of the aircraft and its vision hardware are provided below.

4.2.1 Quadrotor details

The quadrotor is shown in Figure 4.1. Including all mounted vision hardware and battery, the vehicle has a gross take-off weight of 2.5 kg. The aircraft uses a Holybro X500 v2 carbon fiber frame, measuring 354×354 mm (measured from the centers of the motors). It features four 10-inch diameter, two-bladed rotors driven by Holybro 920 KV 2216 brushless electric motors. The motors, flight controller, and flight computer are powered using a single 5200 mAh, four-cell (14.8 V) lithium polymer battery.

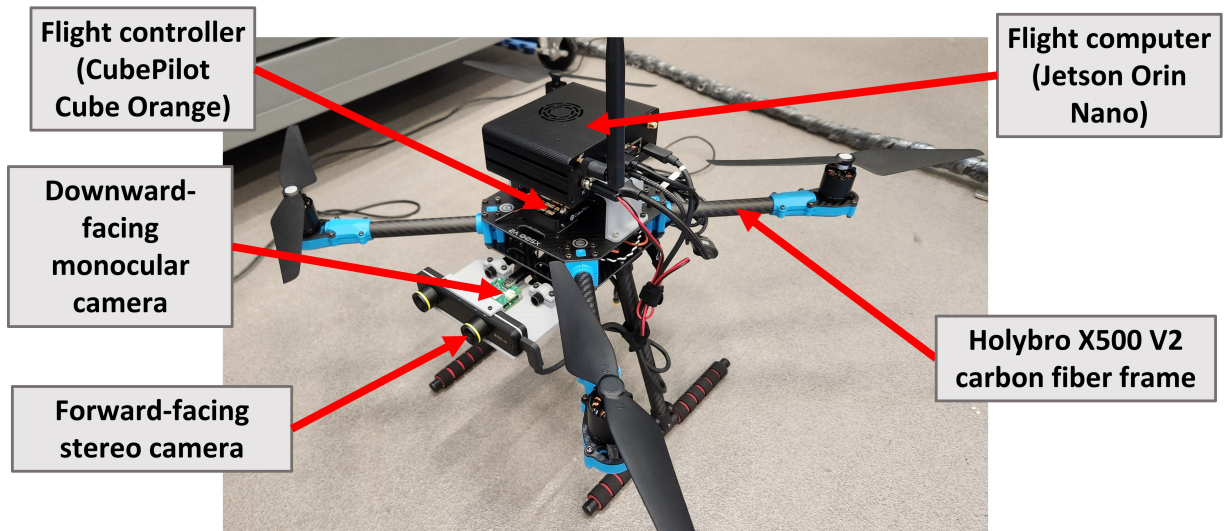


Figure 4.1: The quadrotor UAV incorporating on-board computer and vision hardware.

The aircraft is controlled using a CubePilot Cube Orange flight controller. The controller incorporates an ARM Cortex-M7 processor and uses the ArduPilot open-source autopilot software. It contains redundant integrated sensors, specifically three IMUs (Inertial Measurement Units), two barometers, and one magnetometer. A connected radio receiver, paired to a transmitter, is used to receive manual commands from a pilot.

4.2.2 Cameras

The quadrotor is equipped with a forward-facing stereo camera to evaluate the new stereo vision algorithm. The selected stereo camera is the Stereolabs ZED Mini with a stereo baseline (spacing between left and right cameras) of 63 mm. Parameters such as resolution, frame rate, and exposure time can be modified as needed using manufacturer-provided software. Additionally, the aircraft incorporates a downward-facing monocular camera which is used to obtain images of the ship-deck should the 2D feature-based algorithm be required. The selected camera, the See3Cam USB camera manufactured by e-con Systems, is the same camera used for our previous

vision algorithms. It features a built-in autoexposure algorithm, which allows the the camera to automatically compensate for changes in illumination.

4.2.3 Flight computer

An embedded flight computer is integrated into the quadrotor UAV avionics hardware. The computer runs the vision algorithm in real-time and uses the pose estimation results to control the quadrotor relative position during ship-deck landing. Commands and messages are sent to the flight controller using a serial connection.

In our previous work, all vision processing was performed on an UP Xtreme embedded computer featuring an Intel i5 CPU. The results revealed limitations with running the vision algorithms on CPU only, particularly when matching large numbers of features [37,42]. Stereo cameras also introduce additional processing steps, such as extracting and matching features between the left and right images.

A new flight computer, the Nvidia Jetson Orin Nano Developer Kit, was selected to address this issue. The computer features an Arm Cortex-A78AE processor; unlike the previous flight computer, this embedded computer incorporates an additional Nvidia GPU. In contrast to a CPU, a GPU is capable of running many operations in parallel, making it ideal for tasks such as image processing. The GPU can be accessed and utilized for computational tasks using Nvidia CUDA (Compute Unified Device Architecture) software.

The increase in computational speed was validated using a simple feature extraction and matching example; the test measured the total time to extract and match features between two images. Using CPU-only functions, the task took 0.07 s. With the GPU, the task required only

0.02 s, a significant improvement.

4.3 Feature-Based Stereo Vision Algorithm

The new vision approach is based on the ability of stereo cameras to provide depth and disparity information, which can be used to obtain 3D information using only the images from the camera. Monocular cameras, such as those used in the previous feature-based algorithms described in Chapters 2 and 3, cannot directly provide this information. Using feature matching, the algorithm extracts 3D points from visible ship-structures and uses the motion of these points between camera frames to estimate the motion of the ship. Unlike our previous feature-based algorithms, the stereo vision algorithm works with any arbitrary 3D structures with sufficient features. This allows the algorithm to be potentially used for landing on any ship, with no advance knowledge of the ship's visual appearance or geometry.

4.3.1 Stereo vision

Stereo cameras consist of two separate image sensors with a fixed baseline (distance between them). This is in contrast to monocular cameras, which utilize a single image sensor. Each stereo camera frame consists of images taken simultaneously from both sensors. Typically, the image sensors are placed in a parallel configuration as illustrated in Figure 4.2.

Using the two images obtained in each frame, it is possible to determine the locations of points in 3D space relative to the stereo camera. This requires finding 2D point locations in each image corresponding to the same point in 3D. Because the camera parameters and baseline are known, the 3D location of the point relative to the camera can then be recovered. This process

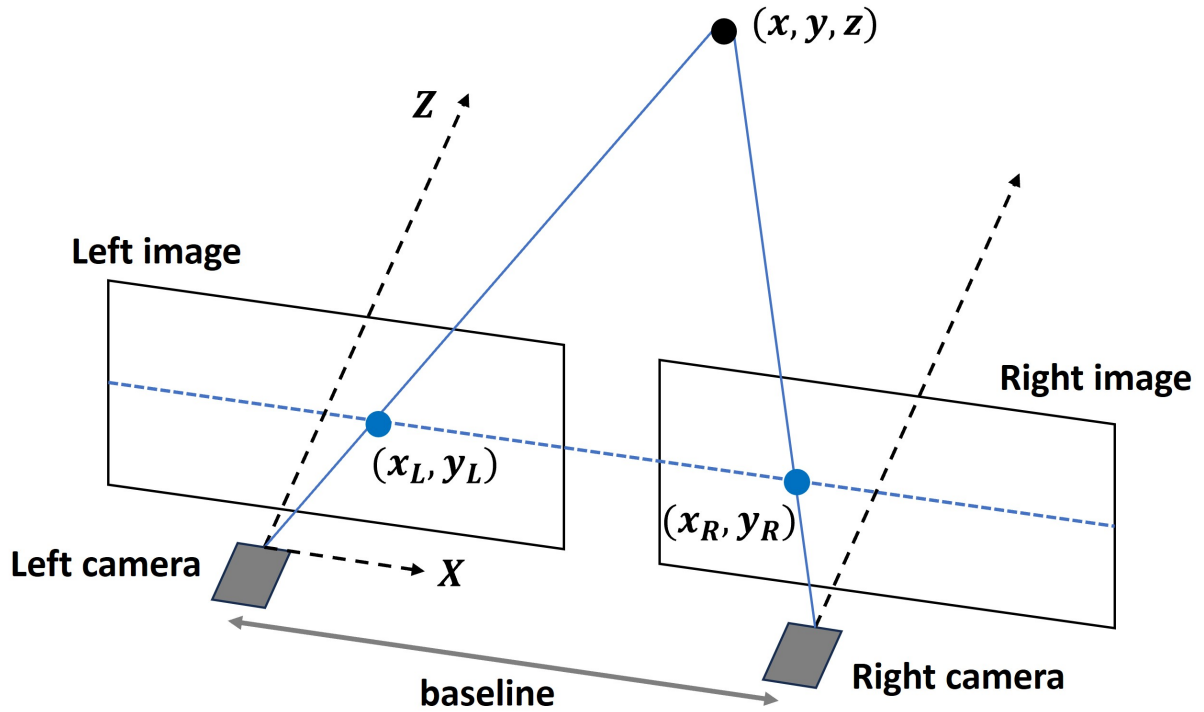


Figure 4.2: Triangulation of a point in 3D space using a stereo camera with two parallel image sensors.

is known as triangulation; more detail on this process is provided in Chapter 3. As before, an OpenCV implementation of triangulation was used. As shown in Figure 4.2, if a parallel stereo camera is used, the corresponding points should fall on the same horizontal line in both images.

Using stereo cameras, it is also possible to generate depth maps using the horizontal disparity between the left and right images. Depth maps consist of an estimated depth (Z-distance from camera) corresponding to each pixel in the camera image. Using manufacturer-provided software libraries, estimated depth maps can be generated in real-time as new camera frames are obtained. These depth maps are used for refinement steps in the algorithm, as described later in this chapter.

4.3.2 Feature matching

As discussed in greater detail in the previous chapters, features are small patches of interest in an image. They are robust to transformations in scale, translation, illumination, and rotation. Locations of features are typically found by detecting strong corners or blobs, which have a strong gradient in pixel intensity in two directions. Image patches at these locations are extracted and converted to a numerical vector representation known as a descriptor. By comparing descriptors, features can be matched to find corresponding points across different images. Good features should be distinctive, meaning that they can be reliably matched across multiple images; low-contrast or nondistinctive features are not desirable for feature matching.

Multiple algorithms have been developed to detect and extract features in images. One commonly used feature detector is the Scale Invariant Feature Transform (SIFT) algorithm, developed by Lowe [14]. It is invariant to changes in translation and scale, and somewhat robust to variations in rotation and illumination. The algorithm provides generally good feature matching results between images, but is computationally demanding [43]. SIFT was used for the 2D feature-based algorithm due to its robustness for feature matching [37].

The Oriented FAST and Rotated BRIEF algorithm is another commonly used feature detector [15]. The algorithm offers a significant improvement in computational speed compared to SIFT, but with lower robustness [43]. Because of the lower computational time required, ORB was utilized for our previous 3D feature-based algorithm, which required matching large numbers of features.

As with the previous vision algorithms for ship-deck pose estimation, the stereo vision algorithm relies on extraction and matching of features from camera frames. Feature matches

provide corresponding points between the left and right camera images, which can then be triangulated to provide 3D point locations. This is illustrated in Figure 4.3 (a) and explained in more detail in the following sections.

4.3.3 Estimating rigid body transformations

For each new stereo camera frame, the algorithm computes a set of 3D points. To estimate the motion of the ship (or other 3D structures) relative to the camera, it is necessary to compute the motion of these 3D points between frames. To accomplish this, a rigid body transformation, consisting of rotation and translation, is computed between corresponding 3D points in the current and previous camera frame. Rotation and translation are applied to a 3D point using Equation 4.1:

$$p' = Rp + T \quad (4.1)$$

where R is a 3×3 rotation matrix from the previous to current frame, T is a 3×1 translation vector relative to the previous frame, and p and p' are 3×1 vectors containing the 3D point coordinates before and after applying the transformation, respectively.

Multiple methods exist to compute the rigid body transformation between two corresponding sets of 3D points [48]. For the vision algorithm, the method described by Arun et. al. is used [49]. The method is summarized here for completeness.

Arun's method computes the optimal rotation R and translation t between two sets of corresponding 3D points $\{p_i\}$ and $\{q_i\}$, $i = 1, 2, \dots, N$, with random noise present in the corresponding point locations. The recovered transformation minimizes the least squared error between the two

sets of corresponding points, given in Equation 4.2.

$$\Sigma^2 = \sum_{i=1}^N ||q_i - Rp_i - T||^2 \quad (4.2)$$

The centroids of the point sets are defined as follows:

$$\bar{p} = \frac{1}{N} \sum_{i=1}^N p_i \quad (4.3)$$

$$\bar{q} = \frac{1}{N} \sum_{i=1}^N q_i \quad (4.4)$$

To isolate the rotation matrix in Equation 4.2 (removing the translation term), the two point sets can be modified to share the same centroid:

$$p_{c_i} = p_i - \bar{p} \quad (4.5)$$

$$q_{c_i} = q_i - \bar{q} \quad (4.6)$$

Equation 4.2 then becomes the following:

$$\Sigma^2 = \sum_{i=1}^N ||q_{c_i} - Rp_{c_i}||^2 \quad (4.7)$$

The rotation matrix that minimizes equation 4.7 can be found using a singular value decomposition (SVD) approach. First, define the 3×3 matrix:

$$H = \sum_{i=1}^N p_{c_i} q_{c_i}^T \quad (4.8)$$

The SVD of this matrix is $H = U\Lambda V^T$. The optimal rotation matrix is calculated as:

$$R = VU^T \quad (4.9)$$

Using this rotation matrix, as well as the centroids of the point sets found earlier, the translation is then calculated as:

$$T = \bar{q} - R\bar{p} \quad (4.10)$$

For an individual pair of corresponding points, the error (distance) after applying the optimal transformation is given by Equation 4.11:

$$e_i = q_i - Rp_i - T \quad (4.11)$$

As this method minimizes the error across all points, the result is sensitive to outliers caused by incorrect point correspondences resulting from poor matches, as well as points not located on the moving ship. To ensure reliable results, it is necessary to mitigate the effect of outliers when computing rotation and translation.

One technique commonly used in computer vision is the RANSAC (Random Sample Consensus) algorithm [39]. The algorithm works by repeatedly taking samples of data and fitting a model to each sample, in order to find a model that maximizes the number of inlier data points. The final result should therefore only be influenced by inlier data points, ignoring outliers entirely. For the vision algorithm, the procedure for estimating rotation and translation is modified accordingly using the following steps:

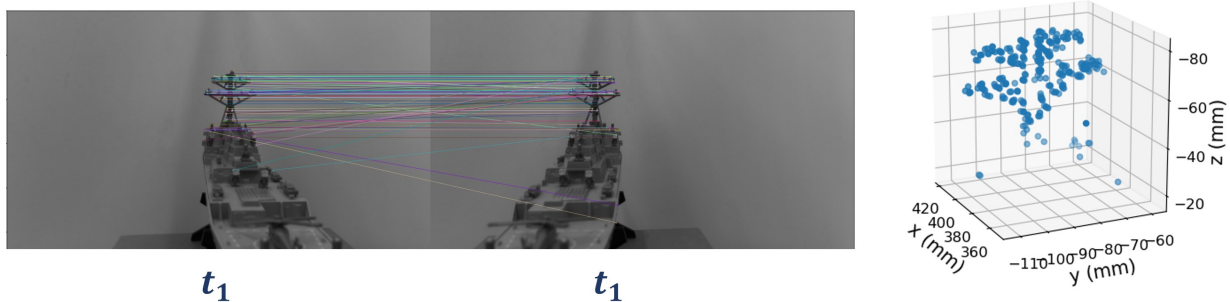
1. Using feature matching and triangulation as discussed above, obtain two sets of corresponding 3D points.
2. Randomly select 3 pairs of corresponding points (the minimum needed to calculate the transformation).
3. Compute an estimated rotation and translation using these selected points.
4. Apply the rotation and translation to all of the points in the first set of 3D points. Compute the error for each pair of corresponding 3D points using Equation 4.11. Any point pairs within a specified error threshold will be marked as inliers, and are considered outliers otherwise.
5. Repeat steps 2-4 a specified number of times. Keep the rotation and translation which results in the highest count of inliers.
6. Using the final set of inlier points (removing any outliers), again calculate the rigid body transformation. This will optimize the rotation and translation estimate across all inlier points.

4.3.4 Algorithm overview

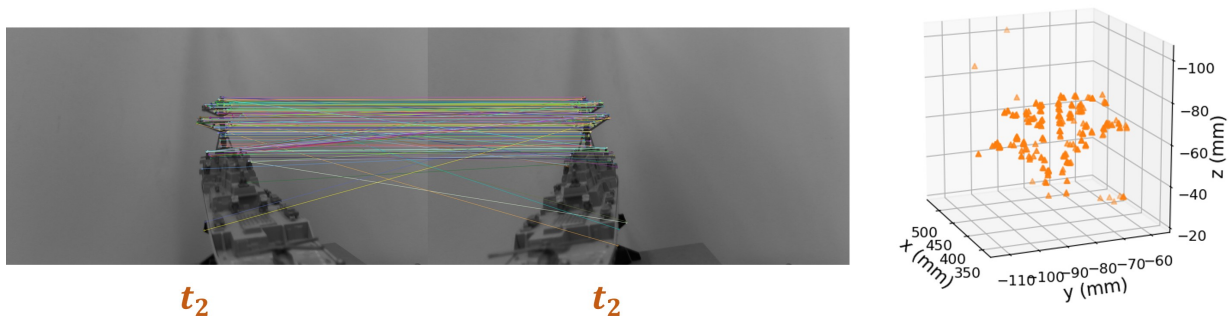
The stereo vision algorithm utilizes these techniques to estimate the motion of the ship.

The steps in this process are described below:

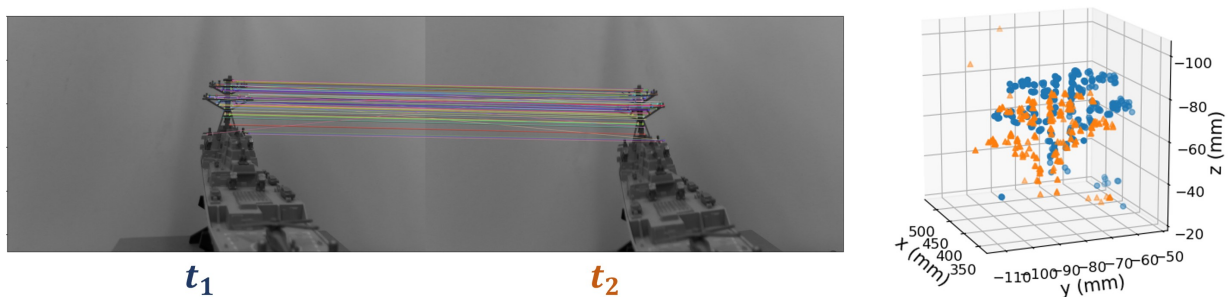
1. Obtain a new frame from the stereo camera. This frame includes the images from both the left and right cameras. The depth maps will also be generated at this time.



(a) Feature matching between the left and right stereo camera images. These features are triangulated to obtain 3D point locations on the ship.



(b) Feature matching and triangulation is repeated for the next stereo camera frame.



(c) Features are matched between the current and previous frames. This generates two sets of corresponding 3D points from the current and previous camera frames. A rigid body transformation between these sets of points is calculated to estimate the motion between frames.

Figure 4.3: Illustration of steps in the stereo vision algorithm.

2. Extract and match features from the left and right camera images to obtain corresponding points, as shown in Figure 4.3a. As discussed above, with stereo cameras, corresponding points should fall on the same horizontal line in both images. Therefore, remove any pairs of corresponding points that fail this condition, within a specified tolerance.
3. Triangulate the corresponding points using the known stereo camera parameters to obtain a set of 3D points, as shown in Figure 4.3a.
4. Use the depth map to further refine the recovered 3D points. For each triangulated point, compute the percent error between the triangulated depth (Z-distance) and the estimated value from the depth map. If the percent error is greater than a threshold, remove the 3D point.
5. The recovered 3D points are defined relative to the stereo camera's origin (center of the left camera), with the Z-direction facing forward. As it is more convenient to calculate motion in the aircraft coordinate frame (Z-down), transform the recovered points to the aircraft coordinate system.
6. Repeat steps 1-5, computing 3D points and their corresponding features for a new camera frame. This is illustrated in Figure 4.3b.
7. Match the features corresponding to the 3D points in the current frame to the stored features from the previous frame. This step obtains the corresponding 3D points in the current and previous frame, as shown in Figure 4.3c.
8. Using the corresponding 3D points, compute the rigid body transformation using the procedure detailed above. This procedure will produce a rotation and translation between the points

in the first and second frame.

9. The recovered rotation and translation is used as an estimate of relative motion between the camera and scene, in this case, the camera and ship.

4.4 Ship-Deck Pose Estimation

The stereo vision algorithm does not estimate the full pose of the ship; rather, it estimates the change in rotation and translation of the ship relative to the camera between frames. Therefore, it is necessary to provide an initial estimate of the ship pose. Without additional corrections, the pose estimate will drift over time. To correct for this, it is necessary to also obtain pose estimates from a second source that do not suffer from drift over time.

Our previous 2D feature-based algorithm is suitable for this purpose. The algorithm provides a full estimate of the ship-deck pose relative to the camera in real-time by matching features from a camera to a reference image. The algorithm can be used with any 2D reference image, allowing it to use existing landing pad or flight deck markings. The disadvantage is that the accuracy of pose estimates will be reduced due to the low number of features, and the algorithm may fail to detect the ship-deck at times during landing. Combining this algorithm with the stereo vision algorithm can counteract these limitations.

4.4.1 Kalman filter for fusing vision algorithm results

Output from both vision algorithms can be fused in a Kalman filter. This can provide more accurate estimates of ship-deck pose than relying on a single vision algorithm, and mitigate noise or errors in individual vision-based estimates. Continuous updates from the 2D feature-based

algorithm ensure that the pose estimate will not drift over time. Additionally, in the event that one of the vision algorithms fails, the filter can continue to generate estimates based on measurements from the other algorithm, providing additional robustness.

An important consideration is that vision algorithm results can have significant time delays due to camera latency and processing time, requiring modifications to the standard Kalman filter to account for this. Larsen et. al. provide a modified form of the discrete Kalman filter which accounts for delayed measurements [50]. These modified equations were used to implement a constant-velocity Kalman filter to fuse measurements from both vision algorithms while accounting for time delays. Individual filters are used to estimate linear position in the X, Y, and Z-directions as well as roll, pitch, and yaw.

For the 1D constant-velocity Kalman filter with no time delay corrections, the equations (for time k) are summarized below. A constant-velocity Kalman filter was also used in Chapter 2 to filter results from the 2D feature-based algorithm.

Predict:

$$\hat{x}_k = F_k \hat{x}_{k-1} + G_k w_k \quad (4.12)$$

$$P_k = F_k P_{k-1} F_k^T + Q_k \quad (4.13)$$

where $\hat{x}_k$ is the state estimate at time k , P_k is the covariance matrix, and w_k is the process noise. Equation 4.12 calculates the predicted state estimate before performing the update (correction) step. The state transition matrix F_k and gain matrix G_k are given as

$$F_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \quad G_k = \begin{bmatrix} \frac{1}{2} \Delta t^2 \\ \Delta t \end{bmatrix}$$

Update:

$$K_k = P_k H_k^T (H_k P_k H_k^T + R_k)^{-1} \quad (4.14)$$

$$\tilde{y}_k = z_k - H_k \hat{x}_k \quad (4.15)$$

$$\hat{x}_k = \hat{x}_k + K_k \tilde{y} \quad (4.16)$$

$$P_k = (I - K_k H_k) P_k \quad (4.17)$$

where K_k is the Kalman gain matrix, $\tilde{y}_k$ is the measurement residual, z_k is the measurement of the state (vision estimate in this case), R_k is the observation noise, and H_k is the measurement model. For position measurements, it is given as

$$H_k = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

while for velocity measurements, it is defined as

$$H_k = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Equation 4.16 provides the final state estimate.

The equations are modified using results in [51]. A delayed estimate will be fused by the filter at time k , but the measurement corresponds to an earlier time s . Additionally, N measurements from another sensor (the other vision algorithm in this case) may be fused by the filter between time s and time k .

Equation 4.15, the residual, is modified as follows:

$$\tilde{y}_k = z_k - H_s \hat{x}_s \quad (4.18)$$

$\hat{x}_s$ is the value of $\hat{x}$ that would have been used if the filter was updated at time s .

Equations 4.14 and 4.16 are also modified accordingly:

$$K_k = M_* P_s H_s^T (H_s P_s H_s^T + R_k)^{-1} \quad (4.19)$$

$$P_k = P_k - K_k H_s P_s M_*^T \quad (4.20)$$

The correction term M_* is given by

$$M_* = \prod_{i=0}^{N-1} (I - K_{k-i} H_{k-i}) F_{k-i-1} \quad (4.21)$$

In these equations, the $k - i$ subscript refers to the values of these parameters during previous filter updates. Note that for measurements with no time delay, these equations simply reduce to the standard (non-modified) update equations.

In the filter implementation, buffers are used to store a set number of previous values of the quantities needed for these modified equations. Separate buffers are used to store previous values of the time t , the state transition matrix F_k , the predicted state estimate $\hat{x}_{k|k-1}$, the Kalman gain K_k , and covariance matrix P_k . When fusing a new measurement, the new values are added to the buffers and the oldest values are discarded. For quantities at time s , the value is interpolated using the stored time values and previous values from the buffers.

To measure computational delay, timestamps are logged at the beginning and end of a vision algorithm update. The camera latency is assumed to be constant. The delay is treated as

the sum of the camera latency and computational delay.

Estimates from the 2D feature-based algorithm are treated as time-delayed position measurements. The time delay correction is crucial for this algorithm due to the high latency of the monocular camera. Individual results from the stereo vision algorithm estimate motion between frames; therefore, results are converted to linear and angular velocity estimates when received by the filter. These are then treated as non-delayed velocity estimates.

As an initial pose is required for the stereo vision algorithm, the filter is initialized with an estimate from the 2D feature-based algorithm. Once initialized, the filter fuses estimates from either vision algorithm as they are received.

Since the actual process noise and observation noise are not known, these parameters were tuned by hand using logged test data to achieve reasonable accuracy and noise filtering. The covariance matrix was initialized with small values along the diagonal.

4.5 Full Algorithm Implementation and Architecture

The full vision algorithm was implemented on the on-board flight computer. ROS 2 (Robotics Operating System 2), a set of robotics libraries, was used to enable asynchronous communication between individual processes, or nodes, and to allow functions within each node to be called at specified time intervals. The stereo vision algorithm and 2D feature-based algorithm run on separate ROS 2 nodes, allowing both algorithms to operate in parallel.

Due to its advantage in computational speed, the ORB feature detector is used for the stereo vision algorithm. As with our previous work, the SIFT algorithm is utilized for the 2D feature-based algorithm because of its improved robustness.

Implementations from the open-source computer vision library OpenCV were used for tasks such as feature extraction and matching and basic image processing. The library was compiled from source to enable CUDA support, allowing some tasks to be performed on the GPU.

The stereo vision algorithm utilizes both CPU- and GPU-based computation to improve performance. Stereo camera images are initially obtained and processed on the CPU. OpenCV CUDA functions are used to upload the images to the GPU and perform ORB feature extraction and matching, the most computationally demanding steps in the stereo vision algorithm. The matched features are then downloaded to the CPU for additional filtering.

OpenCV does not provide a CUDA implementation of SIFT, so the 2D feature-based algorithm runs entirely on the CPU. As the stereo vision algorithm heavily utilizes the GPU, this allows computational load to be distributed between the CPU and GPU.

4.6 Experimental Validation

Before it can be incorporated into a full vision-based landing system, the vision algorithm must first be validated in controlled scenarios.

The test setup for the full vision algorithm simulates a realistic ship landing scenario. A 3D-printed model of ship structures was created to test the stereo vision algorithm. The model consists of the mast and a portion of the superstructure of a DDG-51 ship, simulating the portions of the ship visible to a pilot during an actual landing. The reference image used for the 2D feature-based algorithm is shown in Figure 4.4. This image is representative of the flight deck markings on a DDG-51 ship to create a more realistic test setup. As is the case with typical landing pad

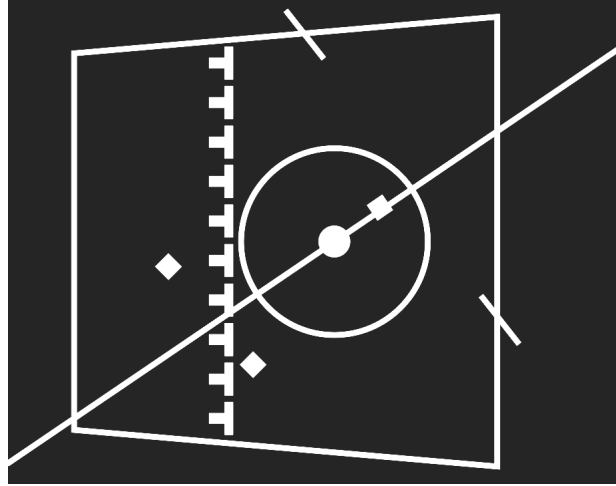


Figure 4.4: The reference image used for the 2D feature-based algorithm. The image is similar to the flight deck markings on a DDG-51 ship.

markings, this image is low in features; our previous tests of the algorithm indicate that pose estimation accuracy will be reduced compared to custom reference images rich in features [37].

Both the 3D model and reference image were incorporated into a simulated ship, shown in Figure 4.5. A backdrop depicting a cloudy sky, also shown in the Figure, was used for all tests unless otherwise specified. The simulated ship is attached to a Stewart platform, which was used to generate ship-deck motions during testing.

Algorithm results provide an estimate of the ship-deck pose relative to the aircraft. For the purposes of the vision algorithm, the aircraft origin is located at the location of the downward-facing monocular camera, with the Z-axis down, X-axis forward, and Y-axis facing right. Note that this is forward of the center of the quadrotor; this location is chosen to simplify algorithm implementation. The origin of the simulated ship is the center of the 2D flight deck image. For simplicity, it uses a Z-down, X-forward (towards the backdrop), and Y-right coordinate system, consistent with the aircraft coordinate axes.

For all tests, stereo camera images were collected at a resolution of 3840×1080 pixels

(1920×1080 pixels for each camera) and processed by the vision algorithm without scaling. Monocular camera images were collected at an original resolution of 1920×1080 pixels. As SIFT is much more computationally expensive at higher resolutions, the images were scaled down to one-third of their original resolution, or 640×360 pixels.

A Vicon motion-capture system was used to collect ground-truth data. The system tracks the true poses of the quadrotor and simulated ship-deck during each test. This data was used to recover the true pose of the ship-deck relative to the camera for validation of vision algorithm results.

All tests were conducted in normal indoor lighting (300 lux).

4.7 Initial Validation and Refinement

Before evaluating more challenging cases, initial tests were completed to validate the algorithm's performance under ideal conditions and ensure that its computational speed is sufficient for real-time performance. Refinements to the algorithm to improve its performance in more difficult scenarios were also investigated.

4.7.1 Baseline performance test

The algorithm was first evaluated under ideal conditions to verify a baseline level of performance. For this test, the simulated ship was secured in a fixed, horizontal position. The quadrotor was held stationary above the 2D reference image for several seconds, with the reference image and ship structures fully in view of the monocular and stereo cameras, respectively.

The results for attitude and position estimation are shown in Figures [4.6](#) and [4.7](#), while the

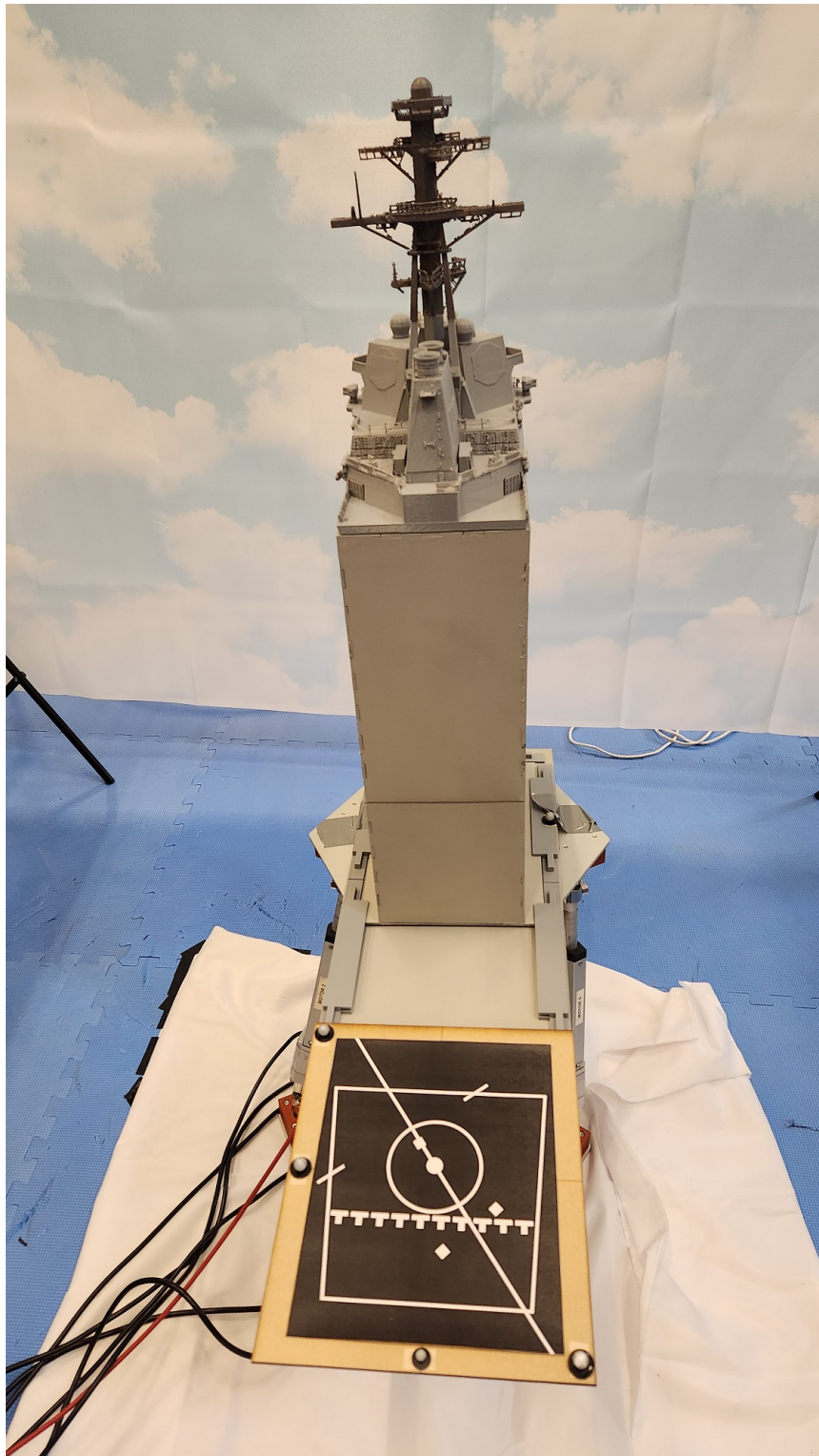


Figure 4.5: The simulated ship used to experimentally validate the new vision algorithm. The simulated ship incorporates the 2D reference image as well as 3D ship structures.

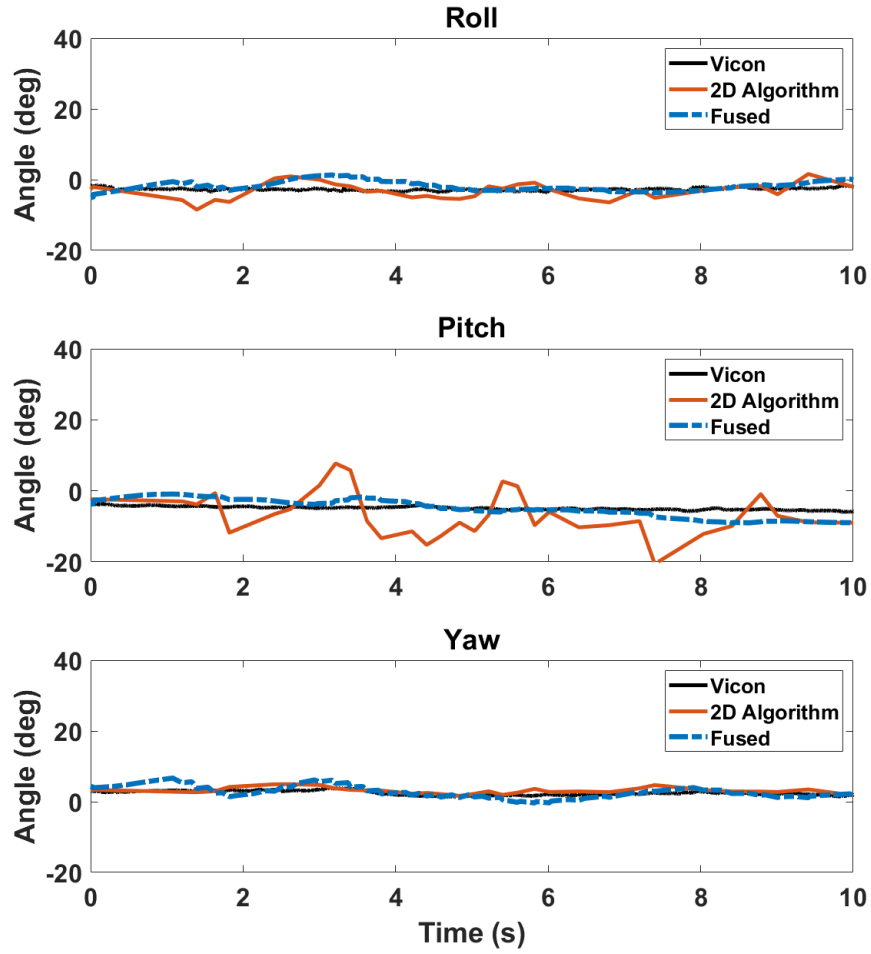


Figure 4.6: Estimation of aircraft attitude angles relative to a stationary simulated ship. Results are compared for the 2D feature-based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.

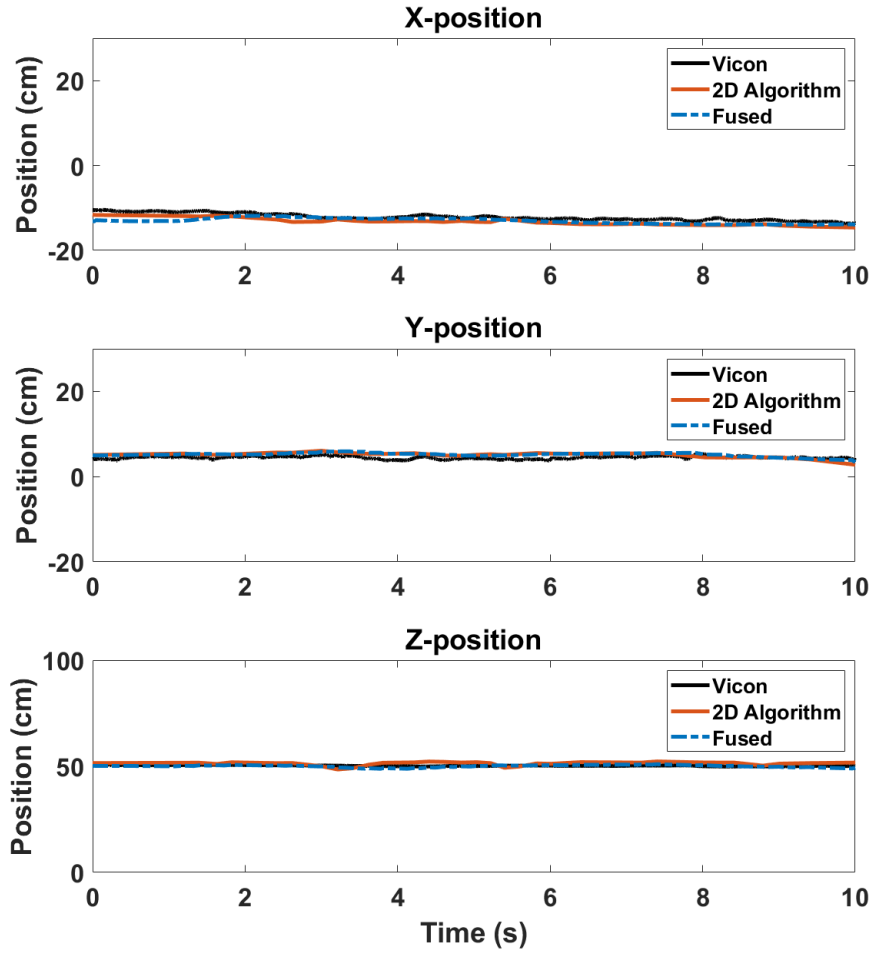


Figure 4.7: Estimation of aircraft position relative to a stationary simulated ship. Results are compared for the 2D feature-based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.

Table 4.1: RMSE for the ideal baseline test case.

Test case	X	Y	Z	Roll	Pitch	Yaw
2D algorithm	1.1 cm	0.9 cm	1.4 cm	2.4°	6.4°	1.0°
Fused	1.1 cm	0.8 cm	0.5 cm	1.8°	2.2°	1.3°

RMSE results are summarized in Table 4.1. The results for X-position, Y-position, and yaw are similar for both the 2D feature-based algorithm (2D Algorithm) and full vision algorithm (Fused) result. However, the fused result shows improvement over the 2D algorithm for Z-position, roll, and pitch; in particular, fusing the 2D and stereo algorithm results mitigates errors of several degrees in the pitch estimation.

Analyzing the stereo vision algorithm results individually can provide additional insight. To obtain these results, an initial pose was obtained from the Vicon ground truth data. The estimated change in position and rotation between each frame, obtained from logged stereo vision algorithm output, was used to generate pose results starting from the initial pose without any additional corrections. This simulates a scenario in which continuous full pose estimates are not available, such as if the 2D feature-based algorithm fails to detect the ship-deck for several seconds.

The results are shown in Figures 4.8 and 4.9. The estimates of X-position, Z-position, and yaw show minimal drift over the 10-second time period, indicating that the estimated motion is very accurate. However, minor drift is observed in the roll and Y-position estimates due to error in the estimated motion, and larger drift occurs in the yaw estimate.

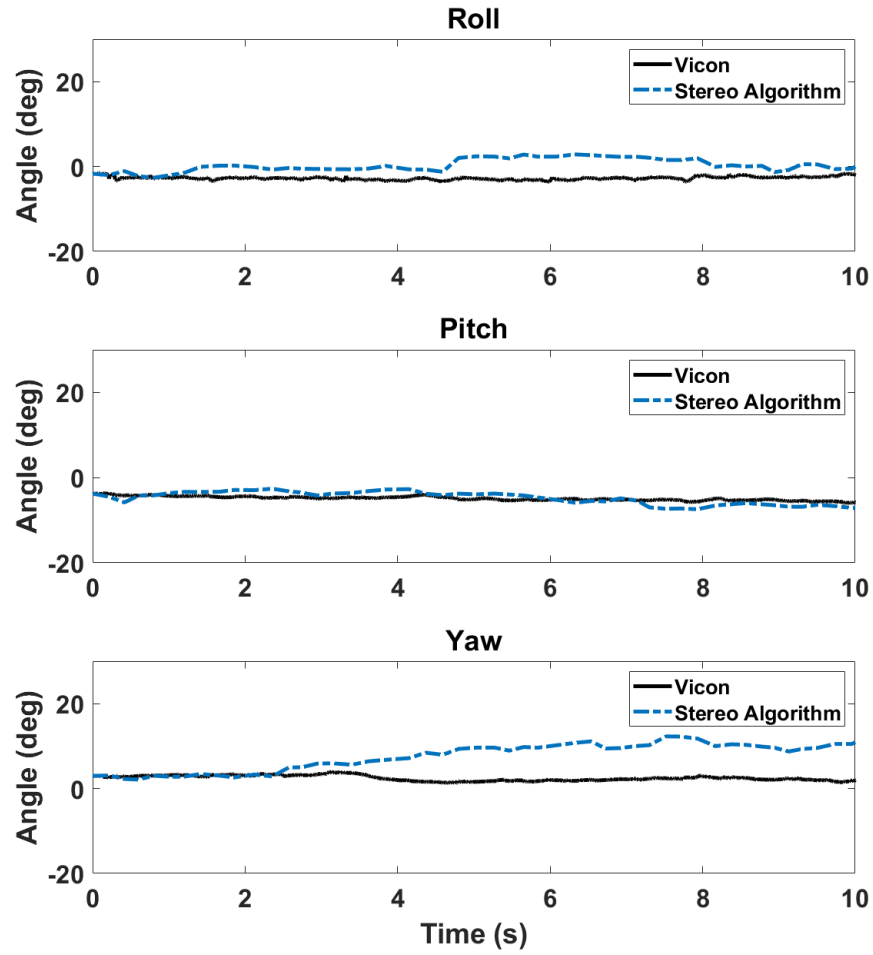


Figure 4.8: Estimation of aircraft attitude angles relative to a stationary simulated ship using only stereo vision algorithm results. The initial pose is obtained from Vicon data.

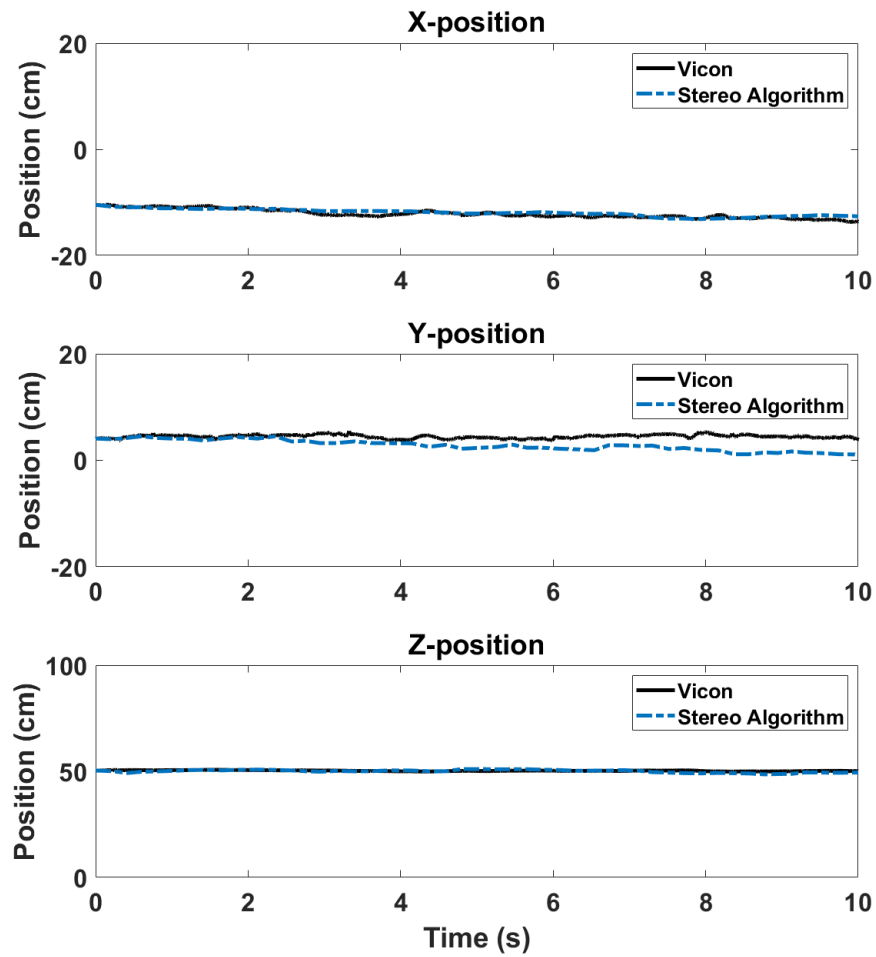


Figure 4.9: Estimation of aircraft position relative to a stationary simulated ship using only stereo vision algorithm results. The initial pose is obtained from Vicon data.

4.7.2 Computational speed

The full ship-deck pose estimation approach requires both the stereo vision algorithm and the 2D feature-based vision algorithm to run in parallel, which is significantly more computationally demanding than running a single algorithm alone. While the stereo vision algorithm utilizes the GPU for the most demanding steps, feature extraction and matching, the other steps in the algorithm are performed using the CPU. The 2D feature-based algorithm exclusively utilizes the CPU.

For practical ship-deck landing, both algorithms must run in real-time with an update rate sufficient to estimate the ship-deck motion. Therefore, the computational speed of each algorithm was evaluated using the selected vision hardware. First, the mean update rate of the algorithm was measured for the 2D feature-based algorithm and stereo vision algorithms alone. Next, the update rate was measured with both algorithms running in parallel. For the parallel scenario, it was further investigated whether the CPU-intensive 2D feature-based algorithm negatively affects the update rate of the stereo algorithm, which utilizes a combination of CPU- and GPU-based computation. Two specific cases were evaluated: 1) No limit on the update rate of the 2D feature-based algorithm; and 2) 2D feature-based algorithm limited to a maximum rate of 5 Hz. For the test, the quadrotor and simulated ship were held stationary to ensure similar conditions for each test case. For all cases, the mean power consumption was also evaluated.

The computational speeds for the individual vision algorithms are summarized in Table 4.2. Both algorithms update at a sufficient frequency to estimate ship-deck motion, which displays frequencies up to 0.34 Hz in the SCONE dataset.

Table 4.3 summarizes the mean update rates with both algorithms operating in parallel. As

Table 4.2: Update speed and power consumption for the two individual vision algorithms.

Algorithm	Update frequency	Power consumption
Stereo algorithm	7.7 Hz	14 W
2D algorithm	13.3 Hz	16 W

Table 4.3: Update speed and power consumption for the two vision algorithms running in parallel.

Setting	Update frequency (Stereo)	Update frequency (2D)	Power consumption
Not limited	5.0 Hz	10.1 Hz	19 W
Limited	4.6 Hz	4.3 Hz	17 W

expected, in the first case (no limit), the update rates for both algorithms decrease compared to operating alone. The update rate for the stereo vision algorithm is similar in both cases, indicating that the 2D feature-based algorithm can be updated at a higher rate if needed. However, limiting the update rate may be advantageous to reduce power consumption and leave sufficient CPU resources available for the autonomous landing controller.

4.7.3 Feature match, inlier, and error thresholds

During each update, the stereo algorithm extracts and matches features from camera frames to estimate the ship motion. However, incorrect matches can result in incorrectly triangulated 3D points and inaccurate 3D point correspondences. This can result in inaccurate estimated motion, especially with a small total number of feature matches.

To improve the quality of estimated motion results, thresholds can be applied to the total number of matched features between each frame, which corresponds to the number of corresponding

3D points used to estimate motion. Thresholds can also be applied to the number of inlier point pairs, as well as the final least squared error (from Equation 4.2) after computing a rigid body transformation. Estimates falling above or below these specified thresholds are then removed, as the resulting estimated motion is more likely to be incorrect.

A vision algorithm test was completed to determine good thresholds for feature matches, inliers, and final error. For the test, the quadrotor was held by hand behind the simulated ship. The quadrotor was moved with rapid, random motions in attitude and position to stress the algorithm performance and produce a mixture of good and poor estimates.

At each update, the stereo algorithm calculates the change in rotation and translation from the previous update. The algorithm output, as well as the timestamp at each update, were logged. Vicon data was used to compute the true change in relative rotation and translation between each timestamp. From this, the error in the total estimated translational motion was calculated. Figure 4.10 illustrates the variation in error over time, as well as the corresponding number of feature matches and the final error in the rigid body transformation.

From the figure, high errors in attitude and position estimation correspond to a decreased number of feature matches. Additionally, increased error in the rigid body transformation is observed at several of these instances. Based on these results, thresholds for number of matches, number of inliers, and maximum error were selected. The selected values were a minimum of 100 feature matches and 60 inliers, and a maximum allowable error of 10 mm. The thresholds for feature matches and final error are shown in Figure 4.10.

Table 4.4 provides the RMSE results for the estimated motion with and without these thresholds applied. With these thresholds, the error in estimated linear and attitude motion decreases significantly.

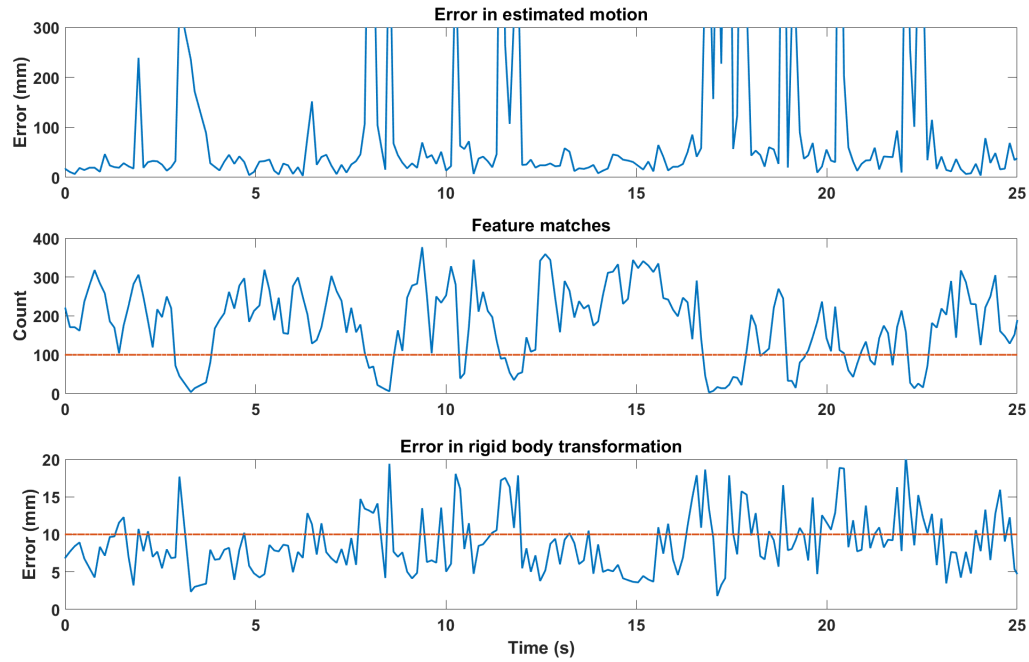


Figure 4.10: The error in estimated translational motion compared with feature match count and final least squared error in rigid body transformation. Horizontal lines illustrate selected thresholds for minimum feature matches and maximum error.

Table 4.4: RMSE with and without thresholding applied.

Test case	X	Y	Z	Roll	Pitch
No thresholding	290.5 mm	133.4 mm	145.4 mm	36.7°	10.1°
Thresholding	28.2 mm	30.1 mm	53.7 mm	2.7°	2.9°

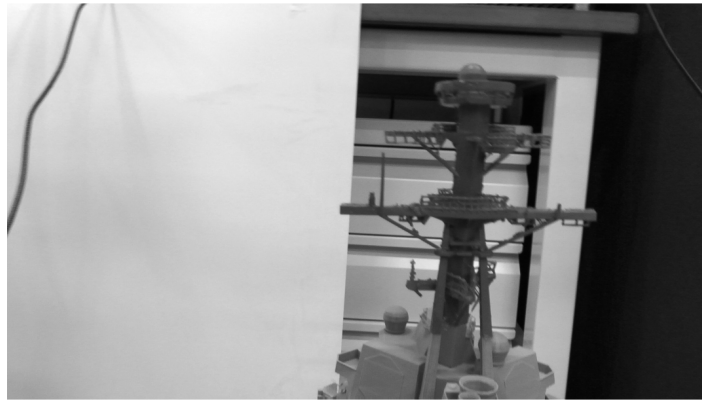
4.7.4 Isolating moving ship structures from background

When estimating ship motion, the stereo vision algorithm may recover features and corresponding 3D points from the stationary background as well as the moving ship structures. If there are few features in the background, the RANSAC-based method for estimating rotation and translation is adequate to exclude background points as outliers. However, if there are a large number of background features, additional steps are necessary to isolate the features and 3D points belonging to the moving ship.

The generated depth maps can be used to remove the background in a simple and computationally efficient manner, while requiring only minimal modification to the stereo vision algorithm. Image pixels with a depth greater than a specified maximum distance (based on the approximate physical dimensions of the ship) are replaced with a set value, as illustrated in Figure 4.11. The background removal is performed prior to feature extraction and matching.

This approach is advantageous because it limits feature extraction to the ship structures, maximizing the number of recovered features and 3D points on the ship and preventing incorrect matches to background features. Additionally, as it only requires the depth maps, it is very simple to implement. However, computational load is increased as depth maps must be generated for both the left and right images (rather than just the left image). Higher-quality depth maps, which increase the accuracy of background removal, also increase computational burden.

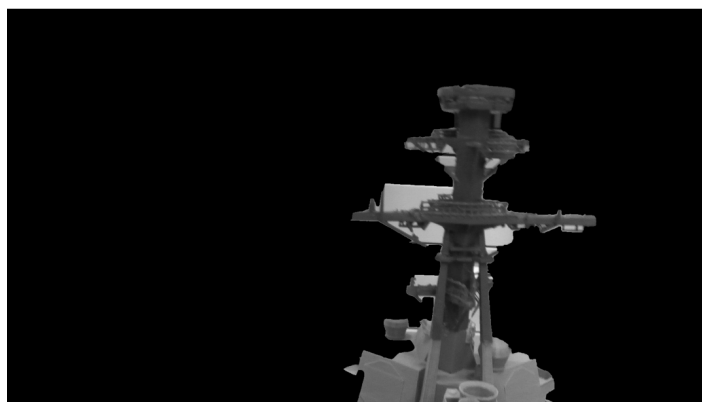
Two depth settings were tested: 1) Performance, which increases computational efficiency with reduced accuracy; and 2) Ultra, which generates more accurate results but is more computationally demanding. As a baseline, results were also collected without background removal performed. Each setting was tested in two scenarios: 1) minimal background features (white background),



(a) Image from left camera. Both the ship and background are visible.



(b) The depth map used to remove the image background. The depth map provides an estimated distance from the camera in millimeters for each pixel in the image.



(c) Image pixels with a depth greater than a specified threshold are replaced with a set value, leaving the ship structures of interest.

Figure 4.11: Using depth map to remove image background and isolate moving ship structures.

Table 4.5: Mean update frequencies for different settings.

	Setting	Update frequency
White background	None	7.8 Hz
	Performance	6.6 Hz
	Ultra	4.0 Hz
Cluttered background	None	7.4 Hz
	Performance	7.3 Hz
	Ultra	4.4 Hz

and 2) significant background detail.

The stereo camera was held stationary, with the ship model in frame. Ship-deck motion was applied to the model to ensure significant motion compared to the static background. For each combination of depth setting and background, the following data was logged during estimation of the rigid body transformation: the number of corresponding pairs of 3D points, the number of inlier 3D point pairs, and the final error across all inlier points (using equation 4.2).

The mean feature correspondences and inliers are illustrated in Figure 4.12. With minimal background detail, a large number of corresponding points were found for all cases, with a high proportion of these points being inliers. Similar numbers of points and inliers were found without background removal and background removal with the Performance depth setting. However, more points were recovered with the Ultra depth map setting.

However, much lower numbers of corresponding point pairs were recovered with the cluttered background with background removal performed. The lowest numbers of points were obtained with the Performance depth map setting. For the cluttered background case with no background removal performed, it was observed that the significant majority of recovered features were

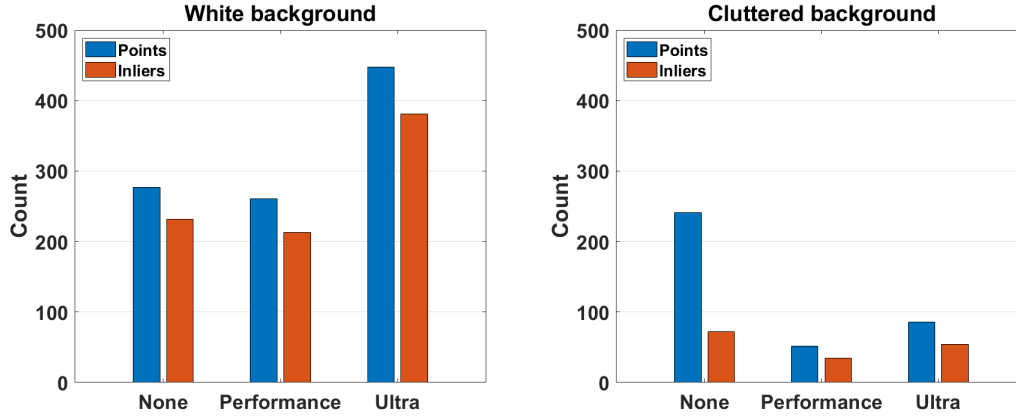


Figure 4.12: Comparison of mean corresponding 3D point pairs and inliers for each setting.

obtained from background details rather than the ship. The high level of detail also resulted in many incorrect matches. This resulted in a high number of corresponding points compared to the cases with background removal, but a lower proportion of inlier points.

The mean update frequencies for each case are provided in Table 4.5. For both the white background and cluttered background, background removal decreases the algorithm update frequency. This decrease is modest for the Performance setting and much greater for the Ultra setting.

4.8 Ship-Deck Motion

The algorithm performance was then evaluated for stochastic Sea-state 6 ship-deck motion generated by the Stewart platform. The ship motion, which is representative of the motions of a full-scale DDG-51 ship, was obtained from time histories provided in the Navy SCONE dataset [6]. The attitude (roll, pitch, and yaw) motions are generated directly from the time history, while the translational (surge, sway, and heave) motions are scaled to 0.5% of their original values due to the smaller experimental setup. For these tests, the update rate of the 2D feature-based algorithm was limited to 5 Hz.

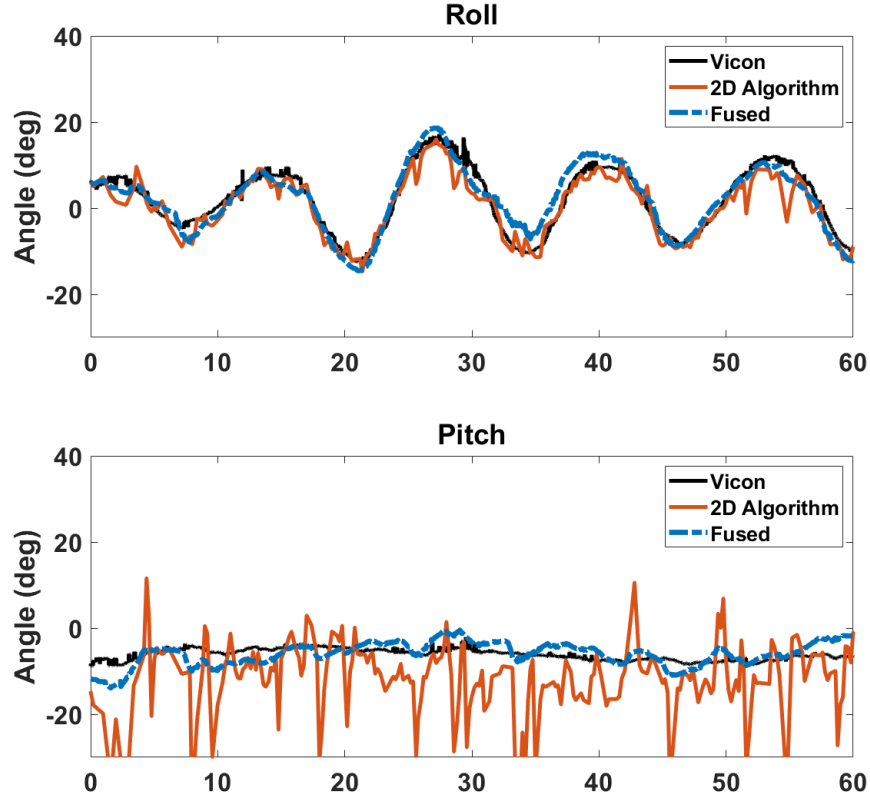


Figure 4.13: Estimation of attitude angles relative to a simulated ship undergoing Sea-state 6 motion, with the quadrotor held stationary. Results are compared for the 2D feature-based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.

For the first test, the quadrotor was held stationary by hand similarly to the baseline test, simulating a stable hover condition. The attitude and position results are shown in Figures 4.13 and 4.14, respectively. With the 2D feature-based algorithm alone, large errors are present in the pitch estimates, while moderate noise is observed in the roll estimate. For both roll and pitch, these errors are significantly reduced in the fused algorithm output, and the estimate follows the ground truth much more closely. For position, both the 2D and fused algorithm results closely follow the ground truth.

For the next test, the quadrotor was moved with varying motions in attitude and position,

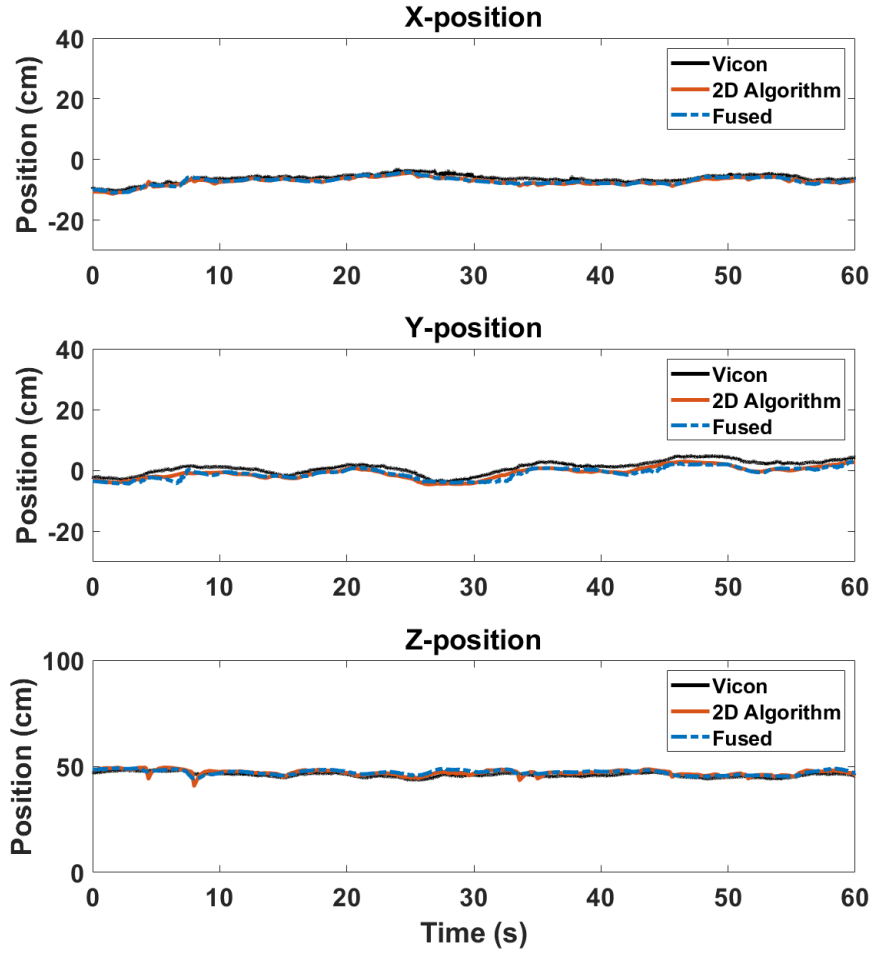


Figure 4.14: Estimation of position relative to a simulated ship undergoing Sea-state 6 motion, with the quadrotor held stationary. Results are compared for the 2D feature-based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.

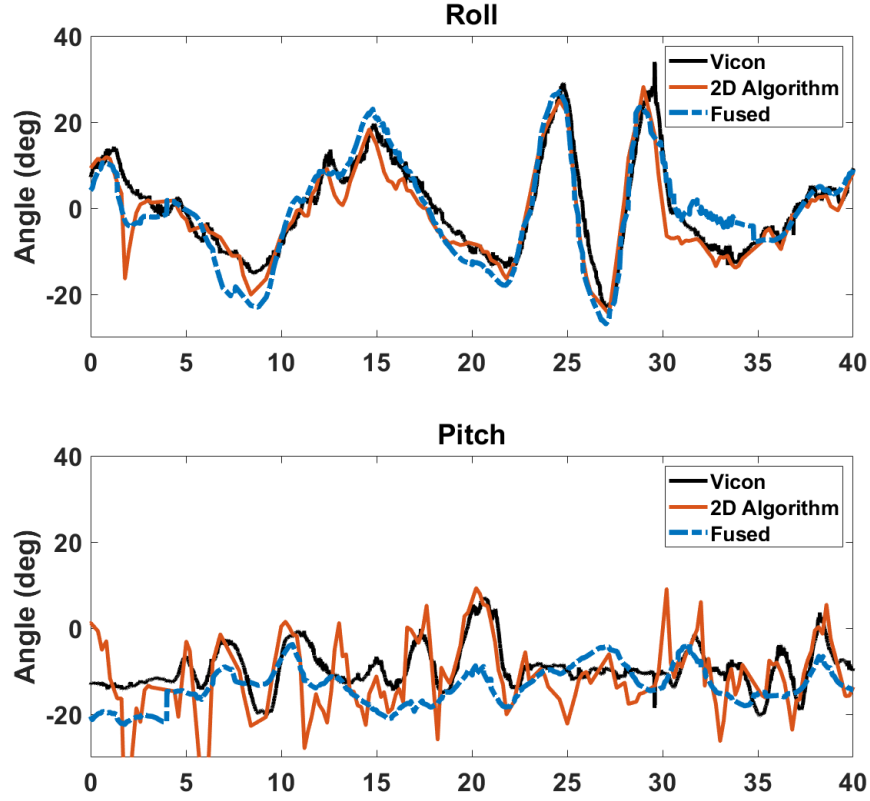


Figure 4.15: Estimation of attitude angles relative to a simulated ship undergoing Sea-state 6 motion, with the quadrotor undergoing large attitude and linear motions. Results are compared for the 2D feature-based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.

creating large and complex relative motion between the camera and simulated ship. The attitude and position results are shown in Figures 4.15 and 4.16, respectively. For roll, X-position, and Z-position, both the 2D feature-based and fused algorithm results reasonably follow the ground truth, while errors are present in the Y-position estimate. As with the previous test, the 2D feature-based algorithm again shows significant noise in the pitch estimate. While this noise is mitigated in the fused algorithm output, the resulting pitch estimate also fails to capture several rapid pitch motions.

The vision algorithm performance in both cases can be quantitatively evaluated using the

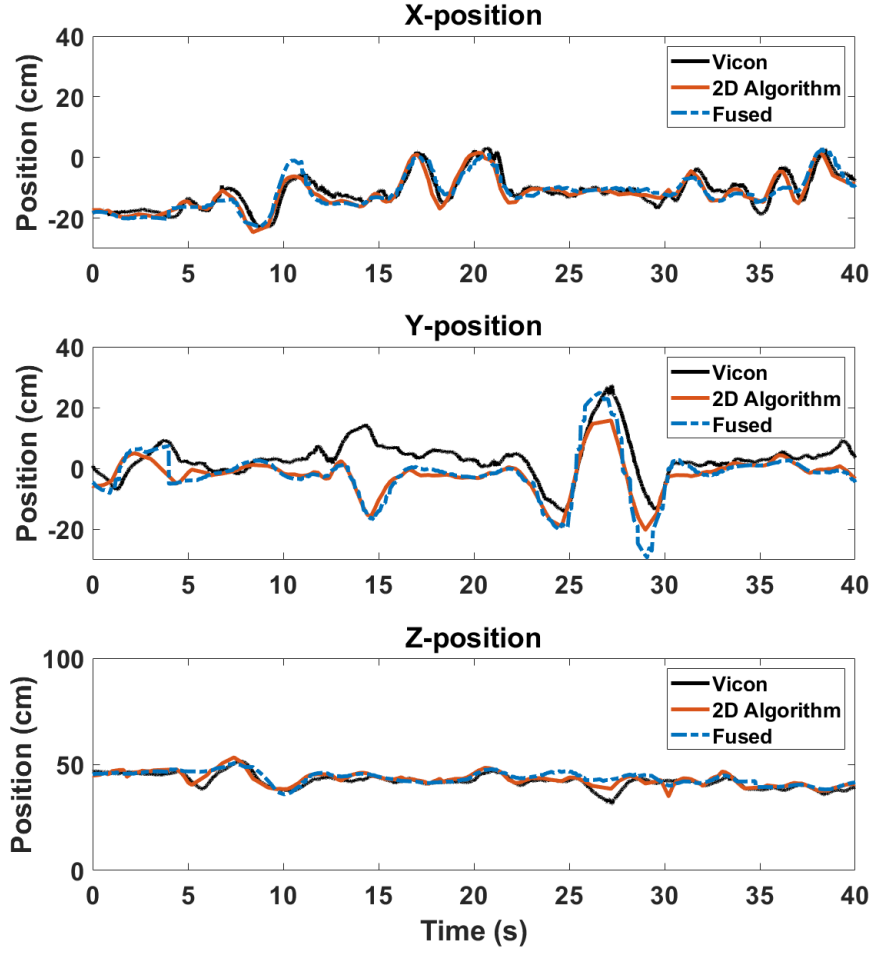


Figure 4.16: Estimation of position relative to a simulated ship undergoing Sea-state 6 motion, with the quadrotor undergoing large attitude and linear motions. Results are compared for the 2D feature-based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.

Table 4.6: RMSE for ship-deck motion cases.

Test case		X	Y	Z	Roll	Pitch
Quadrotor stationary	2D algorithm	1.0 cm	1.8 cm	1.3 cm	2.8°	10.3°
	Fused	1.0 cm	2.0 cm	1.4 cm	2.6°	2.4°
Quadrotor moving	2D algorithm	2.6 cm	7.7 cm	1.9 cm	4.7°	8.1°
	Fused	2.6 cm	8.6 cm	2.6 cm	4.7°	7.0°

RMSE of position and attitude. These results are summarized in Table 4.6. For the first test, with the quadrotor held stationary, both the 2D feature-based algorithm and fused result show similar error in position estimation. However, the fused result demonstrates improved accuracy in attitude estimation, particularly for pitch. For the second case with large-angle motions, the errors increase significantly for both vision algorithms. While the fused result again displays increased accuracy for estimated pitch and similar accuracy for X-position and roll, the error is modestly increased over the 2D feature-based algorithm for Y- and Z-position.

4.8.1 Ship motion estimation with stereo algorithm only

As with the baseline test, additional insight can be gained by evaluating the results of the stereo vision algorithm alone. Similarly to before, the results for the full pose estimate were obtained using an initial pose from Vicon ground truth data. Additionally, using logged timestamps from the stereo vision algorithm, Vicon data was used to obtain the true change in relative pose between each vision algorithm update. These ground truth results were compared to the estimated change in pose from the stereo vision algorithm.

The results for the first test (quadrotor stationary) are shown in Figures 4.17 and 4.18, while the results for the second test (quadrotor moving) are provided in Figures 4.19 and 4.20.

For the first test, minimal drift is observed for the X-position and Z-position estimates, while modest drift is observed for the pitch estimate. Larger errors, however, are present in the estimates of Y-position and roll, with the Y-position estimate in particular showing much larger motions than are present in ground truth. For the second case, all estimates show either drift or significant error over the experimental time window. For position, the stereo vision-based

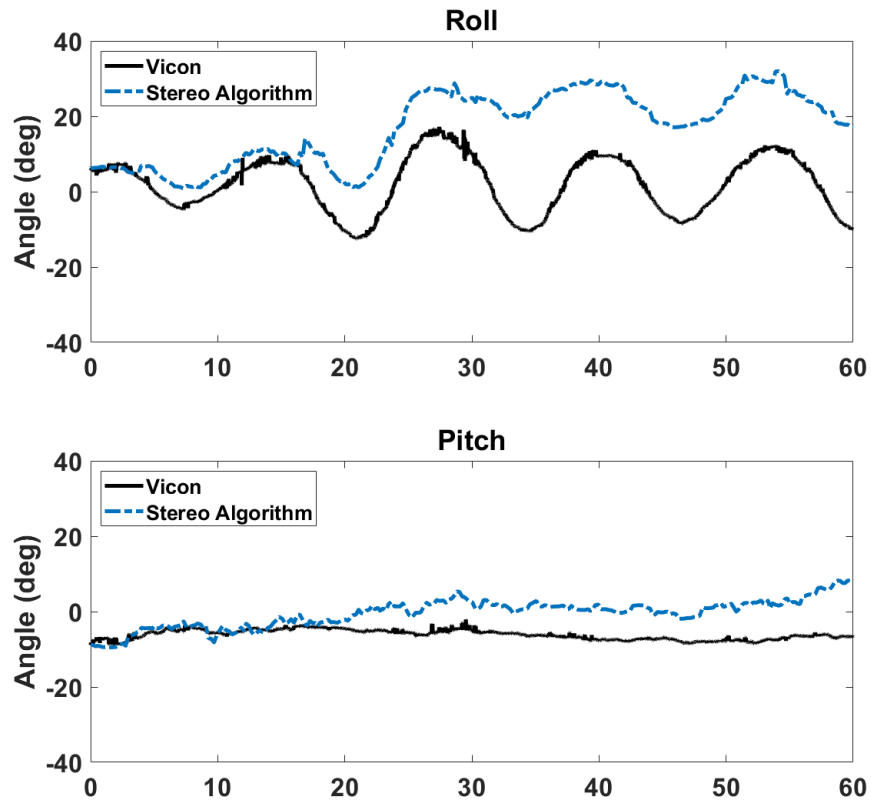


Figure 4.17: Estimation of aircraft attitude angles relative to a stationary simulated ship using only stereo vision algorithm results. The initial pose is obtained from Vicon data.

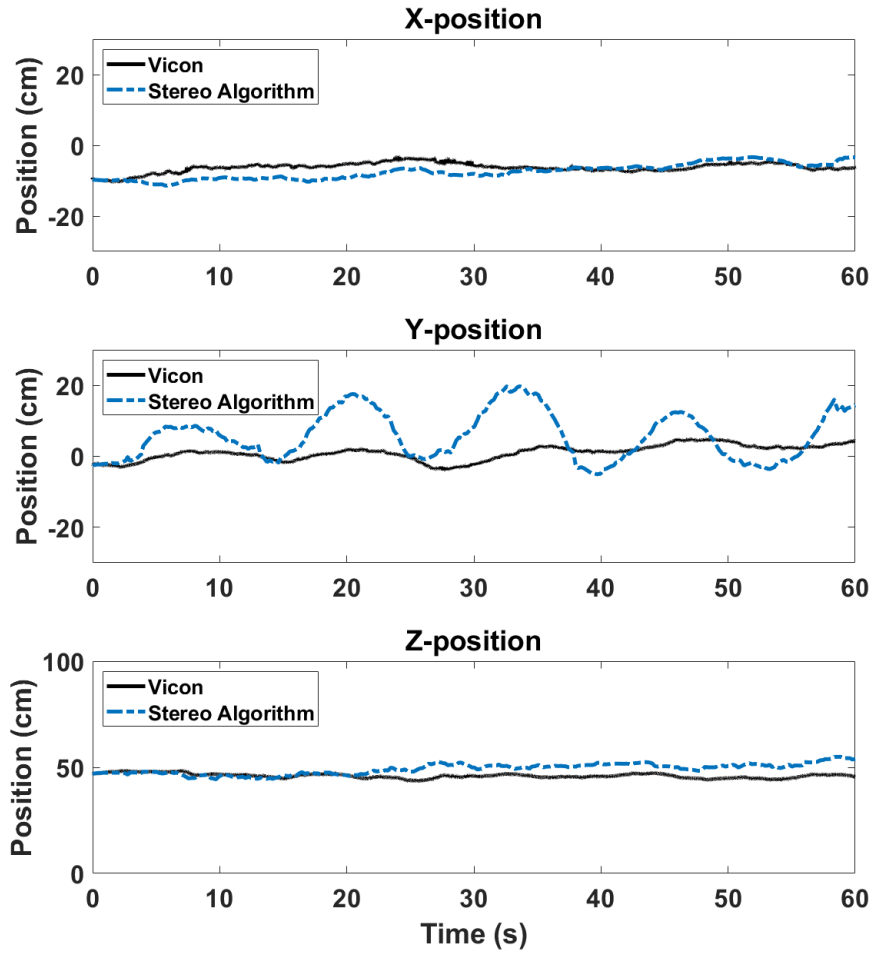


Figure 4.18: Estimation of aircraft position relative to a stationary simulated ship using only stereo vision algorithm results. The initial pose is obtained from Vicon data.

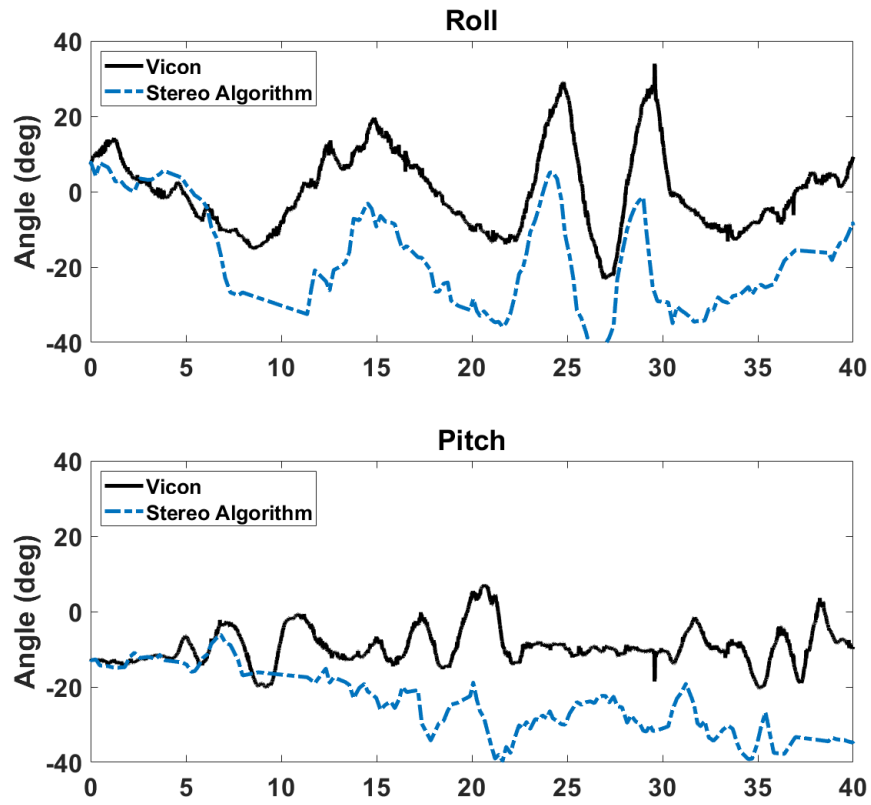


Figure 4.19: Estimation of aircraft attitude angles relative to a stationary simulated ship using only stereo vision algorithm results. The initial pose is obtained from Vicon data.

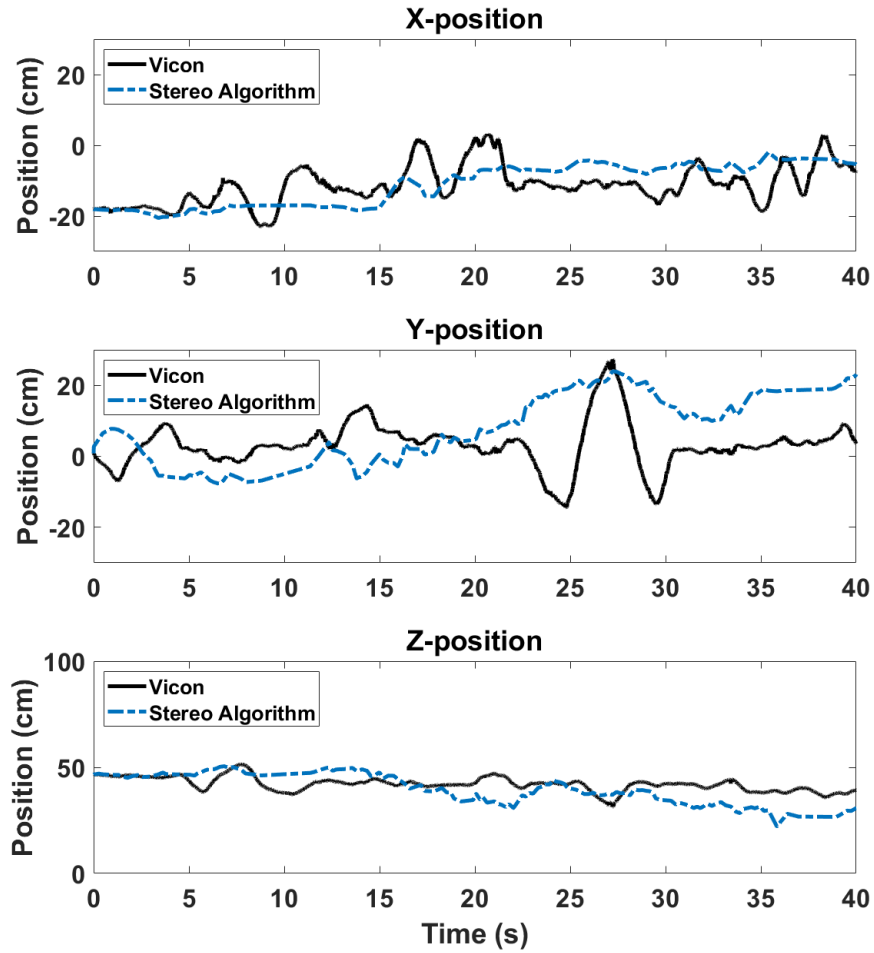


Figure 4.20: Estimation of aircraft position relative to a stationary simulated ship using only stereo vision algorithm results. The initial pose is obtained from Vicon data.

Table 4.7: RMSE of estimated rigid body motion from stereo vision algorithm.

Test case	X	Y	Z	Roll	Pitch
Quadrotor stationary	0.3 cm	0.9 cm	0.5 cm	1.1°	0.9°
Quadrotor moving	2.9 cm	2.9 cm	2.0 cm	4.4°	4.2°

estimates fail to capture many of the small, rapid motions present in ground truth. For pitch and roll, while the estimated result does capture these motions, the estimate drifts significantly over time.

From all results, it is observed that the pose estimate from the stereo vision algorithm alone is satisfactory for approximately 5 seconds, using an accurate initial pose. After this, the estimate may begin to show error or drift over time, particularly with large relative motions.

The RMSE results are summarized in Table 4.7. These results provide a measure of the error in estimated motion from the stereo vision algorithm. For the quadrotor stationary case, the errors are low, with errors below 1 cm along each linear axis and 1.1° or less in rotation. The errors are much larger, however, for the large-angle quadrotor moving test.

4.9 Free-Flight Test

A simple free-flight was conducted to evaluate the performance of the algorithm in manually piloted hover. In this scenario, the vision algorithm must tolerate vibration and constant motions which are not experienced with a hand-held camera. Due to the limitations of manual piloting, the 3D ship structures and 2D image intermittently fall out of view of the stereo camera and monocular camera, respectively, stressing the vision algorithm further.

The pose estimation results for attitude and position are provided in Figures 4.21 and

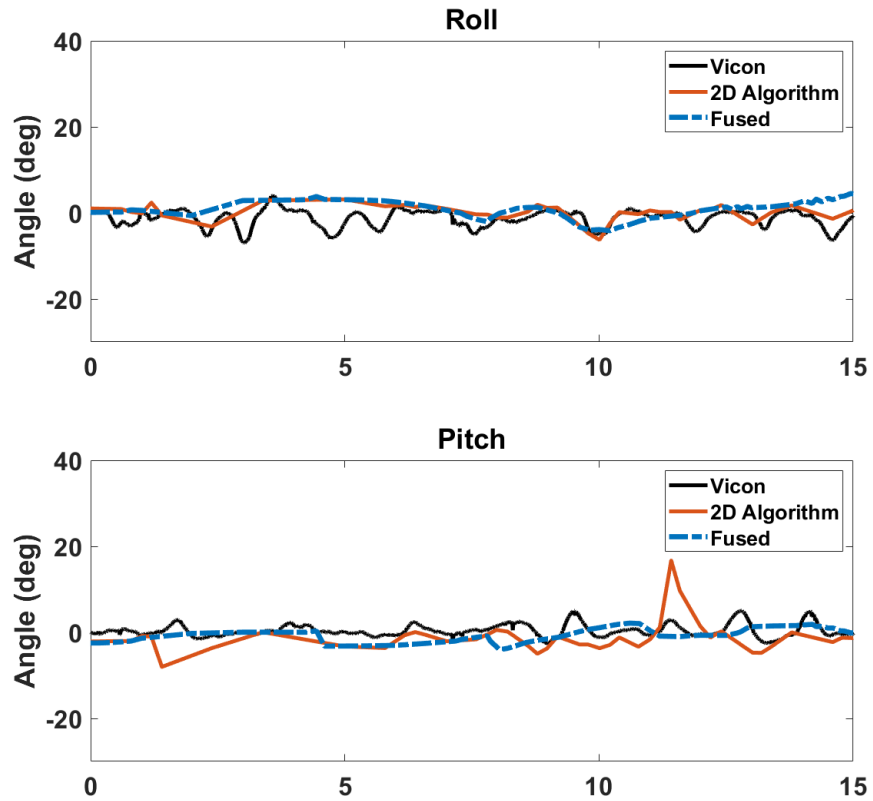


Figure 4.21: Estimation of aircraft attitude angles relative to a stationary simulated ship, with the quadrotor in manually piloted free-flight. Results are compared for the 2D feature based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.

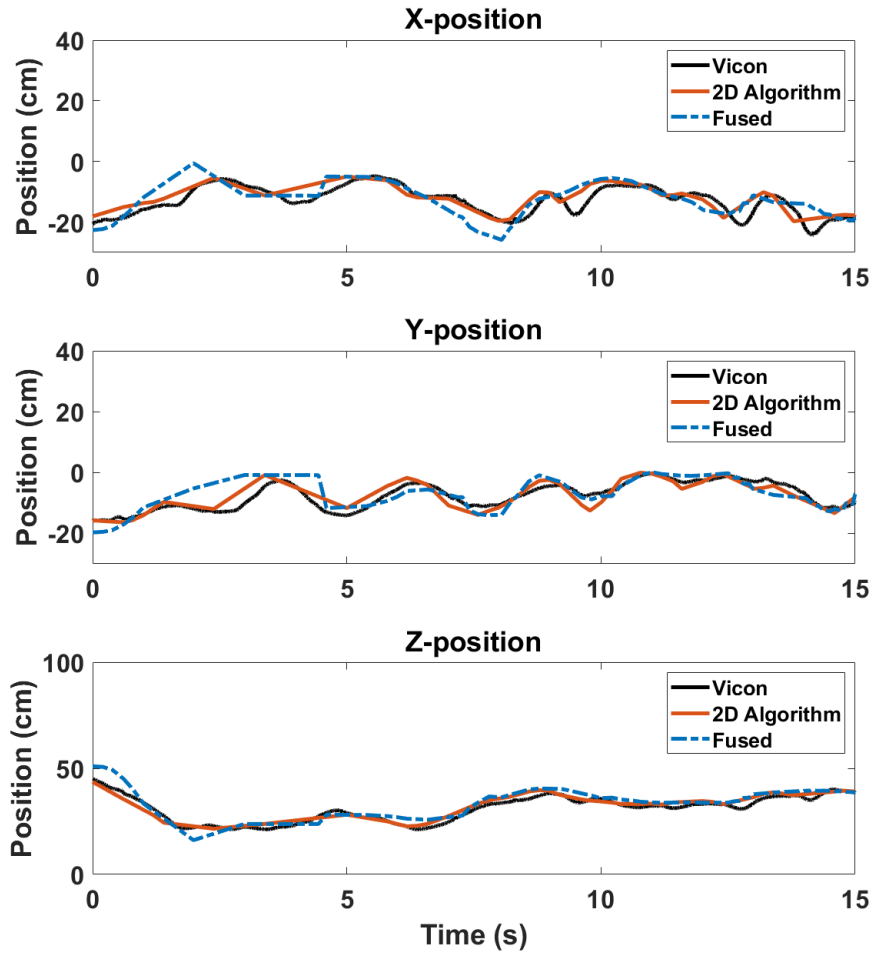


Figure 4.22: Estimation of aircraft position relative to a stationary simulated ship, with the quadrotor in manually piloted free-flight. Results are compared for the 2D feature based algorithm alone and with stereo and 2D feature-based algorithm output fused in a Kalman filter.

Table 4.8: RMSE for simple free flight test.

Test case	X	Y	Z	Roll	Pitch
2D algorithm	2.4 cm	2.0 cm	1.9 cm	2.3°	3.7°
Fused	4.2 cm	3.0 cm	3.2 cm	2.5°	3.7°

4.22, respectively. For position, both the 2D feature-based and fused algorithm results match reasonably with ground truth. For roll and pitch, both algorithms provide generally reasonable results, but fail to capture many of the small, rapid motions caused by manual pilot input. The RMSE for both algorithms is provided in Table 4.8. While the attitude estimates for the fused result show similar error to the 2D algorithm alone, the errors increase for position estimation.

Closer analysis of logged data revealed that during the test, the 2D feature-based algorithm frequently failed to detect the reference image, resulting in a low number of pose estimates provided to the Kalman filter and long delays between many of these estimates. This forced the Kalman filter to rely more heavily on stereo vision algorithm results to generate pose estimates. This reliance on the stereo vision results may explain the increase in error in the fused algorithm result. Additionally, these failures of the 2D feature-based algorithm justify the use of the stereo vision algorithm, as acceptable ship-deck pose estimates can still be continuously generated during flight.

4.10 Summary

This chapter described the development and validation of a stereo vision algorithm to estimate the motion of arbitrary ship structures. The results from this algorithm were fused with output from our previous 2D feature-based algorithm. This allows for estimating the pose

of realistic ships without the need to place custom, feature-rich landing pad images. Results were evaluated for baseline conditions, Sea-State 6 ship-deck motion, large-angle motion, and manual free-flight.

1. The stereo vision algorithm is capable of estimating the motion of arbitrary 3D ship structures relative to the camera without advance knowledge of their visual appearance.
2. By using a Kalman filter to fuse the output of the stereo vision algorithm and our previous 2D feature-based algorithm, it is possible to obtain a full estimate of ship-deck pose. This full algorithm can work with realistic 3D ship structures and 2D flight deck markings.
3. With both the stereo vision and 2D feature-based algorithm running in parallel, using a combination of CPU- and GPU-based processing, the full vision algorithm is able to maintain a sufficient update rate for real-time ship pose estimation.
4. Refinements, including implementing feature match, inlier, and error thresholds or background removal, can improve the quality of stereo vision algorithm output in challenging scenarios.
5. The full vision algorithm can accurately estimate the pose of a stationary simulated ship, with improved accuracy over the 2D feature-based algorithm alone.
6. With the aircraft stationary, the full algorithm can accurately estimate the pose of a simulated ship undergoing Sea-state 6 motion. However, performance was reduced in the large-angle and manual free flight tests.

Chapter 5: Autonomous Landing Algorithm

This chapter describes the initial development of a landing algorithm to autonomously land on a moving ship-deck. The landing algorithm uses the vision algorithms described in the previous chapters to estimate the ship-deck pose.

5.1 Autonomous Ship-Deck Landing Algorithm

After performing initial validation of the vision algorithms for ship-deck landing, the goal is to demonstrate fully autonomous landing on a moving ship-deck. To accomplish this, an autonomous landing algorithm was developed. It is implemented using the same aircraft and vision hardware described in [Chapter 4](#).

5.1.1 Implementation and architecture

The autonomous landing algorithm, in addition to the vision algorithms, runs in real-time on the onboard flight computer. It incorporates a controller to control the aircraft position relative to the ship-deck.

The controller generates desired attitude angles and vertical climb (or descent) rates during flight to control the aircraft's position, based on a desired setpoint position relative to the landing pad or simulated ship. Attitude and climb rate commands are sent to the flight controller, which

handles attitude, angular rate, and climb rate control.

The MAVLink protocol is used for communication between the flight controller and flight computer over a serial connection. MAVLink messages are used to exchange data and commands between the two devices. On the flight computer, the Pymavlink library is used to generate, process, send, and receive MAVLink messages. Each MAVLink message includes an ID number, which corresponds to a defined command or data message.

As stated in Chapter 4, the flight controller uses the ArduPilot open-source autopilot software. The software includes various flight modes and customizable parameters. These affect which MAVLink commands the controller will accept and how the controller will interpret these commands. To allow for direct attitude control from a flight computer, a Guided flight mode must be chosen. Since the aircraft is not equipped with a GPS and is flown indoors, the controller is set to the Guided_NoGPS flight mode. This mode accepts only SET_ATTITUDE_TARGET commands, which contain a desired attitude and thrust (or climb rate).

The algorithm is implemented as a ROS 2 node on the flight computer, allowing it to asynchronously receive pose estimates from the vision algorithm and to call functions within the node at regular time intervals. The full autonomy architecture is illustrated in Figure 5.1. As the autonomous landing algorithm only requires a continuous estimate of the relative ship-deck pose, it can be used with any of our vision algorithms. Variations of the ROS 2 node structure are shown for the 2D feature-based algorithm as well as the fused algorithm described in Chapter 4.

During the autonomous landing procedure, errors from the desired setpoint position relative to the ship-deck are determined using results from the vision algorithm. Individual Proportional-Integral-Derivative (PID) controllers are used to control aircraft position along each linear axis. It was assumed that the vehicle is approximately in level hover during the entire flight, allowing for

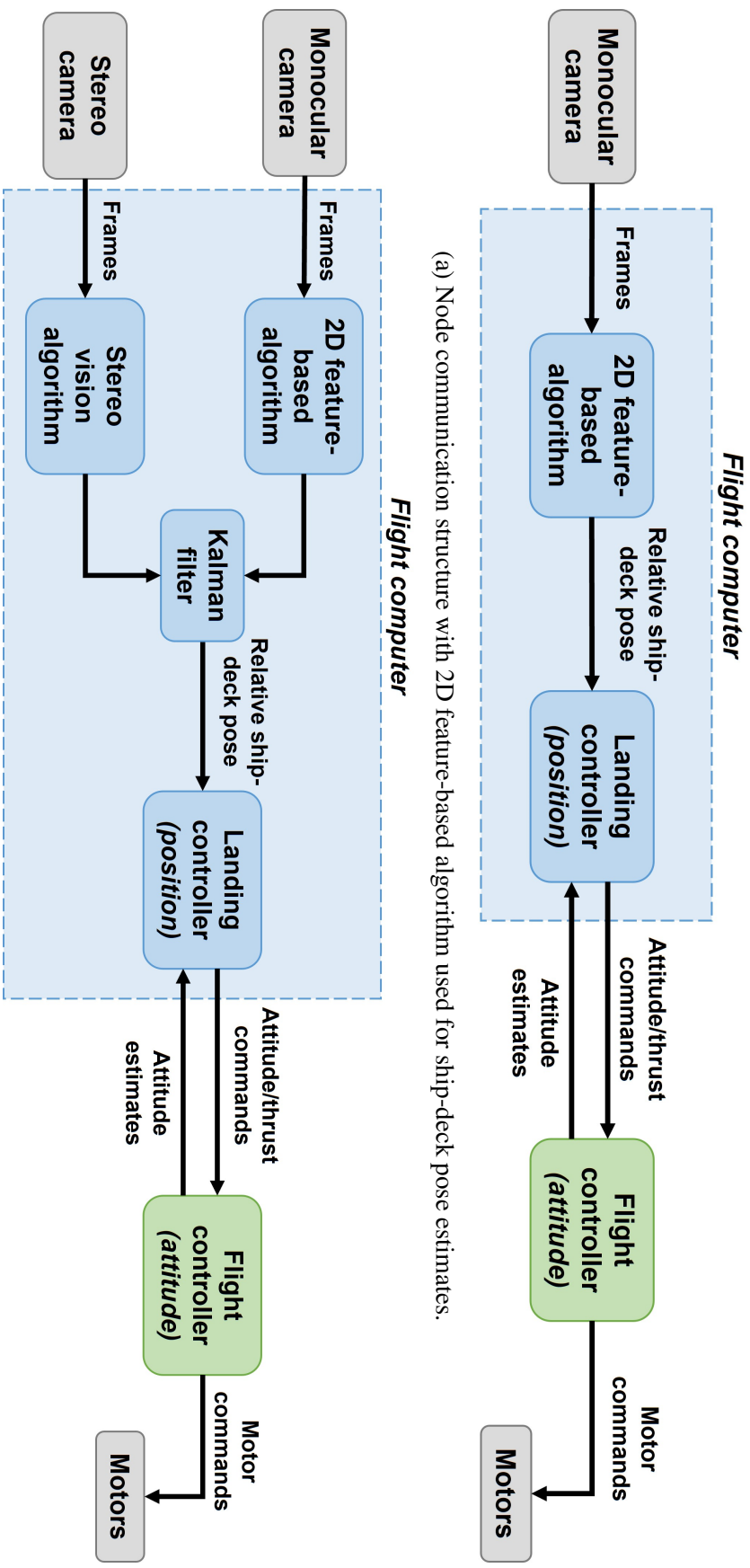


Figure 5.1: Diagram illustrating basic autonomy architecture for vision-based landing.

simplifications in the design of the position controllers. Specifically, the X-position, Y-position, and Z-position controllers output only pitch, roll, and climb rate commands, respectively. The desired yaw, based on the relative yaw of the ship-deck, is sent directly to the flight controller. To prevent integral windup, the integral terms in the controllers are disabled until the aircraft reaches a specified Z-position to ensure it has completed takeoff.

The algorithm completes takeoff, hover, and landing completely autonomously. It uses a landing procedure similar to human pilots, as discussed in Chapter 1. It first tracks the ship-deck, hovering at a set position above it and following its motion in the X- and Y-directions. The controller does not follow the roll or pitch of the deck, only its translational motion. It waits for the motion to be sufficiently "calm" to safely land, as it is dangerous to land when the ship-deck is undergoing large roll or pitch motions. Specifically, when the roll and pitch of the ship-deck are within specified limits for a sufficient length of time, it is determined to be safe to land. These limits are selected based on aircraft operational limits and ship conditions. For this algorithm, the roll and pitch limits are set to 8 and 3 degrees, respectively, based on typical limits for shipboard landings [12]. The aircraft then descends and lands on the ship. If the roll and pitch or position error exceed specified limits during descent, the landing is stopped, and the aircraft returns to hover and again waits for a safe landing opportunity.

As different tasks must be carried out during various portions of the flight, the landing algorithm is divided into a series of stages, representing different portions of the flight. This allows the algorithm to carry out the necessary tasks for each stage, and to switch to different stages as needed. The PI controllers are updated each time a new pose estimate is received, regardless of flight stage. Updated attitude and thrust commands will be sent to the flight controller at this time.

Note that this preliminary design requires that the vision algorithm must be able to detect the ship-deck during the entire flight to provide continuous position estimates. The aircraft will therefore take off from the simulated ship-deck or landing pad image.

1. Initialization: The algorithm begins and performs a sequence of initial setup steps.

- (a) The algorithm initializes the MAVLink connection with the flight controller. It waits for a heartbeat message, confirming that the flight controller is active and that communication is functioning properly.
- (b) As the aircraft is not equipped with a GPS, it is necessary to first set the EKF (Extended Kalman Filter) origin. The EKF is used internally by the flight controller to estimate its attitude. A MAVLink message is sent to the flight controller containing the current latitude and longitude. The flight controller uses this information to apply appropriate corrections to magnetometer (compass) measurements.
- (c) A MAVLink message is sent to the flight controller to request attitude estimates, which will then be sent to the flight computer at regular intervals.
- (d) The landing algorithm then waits until pose data is received from the vision algorithm. When the first pose estimate is received, the desired yaw is set using yaw data from the aircraft (relative to North) as well as the ship-deck yaw relative to the aircraft. This desired yaw will remain constant through the entire flight. The initial setpoint for the position controllers is also set to the current position.
- (e) After completing these steps, an arming message is sent to the flight controller (enabling the motors). The algorithm moves to the next stage.

2. Takeoff: Starting from the initial relative position, the landing controller gradually moves the desired setpoint position at a set rate until reaching the setpoint for hover (zero X- and Y-position relative to the ship-deck origin, and Z-position equal to the desired distance above the ship-deck). This will ensure the aircraft smoothly ascends while following the setpoint. When the final setpoint is reached, the algorithm moves into the hover stage.
3. Hover: The aircraft maintains a stable hover above the ship-deck. This ensures that the aircraft is adequately tracking the ship-deck and maintaining hover before attempting a landing. After a specified time has passed (several seconds), the algorithm begins checking for a landing opportunity.
4. Checking for landing opportunity: The aircraft continues to hover above the ship-deck. The landing algorithm checks the estimates of roll and pitch returned by the vision algorithm. If these values are within specified limits for a set length of time, the aircraft moves to the landing stage.
5. Landing: The setpoint position is moved downward at a set rate, causing the aircraft to descend safely onto the ship-deck. If the roll and pitch exceed safe limits, or if the error in the relative X- or Y-position exceeds an acceptable threshold, the algorithm returns the aircraft to hover, and goes back to the previous stage to attempt another landing. Otherwise, if the aircraft completes the descent, the algorithm moves to the final stage.
6. End of flight: The algorithm sends a disarm command to the flight controller, disabling the motors. The landing is completed and the landing controller node is exited.

For safety, if the landing controller does not receive pose data from the vision algorithm

for a set period of time (1 second in the current implementation), it will reduce the commanded thrust to quickly descend and land, then disarm the motors. In the future, this failsafe will be modified to instead return manual control to a pilot.

In benchtop testing, the landing controller was able to progress through these flight stages and successfully send the necessary MAVLink commands to the flight controller. A simple takeoff and landing was also validated using MAVLink commands only. For this initial testing, the 2D feature-based algorithm was used with a stationary landing pad image.

5.2 Summary

This section described the initial development of an autonomous landing algorithm, discussing the autonomy architecture, implementation, and landing procedure.

Chapter 6: Summary and Conclusions

This chapter provides a summary of the dissertation and key conclusions, as well as a number of recommendations for future work.

6.1 Summary

The objective of this dissertation was to investigate feature-based algorithms to estimate the pose of a moving ship-deck, with a focus on high sea-states and visually-degraded conditions.

A 2D feature-based algorithm previously developed at UMD [5] was investigated first. Refinements were made to improve algorithm performance. The algorithm was then extensively characterized in multiple scenarios, including for Sea-state 6 motion and visually-degraded conditions, in both controlled and free-flight tests.

A new 3D feature-based algorithm, based on the structure-from-motion computer vision technique, was then developed. The new algorithm can accurately estimate the pose of a known 3D object, rather than a 2D reference image. The algorithm was validated for Sea-state 6 motion, visually-degraded conditions, and free-flight.

Next, a new feature-based stereo vision algorithm was developed. The algorithm can estimate the motion of arbitrary 3D ship structures, with no prior knowledge of their appearance. A Kalman filter was used to fuse the output of this algorithm with results from the 2D feature-

based algorithm; this full algorithm is capable of estimating the pose of realistic ships with no added tags or markers and minimal stored data. Both the stereo vision and full (fused) algorithms were validated for Sea-state 6 motion and free flight.

Initial development of an autonomous landing algorithm was then carried out. The algorithm uses pose estimation data from the previously developed vision algorithms to control the aircraft relative position during landing. A landing procedure similar to human pilots is implemented, where the aircraft descends and lands during a quiescent period of ship motion.

6.2 Conclusions

From this work, the following key conclusions were found:

1. The 2D feature-based algorithm is capable of accurately estimating the pose of a stochastically-moving platform at high sea states in both controlled and free-flight conditions. In ideal visual conditions, the algorithm can successfully estimate attitude motions in excess of 20 degrees.
2. A reference image rich in features is necessary for good algorithm performance. While any image may be used as a reference, including typical ship-deck or helipad markings, performance is reduced due to the low number of features.
3. The pose estimation results can be significantly improved, particularly in visually challenging scenarios, by incorporating thresholds for feature matches and inliers. These thresholds automatically reject poor or inaccurate pose estimates.
4. The algorithm can accurately estimate the ship-deck pose under visually degraded conditions,

including low illumination, glare, and occlusion of the landing pad, as well as with increased camera distance.

5. Using only on-board computer and vision hardware, the algorithm updates at a speed of 9-11 Hz, which is adequate for estimating ship-deck motion.
6. The accuracy of the 2D feature-based algorithm is comparable to fiducial-based ship landing approaches.
7. It is possible to use 3D computer vision techniques and feature matching to estimate the pose of a 3D ship-deck.
8. In ideal visual conditions, the 3D feature-based algorithm can estimate the pose of a stationary 3D ship-deck, as well as a ship-deck undergoing Sea-state 6 motion. The pose estimation accuracy is comparable to 2D vision methods.
9. The algorithm can accurately estimate the pose of the moving-ship deck in low illumination and occlusion conditions. However, the algorithm performance degraded in the glare condition, as it failed to detect the ship-deck for several seconds during the test.
10. Different feature detectors present trade-offs for real time performance. SIFT is accurate even at longer distances from the ship-deck, but is very computationally intensive and therefore slow. ORB performs poorly at longer distances, but is much faster, making it viable for real-time pose estimation using on-board computer hardware.
11. Similarly to the 2D feature-based algorithm, filtering pose estimates based on the number of inlier points can improve the quality of algorithm output by removing poor quality results.

12. As the 3D feature extraction process recovers a large number of features, the algorithm updates at a speed of 5-6 Hz. This is still adequate to capture ship motion, but is much slower than the 2D feature-based algorithm. By determining which features are visible based on an initial orientation, and matching only to these features, it is possible to improve algorithm update speed significantly. However, this comes at the cost of increased error in pose estimation.
13. A feature-based algorithm utilizing stereo cameras is capable of estimating the motion of arbitrary 3D ship structures relative to the camera without advance knowledge of their visual appearance or geometry.
14. The stereo vision algorithm requires an initial pose to generate full estimates of ship-deck pose. As the algorithm estimates motion between frames, the pose estimates suffer from drift over time.
15. By using a Kalman filter to fuse the output of the stereo vision algorithm and our previous 2D feature-based algorithm, it is possible to obtain a full estimate of ship-deck pose. The 2D feature-based algorithm provides an initial ship-deck pose and corrects for drift in the pose estimate. This full algorithm can work with realistic 3D ship structures and 2D flight deck markings.
16. With both the stereo vision and 2D feature-based algorithm running in parallel, using a combination of CPU- and GPU-based processing, the full vision algorithm is able to maintain a sufficient update rate for real-time ship pose estimation.
17. Refinements, including implementing feature match, inlier, and error thresholds or background

removal, can improve the quality of stereo vision algorithm output in challenging scenarios.

18. The full vision algorithm can accurately estimate the pose of a stationary simulated ship, as well as a simulated ship undergoing Sea-state 6 motion with the aircraft held stationary. Accuracy was improved over the 2D feature-based algorithm alone.
19. The full algorithm performance was reduced in the large-angle and manual free-flight tests. For the free-flight test, the 2D feature-based algorithm was observed to fail at multiple instances, forcing the vision algorithm to rely heavily on the stereo vision algorithm to obtain acceptable estimates of ship pose.

6.3 Recommendations for Future Work

Several recommendations are made for future work. These are separated into further work into vision-based autonomous landing and recommendations for future development and testing of the vision algorithms.

6.3.1 Autonomous landing

1. The autonomous landing algorithm must be thoroughly tested and tuned so that it can land accurately on both stationary and moving landing pads.
2. Future work should demonstrate vision-based landing on a large Stewart platform (Motion Systems PS-6TL-1500) with a 6×6 foot attached landing surface, shown in Figure 6.1. This will validate the full vision system under realistic ship-deck conditions as well as higher sea states. Additionally, the effect of scaling the vision system to larger platforms can be

investigated. Autonomous landing should be carried out for each of the vision algorithms described in Chapters 2, 3, and 4.

3. When identifying safe periods of motion for landing, it would be useful to have a measure of uncertainty of the ship pose estimate. The covariance matrix P_k from the Kalman filter could be used as an uncertainty measure, in addition to the number of feature matches and inliers.
4. Future work can investigate more advanced control techniques for aircraft position control during landing. In real ship landing scenarios, particularly in rough sea conditions, wind and gusts are present. Controller performance and gust rejection in these conditions can be investigated. Additionally, vision-based estimates of ship position are always time-delayed due to camera latency and processing time. Techniques to compensate for this delay and reduce deck tracking lag should be explored.
5. As described in the previous chapter, the current autonomous landing algorithm requires continuous pose estimates from a vision algorithm during all stages of the flight. The algorithm can be modified to allow a pilot to takeoff and fly the aircraft manually. When the vision algorithm detects the ship-deck, the flight computer should automatically switch to autonomous control and carry out the landing procedure.
6. The vision algorithms are used during the final landing stage, where the aircraft is hovering closely above the ship-deck. A potential extension would be to incorporate detection of the ship or platform at longer ranges. This would allow the aircraft to autonomously approach the landing pad before initiating the final landing stage, removing the need for manual



Figure 6.1: Large Stewart platform with attached 6×6 foot fiberglass surface to be used for vision-based autonomous landing. The platform can recreate severe sea states up to Sea-State 7.

pilot intervention. Neural network-based object detection algorithms are suitable for this purpose. A similar technique is implemented by Lee et al. [35].

To develop this, it is necessary to create training data consisting of labeled images of the landing platform. This data can then be used to train an existing object detector, such as the such as the You Only Look Once (YOLO) algorithm [16]. In later stages of the research, training data can also be created for real ships, and can incorporate multiple classes of ship to allow for identification of ship type in addition to detection.

An important note is that the object detector does not provide scale or physical distance information on its own. While an approach can still be carried out based on the size and position of the object bounding box in the camera image [35], it is desirable to have actual position and scale data. Depth and disparity information from a stereo camera can be used to approximate the scale and distance to the platform.

6.3.2 Additional vision algorithm development

1. The stereo vision algorithm was validated with a 3D model representative of a DDG-51 type ship. This algorithm should be further validated with different ship types. Additionally, the performance of the stereo vision algorithm and full (fused) algorithm should be evaluated for visually-degraded conditions.
2. Ship-deck rotation estimates are used to determine quiescent periods of motion for safe landing. Once a quiescent period is detected, only relative position estimates are needed to land. Therefore, it can be investigated whether it is feasible to determine quiescent periods using only the estimated rotational motion from the stereo vision algorithm without an initial estimated pose. This would remove the need for accurate estimates of ship roll and pitch.
3. The full vision algorithm described in Chapter 4 fuses results from the stereo algorithm and the 2D feature-based algorithm. The 2D feature-based algorithm could, however, be replaced with any alternate vision algorithm that provides full estimates of ship-deck pose. A neural network could be developed for this purpose. Unlike the 2D feature-based algorithm, which requires a stored reference image, the neural network can be trained on a wide range of flight deck and landing pad markings. This would remove the requirement for a stored image of the ship's flight deck markings, allowing landing on a wide range of ships not known in advance.

The neural network will need to first detect whether a landing pad or flight deck is present in the camera image, and if so, predict the origin, rotation, and bounding box of the landing

pad. Custom datasets of flight decks and landing pads will need to be created for this purpose.

Flight decks and landing pads may vary in size and scale. Therefore, additional information is needed to recover the position of the landing pad relative to the camera in physical units. Stereo vision techniques, such as depth maps, can be used to estimate scale and position information.

4. The 3D feature-based algorithm was validated with a simplified 3D object. Future work can evaluate algorithm performance with more realistic ship models.
5. The Kalman filters described in Chapters [2](#) and [4](#) utilize only vision-based estimates. For all vision algorithms, the pose estimation performance degrades in cases with significant motion of both the aircraft itself and the ship deck, particularly for the large-angle test cases. To improve estimation performance, Kalman filters incorporating both vision algorithm output and data from onboard IMUs can be developed. This could potentially improve estimation by accounting for aircraft linear and rotational motion.

Bibliography

- [1] Curtiss-Wright, “Aircraft Ship Integrated Secure and Traverse System (ASIST),” <https://www.cw-ems.com/indal/products/helicopter-securing-and-traversing/asist>.
- [2] Sierra Nevada Corporation, “UAV Common Automatic Recovery System - Version 2,” https://www.sncorp.com/media/blof0nnm/ucars-v2_product-sheet_2024-10-31_web.pdf.
- [3] Airbus, “DeckFinder,” <https://www.airbus.com/en/products-services/defence/uas/deckfinder>.
- [4] Shastry, A., Datta, A., and Chopra, I., “Vision-based Autonomous UAS Landing on a Stochastically Moving Platform,” *Vertical Flight Society 77th Annual Forum and Technology Display*, Palm Beach, Florida, May 2021.
- [5] Shastry, A., Datta, A., and Chopra, I., “Feature-Based Vision For Stochastic Motion Tracking Under Partial Occlusion,” *Vertical Flight Society 78th Annual Forum and Technology Display*, Fort Worth, Texas, May 2022.
- [6] Schwartz, A., “SCONE - Standard Deck Motion Data for a Generic Surface Combatant,” Office of Naval Research, March 2017.
- [7] “Beartrap,” *The Crowsnest: The Royal Canadian Navy’s Magazine*, Vol. 17, No. 3 and 4, Mar. 1965.
- [8] de Bruin, T., “Beartrap (Helicopter Hauldown and Rapid Securing Device),” *The Canadian Encyclopedia*, Feb. 2022.
- [9] Bathurst, D. B., “The Development of Small Ship Helicopter Operations in the Royal Navy,” *SAE Transactions*, Vol. 83, No. 3, 1974, pp. 2797–2811.
- [10] Swartz, K. I., “The History of the Canadian Maritime Helicopter Project,” *Vertiflite*, Vol. 62, No. 3, May 2016.

- [11] Colwell, J., “Real Time Ship Motion Criteria for Maritime Helicopter Operations,” *23rd International Congress of Aeronautical Sciences*, Toronto, Canada, Sep. 2002, pp. 1–10.
- [12] Tadros, A., Langlois, R., and LaRosa, M., “Ship Flight Deck Motion Parameters for Ensuring Safety of Helicopter Operation,” Indal Technologies Inc., 2003.
- [13] Fukushima, K., “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, Vol. 36, No. 4, April 1980, pp. 193–202.
- [14] Lowe, D. G., “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, Vol. 60, No. 2, Nov. 2004, pp. 91–110.
- [15] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G., “ORB: An efficient alternative to SIFT or SURF,” *2011 International Conference on Computer Vision*, IEEE, Barcelona, Spain, Nov. 2011, pp. 2564–2571.
- [16] Jocher, G., Chaurasia, A., and Qiu, J., “Ultralytics YOLO,” Jan. 2023.
- [17] Soltani, A. A., Huang, H., Wu, J., Kulkarni, T. D., and Tenenbaum, J. B., “Synthesizing 3D Shapes via Modeling Multi-view Depth Maps and Silhouettes with Deep Generative Networks,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, Hawaii, Jul. 2017, pp. 2511–2519.
- [18] Araar, O., Aouf, N., and Vitanov, I., “Vision Based Autonomous Landing of Multirotor UAV on Moving Platform,” *Journal of Intelligent & Robotic Systems*, Vol. 85, No. 2, Feb. 2017, pp. 369–384.
- [19] Badakis, G., Koutsoubelias, M., and Lalis, S., “Robust Precision Landing for Autonomous Drones Combining Vision-based and Infrared Sensors,” *2021 IEEE Sensors Applications Symposium (SAS)*, IEEE, Sundsvall, Sweden, Aug. 2021, pp. 1–6.
- [20] Lin, J., Wang, Y., Miao, Z., Zhong, H., and Fierro, R., “Low-Complexity Control for Vision-Based Landing of Quadrotor UAV on Unknown Moving Platform,” *IEEE Transactions on Industrial Informatics*, Vol. 18, No. 8, Aug. 2022, pp. 5348–5358.
- [21] Lee, H., Jung, S., and Hyunchul, D., “Vision-based UAV landing on the moving vehicle,” *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, Arlington, VA, USA, Jun. 2016, pp. 1–7.
- [22] Falanga, D., Zanchettin, A., Simovic, A., Delmerico, J., and Scaramuzza, D., “Vision-based autonomous quadrotor landing on a moving platform,” *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, IEEE, Shanghai, China, Oct. 2017, pp. 200–207.
- [23] Kawamura, E., Kannan, K., Lombaerts, T., and Ippolito, C. A., “Vision-Based Precision Approach and Landing for Advanced Air Mobility,” *AIAA SCITECH 2022 Forum*, San Diego, CA & Virtual, Jan. 2022.

- [24] Cesetti, A., Frontoni, E., Mancini, A., Zingaretti, P., and Longhi, S., “A Vision-Based Guidance System for UAV Navigation and Safe Landing using Natural Landmarks,” *Journal of Intelligent and Robotic Systems*, Vol. 57, No. 1-4, Jan. 2010, pp. 233–257.
- [25] DeAngelo, M. P., *Landmark-Aided Localization for Air Vehicles Using Learned Object Detectors*, Ph.D. dissertation, Pennsylvania State University, Aug. 2016.
- [26] Truong, N. Q., Nguyen, P. H., Nam, S. H., and Park, K. R., “Deep Learning-Based Super-Resolution Reconstruction and Marker Detection for Drone Landing,” *IEEE Access*, Vol. 7, May 2019, pp. 61639–61655.
- [27] Lee, M., Shin, S. G., Jang, S., Cho, W., Kim, S., Han, S., Choi, C., Kim, J., Kim, Y., and Kim, S. H., “Visual-based Landing Guidance System of UAV with Deep Learning Technique for Environments of Visual-detection Impairment,” *International Journal of Control, Automation and Systems*, Vol. 20, No. 5, May 2022, pp. 1735–1744.
- [28] Cho, G., Choi, J., Bae, G., and Oh, H., “Autonomous ship deck landing of a quadrotor UAV using feed-forward image-based visual servoing,” *Aerospace Science and Technology*, Vol. 130, Nov. 2022, pp. 107869.
- [29] Nicholson, D., Hendrick, C. M., Jaques, E. R., Horn, J., Langelaan, J. W., and Sydney, A. J., “Scaled Experiments in Vision-Based Approach and Landing In High Sea States,” *AIAA AVIATION 2022 Forum*, Chicago, IL & Virtual, Jun. 2022.
- [30] Sanchez-Lopez, J. L., Pestana, J., Saripalli, S., and Campoy, P., “An Approach Toward Visual Autonomous Ship Board Landing of a VTOL UAV,” *Journal of Intelligent & Robotic Systems*, Vol. 74, No. 1-2, Apr. 2014, pp. 113–127.
- [31] Lin, S., Garratt, M. A., and Lambert, A. J., “Monocular vision-based real-time target recognition and tracking for autonomously landing an UAV in a cluttered shipboard environment,” *Autonomous Robots*, Vol. 41, No. 4, Apr. 2017, pp. 881–901.
- [32] Holmes, W. K. and Langelaan, J. W., “Autonomous Ship-board Landing using Monocular Vision,” *American Helicopter Society 72nd Annual Forum*, American Helicopter Society, West Palm Beach, Florida USA, May 2016, pp. 1–15.
- [33] Truong, Q. H., Rakotomamonjy, T., Taghizad, A., and Biannic, J.-M., “Vision-based control for helicopter ship landing with handling qualities constraints,” *IFAC*, Vol. 49, No. 17, Aug. 2016, pp. 118–123.
- [34] Lee, B., Saj, V., Benedict, M., and Kalathil, D., “A Deep Reinforcement Learning Control Strategy for Vision-based Ship Landing of Vertical Flight Aircraft,” *AIAA AVIATION 2021 FORUM*, VIRTUAL EVENT, Aug. 2021.
- [35] Lee, B., Saj, V., Kalathil, D., and Benedict, M., “Intelligent Vision-based Autonomous Ship Landing of VTOL UAVs,” *Journal of the American Helicopter Society*, Apr. 2023, pp. 113–126.

- [36] Wickramasuriya, M., Lee, T., and Snyder, M., “Deep Monocular Relative 6D Pose Estimation for Ship-Based Autonomous UAV,” *AIAA SCITECH 2024 Forum*, Orlando, FL, Jan. 2024.
- [37] Britcher, V., Datta, A., and Chopra, I., “Refinement and Characterization of Feature-Based Vision Algorithm for Ship-Deck Landing under Degraded Visibility,” *Journal of the American Helicopter Society*, Vol. 70, No. 1, Jan. 2025, pp. 1–17.
- [38] Harris, C. G. and Stephens, M. J., “A Combined Corner and Edge Detector,” *Proceedings of the Alvey Vision Conference*, Manchester, UK, Sep. 1988.
- [39] Fischler, M. A. and Bolles, R. C., “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, Vol. 24, No. 6, June 1981, pp. 381–395.
- [40] Collins, T. and Bartoli, A., “Infinitesimal Plane-Based Pose Estimation,” *International Journal of Computer Vision*, Vol. 109, No. 3, Sep. 2014, pp. 252–286.
- [41] Britcher, V., Shastri, A., and Chopra, I., “Exploration of Feature-Based Algorithm for Autonomous Ship-Deck Landing under Visually Degraded Conditions,” *Vertical Flight Society 79th Annual Forum and Technology Display*, West Palm Beach, Florida USA, May 2023, pp. 1–14.
- [42] Britcher, V. R., Chopra, I., and Datta, A., “Development of Three-Dimensional Feature-Based Vision Algorithm for Ship-Deck Landing,” *Journal of Aircraft*, Jun. 2025, pp. 1–13.
- [43] Tareen, S. A. K. and Saleem, Z., “A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK,” *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, IEEE, Sukkur, Pakistan, Mar. 2018, pp. 1–10.
- [44] Hartley, R. I. and Zisserman, A., *Multiple View Geometry in Computer Vision*, Cambridge University Press, ISBN: 0521540518, 2nd ed., 2004.
- [45] Ma, Y., Soatto, S., Košecká, J., and Sastry, S. S., *An Invitation to 3-D Vision: From Images to Geometric Models*, Springer New York, New York City, NY, 1st ed., 2004.
- [46] Seitz, S., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R., “Dino data set,” *Multiview Stereo Evaluation*, University of Washington, Oct. 2005.
- [47] Markley, F. L., Cheng, Y., Crassidis, J. L., and Oshman, Y., “Averaging Quaternions,” *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 4, Jul. 2007, pp. 1193–1197.
- [48] Eggert, D., Lorusso, A., and Fisher, R., “Estimating 3-D rigid body transformations: a comparison of four major algorithms,” *Machine Vision and Applications*, Vol. 9, No. 5, Mar. 1997, pp. 272–290.
- [49] Arun, K. S., Huang, T. S., and Blostein, S. D., “Least-Squares Fitting of Two 3-D Point Sets,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 9, No. 5, 1987, pp. 698–700.

- [50] Larsen, T., Andersen, N., Ravn, O., and Poulsen, N., “Incorporation of time delayed measurements in a discrete-time Kalman filter,” *Proceedings of the 37th IEEE Conference on Decision and Control*, Vol. 4, 1998, pp. 3972–3977.
- [51] Larsen, T., Andersen, N., Ravn, O., and Poulsen, N., “Incorporation of time delayed measurements in a discrete-time Kalman filter,” *Proceedings of the 37th IEEE Conference on Decision and Control*, Vol. 4, Dec. 1998, pp. 3972–3977 vol.4.