



**NAVAL
POSTGRADUATE
SCHOOL**

MONTEREY, CALIFORNIA

THESIS

**VISION-BASED 3D MOTION ESTIMATION FOR ON-
ORBIT PROXIMITY SATELLITE TRACKING AND
NAVIGATION**

by

Alessio A. Grompone

June 2015

Thesis Advisor:
Co-Advisor:

Roberto Cristi
Marcello Romano

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE June 2015	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE VISION-BASED 3D MOTION ESTIMATION FOR ON-ORBIT PROXIMITY SATELLITE TRACKING AND NAVIGATION		5. FUNDING NUMBERS	
6. AUTHOR(S) Alessio A. Grompone		8. PERFORMING ORGANIZATION REPORT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ___N/A___.	
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) The main challenge addressed in this work is to develop and validate an algorithm able to track and estimate the relative position and motion of on-orbit, un-modeled targets by using only passive vision. The algorithm developed is based on well-known image processing techniques. To achieve this goal, a number of different approaches were analyzed and compared to assess their performance for a satisfactory design. The code also has a modular general structure in order to be more flexible to changes during the implementation until best performance is reached. Artificially rendered high quality, animated videos of satellites in space and real footage provided by NASA have been used as a benchmark for the calibration and test of the main algorithm modules. The final purpose of this work is the validation of the algorithm through a hardware-in-the-loop ground experiment campaign. The development of the Floating Spacecraft Simulation Test-bed used in this work for the validation of the algorithm on real-time acquisition images was also documented in this thesis. The test-bed provides space-like illumination, stereovision and simulated weightlessness frictionless conditions. Insight on the validity of this approach, describing the performance demonstrated by the experiments, the limits of the algorithm and the main advantages and challenges related to possible future implementations in space applications, were provided by this research.			
14. SUBJECT TERMS Spacecraft detection, computer vision, autonomous navigation, image processing, vision-based algorithm, non-modeled non-cooperative target		15. NUMBER OF PAGES 185	
		16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

**VISION-BASED 3D MOTION ESTIMATION FOR ON-ORBIT PROXIMITY
SATELLITE TRACKING AND NAVIGATION**

Alessio A. Grompone
Civilian, Department of Defense
M.S., University of Rome 'La Sapienza,' 2010

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN ENGINEERING SCIENCE
(ELECTRICAL ENGINEERING)**

from the

**NAVAL POSTGRADUATE SCHOOL
June 2015**

Author: Alessio A. Grompone

Approved by: Roberto Cristi
Thesis Advisor

Marcello Romano
Co-Advisor

Clark Robertson
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The main challenge addressed in this work is to develop and validate an algorithm able to track and estimate the relative position and motion of on-orbit, un-modeled targets by using only passive vision. The algorithm developed is based on well-known image processing techniques. To achieve this goal, a number of different approaches were analyzed and compared to assess their performance for a satisfactory design. The code also has a modular general structure in order to be more flexible to changes during the implementation until best performance is reached.

Artificially rendered high quality, animated videos of satellites in space and real footage provided by NASA have been used as a benchmark for the calibration and test of the main algorithm modules. The final purpose of this work is the validation of the algorithm through a hardware-in-the-loop ground experiment campaign. The development of the Floating Spacecraft Simulation Test-bed used in this work for the validation of the algorithm on real-time acquisition images was also documented in this thesis. The test-bed provides space-like illumination, stereovision and simulated weightlessness frictionless conditions.

Insight on the validity of this approach, describing the performance demonstrated by the experiments, the limits of the algorithm and the main advantages and challenges related to possible future implementations in space applications, were provided by this research.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	SPACE APPLICATIONS FOR UNMANNED AUTONOMOUS SPACECRAFT.....	1
B.	FOCUS OF THIS RESEARCH.....	2
C.	VISION BASED TRACKING AND POSE ESTIMATION IN SPACE	2
D.	THESIS OUTLINE.....	4
II.	BACKGROUND	7
A.	ON-ORBIT RELATIVE NAVIGATION SYSTEMS	7
B.	LASER-BASED RADARS AND SENSORS.....	7
C.	VISION-BASED SPACE SENSORS	9
D.	ARTIFICIAL VISION DETECTION AND TRACKING METHODS ...	10
	1. Region-of-Interest Selection Methods	11
	2. Features Extraction	11
	3. Motion-Based Detection Methods	13
E.	KALMAN FILTER APPLICATIONS TO VISION ALGORITHMS	14
III.	SELECTED IMAGE PROCESSING TECHNIQUES	15
A.	REGION-OF-INTEREST DETERMINATION.....	15
	1. Background Segmentation	15
	2. Static Background Subtraction	17
	3. Optical Flow	18
B.	FEATURE DETECTION METHODS.....	19
	1. Harris Corner Detection.....	19
	2. Gaussian Blob Detection	20
	3. Adaptive Non-maximal Suppression (ANMS)	21
	4. Speeded-Up Robust Features (SURF).....	22
	5. Histogram of Oriented Gradients (HOG).....	23
C.	FEATURE TRACKING: THE KANADE LUCAS TOMASI METHOD	24
D.	BASIC POSE ESTIMATION TECHNIQUE	25
E.	STEREO AND GEOMETRY RANGE ESTIMATION	26
IV.	ARTIFICIAL VISION ALGORITHM	29
A.	ALGORITHM STRUCTURE AND LOGIC	29
	1. Main Logic.....	30
	2. Initialization.....	32
	3. Target Tracking	33
	4. Estimation	34
B.	ALGORITHM'S LIBRARIES	36
	1. Initializer.....	36
	2. MAIN_AViATOR.....	37
	3. FUN_BACKGROUNDSUB	37
	4. FUN_DETECTION	40

5.	FUN_SURF	40
6.	FUN_BLOB	41
7.	FUN_KLT	42
8.	FUN_EPIPOLAR.....	42
	<i>a. Linear Eight-Point Algorithm</i>	<i>42</i>
	<i>b. Continuous Eight-Point Algorithm</i>	<i>43</i>
	<i>c. Linear Four-Point Algorithm</i>	<i>45</i>
	<i>d. Continuous Four-Point Algorithm.....</i>	<i>47</i>
9.	FUN_STEREO_RANGE.....	49
10.	FUN_GEOMETRIC_RANGE.....	51
C.	ON-ORBIT TIMELAPSE AND COMPUTER RENDERED VIDEOS ..	52
	1. Computer-Rendered 3D Videos.....	52
	2. NASA On-orbit Videos	55
V.	HARDWARE-IN-THE-LOOP EXPERIMENTS.....	57
	A. THE FLOATING SPACECRAFT SIMULATOR TEST-BED	57
	1. High Precision Flat Floor	57
	2. UDP Network	58
	3. Telemetry Computer	60
	4. Floating Units	61
	<i>a. Propulsion System.....</i>	<i>63</i>
	<i>b. Electronics</i>	<i>64</i>
	<i>c. On-board Sensors.....</i>	<i>68</i>
	5. FSS Software	70
	<i>a. Main Model</i>	<i>71</i>
	<i>b. Sensor Package</i>	<i>73</i>
	<i>c. State Estimator</i>	<i>73</i>
	<i>d. Guidance Block.....</i>	<i>73</i>
	<i>e. Actuator Package</i>	<i>73</i>
	<i>f. Variable Collect and Send</i>	<i>74</i>
	<i>g. Target Package.....</i>	<i>74</i>
	B. EXPERIMENTS AND RESULTS	74
	1. Test Videos.....	74
	<i>a. Detection and Tracking Calibration.....</i>	<i>75</i>
	<i>b. Epipolar Transformation Test.....</i>	<i>79</i>
	<i>c. Stereovision Algorithm Test.....</i>	<i>86</i>
	2. NASA Videos	88
	3. Live Target	103
VI.	CONCLUSIONS	113
	A. FUTURE WORK	114
	APPENDIX.....	115
	A. ARTIFICIAL VISION ALGORITHM	115
	1. Initializer (initializer.m)	115
	2. Main script (MAIN_AViATOR.m)	118
	3. Background Subtraction (FUN_BACKGROUNDSSUB.m).....	124

4.	HARRIS Detection (FUN_DETECTION.m).....	126
5.	BLOB Selection (FUN_BLOB.m).....	128
6.	KLT Tracking (FUN_KLT.m).....	131
7.	Epipolar Transformation (FUN_EPIPOLAR.m)	133
8.	Continuous Eight-Points Algorithm (epipolar1.m)	135
9.	Continuous Four-Points Algorithm (epipolar3.m)	138
10.	Stereovision Range Estimation (FUN_STEREO_RANGE.m)	140
11.	Optical Flow Estimation (FUN_OPTFLOW.m)	143
12.	Computed ROI Limits Validation (FUN_ROILIMITER.m)	143
13.	Image Indexing Transformation (indextolinear.m).....	145
B.	MATLAB RIGID CLOUD.....	145
	LIST OF REFERENCES.....	151
	INITIAL DISTRIBUTION LIST	159

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1.	Original (on the left) and processed image (on the right) of Orbital Express using edge detection, after [44].	16
Figure 2.	Edge detected Orbital Express image processed using the dilation and fill hole (on the left) and erosion (on the right), after [44].	17
Figure 3.	Selection (in blue) of background regions on an ISS time-lapse frame, after [52].	18
Figure 4.	Corner feature in a window of pixels, from [53].	20
Figure 5.	Region of interest derived by a BLOB Gaussian filter on a frame of Orbital Express time-lapse, after [44].	21
Figure 6.	Examples of box filter approximation (two images on the right) on Gaussian second-order derivatives (two images on the left), from [48].	22
Figure 7.	SURF features detected using an Orbital Express image, after [44].	23
Figure 8.	Example of a histogram of gradients classified feature, after [44].	24
Figure 9.	Example of geometric estimation features for the distance tracking in closer-than-stereo-vision range, after [44].	27
Figure 10.	Representation of the four stages of the benchmark scenario from Orbital Express time-lapse data, after [44].	30
Figure 11.	Logic schematic of the vision-based algorithm.	31
Figure 12.	Example of static background subtraction, Harris detection and ROI selection on a time-inverted ISS Cygnus time-lapse, after [52].	33
Figure 13.	ISS-Cygnus tracking and update using Harris features detection and KLT, after [52].	35
Figure 14.	ISS-Cygnus tracking and update using Harris features detection and KLT at a close range, after [52].	35
Figure 15.	Logic schematic of the main script MAIN_AViATOR.m	39
Figure 16.	Example of estimation of the physical distance between two features using range, focal length and projected pixel distance.	51
Figure 17.	Computer rendered video of an on-orbit rendezvous maneuver for the debugging and first calibration of the vision algorithm.	53
Figure 18.	Frames from the two simulated cameras of the computer rendered stereovision video.	54
Figure 19.	Examples of rotating objects in computer rendered videos for the debugging and calibration of the epipolar algorithm.	55
Figure 20.	Granite table of the FSS test-bed at the Naval Postgraduate School.	58
Figure 21.	ARRI temperature lamp used in the FSS testbed to simulate changes in illumination conditions.	59
Figure 22.	Example of the space-like illumination simulated on the FSS testbed.	59
Figure 23.	FSS network communication schematic.	60
Figure 24.	Desktop screenshot of the telemetry software and the SSH terminals.	61
Figure 25.	View of the VICON cameras above the granite flat floor of the FSS.	62
Figure 26.	One of the VICON cameras connected to the ceiling of the Spacecraft Robotics Laboratory.	62

Figure 27.	Screenshot of the VICON software tracker. It is possible to recognize (as green squares) the position of the cameras installed along the walls of the laboratory.	63
Figure 28.	Picture of a fourth generation FSS floating unit.	64
Figure 29.	Main components of the FSS units on the four side views.	65
Figure 30.	Representation of the hovering and propulsion system. The air flow is represented with yellow arrows.	66
Figure 31.	Schematic of all the components of the compressed-air hovering and propulsion system of the FSS floating unit.	67
Figure 32.	Schematic of the FSS unit electronic system.	68
Figure 33.	Point Grey Bumblebee stereovision camera, from [64].	69
Figure 34.	Fiber-optic gyroscope DSP-3000 from KVH [65].	69
Figure 35.	Hokuyo laser scanner, from [66].	70
Figure 36.	Leap Motion, from [67].	70
Figure 37.	Main Simulink model before the compilation into an executable. Each Atomic Block is identified with a different color.	72
Figure 38.	Sequence of frames from the detection and tracking test on video 1. Harris corner features are represented in green, KLT tracked features in red and the ROI is the yellow square.	77
Figure 39.	Sequence of frames from the detection and tracking of video 2. Harris corner features are represented in green, KLT tracked features in red and the ROI is the yellow square.	78
Figure 40.	Sequence of frames from the detection and tracking of video 3. Harris corner features are represented in green, KLT tracked features in red and the ROI is the yellow square.	79
Figure 41.	The four test cases to verify the epipolar transformation algorithm.	80
Figure 42.	Three frames representing the rotation and translation of a group of computer rendered objects created to test the epipolar transformation reducing planarity and increasing parallax of the features.	82
Figure 43.	Time-history of linear and angular velocity for the test case 1 where the target is only translating along the y -axis.	83
Figure 44.	Time-history of linear and angular velocity for the test case 2 where the target is only translating along the x -axis.	83
Figure 45.	Time-history of linear and angular velocity for the test case 3 where the target is only translating along the z -axis.	84
Figure 46.	Time-history of linear and angular velocity for the test case 4 where the target is only rotating along the y -axis.	84
Figure 47.	Time-history of linear and angular velocity for the test case 5 where the target is only rotating along the x -axis.	85
Figure 48.	Time-history of linear and angular velocity for the test case 6 where the target is only rotating along the z -axis.	85
Figure 49.	Birds-eye view and stereovision distance estimation results from the test on the computer rendered video.	87
Figure 50.	Frame taken from video 1 while the algorithm is tracking the features marked in red.	96

Figure 51.	Another frame from video 1. Because of the change in illumination the algorithm tracks only edge features.	96
Figure 52.	Also during docking the algorithm tracks only features of the Soyuz on the last frames form video 1.	97
Figure 53.	Tracking of the features of the Shuttle with the Earth as background on a frame from video 2.....	97
Figure 54.	The algorithm tracks only a few features when the camera faces towards the thermal shields of the Space Shuttle.	98
Figure 55.	View of cluttered features on the progress and obstructions on the edges of the image from video 3.	98
Figure 56.	The algorithm is able to automatically detect the Progress also in this challenging frame where the Earth features spin almost at the same speed as the target and have the same luminosity intensity.	99
Figure 57.	Changes in illumination causes the algorithm to lose most of the features tracked, but it automatically recovers.	99
Figure 58.	The algorithm fully recovers from illumination changes and provides strong features and a correct ROI of the target.	100
Figure 59.	When two spacecraft with identical features are close, the algorithm expands the ROI to include and track both. This causes also other unwanted features to be captured.....	100
Figure 60.	Detection of a moving target over the static features of the ISS on the initial frames of video 5.	101
Figure 61.	The ROI expands over clouds with high defined edges, confused for target features and tracked.	101
Figure 62.	The static background removal method is able to mask the obstructed areas and non-target features.	102
Figure 63.	The algorithm is able to track the features of the target behind the docking interface.....	102
Figure 64.	The target docking interface of video 6 tracked by the algorithm.....	103
Figure 65.	View of the live desktop + live-target experiment setup and the main components of the test-bed at the Spacecraft Robotics Laboratory of NPS. .	104
Figure 66.	Bird's-eye view of the trajectories of the FSS unit in four experiments.	105
Figure 67.	Sequence of frames acquired during one of the experiments on the FSS test-bed. The tracked features are marked in red and the detected features in green.....	106
Figure 68.	Measured and estimated time-history comparison of the distance between camera and target in experiment 1.	107
Figure 69.	Zoomed view of the distance-error time history in the first 16 seconds of experiment 1.....	107
Figure 70.	Measured and estimated time-history comparison of the distance between camera and target in experiment 2.	108
Figure 71.	Zoomed view of the distance-error time history in the first 16 seconds of experiment 2.....	108
Figure 72.	Measured and estimated time history comparison of the distance between camera and target in experiment 3.	109

Figure 73.	Zoomed view of the distance-error time history in the first 16 seconds of experiment 3.....	109
Figure 74.	Measured and estimated time history comparison of the distance between camera and target in experiment 4.....	110
Figure 75.	Zoomed view of the distance-error time history in the first 16 seconds of experiment 4.....	110

LIST OF TABLES

Table 1	List of algorithm initialization options.	38
Table 2.	The four possible solutions for the linear four-point algorithm.....	47
Table 3.	The four possible solutions for the continuous four-point algorithm	49
Table 4.	List of the main guidance logic implemented on the FSS.	73
Table 5.	Detection and tracking calibration values.....	76
Table 6.	Description of results for the Epipolar analysis on the six cases.....	86
Table 7.	Video 1 properties and calibration values.....	89
Table 8.	Video 2 properties and calibration values.....	90
Table 9.	Video 3 properties and calibration values.....	91
Table 10.	Video 4 properties and calibration values.....	92
Table 11.	Video 5 properties and calibration values.....	93
Table 12.	Video 6 properties and calibration values.....	94
Table 13.	Description of the performances of the algorithm applied to the NASA videos	95
Table 14.	Detection and tracking calibration values.....	111

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

2D	two-dimensional
3D	three-dimensional
ANMS	adaptive non-maximal suppression
APF	artificial potential function
AR&D	automated rendezvous and docking
ATV	automated transfer vehicle
AVGS	advanced video guidance sensor
AViATOR	artificial vision algorithm for tracking on-orbit relative motion
BOW	bag of words
CAD	computer aided design
CBCS	centerline berthing camera system
DoG	difference of Gaussian
DPM	deformable parts model
EKF	extended Kalman filter
EPOS	European proximity operations simulator
ESA	European space agency
ETS-VII	engineering test satellite number seven
FSS	floating spacecraft simulator
GN&C	guidance, navigation and control
GPS	global positioning system
HOG	histogram of oriented gradients
ID	inverse dynamics
IDVD	inverse dynamics in virtual domain
IEKF	multiple-iterated Kalman filter
ISS	international space station
JAXA	Japan aerospace exploration agency
JEM	Japanese experiment module
KLT	Kanade Lucas Tomasi
LCDE	laser communication demonstration equipment
LCS	laser camera system

LED	light-emitting diode
LIDAR	light detection and ranging system
LoG	Laplacian of Gaussian
LQR	linear quadratic regulator
MAE	mechanical and aerospace engineering
MAP	maximum a posteriori
MPVC	multiple purpose crew vehicle
NASA	National Aeronautics and Space Administration
NASDA	national space development agency of Japan
NPS	Naval Postgraduate School
OPS	operations
PID	proportional integral derivative
POI	point of interest
ROI	region of interest
RPM	r-bar pitch maneuver
RSO	resident space object
RTAI	real-time application interface
RVDM	radar videometer
RVR	rendezvous laser radar
SIFT	scale-invariant feature transform
SM	service module
SRL	spacecraft robotics laboratory
SSH	secure shell
STS	space transportation system
SURF	speeded-up robust features
SVD	singular value decomposition
TriDAR	triangulation and LIDAR automated rendezvous and docking
UDP	user datagram protocol
UKF	unscented Kalman filter
VERTIGO	virtual estimation for relative tracking and inspection of generic objects
VIBANASS	vision based navigation system
XSS-11	experimental satellite system number eleven

EXECUTIVE SUMMARY

The main goal of this thesis is the development of an algorithm able to estimate relative position, attitude and motion using only monocular camera images in the far range, stereovision in the medium range and monocular images for the docking. On-orbit proximity maneuvering using autonomous spacecraft is today one of the major topics of interest within the space community. The potential capability of rescuing, repairing or recharging orbiting spacecraft, harvesting for orbiting components or removing space debris using unmanned robotic vehicles has proven commercial, military and scientific interest despite the complexity of such operations.

One of the main challenges of autonomous on-orbit proximity maneuvering is the relative navigation. A promising solution for relative navigation is the use of mono or stereovision for the detection and tracking of a target and for the estimation of relative position and attitude. Camera systems can have small form factors, are usually relatively inexpensive and do not require too much power. Another advantage of vision systems is that image processing can be used to define features without *a priori* information on the target. This characteristic extends the applicability to unknown or damaged targets whose features and shape are not *a priori* known.

The main challenges related to vision based systems are the following:

- image processing can be computationally demanding,
- vision systems are affected by changes in the illumination conditions,
- cluttered background and repeated patterns can cause false positive matching and detection,
- range information for unknown targets is available only within the stereovision interval of applicability, and
- the tracking can be affected by frame rate and resolution.

The main objective of this research was to investigate the use of vision-based systems for space applications through the development and test of an artificial vision algorithm.

The algorithm was designed by using well-known image processing and estimation techniques and implemented in a modular fashion to provide a “machine learning” like capability to adapt to the scenario.

The overall algorithm logic can be summarized by the following four main tasks:

1. Region-of-interest determination: Background subtraction processes the acquired images by masking the background and the obstructing features. The process uses several techniques, such as static background subtraction and optical flow.
2. Feature detection: The algorithm uses Harris corner detection to detect and classify the features of the target.
3. Feature tracking: The detected features are tracked by the Kanade Lucas Tomasi (KLT) algorithm. Tracking provides the information necessary for the optical flow used in the estimation phase.
4. Position/Motion Estimation: The algorithm estimates linear and angular velocities through the epipolar constraint, while range is estimated using the image offset of two cameras in stereovision. The attitude is estimated defining a reference frame fixed with the main tracked features and integrating the estimated rigid body rotations in time.

The algorithm was tested by using computer rendered animations that simulate the space environment, features and illuminations. The finalized algorithm was calibrated on real on-orbit footage provided by NASA, showing rendezvous and docking maneuvers of Soyuz, Space Shuttle and Progress missions in the proximity of the International Space Station.

A fourth generation of the floating spacecraft simulator test-bed (FSS) was also developed. The test-bed was used for the hardware-in-the-loop validation of the algorithm. The experiments were designed to verify the performance of the stereovision system with real-time acquisition, planar orbital-like dynamics and space-like illumination conditions, providing detection, tracking and relative position and attitude estimation (usually called pose estimation).

From the results of the experimental testing, it was possible to show the reliability of the algorithm in detecting and tracking the features on the hovering FSS test-bed unit and the capability to estimate the distance within the stereovision range with an average

error of 2.5 cm. The tests also proved that the image acquisition rate can be reduced to about 3.0 frames per second (fps) thanks to the typical low relative speed of on-orbit maneuvers.

The epipolar transformation algorithm did not provide the full estimation of the pose due to unsolved bugs, but some partial results (limited to linear and angular velocities along certain axes) do show that the method is promising, and correction of the algorithm may provide the capabilities wanted.

It was shown in this research that a vision-based algorithm can be used in real-time to detect and track on-orbit spacecraft for a wide range of illumination conditions and background scenarios with a low frame-rate.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Thanks to Professor Roberto Cristi, for his outstanding academic and moral support.

Thanks to Professor Romano and his SRL team for this great experience.

Special thanks to my soon-to-be wife, Suzi, for supporting my efforts in the good and in the tough times.

Thanks to all the friends who shared this Monterey experience with me.

Also special thanks to my parents on the other side of the world and those people who helped me become who I am today.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. SPACE APPLICATIONS FOR UNMANNED AUTONOMOUS SPACECRAFT

The advantages of on-orbit proximity operations have been widely discussed in several works [1–5] and have inspired a large number of studies on proximity maneuvers, space robotics, range sensors, teleoperation [6], [7] and more. Many studies highlight the commercial interests that could derive from on-orbit proximity operations extending the lifetime of satellites through refueling, upgrade, maintenance or harvesting parts from decommissioned orbiting spacecraft. Furthermore, system reconfiguration, rescue, removal of resident space objects [5] or safety inspection are other extremely important topics [1]. Despite the complexity of this type of mission, the progress in space robotics has made unmanned missions the most attractive option for on-orbit services [4], [6]. For example, the fact that on-board human operators are not involved drastically reduces costs, risks and mission complexity. In addition, robotic systems can be kept inactive in space or work with no interruption, resulting in extending a mission’s lifetime. The only unmanned orbiting proximity operations that have been performed were demonstration missions, but missions like ETS-VII, Orbital Express and XSS-11 have successfully demonstrated the level of maturity of several technologies for rendezvous, docking and proximity operations on resident space objects (RSO) [8], [9].

In the past few years an increasing number of studies [5], [10–13] have proposed autonomous unmanned spacecraft provided only with on-board monocular or stereovision cameras as a possible answer for an effective, low cost, low power and low weight solution for on-orbit proximity operations. The interest in vision-based systems is also motivated by the need to develop systems which are able to track passive, unmodeled, non-cooperative spacecraft. Indeed, targets with active sensors, beacons, markers or known features represent only a small number of the space objects that require on-orbit proximity operations technology as stated in [13], [14]. Many on-orbit proximity operations missions may require tracking and estimation of the relative attitude of decommissioned satellites with unknown dynamics, structure, shape and mass properties.

Another important example of potential application for tracking and state estimation of non-cooperative targets is the avoidance or active deorbiting of space debris such as damaged satellites, broken components, abandoned launcher stages or other potentially hazardous objects orbiting the earth. So far, avoidance of resident space objects has been performed through ground-based detection and control, but the importance of having on-board autonomous detection systems has been discussed in [15] and [16]. Effective on-board autonomous detection systems will reduce risks and costs associated with RSO detection, increase the range of the RSO sizes detectable and eventually be usable in outer space missions.

B. FOCUS OF THIS RESEARCH

As mentioned before, a number of approaches in the literature investigate the use of Vision-Based systems for on-orbit tracking and relative position and attitude estimation (also called pose-estimation). Only a few studies exist on the use of vision sensors on a completely unmodeled and non-cooperative target.

The main challenges of on-orbit camera sensing related to fundamental issues such as illumination and reflections, optical deformation, frame rate, stereovision's limited range, noise and cluttered background are investigated in this work. In particular, the challenge of detecting and tracking an unknown, non-cooperative target is focused on in this thesis. No *a priori* information is considered available other than the target being man made (with straight lines, regular patterns and evident corners).

The development of a real-time, vision-based algorithm and the new generation of the floating spacecraft simulator (FSS) test-bed installed in the Spacecraft Robotics Laboratory (SRL) at the Naval Postgraduate School (NPS) is used to provide experimental data and demonstrate the feasibility of the various approaches described in this work.

C. VISION BASED TRACKING AND POSE ESTIMATION IN SPACE

Model uncertainty on a non-cooperative target is considered in [13]. The use of multiple-iterated Kalman filters (IEKFs) combined with a Bayesian maximum *a*

posteriori (MAP) estimator that estimates the inertia tensor is proposed in this study. A numerical simulated comparison between the robust multiple-IKEF scheme and a plain IEKF (aware of the true target inertia) are provided, demonstrating significant robustness improvements using the first approach.

An IKEF is used also in [17] and [18] combined with optical-flow and disparity techniques to estimate the three-dimensional (3D) structure of the target. The attractiveness of this method is that it does not require a known model of the target since it uses point-wise kinematic models. The pose of the 3D structure is then estimated using a dual quaternion method [19]. The robustness and validity of this method have also been validated through hardware experiments on simulation mockups. The same image-processing technique was used in the closed loop vision-based control algorithm for the Vision based Navigation System (VIBANASS) experiments on the European Proximity Operations Simulator (EPOS) [20]. These experiments demonstrated the robustness of the guidance, navigation and control (GN&C) algorithm with variable illumination conditions and luminosity ranges using image processing to hold a position, to autonomously navigate the docking maneuver or to aid a delayed teleoperation. Several useful observations came out of this work: a) calibration is needed to transform camera measurements to world coordinates; b) time delay affects mostly distance measurements; c) experiments with autonomous systems equipped with vision show improved performance compared to systems with delayed teleoperation.

A vision-based control algorithm to keep the camera always pointing towards the detected unknown RSOs is introduced in [5]. The main contribution is given by the comparison between the use of monocular and stereovision in the control algorithm. According to [5], we find that stereovision improves the robustness and speed of the tracking while reducing fuel consumption.

Stereovision is also considered in [11], this time for the inspection of an unknown object. The vision-based algorithm is required to guide the spacecraft around the target at a desired distance while pointing at it during inspection. The only information available to the GN&C algorithm is given by the on-board Gyroscope and stereo-camera raw images. This works was then successfully validated through hardware simulations using

the Visual Estimation for Relative Tracking and Inspection of Generic Objects (VERTIGO) ISS-Based research test-bed, making it the first on-orbit demonstration of an autonomous, vision-based, non-cooperative inspection.

A feasibility study for autonomous rendezvous with an unknown space object using a monocular camera is presented in [12]. The method proposed implements two different extended Kalman filters (EKF) for the far-range relative orbit estimation and close-range relative position and attitude estimation. Simulation results provide a measure of the estimated errors during the maneuver, proving the convergence of the estimation of the full state with the applicability constraint of orbital maneuvers only.

The method proposed in [21] for the pose estimation of a non-cooperative Satellite defines a target body-fixed frame through the identification of features on the surface and using two of the most common attitude estimation algorithms, TRIAD and QUEST, for the relative attitude measurements. The translational parameters and the center of mass are estimated through a Kalman filter. The unscented Kalman filter (UKF) and the EKF are then compared for the estimation of the moment-of-inertia ratios. All these steps have been validated through numerical simulations proving the feasibility of the method. In particular the UKF has shown to converge faster than the EKF in the moment-of inertia estimation phase, achieving similar accuracy in the long term.

D. THESIS OUTLINE

The core of this research is the design and implementation of a vision algorithm using approaches available in the literature. Several image processing techniques are analyzed in order to provide a well-informed, efficient foundation in the development phase.

In the first part of the thesis, a short survey of the systems and methods used in the literature is provided as support for the choices adopted during the algorithm implementation. The most common image processing techniques of interest for this research and some results provided by analogous and interesting studies found in literature are briefly introduced.

The logic behind the image processing techniques used in this work, and why these methods have been chosen, is discussed in Chapter III.

In Chapter IV, the artificial vision algorithm for tracking on-orbit relative motion (AViATOR) is presented. Here the algorithm logic and modules are discussed, introducing also the preliminary results obtained through virtual image rendering and real videos provided by NASA.

In Chapter IV and V the implementation of the hardware-in-the-loop validation experiments on the floating spacecraft simulator are described and explained, and experimental data and plots are provided.

Observations derived from both the analysis of the experimental results and the experience acquired during the development process of the algorithm and the test-bed are given in the conclusions.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. ON-ORBIT RELATIVE NAVIGATION SYSTEMS

The list of applications of Artificial-Vision is practically endless. Automated computer image processing has been extensively used in many types of unmanned systems and in a variety of environments (on-water, under-water, on-ground, air and space). It is used for surveillance, safety systems, inspection, vision-aided navigation and more. In particular, recent advancements in computer technology and image processing techniques have contributed to the increased reliability of real-time, artificial-vision systems for detection and objects identification, human face and pose tracking, traffic flow analysis, environment mapping, human senses enhancement or replacement, surgical support and robotic manipulator servicing, GPS-denial localization, autonomous mobile robots chase, capture and formation control [22], [23].

While relative navigation based on GPS or radio transponders is the most widely validated approach being applied on many manned and unmanned rendezvous and docking missions between cooperative spacecraft [24], [25], most of the potential non-cooperative targets are not provided with on-board active sensors and require an accurate estimation of the state, the shape and the mass and inertia characteristics of the target.

B. LASER-BASED RADARS AND SENSORS

The most studied, developed and tested space sensors for on-orbit relative navigation and target tracking are laser based active radars and range sensors that use the collimated beam reflection to estimate the distance from the surface of the object [1], [26].

Several radar solutions, radar imaging processing techniques and constellation of scanning satellites have been proposed and tested for on-orbit RSO detection and cataloging [27], [28]. Of particular interest are the studies on laser radars and laser range sensors.

It is worth mentioning the Laser Communications Demonstration Equipment (LCDE) implemented on the Japanese Experiment Module (JEM) of the ISS [29]. This laser communication device has been used to demonstrate the ability of being converted into a range sensor for RSOs. The basic idea was to point the LCDE towards a debris surface (tracked previously using only sunlight reflection) and read the reflected return of the laser with the receiver of the LCDE.

Laser radar sensors have also been widely tested for docking and proximity maneuvers in space. An example of laser radar is the Rendezvous Laser Radar (RVR), the primary sensor of the demonstration mission ETS-VII. Based on near-infrared laser diodes, the RVR measures the distance between transmitter and reflector within 660 meters and with a line-of-sight angle of four degrees. With no moving components, the RVR was shown to be reliable, easy to test on the ground and cost effective [30].

Light Detection and Ranging systems (LIDAR) have been used for many years in space to support relative navigation during rendezvous of the Space Shuttle with ISS, MIR and HST [31]. Furthermore, many LIDAR based experiments have shown high performance and robustness in target detection, characterization, relative state estimation, rendezvous and docking. An example is the Videometer (RVDM) used on the ESA Automated Transfer Vehicle (ATV) for the last 250 meters of autonomous docking [32].

These qualities make the LIDAR technology the most appealing technology for many future large spacecraft missions like the Orion Multiple-Purpose Crew Vehicle (MPVC), which will be provided with a LIDAR sensor as primary relative navigation system [26, 33, 34].

A drawback of LIDAR systems is that their performance is affected by the reflectiveness of the target, and most of the applications require retro-reflectors or specific features placed on the surface in a specific configuration known to the navigation algorithm. It is through the tracking of these markers that most LIDAR systems derive the information relative to the target pose. A few exceptions, like the STS experiment system “TriDAR” [35], have demonstrated the use of LIDAR technology without retro-

reflective markers by using 3D models of the target shape to retrieve pose information through virtual and real images comparisons [10], [33], [36].

C. VISION-BASED SPACE SENSORS

Compared to the systems mentioned above, cameras are passive devices, requiring less power than Laser-Based radars and having a smaller form factor [12], [37]. Camera systems are usually more compact and less complex and, therefore, less expensive, mechanically simple, reliable and easier to test [5], [13]. Visual Imaging capabilities can easily be integrated with human-in-the-loop teleoperations in partially automated control systems to overcome delay and connection loss [38].

On the other hand, implementing camera systems as navigation sensors presents some challenges such as the dependency on ambient illumination and the configuration limits for range estimation through stereovision [10]. Nevertheless, camera systems have been widely used in space, mostly integrated with lasers or range sensors for spacecraft inspection, teleoperation activities and to aid navigation [38]. Vision-based is the solution adopted by the Orbital Express demonstration mission in 2007 for the Automated Rendezvous and Docking (AR&D), called the Advanced Video Guidance Sensor (AVGS). The AVGS fires two sets of laser beams onto retro-reflective markers positioned on the target and captures the images of the laser projection on visible and infrared cameras [9, 39]. Similar to LIDAR systems, retro-reflective patterns have been used by the software of these demonstration systems to reconstruct the relative pose.

An example of a vision-based system that does not require retro-reflective markers is the Canadian Laser Camera System (LCS) used in the STS programs to detect possible damages on the Space Shuttle. The LCS combines the cameras' photographic information with projected laser patterns on the surface to reconstruct the 3D image of the target [40].

A category of vision-based pose estimation methods uses models to either detect and match known two-dimensional (2D) and 3D features or to render 3D images (provided, for example, by computer aided design (CAD) data) and compare them with the real acquired views of the target. The use of 3D models or features is less affected by

change in illumination, shadows, optical deformations and view occlusion and is, therefore, more robust than 2D features [10].

A more challenging approach is assuming no *a priori* knowledge about the target mass, shape and structure and assuming that no retro-reflective markers or known features are present on the surface. Only a few studies have proposed an approach where the target is completely unknown [11], [12], [17] or has uncertain properties [13].

For the pose estimation without markers or models, the main challenge is given by the difficulty in retrieving range information. It is possible to retrieve the distance when two or more cameras are available and the target is in the stereovision range of the chaser [10] or when the images are collected from different known positions. This second option is likely to be the case when the chaser and target travel at different speeds on different orbits.

Another challenge is to reliably acquire and track enough features to be able to provide a cloud of points for the pose estimator. Expected difficulties acquiring features can be due to reflective or featureless surfaces of space vehicles, large changes in illumination and high contrasting shadows, presence of repetitive patterns on the target and rich and shifting background objects (e.g., Earth) [10].

Finally, an important limit to be considered is the computational load required by the image processing algorithms. Usually, the computational power of space systems is limited, but a real-time tracking and pose estimation is needed in order to reliably use the system as a navigation sensor.

D. ARTIFICIAL VISION DETECTION AND TRACKING METHODS

Given the wide interest and artificial-vision's many fields of application, many studies have approached image processing in different ways using different techniques. In The goals of the techniques we investigate are the following: a) select a region-of-interest within which the target is fully contained; b) detect and track a target using natural features; c) acquire information through the relative motion between camera, target and other objects/background.

The survey on “Object Detection Techniques” in [41] is an extremely useful tool for identifying a useful classification for vision-based methods; therefore, a similar classification is used here.

1. Region-of-Interest Selection Methods

The first phase of image processing usually requires the selection of an area of the image where the algorithm has a high confidence of detecting the target. A valid definition of a region-of-interest (ROI) is one where false positive detections and the computational load of the algorithm are reduced, giving a first estimate of the 2D localization of the object in the camera plane. Several methods can be used, but combined techniques often give the best results. Besides search methods that require *a priori* information, the most common techniques to define a ROI are based on static background subtraction and edge detection.

Static background subtraction is usually implemented when the background is fixed with respect to the camera and only moving objects must be detected. Edge detection can be used to estimate the distribution of features along the image and discard areas where no edges are detected.

In references [42] and [43], it is stated that methods based on basic segmentation (Bottom-Up Approaches) are known to run faster and use less computer resources as compared to methods that require known features.

Bottom up approaches can also be integrated with Gaussian distribution or Fourier transform filters (BLOB) in order to refine and improve the quality of the ROI as described in the following sections.

2. Features Extraction

One of the main constraints is usually given by the amount of available *a priori* information about the target. According to [41] vision-based models can be grouped in three major classes: “Holistic Generative Models,” “Holistic Discriminative Models,” and “Multi-part Representation.” The first class includes all those methods that need *a priori* 2D and 3D shape information or surface texture information. As discussed in the

introduction, these methods cannot be used in the hypothesis for an unknown, unmodeled target. The “Multi-part representation” uses classification processes, hybrid techniques and decomposition methods that are at the moment beyond the scope of this work but could lead to interesting future research.

The Holistic Discriminative Models use feature extraction techniques to analyze small regions and then machine-learning techniques to classify the location without the need of *a priori* models. According to [41] discriminative approaches are easier to implement and usually require less computation than other approaches. Only this last class of methods can be used in case of non-modeled targets.

Features extraction methods can vary based on how the algorithm recognizes and classifies the differences of neighbor pixels in the image. Statistical distribution methods, pattern recognition and local shape filters are the most common techniques. Most commonly used are the Haar-like features and the Histogram of oriented gradients (HOG) [44].

Some advanced feature extraction methods combine a detector algorithm to find the features that match a numerical constraint and a descriptor to classify the feature and some other useful information (orientation, intensity, etc.).

Very useful performance evaluations and comparisons between three Features extraction methods, or detectors, called “Bag of words” (BOW), HOG and “Deformable parts Model” (DPM) are given in [45]. These methods have been compared using several kinds of descriptors for ship detection. The performance of image processing techniques are usually strictly bounded to the parameters of the specific application; however, some considerations made in [45] were found useful as a starting point for the determination of the most suitable detector and descriptor for the research topic of this thesis.

In [45], the comparison of several methods lead to the conclusion that a Hybrid method has a slightly better average performance in terms of small false-positive detection and low computational speed with respect to the BOW, the DPM and the HOG method, but the HOG detector is easy to implement, the fastest computationally and provides very good results as compared to the other techniques.

Selection of the right combination of detector and descriptor is usually based on the type of features the algorithm must detect, on the kind of deformations expected or changes in size and orientation, or simply on the computational cost constraints of the system [46].

From comparisons between the large variety of keypoint detection and description algorithms, one of the best performing is called the Scale-invariant feature transform (SIFT). Like other detection algorithms, SIFT uses detection windows to estimate gradients, orientation and other local characteristics to define points-of-interest (POI). Scaling these detection windows can be done using the Laplacian of Gaussian (LoG) as a blob detector. SIFT approximates the LoG using a faster difference of Gaussians (DoG) approach. Once DoG are defined, *local extrema* are computed to find keypoints [47].

Similarly to the SIFT, the speeded-up robust feature (SURF) approximates LoG to define the scaled windows and uses a box filter, which are extremely fast to compute with integral images [48]. According to [48] SURF is much faster than the SIFT while comparable in terms of robustness and repeatability under different viewing conditions. The SURF method is discussed in detail in Section 3.

3. Motion-Based Detection Methods

Static background subtraction, already mentioned as a ROI selection technique, is fast and easy to implement but can be affected by change in background luminosity or movement of the camera. Other motion-based detection methods worthy of mention are “optical flow” and “frame differencing” [49].

With the optical flow, the speed of a feature is tracked on the 2D image plane, which corresponds to the projection of the 3D velocity vector of the target. The direction and speed information derived by this technique can be used to group or filter the features.

Frame differencing compares two or more sequential frames in order to detect the change in location of a pixel or a feature. If the object moves slowly enough with respect to the frame-rate, it is possible to assume that in two different frames the similar images

close in location belong to the same translated object. This method becomes more robust, but slower, when more frames are used.

E. KALMAN FILTER APPLICATIONS TO VISION ALGORITHMS

Most of the vision-based algorithms are implemented with a Kalman filter [50] either for correcting the tracking errors, increasing the robustness and integrate measures from different sensors, or to reconstruct the 3D information and estimate the state and inertia properties of a target. According to the application and the approach taken, Kalman filters have been implemented in several ways. In the cases when the fundamental assumptions do not hold (as in non-linear/or non-Gaussian cases), the most common approaches are the extended Kalman filter (EKF) and the unscented Kalman filter (UKF).

Nowadays the EKF is considered a standard method for the estimation of a non-linear system's parameters and state through a maximum likelihood approximation. The UKF has been designed to reduce the approximation errors of the EKF; extending the concept of unscented transformation [51] and several implementations also show better performance [47].

How to develop an algorithm that detects and tracks an object but also estimates the state of the target is investigated in this thesis. From the survey provided in [50], these two classes of applications are usually implemented with an EKF.

III. SELECTED IMAGE PROCESSING TECHNIQUES

The main goal of this thesis is the development of an algorithm to estimate relative pose, position and motion using only monocular camera images in the far range, stereovision in the medium range and then monocular for docking. Each of these phases require the use of image processing techniques to retrieve valid data and a Kalman filter to reduce the error and estimate the full state of the target.

The image processing itself can be divided into several subsets of operations, those which are necessary during the entire tracking and those required only for specific phases of the maneuver (detection, docking etc.). On-orbit time-lapse data from the Orbital Express mission [44] and taken from the International Space Station (ISS) [52] were used in this part of the work to demonstrate results of the implementation of the image processing methods described.

A. REGION-OF-INTEREST DETERMINATION

Preprocessing the image prior to target detection is extremely important. The creation of a ROI or the subtraction of the background reduces the probability of false-positive detections and reduces the computational load of the algorithm. Considering that the position of the camera on the chaser is known, it is usually possible in space applications to predict the background features. Background objects that can be recognized and filtered out of the image are, for instance, the Earth, manipulators, antennas and other objects mounted on the chaser or any other object with known state, features or relative velocity. The following methods were implemented and tested during the development phase of this thesis work.

1. Background Segmentation

Segmentation requires some knowledge of the target shape, illumination gradients or patterns in order to be able to recognize the target from the background (e.g., color, geometry, luminosity). The segmentation process tested in this work was mainly structured in three phases: edge detection, dilation, fill holes and erosion.

The edge detection recognizes the boundaries of objects in the image by detecting discontinuities in brightness [46]. The result of edge detection on an image taken from the Orbital Express mission [44] is shown in Figure 1.

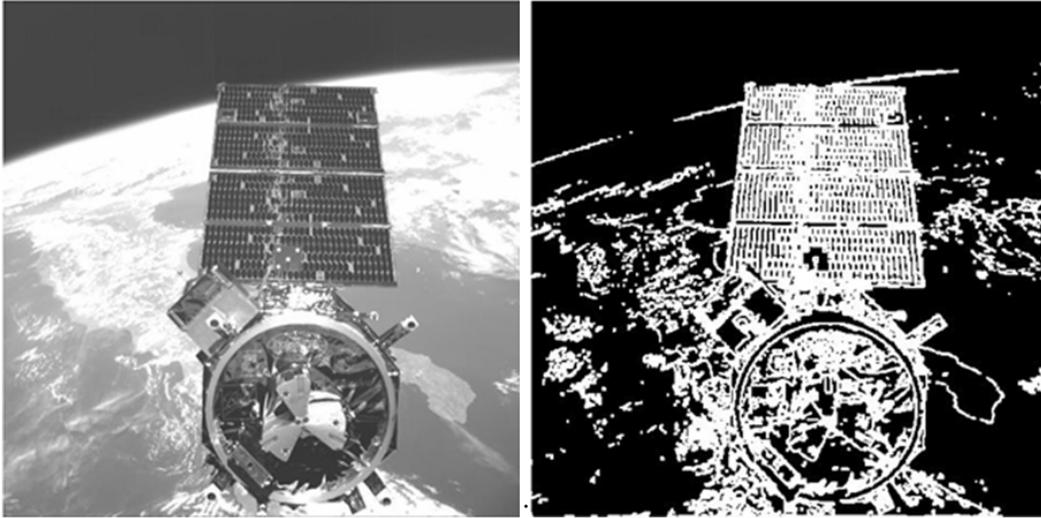


Figure 1. Original (on the left) and processed image (on the right) of Orbital Express using edge detection, after [44].

Dilation and “fill hole” processes are the equivalent of making a convolution in the binary domain. The process expands the detected edges and fills the smaller regions enclosed by the edges in order to obtain a more uniform and unified shape. Erosion, on the contrary, subtracts all the edges and small specks that are far from the main figure, providing a cleaner final result. Dilation, fill hole and erosion were applied in this order on the image previously obtained using edge detection (shown in Figure 1). Final results are shown in Figure 2 where it is possible to see that, due to errors in the segmentation of the background features, the shape of the satellite is not accurate and blends with portions of the background.

This method proved to be difficult to calibrate, inefficient and computationally intensive, therefore, it was not used in the final implementation of the algorithm.



Figure 2. Edge detected Orbital Express image processed using the dilation and fill hole (on the left) and erosion (on the right), after [44].

2. Static Background Subtraction

This method filters the background by subtracting from the image all the pixels that do not change in sequential frames. Many spacecraft cameras have a fixed angle with respect to the Earth, allowing high resolution cameras, located on artificial satellites, to recognize features on Earth regardless of the fact that the Earth cannot be considered a fixed background. For low resolution cameras, this technique can still be used to classify regions of the image in such a way that different detection techniques can be used.

For test purposes this technique was implemented to impose the condition of only detecting features approaching from the dark regions of the images, such as objects coming from outer space, on higher orbits, or appearing above the Earth horizon. Although this constraint limits the use of the algorithm, when applicable it yields a much faster and more reliable detection than other, more sophisticated techniques; therefore, this method was implemented as an initialization option for the algorithm. As an example, it was tested on a time-lapse from an ISS camera pointing constantly towards the Earth's horizon. An example of original image is on the left in Figure 3, while the processed image on the right. The processed image shows, in blue monochromatic scale, all the regions that have been discarded as background because of null or minimum relative motion. This technique was shown to be extremely useful when the camera has

an almost constant relative motion with respect to background objects, while the relative motion of the target is more relevant. In the algorithm this technique was implemented as an option that could be activated or not according to the scenario of the simulation.

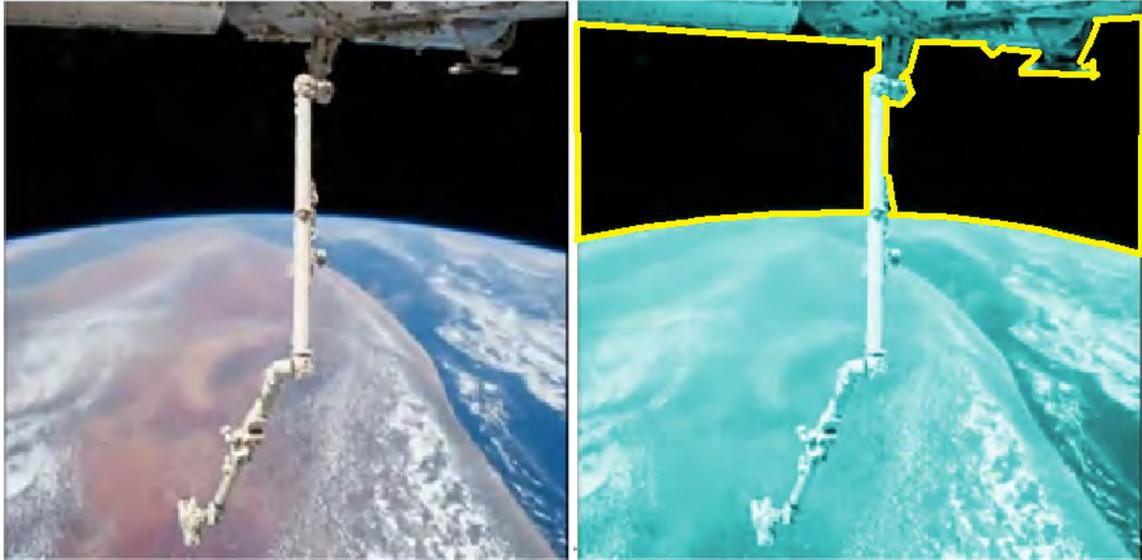


Figure 3. Selection (in blue) of background regions on an ISS time-lapse frame, after [52].

3. Optical Flow

With the method of the optical flow, the projected velocity of the pixels on the image 2D plane is calculated. Entire regions can be classified based on these measurements and removed from the image if considered belonging to known background objects. This method was shown to be computationally demanding but extremely useful for the initialization of the algorithm or after a ROI has been selected. The optical flow was used in this research mainly for the tracking phase of the algorithm, after the detection and description of the features. During tracking, the orientation of the velocity vector of the valid features provides a first estimate of the relative motion of the target. The use of the optical flow in the detection phase or in background subtraction is more challenging because it requires knowledge of the relative motion of the objects that need to be excluded from the detection analysis.

B. FEATURE DETECTION METHODS

After the determination of a ROI has been achieved, it is necessary to detect and define the points-of-interest (POIs) in order to obtain measurements regarding a target's orientation or position based on optical images. This operation can be the most challenging phase of image processing even when preprocessing techniques of background subtraction and ROI selection are implemented perfectly [41].

POIs must exhibit two important properties:

- the detection of the POIs must be robust (i.e., detectable for different illumination and resolution, from different viewpoints and with noise or deformations);
- the description has to be distinctive, which means the algorithm must be able to recognize one POI from another.

In this section, detection and description techniques used in the development of the algorithm are described and explained.

1. Harris Corner Detection

In image-processing the implementation of a reliable feature detection method is essential for a correct identification and tracking of physical points. As mentioned before the hypothesis of artificial satellites simplifies the problem of detecting the target because of the straight lines and regular patterns on the surface. One common method to select features is to identify cells (small regions of pixels) where the illumination gradient changes in two directions, as can be seen in Figure 4.

Defining I_x and I_y as the image intensity gradients along the x - and y -axes, we base a numerical algorithm to select gradient changes on the invertibility of the matrix [53]

$$G(x) = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}. \quad (1)$$

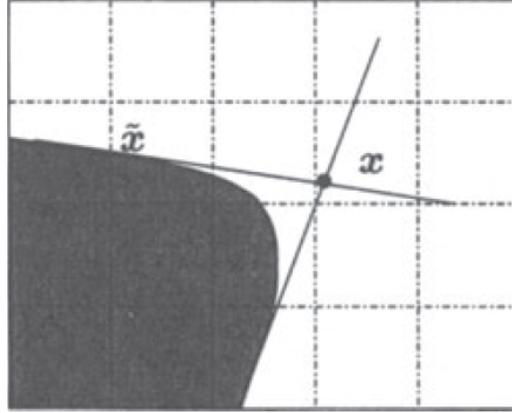


Figure 4. Corner feature in a window of pixels, from [53].

A more advanced version of this algorithm is the well-known Harris corner detection method. The matrix in Equation (1) is used to determine a threshold C that defines whether a window of pixels can be considered a corner feature using

$$C(G) = \det(G) + k \times \text{trace}^2(G), \quad (2)$$

where k is an arbitrary small scalar.

Given the hypothesis of artificially regular shapes, the Harris corner detector has been shown to perform extremely well and is, therefore, used in almost all the phases of the detection, tracking and estimation algorithm for the thesis work.

2. Gaussian Blob Detection

Detected corners can be used to estimate the position and size of the target and build an initial region-of-interest. In order to build a robust and reliable region-of-interest, a Gaussian distribution of the detected feature was implemented and filtered. This method measures the Gaussian distribution of the detected features and highlights only the regions on the images corresponding to Gaussian peaks above a certain arbitrary threshold. A well calibrated filter provides a highlighted blob-like region where a higher density of corners has been detected, giving a first rough estimate of the position and size of the target [46].

In Figure 5, it is possible to see the region-of-interest derived by the computation of a Gaussian filter for an on-orbit time-lapse of Orbital Express. The ROI is represented in blue, the features in green and the blob-like object in white.

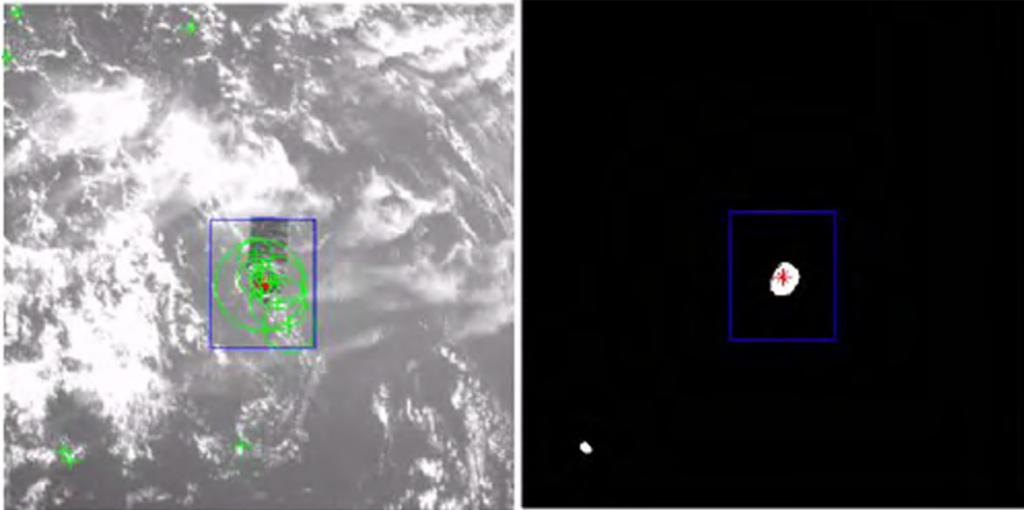


Figure 5. Region of interest derived by a BLOB Gaussian filter on a frame of Orbital Express time-lapse, after [44].

This method was shown to perform extremely well with Harris corner detection and adaptive non-maximal suppression (ANMS).

3. Adaptive Non-maximal Suppression (ANMS)

The above mentioned adaptive non-maximal suppression is a technique that measures the relative distance between detected features with the scope of discarding some POIs when they are too close together. This method reduces the density of features in certain regions, where the large number of detections can actually decrease the performance of the algorithm. Too many close features do not give much valid information compared to the computational load that they can cause. Furthermore, Gaussian distribution filters, such as the one mentioned above, were shown in this work to be negatively affected by this problem, making the ROI focus on a very complex feature instead of the entire target spacecraft [54].

4. Speeded-Up Robust Features (SURF)

Corner detection is computationally efficient but does not consider local illumination intensity normalization, scale factors and orientation. As mentioned in the previous chapter, in order to classify features a more robust method is to use a combination of detectors and descriptors.

The SURF detector is based on the fast-hessian detection method [48] which requires the computation of the determinant of the hessian matrix

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(\chi, \sigma) & L_{xy}(\chi, \sigma) \\ L_{xy}(\chi, \sigma) & L_{yy}(\chi, \sigma) \end{bmatrix} \quad (3)$$

to define location and scale, where the elements L_{xx} , L_{yy} , L_{xy} , and L_{yx} are the convolution elements of the Gaussian, respectively, along the x -axis, the y -axis and on both the x -axis and the y -axis, all functions of the point coordinates $\chi = (x, y)$ and of the scale σ .

In the SURF algorithm the second-order Gaussian derivatives are approximated with box filters in order to make the algorithm faster to compute using integral image of different sizes. The box filter approximation is shown in Figure 6.

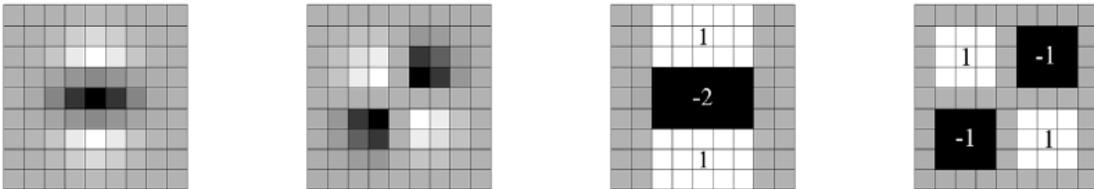


Figure 6. Examples of box filter approximation (two images on the right) on Gaussian second-order derivatives (two images on the left), from [48].

The advantage of this method is given by the fact that, in order to detect features at different scales, the Gaussian derivatives must be computed only once while the image is iteratively filtered with sequentially bigger masks.

The SURF descriptor identifies circular regions around the POI and computes Haar-wavelet responses. The responses are weighted with a Gaussian window and used to define the dominant orientation. The orientation is then used to define a square region

where Haar-wavelets are computed and weighted in a locally oriented reference frame. These oriented wavelets are used to retrieve a four-dimensional vector that describes the distribution of the intensities changes that characterize the feature. An example of SURF features detection is shown in Figure 7.

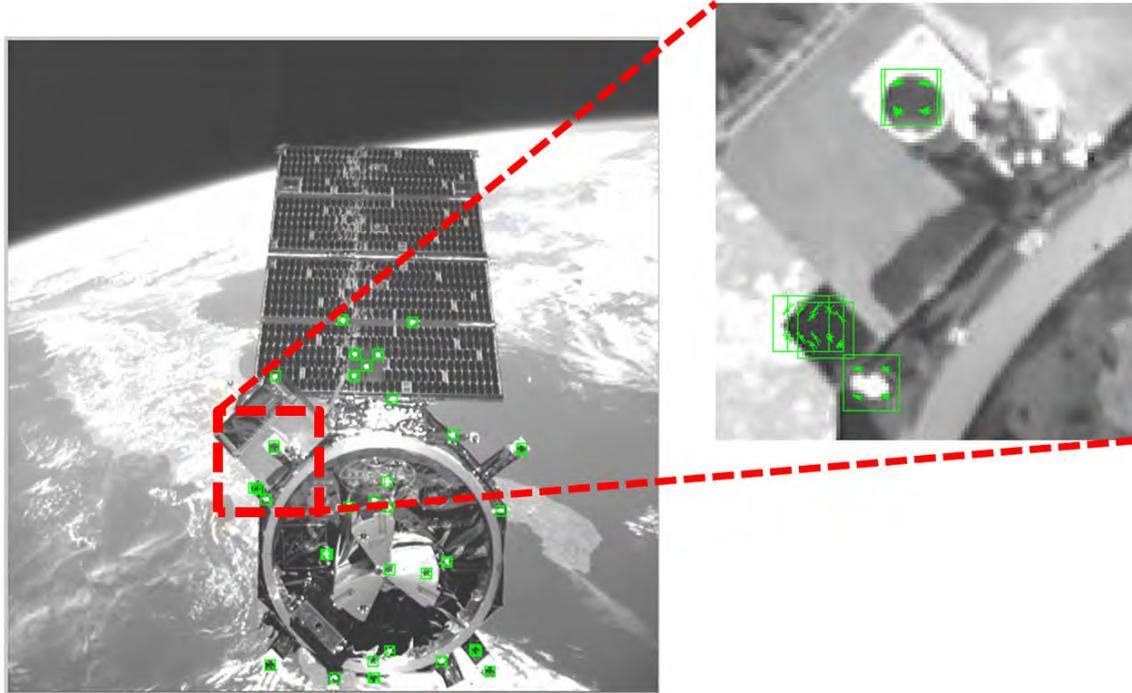


Figure 7. SURF features detected using an Orbital Express image, after [44].

5. Histogram of Oriented Gradients (HOG)

The histogram of oriented gradients (HOG) descriptor has been shown to outperform other feature detection methods in other applications [45], [55] given its simplicity and robustness. The HOG descriptor maps the image in small, equal-size-cell grids and normalizes the illumination with respect to local regions, describing the features through the distribution of local intensity gradients or edges.

This method is simple, fast and robust but performs better in combination with detectors such as SIFT or SURF [55]. An example of HOG feature and HOG classification are represented in Figure 8.

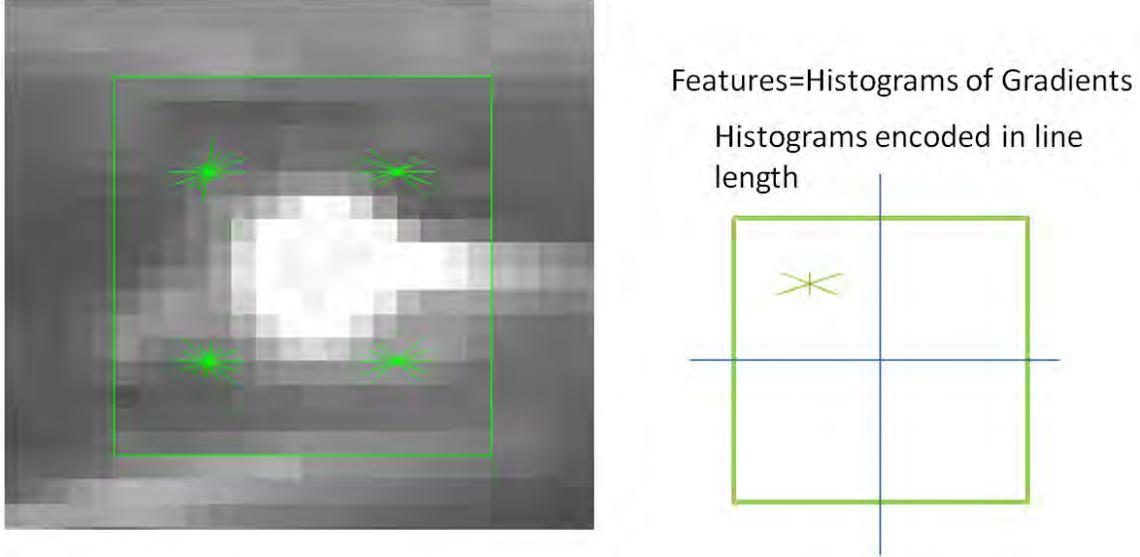


Figure 8. Example of a histogram of gradients classified feature, after [44].

C. FEATURE TRACKING: THE KANADE LUCAS TOMASI METHOD

The detection methods described above have been shown to be some of the most efficient algorithms for feature detection, being robust, repetitive and relatively fast. Nevertheless, the use of these methods on every camera frame is prohibitive when high sampling rates and low computational power are involved. A solution is to implement the KLT (Kanade, Lucas, Tomasi) tracking method [56], [57]. The KLT tracks frame-by-frame only the features that have been detected during the initialization of the algorithm. With this approach the code is not required to use detection computation on all frames but instead estimates the new location of old features by analyzing changes in windows of pixels.

The KLT method detects only planar translations of the tracked features, measured through the definition of a displacement vector d . A matching threshold is used to either discard or accept the new location to overcome small errors due to noise and changes in attitude, distance and illumination conditions. The displacement vector is the vector that minimizes

$$\varepsilon = \int [I(x-d) - J(x)]^2 w dx, \quad (4)$$

where $I(x-d)$ and $J(x)$ are the functions representative of the same feature on two sequential frames and $w(x)$ is a weighting function.

More details on how this solution is approximated are provided in [57]. The limit of this method is given by the loss of the features due to obstruction or complex and unknown change of patterns due to the motion of the relative view. To overcome this limit, a periodic detection-feature initialization might be necessary, with a period function of resolution, frame rate and relative velocity.

D. BASIC POSE ESTIMATION TECHNIQUE

Given a number of reliable features belonging to the same rigid body, the state of the detected points is constrained by the common dynamics of the entire body. A method to extract the state of the entire body through its features is described in [53] and called “the linear eight-point algorithm.” This method is based on the epipolar constraint according to which, given two different image planes, one being the reference and the second defined by a translation vector T and rotation matrix R , two projections images of the same point x_1 and x_2 are related by

$$x_2^T \overset{\square}{T} R x_1 = 0 \quad , \quad (5)$$

where $\overset{\square}{T}$ is defined as

$$\overset{\square}{T} x_2 = T \times x_2 \quad . \quad (6)$$

and $E = \overset{\square}{T} R$ is called the essential matrix.

Several epipolar-based methods to retrieve the relative position of two cameras with respect to the same target are proposed in [53]. In this work this approach is inverted without changing the main outline of the algorithm, and the epipolar measurements are used to retrieve the motion or attitude of a moving target with respect to a fixed camera.

The “linear eight point algorithm” is ill-conditioned when the rate of change between two frames is low, which is the case of high quality videos and must be properly modified. When this is the case, the tracked points provide only a small parallax displacement, and the motion can be considered almost continuous. If, in addition, the

features in the 3D space are aligned on a plane, an extra constraint must be considered in the computation. These problems are dealt with in [53] where four different algorithms are proposed, depending on the motion (discrete or continuous) and whether the features in 3D are planar or not.

Details on these algorithms and a step-by-step description of the implementation are provided in Chapter IV.

E. STEREO AND GEOMETRY RANGE ESTIMATION

The range measurements can be retrieved only after the target enters in the stereovision range, which is function of the relative position of the two cameras. Once features are detected on both images, matching algorithms are necessary to recognize the same POI in different 2D frame coordinates. A method used to retrieve range information from these stereo coordinates was proposed in [53], estimating a depth gain λ_1 from

$$\lambda_1 x_2 \times R_{12} x_1 + x_2 \times T_{12} = 0 \quad (7)$$

where R_{12} and T_{12} are, respectively, the rotation matrix and the translation between the two cameras, while x_1 and x_2 are the POI coordinates in the 2D frames.

Several matching techniques can be used. SURF integrated matching algorithms that are useful as starting points for this technique are provided in MATLAB [46]. Another method investigated is based on the condition that, knowing rotation matrix R_{12} between the two stereo-cameras, valid matching points must satisfy

$$x_2^T (R_{12} \times) x_1 = 0 \quad (8)$$

As stated before, the full 3D target location can be estimated by stereovision provided the distance between target and cameras are within certain limits. If it is too far, the algorithm is ill-conditioned and unreliable, and if it is too close, the two images do not match. While there is no other method to estimate the distance of a feature in the far range, in this work an idea to extend the range estimation in the close range is proposed.

For the monocular camera close range estimation, the proposed approach is to collect range and size information on specific geometries and features while the target is within stereovision range. This information is then used to compute the range from single projections. Examples of some possible geometries that can be tracked in close proximity range are provided in Figure 9.

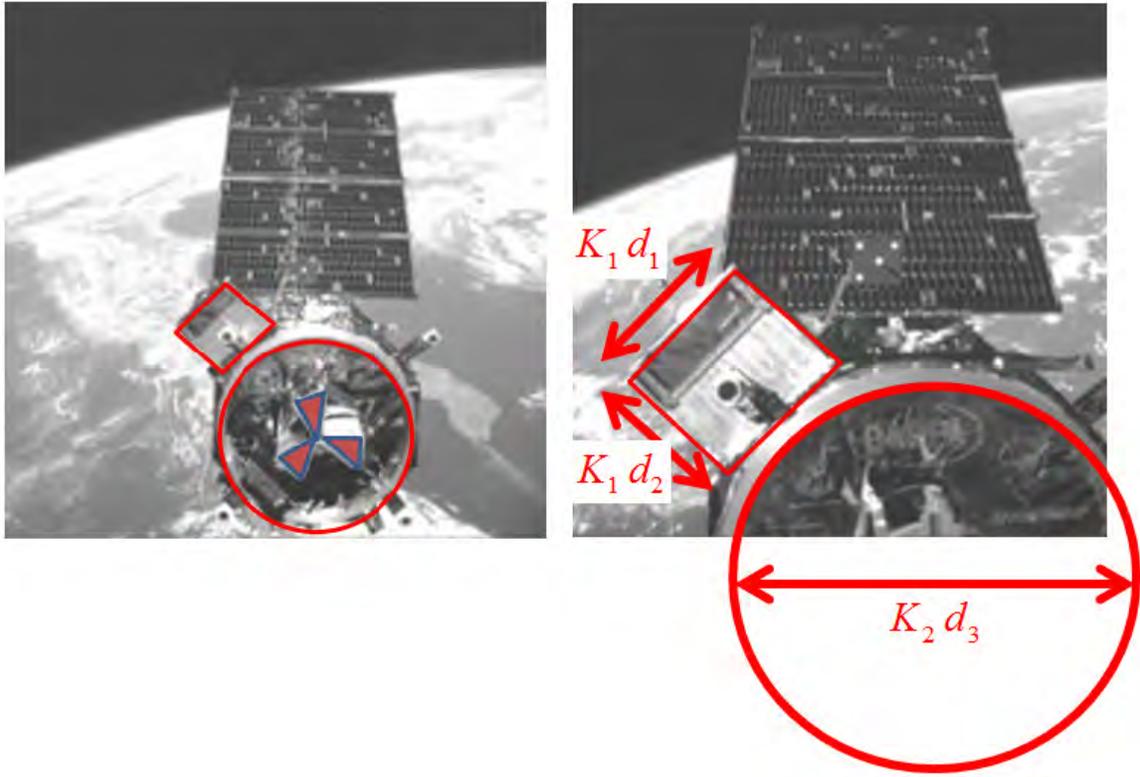


Figure 9. Example of geometric estimation features for the distance tracking in closer-than-stereo-vision range, after [44].

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ARTIFICIAL VISION ALGORITHM

The main goal of this thesis is the development of an algorithm to estimate relative position and motion of a satellite for on-orbit navigation and proximity operations through real-time image processing. In order to accomplish this task, the algorithm must be able to recognize when to activate and perform all the subsets of operations mentioned in the previous chapter. In this chapter, the implementation of these operations and the logic structure of the main algorithm are presented, followed by a description of the videos used for the first test, debugging and calibration phase.

A. ALGORITHM STRUCTURE AND LOGIC

The algorithm has been developed in a modular structure, as a collection of MATLAB scripts. This solution keeps the specific image processing tasks separate from the main logic that activates and combines them together and makes the entire algorithm adaptable to different applications and scenarios.

An on-orbit proximity operations scenario was created as a benchmark maneuver for the development of the main logic and for the test of the image processing operations. The scenario considers an operation of detection, rendezvous and docking with an on-orbit non-cooperative target. The maneuver has been divided into four main stages:

- Far-range detection and tracking is when the chaser and the target are too far apart to recognize specific features. In this case the algorithm detects the presence of the target and separates it from the background.
- Monocular middle-range motion estimation is when features of the target are recognizable and monocular estimation of the angular and linear velocities can be computed.
- Stereo-range is when the target is within the stereovision range, and the algorithm can provide range measurements.
- Monocular close-range is when the target is too close for stereovision computation, and previously detected geometries are used to estimate the range during the docking phase.

An example of the four stages is shown in Figure 10.

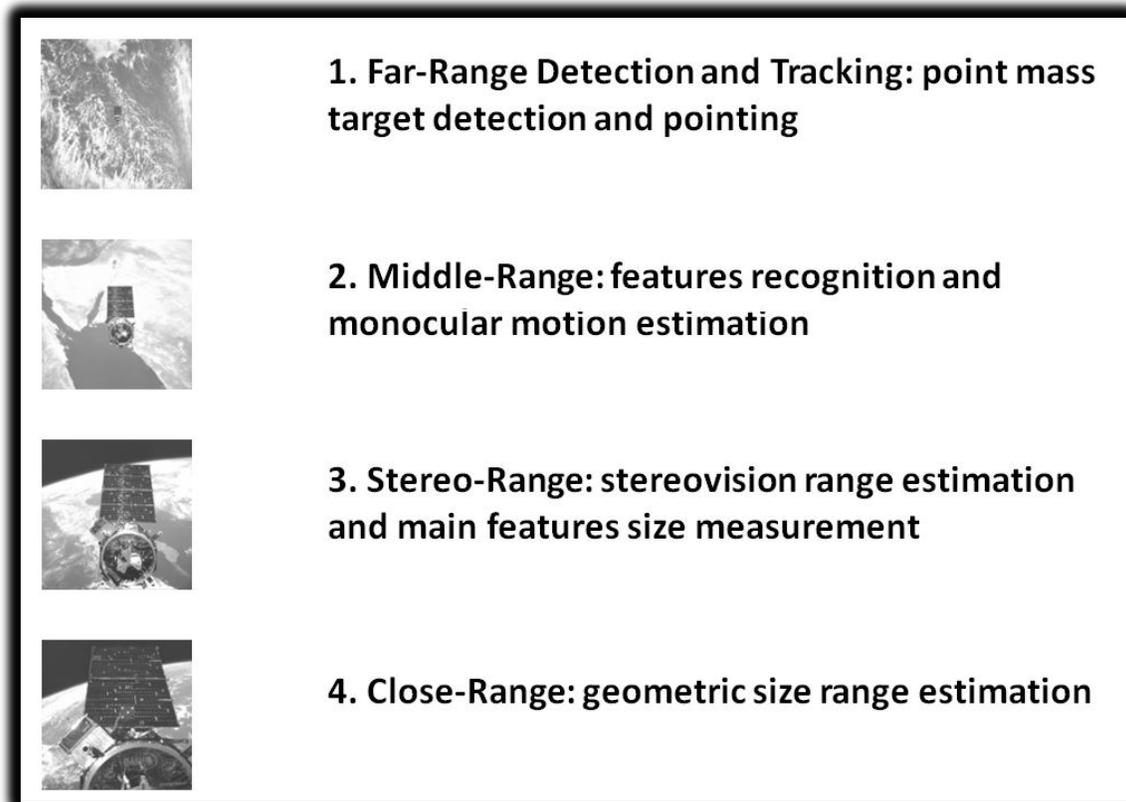


Figure 10. Representation of the four stages of the benchmark scenario from Orbital Express time-lapse data, after [44].

The algorithm must be able to recognize the stage of the maneuver and adapt accordingly, activating and modifying the appropriate sub-functions. A description of how the algorithm performs this task is provided below, followed by the description of the structure logic of the other functions.

1. Main Logic

The **Main Logic** is defined through three modes of operation: initialization, tracking and estimation. Within these modes, the main logic triggers different sub-functions according to the stage of the maneuver. A general schematic of the logic is provided in Figure 11.

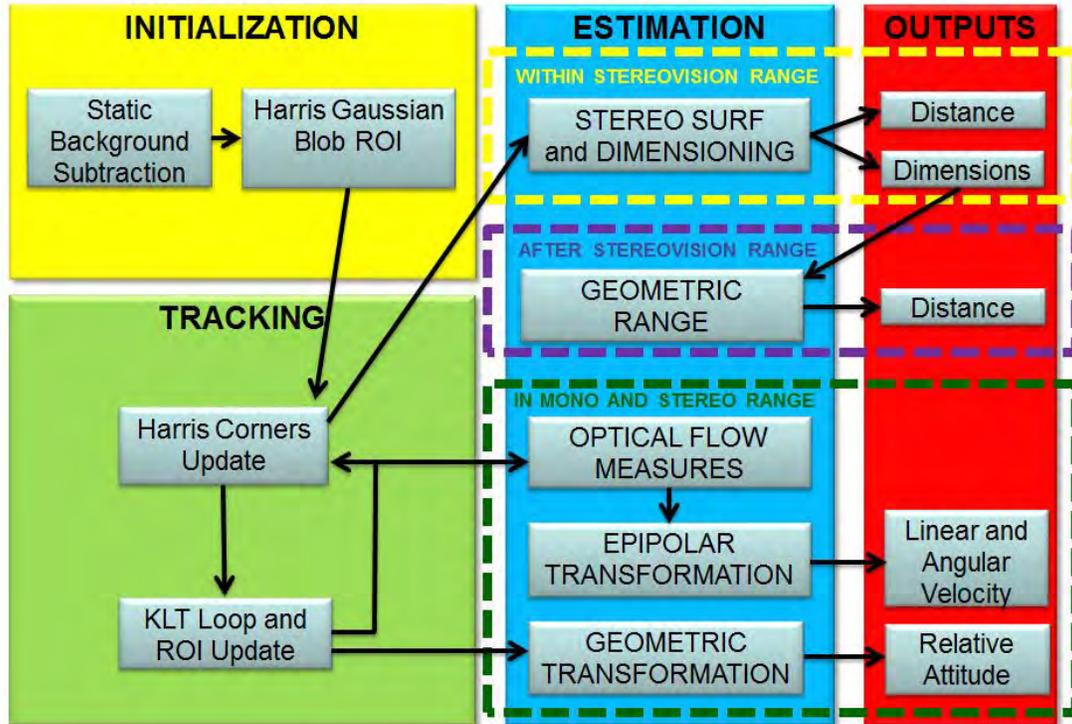


Figure 11. Logic schematic of the vision-based algorithm.

The **initialization phase** includes the preprocessing of the image and the first ROI selection. The preprocessing options must be calibrated based on known initial-view conditions (like Earth horizon position and on-board camera occlusions), while the ROI is automatically generated when an object is detected.

The **tracking phase** utilizes the first ROI generated to run the Harris corner features detection only within the defined limits. The target is tracked through the features detected as a point mass, updating the ROI until a sufficient number of high quality features is collected. When the features are recognizable, the KLT tracking activates on every frame, while the detection algorithm is disabled. The detection is re-enabled only periodically to search for new features. The ROI is a function of the KLT features and updates on every frame.

The **estimation phase** activates with the KLT output. Projected feature translations on the 2D image plane are computed with optical flow measurements. Relative pose and relative velocity of the target are estimated through epipolar and geometric transformations. The stereovision range estimation is activated either by a

threshold dimension of the ROI or a periodic stereo-frame comparison. When range measurements are available through stereovision, the algorithm computes the geometrical dimensions of the features. In the final phase, when stereovision is no longer available, this information is transferred to a geometric range estimator. The outputs change according to the stage of the maneuver:

- Pointing information is available from the first detection of the point mass object.
- Relative pose and velocity information are available only when the KLT is active and tracking of features is possible.
- Range information is collected from the beginning of the stereovision range until docking.

2. Initialization

In this work the initialization phase proved to be extremely important for the performance of the subsequent tasks, but the quality of the initialization is a function of the initial condition information provided to the camera. Several initialization options were implemented for the different videos and experiments performed.

If it is known that the background does not change in time and that the target is not visible during initialization, a first fast option is to mask all the gradients of illumination that appear in the initial images acquired. This method hides the regions where images are detected in the first frames and instructs the algorithm to search for a target only in the empty regions of the image. This method is fast and valid only if the target is expected to appear above the Earth horizon and is not visible from the beginning of the acquisition. Furthermore, the view angle with the horizon must be almost constant.

A second option is to improve the static background by removing the mask over the background features once the target is detected and/or a ROI is created. In this way, if the tracked features cross over the masked regions after the initialization phase, the algorithm is still able to follow the tracked features within the ROI.

Another possibility is to use the optical flow in order to mask all of the features that have a projected velocity not compatible with the expected velocity of the target on

the first frames acquired. This masking technique hides the features that may otherwise be confused with the target, including Earth's surface or other spacecraft. The limit of this option is that relative speed information is not always available, but it is an efficient method to discard Earth features.

An example of the implementation of the initialization is provided in Figure 12. The algorithm has been tested on a time-inverted video from an ISS time-lapse that shows the Cygnus spacecraft approaching the ISS, while the earth and the ISS robotic arm are background features. The Cygnus appears only after a few frames from the initialization. Static background subtraction was used to exclude arm and Earth features from the detection. Harris corner detection was used to detect the point mass target (the detected feature is indicated in green) and the ROI is initialized (indicated with the yellow box). The red arrow indicates magnified areas from the same frame.

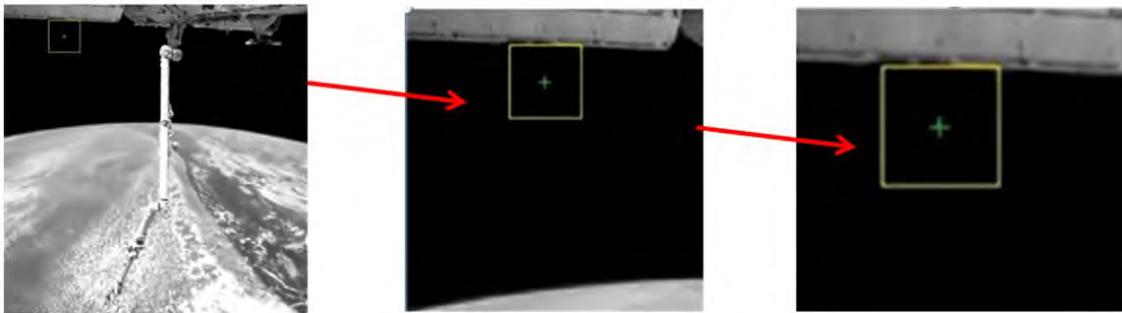


Figure 12. Example of static background subtraction, Harris detection and ROI selection on a time-inverted ISS Cygnus time-lapse, after [52].

3. Target Tracking

Target tracking can be divided in two phases. In the first phase, when the target is too far away to recognize specific features, detected points of interest are simply treated as point masses. Harris features are used to update the ROI location using the same method implemented for the first detection during initialization.

The algorithm activates the KLT tracking of distinct features when the target is closer. KLT provides the position of each valid feature in sequential frames, making it

possible to track them and use optical flow analysis to estimate the 2D image projected velocity. The points detected are also used to update position and dimensions of the ROI.

A periodic loop is implemented that activates the Harris detection in order to search for new features in the updated ROI and to reinitialize the KLT tracking. Deactivating the Harris detection within the periods reduces the computational complexity of the algorithm. The period must be calibrated based on the quality of the acquisition and was found to be mostly a function of frame rate and resolution.

Images from the ISS-Cygnus video that describe this phase are provided in Figure 13 and Figure 14. The KLT tracked points are indicated in red, while the updated Harris points are indicated in green. The ROI is indicated with the yellow box. The red arrow indicates the magnified view of the ROI.

4. Estimation

The KLT tracking provides the 2D image coordinates of a set of valid features for each frame. This information is used in the estimation phase to measure the projected velocity of each feature through optical flow analysis. This measure is then used for the Epipolar transformation algorithm to estimate the angular and linear velocity of the rigid body detected. Details on the algorithm used can be found in [53].

Relative attitude information can be defined using the geometric transformation algorithm, which defines a coordinate frame fixed to the rigid body and estimates the transformation between the body and the camera frame [46].

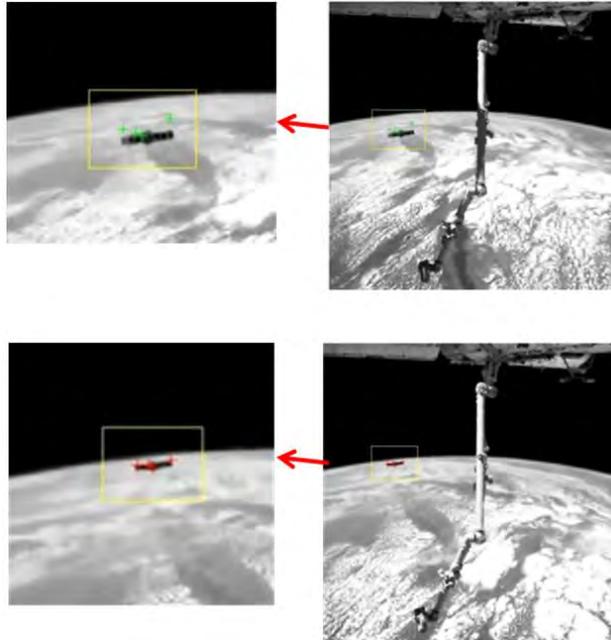


Figure 13. ISS-Cygnus tracking and update using Harris features detection and KLT, after [52].

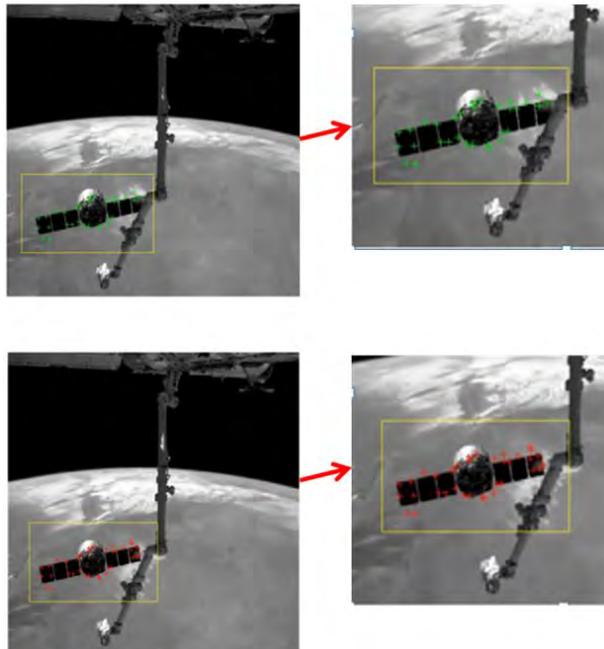


Figure 14. ISS-Cygnus tracking and update using Harris features detection and KLT at a close range, after [52].

While these tasks are performed, the stereo images are compared in order to detect when the target enters the stereovision range. The stereovision sub-function

activates when a large-enough threshold distance between the same features on two images is detected.

The stereovision uses SURF to describe and match points between the left and right images. The displacement is then used for the range estimation. The range is not only used as an output of the algorithm but is also necessary to estimate the geometrical dimensions of shapes and strong features on the target. This information is then classified and reused in reverse to estimate the range when the stereovision capabilities are no longer available. Indeed, when the target is too close to the chaser, some features might be outside the field-of-view of a camera or one camera might be occluded by the docking body.

B. ALGORITHM'S LIBRARIES

The algorithm discussed in this section is a collection of MATLAB scripts developed to implement the capabilities described above. MATLAB was chosen as the initial development and test coding language for several reasons:

- The MATLAB image processing toolbox is provided with most of the algorithms analyzed in this work;
- MATLAB code can be integrated with C code and Simulink models for hardware-in-the-loop implementations;
- MATLAB/Simulink code can be compiled as a C real-time executable with the open-source RTAI Linux OS.

Future work is required to implement this algorithm as a single Simulink block, making the algorithm easily implementable in RTAI GN&C Simulink models. The MATLAB scripts described here are all collected in Section A of the Appendix of this thesis.

1. Initializer

The “initializer.m” file is a script used only at the beginning of the algorithm to upload all the initial conditions, calibration gains and motions that define how the algorithm performs the image processing and the estimation.

This script begins with a list of options that define the performance of Harris, SURF, ROI and stereovision. The main options are provided in Table 1. All the other values defined in the initializer are only necessary to pre-allocate initial variables.

A second part of the script defines and loads the input camera or the input video. Most of the values in the initializer are defined as global variables in order to use them in all the other subscripts of the algorithm.

2. MAIN_AViATOR

The main script called “MAIN_AViATOR.m” has the function of connecting, activating and deactivating all of the functions of the algorithm. The main file keeps track of the number of frames computed and triggers the periodic functions. The schematic of the main algorithm is provided in Figure 15 where it is possible to see the periodic loops, the optional tasks and the functions. The main script also has the task of detecting when the target enters or leaves the stereovision range or when the target is no longer tracked.

3. FUN_BACKGROUNDSUB

Two methods for the subtraction of the background are implemented in the function called FUN_BACKGROUNDSUB.

The static background subtraction is used to detect the gradients of illumination due to features in the first frame and mask these features on subsequent frames until the approaching target is detected. This function is activated only when it is known that the initial frames do not contain the target and most of the background is static or slow relative to the camera. This function is extremely powerful because it drastically reduces initial detection error and computational load.

Table 1 List of algorithm initialization options.

Name	Function
CreateVideo	set to 0 or 1 to activate the creation of a video output
CreateImage	set to 0 or 1 to activate the creation of a frames output
Refreshperiod	number of frames between detection updates during the tracking
HFOV	camera horizontal field of view
fl	focal length of the camera measured in meters
Dstereo	horizontal distance between two cameras
pix	square pixel dimensions in micrometers
BackgroundSub	set to 0 or 1 to activate the static background subtraction
Detect	set to 0 or 1 to hold the detection until tracking is possible
Hstrongest	number of strongest Harris points that the algorithm will classify
Hquality	threshold quantity. Harris detector discards corners with a quality below this value
SurfSwitch	set to 0 or 1 to activate SURF as detector/descriptor
Sstrongest	threshold quantity. SURF detector discards features with a quality below this value
ANMSSwitch:	set to 0 or 1 to activate ANMS in the detection
ANMSdistance	defines the radius in pixels of the ANMS
Blength	length added to the Blob Gaussian distribution
Bsigma	standard deviation of the Blob Gaussian distribution
Bnumber	number of strongest Blobs that the algorithm will classify
Bmode	defines the method of selection of the Blob (numbered from 1 to 3)
BroiDim	pixel sides dimensions of the ROI created around the first Blob
KLTroi	set to 1 discards all the tracked points too far from the ROI
KLTvalue	maximum distance to discards KLT points too far from the center of the ROI
KLTroiDim	number of pixel to make the KLT ROI bigger than the farthest KLT point.
Distance	stereovision threshold activation distance
Stereovision	set to 0 or 1 to activate Stereovision
Stereoperiod	number of frames between activation of the detection during the stereovision

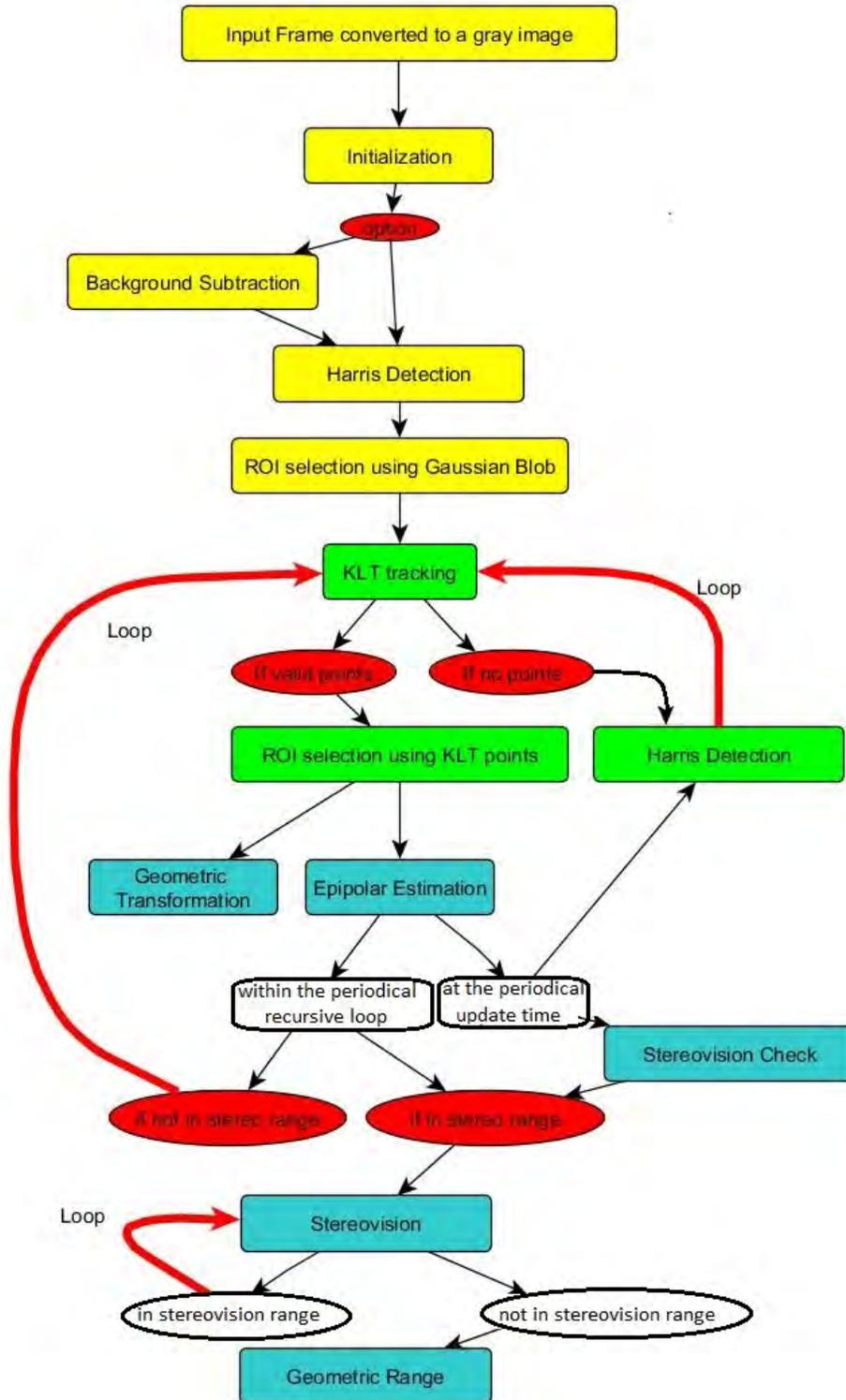


Figure 15. Logic schematic of the main script MAIN_AViATOR.m .

The background segmentation uses a combination of MATLAB built-in commands. The edges are detected with “edge” command and dilated using “imdilate.” The interior gaps are filled with “imfill,” and the final blob image is adjusted with “imclearborder.” This method is extremely sensitive to the illumination conditions and to the sensitivity parameters chosen, requires more computation than static background subtraction and is less reliable.

For some videos tested in this work, an optical flow background subtraction was needed, where the features were selected based on the speed. When the methods mentioned above could not be used or were not necessary, the detection was implemented for the entire image or only in manually selected ROIs.

4. FUN_DETECTION

Harris corner detection is activated through the function called “FUN_DETECTION.” The function uses the built-in MATLAB command “detectHarrisFeatures” for detection, selects the strongest points and activates the ANMS sub-function to reduce the number of point in overcrowded locations. The code reorders the detected points based on the quality metric computed by the detection and eliminates all points within a circle of arbitrary radius centered on the strongest points.

The coordinates are then used for the Blob ROI selection during the initialization and for the KLT periodic update.

5. FUN_SURF

The SURF code is identical to the Harris corner detection code but activates the MATLAB built-in function “detectSURFFeatures” using the HOG description through the “extractfeatures” command. The performance of this function was compared with the performance of the Harris corner detection to analyze the difference.

Tests in this work showed that SURF is more precise and provides more information about the features, making it a stronger descriptor for matching features in different frames (or cameras), but is computationally more demanding. During the detection phase the high quality information of the SURF is not necessary, but a low

computational load is essential; therefore, SURF was used only in the stereovision phase, where robust features-matching is essential, while the Harris corner detection code is preferred for the initial detection and periodic updates of the KLT tracker.

6. FUN_BLOB

The Gaussian blob filter is activated by the Harris function only during initialization in order to create blob-like figures and select the one that is the most likely target. The blobs are created through a Gaussian analysis of the distribution of features provided by the detection. The density determines the peaks and valleys of the 3D Gaussian over the 2D image plane. A threshold filters the lower regions of the Gaussian curve and forms the blob regions as white areas over a black image.

The best selection of the blobs depends on what kind of information the user has on the target (dimensions, trajectory, etc.), but several tests have shown that if good background segmentation is implemented, simply choosing the bigger blob is sufficient. In case the blob represents only a part of the target or it is bigger than the target, new detection automatically updates the region-of-interest and eventually adapts to the features tracked.

The blob function creates a zero matrix with the dimensions of the frame and updates the value within a range from the detected points according to the Gaussian distribution. If K is the metric vector of each point provided by Harris feature detection, σ the standard deviation of the Gaussian window, and n a function of the arbitrary range defined in the initializer as “Blength,” it is possible to calculate for each element of the matrix a value M_i defined as

$$M_i = K \left[\frac{1}{\sigma\sqrt{2\pi}} e^{(-n^2/2\sigma^2)} \right]^T \left[\frac{1}{\sigma\sqrt{2\pi}} e^{(-n^2/2\sigma^2)} \right]. \quad (9)$$

The blob selection instead uses the “bwconncomp” MATLAB command to detect connected regions and provide information like size and position. The classification of the blobs is then used to choose the larger one and to build a proportional ROI around it.

In this function a security “if statement” is also created to keep the ROI within the limits of the image in order to avoid errors with the detectors and the tracker.

7. FUN_KLT

The KLT tracker is based on the MATLAB built-in “step” command. The “step” command with the option “tracker” provides KLT points and a validity vector that indicates when a feature is no longer tracked. The KLT requires an initial set of features to begin tracking. The initial set of features is provided by the Harris detection during initialization and during the periodic updates. The mean value of the coordinates of the valid KLT points and the maximum distance from this value are used to build a new ROI for each frame. The ROI translates, expands or reduces its size according to the location of the features tracked. This method increases the robustness of the tracking and reduces the computational load and the error during periodic detection.

8. FUN_EPIPOLAR

The Epipolar transformation uses the tracked features of the KLT to estimate relative attitude and relative motion between the rigid body and the camera. The algorithm was developed from basic principles following the four methods provided in [53] and is reported in the following subsections. All results are demonstrated in [53].

a. *Linear Eight-Point Algorithm*

The basic estimation method is the “linear-eight point algorithm” where, given the 2D points coordinates in the image reference frame at two different times, we define the matrix X as

$$X = [a^1, a^2 \dots a^n]^T \quad (10)$$

where each column a^i is the Kronecker product $x_1^i \otimes x_2^i$ defined as

$$a = [x_1x_2, x_1y_2, x_1z_2, y_1x_2, y_1y_2, y_1z_2, z_1x_2, z_1y_2, z_1z_2]^T. \quad (11)$$

The stacked essential matrix E^S is computed as the ninth column of V_x , obtained by minimizing $\|XE^S\|$ based on the singular-value decomposition (SVD) of X :

$$X = U_X \Sigma_X V_X^T. \quad (12)$$

The solution matrix E^S must then be projected in the essential space. This is obtained by computing the SVD of the unstacked E^S and obtaining

$$E = U \text{diag}\{\sigma_1, \sigma_2, \sigma_3\} V^T. \quad (13)$$

The values of U and V are necessary for the estimation of the rotation matrix R and the translation vector T as

$$R = U R_z^T \left(\pm \frac{\pi}{2} \right) V^T \quad (14)$$

and

$$\hat{T} = U R_z \left(\pm \frac{\pi}{2} \right) \Sigma U^T, \quad (15)$$

where

$$R_z^T \left(\pm \frac{\pi}{2} \right) = \begin{bmatrix} 0 & \pm 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (16)$$

This method provides a unique solution only if the following conditions are satisfied:

- The number of points tracked is equal to or larger than eight.
- The points are not aligned or on the same plane.
- The rotation and translation provide sufficient parallax.
- The parallax values must be larger than the noise.
- A positive depth constraint is used.

This algorithm was modified in [53] to overcome some of these limitations for the continuous and planar cases.

b. Continuous Eight-Point Algorithm

If the motion is slow compared to the frame rate, the algorithm does not have sufficient parallax distance between features to estimate the essential matrix with the method described in Subsection *a* and must be modified as follows.

Indicating with the symbol $(\alpha \times)$ the skew-symmetric matrix of a generic vector α , we define the skew-symmetric matrix of the linear velocity v as

$$B = (v \times) = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \quad (17)$$

and the product of skew-symmetric matrices of the angular velocities ω and of the linear velocities v as

$$A = (\omega \times)(v \times). \quad (18)$$

We can express the “continuous Epipolar constraint” for a x_i position vector and u_i velocity vector of each i feature as

$$u_i^T B(t)x_i + x_i^T A x_i = 0. \quad (19)$$

Based on this constraint, the a^i necessary to build the X matrix in (10) is a function of x_i and u_i :

$$a = [u_3 y - u_2 z, u_1 z - u_3 x, u_2 x - u_1 y, x^2, 2xy, 2xz, y^2, 2yz, z^2]^T. \quad (20)$$

In the MATLAB function the values of the points' velocity are obtained through the optical flow, measuring the distance traveled of each feature on the 2D projection of two subsequent frames. The ratio between distance and frame rate provides the projected velocity in pixels per second. The SVD of X provides the E^S stacked vector. The vector E^S is used to form a vector v_0 with the first three elements and a matrix s with the remaining six. The SVD of s provides V_s , λ_1 , λ_2 and λ_3 for

$$s = V_s \text{diag} \{ \lambda_1, \lambda_2, \lambda_3 \} V_s^T, \quad (21)$$

and computing $\sigma_1 = (2\lambda_1 + \lambda_2 - \lambda_3)/3$, $\sigma_2 = (\lambda_1 + 2\lambda_2 + \lambda_3)/3$ and $\sigma_3 = (2\lambda_3 + \lambda_2 - \lambda_1)/3$, we define

$$\lambda = \sigma_1 - \sigma_3 \quad (22)$$

and

$$\theta = \arccos\left(\frac{-\sigma_2}{\lambda}\right). \quad (23)$$

The values λ and θ are necessary to compute $V = -V_s R_y^T(\theta/2 - \pi/2)$ and $U = -VR_y(\theta)$ with $R_y(\alpha)$ being a rotation matrix along the y -axis of angle α .

Four possible 3D velocities can be computed from

$$\omega = UR_z\left(\pm\frac{\pi}{2}\right)\Sigma_1 U^T, v = VR_z\left(\pm\frac{\pi}{2}\right)\Sigma_1 V^T \quad (24)$$

and

$$\omega = VR_z\left(\pm\frac{\pi}{2}\right)\Sigma_1 V^T, v = UR_z\left(\pm\frac{\pi}{2}\right)\Sigma_1 U^T. \quad (25)$$

The method to obtain a unique solution is to choose the pair of angular and linear velocity vectors such that the product $v_i^T v_0$ is the maximum of the i possible values as

$$v^{*T} v_0 = \max_i \{v_i^T v_0\}. \quad (26)$$

This method overcomes the problem of the small parallax displacement with the hypothesis of continuous motion but still requires at least eight non-planar features.

c. *Linear Four-Point Algorithm*

The four-point algorithm overcomes the limitation of the eight-point algorithm by introducing the planar constraint

$$\hat{x}_2 H x_1 = 0 \quad (27)$$

where H is the planar homography matrix defined as

$$H = R + \frac{1}{d} T N^T, \quad (28)$$

with the variable N being the unit normal vector of the target plane with respect to the camera frame, d the distance from the optical center of the camera, and R and T rotation and translation, respectively, as defined in the previous subsections.

The homography matrix can be approximated by building the matrix $X = [a^1, a^2 \dots a^n]^T$ with a^i defined as the Kronecker product $x_1^i \otimes x_2^i$ and computing the SVD of X . The nine output elements are used to form a 3×3 matrix H_L . The SVD of H_L provides the σ_2 that is used to normalize H_L and obtain the homography matrix H as

$$H = \frac{H_L}{\sigma_2}. \quad (29)$$

The homography matrix is used to define several vectors and matrices necessary for the computation of the solution equations. The vectors v_1, v_2 and v_3 are the column vectors of the matrix V computed through the SVD of $H^T H$ as

$$H^T H = V \Sigma V^T. \quad (30)$$

The vectors u_1 and u_2 are defined as

$$u_1 = \frac{\sqrt{1 - \sigma_3^2} v_1 + \sqrt{\sigma_1^2 - 1} v_3}{\sqrt{\sigma_1^2 - \sigma_3^2}} \quad (31)$$

and

$$u_2 = \frac{\sqrt{1 - \sigma_3^2} v_1 - \sqrt{\sigma_1^2 - 1} v_3}{\sqrt{\sigma_1^2 - \sigma_3^2}}. \quad (32)$$

Based on these vectors, we define four matrices as

$$U_1 = [v_2, u_1, \sqrt{v_2^T u_1}], \quad (33)$$

$$U_2 = [v_2, u_2, \sqrt{v_2^T u_2}], \quad (34)$$

$$W_1 = [Hv_2, Hu_1, \sqrt{Hv_2^T Hu_1}], \quad (35)$$

and

$$W_2 = [Hv_2, Hu_2, \sqrt{Hv_2^T Hu_2}]. \quad (36)$$

The equations listed in Table 2 can be computed to retrieve the four solutions for R , T and N . The solutions can be reduced to two identifying the one which is consistent with the positive depth constraint

$$N^T e_3 > 0 \quad (37)$$

where $e_3 = [0, 0, 1]^T$.

The implementation of planar methods was found extremely useful due to the quasi-planarity of the features detected in most of the scenarios analyzed in the experiments of this work.

Table 2. The four possible solutions for the linear four-point algorithm

	R	N	T
Solution 1	$R_1 = W_1 U_1^T$	$N_1 = \sqrt{d} u_1$	$\frac{T_1}{d} = (H - R_1) N_1$
Solution 2	$R_2 = W_2 U_2^T$	$N_2 = \sqrt{d} u_2$	$\frac{T_2}{d} = (H - R_2) N_2$
Solution 3	$R_3 = R_1$	$N_3 = -N_1$	$\frac{T_3}{d} = -\frac{T_1}{d}$
Solution 4	$R_4 = R_2$	$N_4 = -N_2$	$\frac{T_4}{d} = -\frac{T_2}{d}$

d. Continuous Four-Point Algorithm

For small parallax distances and high frame rate, the hypothesis of continuous motion was also applied to the planar algorithm. The matrix X is computed in the same way as for the linear case, while another matrix B is defined as

$$B = [b^{1T}, b^{2T}, \dots, b^{nT}]^T \quad (38)$$

where $b = \hat{x}u$, with \hat{x} being skew-symmetric of the coordinate vector and u the optical flow velocity vector. The stacked not-normalized homography matrix H_{L_s} is computed using

$$H_{L_s} = X^p B \quad (39)$$

where X^p is the pseudo-inverse of the matrix X . The homography matrix is computed using

$$H = H_L - \frac{1}{2}\gamma_2 I \quad (40)$$

where $\{\gamma_1, \gamma_2, \gamma_3\}$ are the eigenvalues of the matrix $H_L^T + H_L$. The matrix $H + H^T$ has eigenvalues λ_i and eigenvectors u_i . If all eigenvalues are zero, the linear velocity v is a zero vector and the skew-symmetric matrix of the angular velocity $\hat{\omega} = H$.

The solution is computed by first defining α , $\tilde{v}_1, \tilde{v}_2, \bar{N}_1$ and \bar{N}_2 as

$$\alpha = \frac{1}{2}(\lambda_1 - \lambda_3), \quad (41)$$

$$\tilde{v}_1 = \frac{1}{2}(\sqrt{2\lambda_1}u_1 + \sqrt{-2\lambda_3}u_3), \quad (42)$$

$$\tilde{v}_2 = \frac{1}{2}(\sqrt{2\lambda_1}u_1 - \sqrt{-2\lambda_3}u_3), \quad (43)$$

$$\bar{N}_1 = \frac{1}{2}(\sqrt{2\lambda_1}u_1 - \sqrt{-2\lambda_3}u_3), \quad (44)$$

and

$$\bar{N}_2 = \frac{1}{2}(\sqrt{2\lambda_1}u_1 + \sqrt{-2\lambda_3}u_3). \quad (45)$$

The equations for the four solutions of the continuous epipolar matrix are provided in Table 3.

Table 3. The four possible solutions for the continuous four-point algorithm

	v_1	N	ω_1
Solution 1	$\frac{v_1}{d} = \sqrt{\alpha} \tilde{v}_1$	$N_1 = \tilde{N}_1 / \sqrt{\alpha}$	$\omega_1 = H - \tilde{v}_1 \tilde{N}_1^T$
Solution 2	$\frac{v_2}{d} = \sqrt{\alpha} \tilde{v}_2$	$N_2 = \tilde{N}_2 / \sqrt{\alpha}$	$\omega_2 = H - \tilde{v}_2 \tilde{N}_2^T$
Solution 3	$\frac{v_3}{d} = -\frac{v_1}{d}$	$N_3 = -N_1$	$\omega_3 = \omega_1$
Solution 4	$\frac{v_4}{d} = -\frac{v_2}{d}$	$N_4 = -N_2$	$\omega_4 = \omega_2$

9. FUN_STEREO_RANGE

The estimation of the distance between target and chaser starts when the target is in stereovision range. Periodically, the algorithm measures the distance in pixels between the features detected on the frames collected on the left and the right cameras. When the distance is above a certain threshold, reliable estimation of the range is computed through stereovision.

The points are detected on the left and the right frames with the MATLAB built-in “detectSURFFeatures” within the ROI built from the KLT tracking function. The features are then extracted with the built-in “extractFeatures” and matched with the “matchFeatures” command. A description of how these built-in MATLAB commands work can be found in the MATLAB documentation [46].

The i coordinates are multiplied by the dimension of the camera pixel (the “pix” value in the initializer) to convert the coordinate into meters. The new coordinate values are defined as $X_l(i)$ and $Y_l(i)$ for the left frame and $X_r(i)$ and $Y_r(i)$ for the right frame. The focal length f is added as the third element of the vector defined by

$$x_l(i) = [X_l(i), Y_l(i), f] \quad (46)$$

and

$$x_r(i) = [X_r(i), Y_r(i), f]. \quad (47)$$

The skew-symmetric matrix of $x_1(i)$ is indicated with $\hat{x}_1(i)$. If the camera is not rotating with respect to the chaser reference system, the rotation matrix between the frames is an identity matrix ($R_r = I$). For horizontal stereovision, the translation vector T_r has only the horizontal element different from zero and represents the distance between the optical centers of the cameras.

With these definitions it is possible to estimate the values of the depth scale λ applying a least-squares operation that optimizes

$$\lambda \hat{x}_2(i)T + \hat{x}_2(i)R_r x_1(i) = 0. \quad (48)$$

The range Z is estimated multiplying the mean of the n values of λ by the focal length as

$$Z = \frac{f}{n} \sum_{i=1}^n \lambda(i). \quad (49)$$

The distance between strongest features is then measured knowing the distance Z and the dimensions of the pixels as in Figure 16.

The range distance measures the segment indicated in Figure 16 as BG. The segment EG is the focal length which is provided with the camera specifications or can be estimated with camera calibration. The segment DF can be retrieved from the image measuring the difference in pixel coordinates and multiplying by the pixel width of the sensor. It is possible to build similar triangles and measure the physical distance AC between feature A and B with simple proportions:

$$AB = \frac{DE \cdot BG}{EG} \quad (50)$$

and

$$BC = \frac{EF \cdot BG}{EG} \quad (51)$$

The AC segment measures are stored in a memory array and used to retrieve the range when the stereovision estimation is not available.

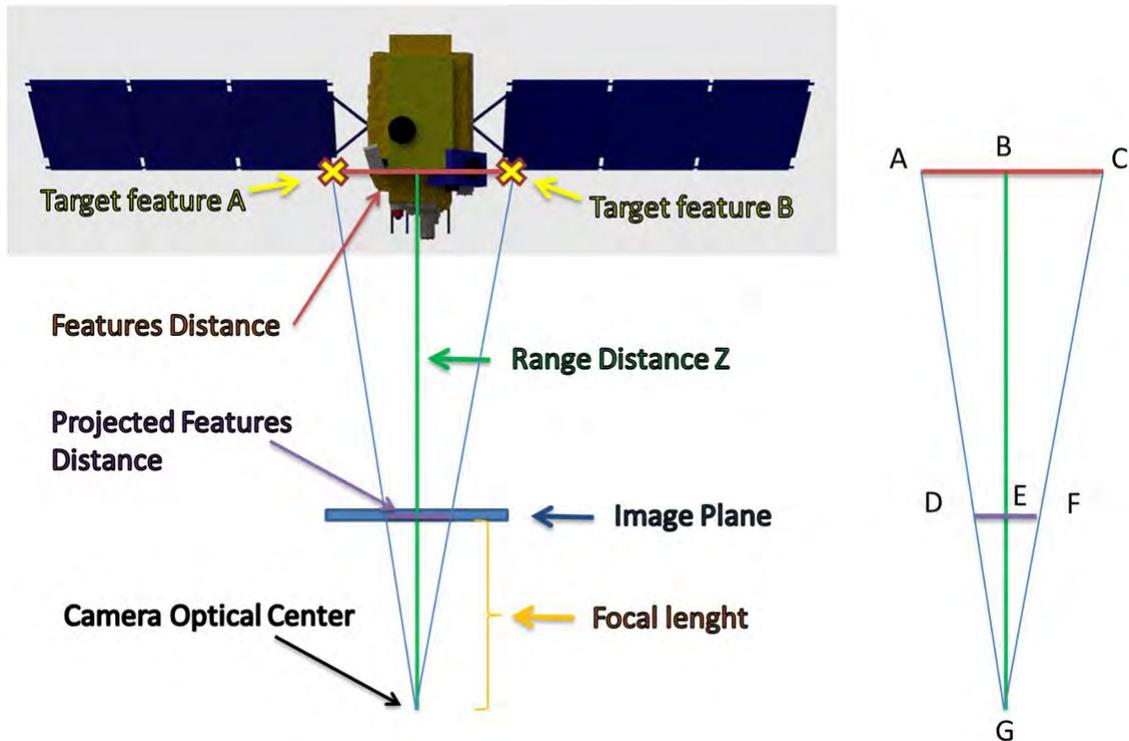


Figure 16. Example of estimation of the physical distance between two features using range, focal length and projected pixel distance.

10. FUN_GEOMETRIC_RANGE

The geometric range function uses the classified distances between strong features measured with the stereovision function and tracked with KLT or matching SURF. When the target is very close to the camera, changes in distance between these features is the only source of information for the estimation of the range. Supposedly, in this proximity phase the relative angular velocities are low, and the chaser is slowly approaching the target for docking. The dominant variable is the linear velocity along the axis orthogonal to the 2D image, and the changes in projected distance between features are considered mostly due to variations in range. As mentioned before, the estimation of the range is obtained from the inverted operation implemented in the stereovision function to estimate the distance between features.

C. ON-ORBIT TIMELAPSE AND COMPUTER RENDERED VIDEOS

In the development of the algorithm, the use of recorded or computer rendered videos was essential for the debugging and calibration of the code and for a first understanding of the constraints, limits and performance of the techniques implemented.

In the first phase of the development, computer rendered 3D videos were created in order to test and debug the algorithm. Computer rendered videos allow full control of all the parameters that affect the detection, tracking and estimation of a target. Ideal conditions with no background, wanted rotations and known features can be simulated as well as more complex scenarios where tumbling objects, moving background and reflections are introduced.

In the calibration phase the use of real, on-orbit footage is essential for testing the algorithm with real illumination conditions, real target features and real on-orbit background. To accomplish this task, NASA Johnson Space Center provided a collection of videos of on-orbit rendezvous and docking maneuvers with footage of the Space Shuttle, the ISS and Soyuz.

1. Computer-Rendered 3D Videos

The first debugging phase required a simple video, with no noise, high resolution and high frame rate in order to debug the detection and tracking algorithms.

The open source software Blender 2.72 [58] was used for the creation of the computer rendered videos described. The Blender 2.72 software includes all the tools necessary to create a 3D object, add texture and material characteristic to the surface and then record animated videos with adjustable background and illumination conditions.

The first video created was a simulated rendezvous and docking between two on-orbit spacecraft. Lighting conditions, reflections and background were added to make the video more realistic and to make the detection and the tracking more challenging for the algorithm. Some example frames are provided in Figure 17. This video was used for the debugging and first calibration of the Harris corner detection, the ROI selection, SURF description and the KLT tracking algorithms.

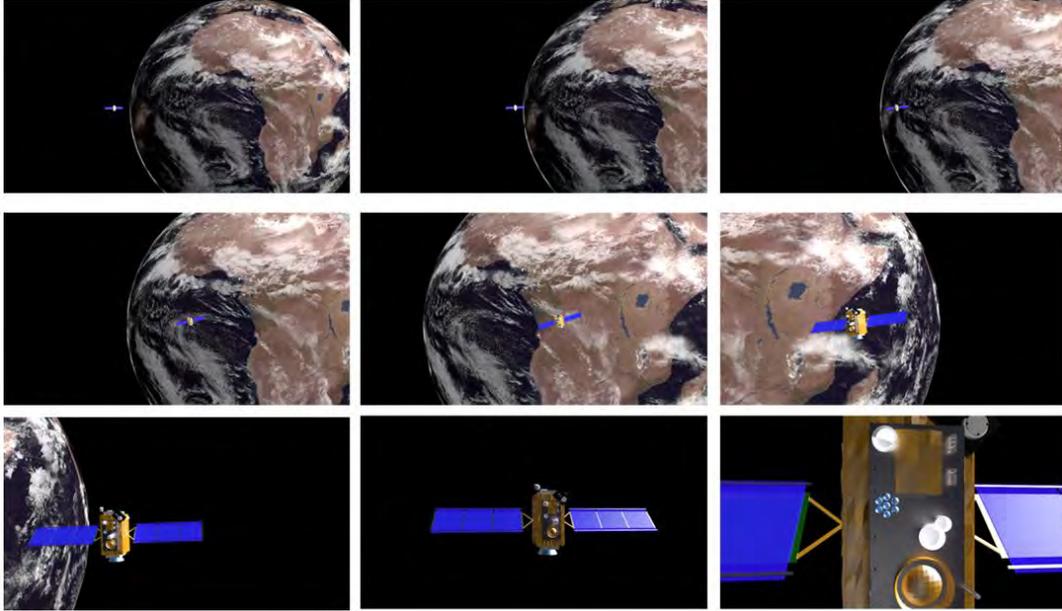


Figure 17. Computer rendered video of an on-orbit rendezvous maneuver for the debugging and first calibration of the vision algorithm.

A second video was implemented using the same model and maneuver simulated in the first one. The only addition to the second video was the simulation of a stereovision camera, obtained recording rendered videos from two virtual locations with a known offset. The stereovision offset is seen in the two frames (left and right cameras) shown in Figure 18.

Videos with only rotations or translation along known axes and easy to track features were created in order to debug the epipolar transformation algorithm. Several rendered videos were used to decouple linear and angular velocities in order to be able to detect errors in the code and test the estimation performance. Examples of videos used for this task are shown in Figure 19, where the rotation and translation of the objects rendered was changed accordingly to the measurement investigated.

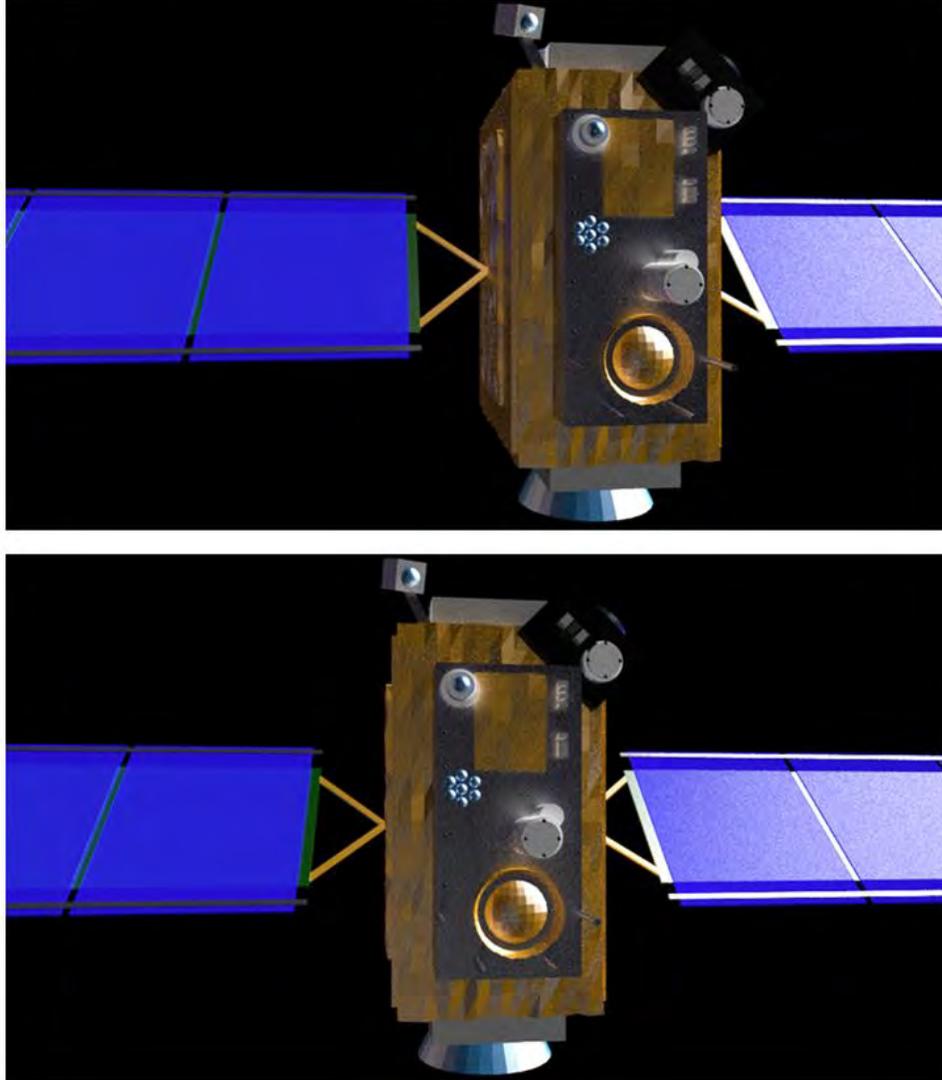


Figure 18. Frames from the two simulated cameras of the computer rendered stereovision video.

Another method was used to decouple the tracking error from the pose estimation error. A MATLAB script was developed to create a rigid rotating cloud of points. The script has as inputs the initial and final state vectors of the rigid body frame and generates arrays of geometrically organized or random points. The objects generated are then rigidly translated and/or rotated according to the initial and final condition. The output array of coordinates of the points before and after the rotation was used as error-free input to measure the quality of the estimation for linear and angular velocities. The algorithm is provided in Section B of the Appendix.

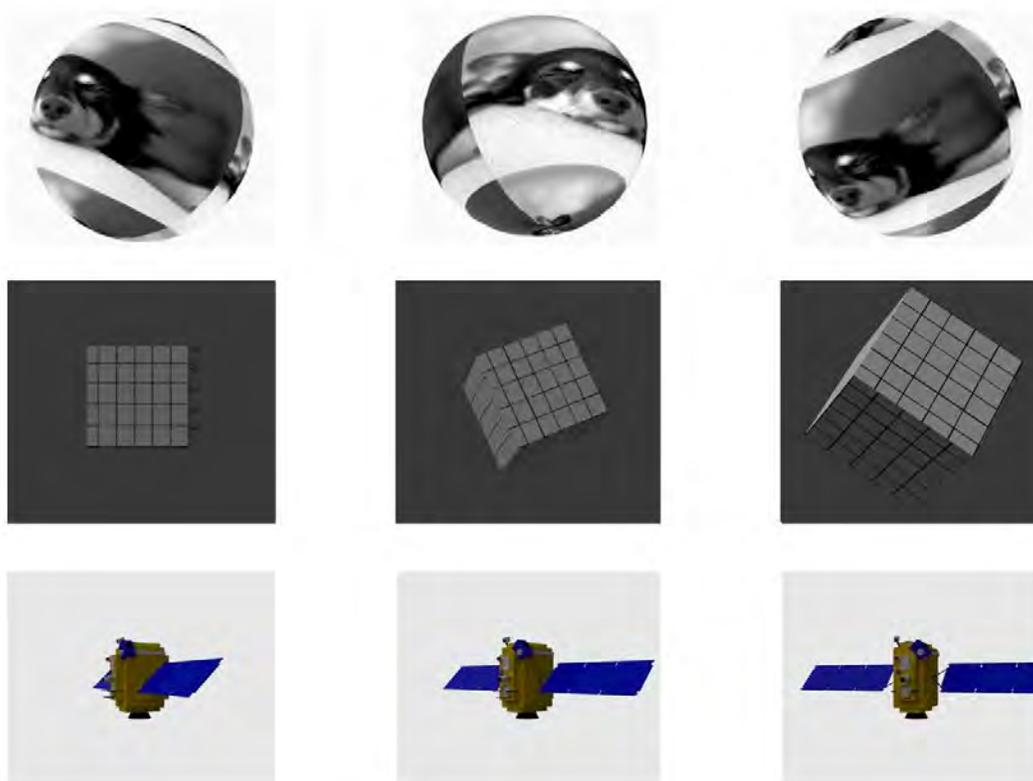


Figure 19. Examples of rotating objects in computer rendered videos for the debugging and calibration of the epipolar algorithm.

2. NASA On-orbit Videos

The “moving image repository” team at NASA Johnson space center provided a collection of nine video recordings from orbiting spacecraft’s during rendezvous, docking and relocation maneuvers. The videos have different illumination conditions, background and target features, matching the required generality necessary to calibrate and test the algorithm over several challenging conditions.

The videos were not provided with relative attitude, relative velocity between camera and target or stereovision information, and in most of the videos the acquisition parameters are not constant since the camera is adjusted in magnification, focus, aperture and orientation. For these reasons these videos were mostly used in this work to test the detection and tracking function of the algorithm.

The high resolution and frame rate of these videos simplifies the detection and tracking tasks but increases considerably the computational load and the memory required to process the data; therefore, the videos implemented in this research were degraded in terms of frame rate. In this work the reliable detection and tracking performance of the algorithm over the degraded videos showed that the relative velocities in space are in general slow with respect to the frame rate of the cameras, and lower acquisition rates can be used to reduce computational load and power consumption. Description, calibration parameters and observation of the test implemented on these videos are provided in Chapter V.

V. HARDWARE-IN-THE-LOOP EXPERIMENTS

The Hardware-in-the-loop experiments were conducted in the Spacecraft Robotics Laboratory at the Naval Postgraduate School in Monterey, CA on the Floating Spacecraft Simulator Test-bed (FSS). This thesis represents the latest of a series of research efforts dedicated to the investigation and development of autonomous spacecraft GN&C algorithms for rendezvous, docking, formation flying, collision avoidance, on-orbit assembly and robotic manipulation [59], [60]. In particular, previous efforts on an early version of the floating simulator was reported in [61] using single-camera vision and inertia measurement units for autonomous cooperative rendezvous and docking experimentation.

The current experiments are held on the fourth generation FSS test-bed, the product of several iterations and upgrades implemented over the years. A detailed description of the test-bed and of the experimental setup is provided in the following subsections.

A. THE FLOATING SPACECRAFT SIMULATOR TEST-BED

The FSS test-bed is a two-dimensional, three-degrees of freedom experimental facility for the dynamic simulation of on-orbit maneuvers. The test-bed is mainly composed of a high precision flat surface and a set of compressed-air based hovering units. The dynamics of the FSS on the flat surface reproduces closely, in 2D, the weightlessness and frictionless conditions of the relative orbital flight

1. High Precision Flat Floor

The high-precision flat surface, shown in Figure 20, is a 4 m \times 4 m granite table with a AAA surface precision grade, a planar accuracy of ± 0.0005 inch ($\pm 1.27 \cdot 10^{-5}$ mm) and a horizontal leveling precision of 0.01 deg.



Figure 20. Granite table of the FSS test-bed at the Naval Postgraduate School.

The granite table is located in a clean/low-reflective room and provided with an ARRI LED temperature lamp to simulate several illumination conditions. An image of the temperature lamp and an example of the illumination effect are provided in Figure 21 and Figure 22.

2. UDP Network

An ad-hoc internal wireless network is used for the stream of information between the VICON camera system, a Telemetry computer and the FSS units. More information on the development and implementation of this network can be found in [62].

The computers and the FSS units are provided with D-Link routers to connect with the wireless network. The executables that run on the FSS units and the Simulink models on the telemetry computer interface with the routers through customized Simulink UDP blocks (user datagram protocol), as described in [62], to compress, stream and receive telemetry information. A schematic of the network communication is provided in Figure 23. Wireless communication is indicated with black dash lines. Red

arrows indicate the infrared reflection on the passive markers of the VICON, and yellow arrows indicate wired connections.



Figure 21. ARRI temperature lamp used in the FSS testbed to simulate changes in illumination conditions.

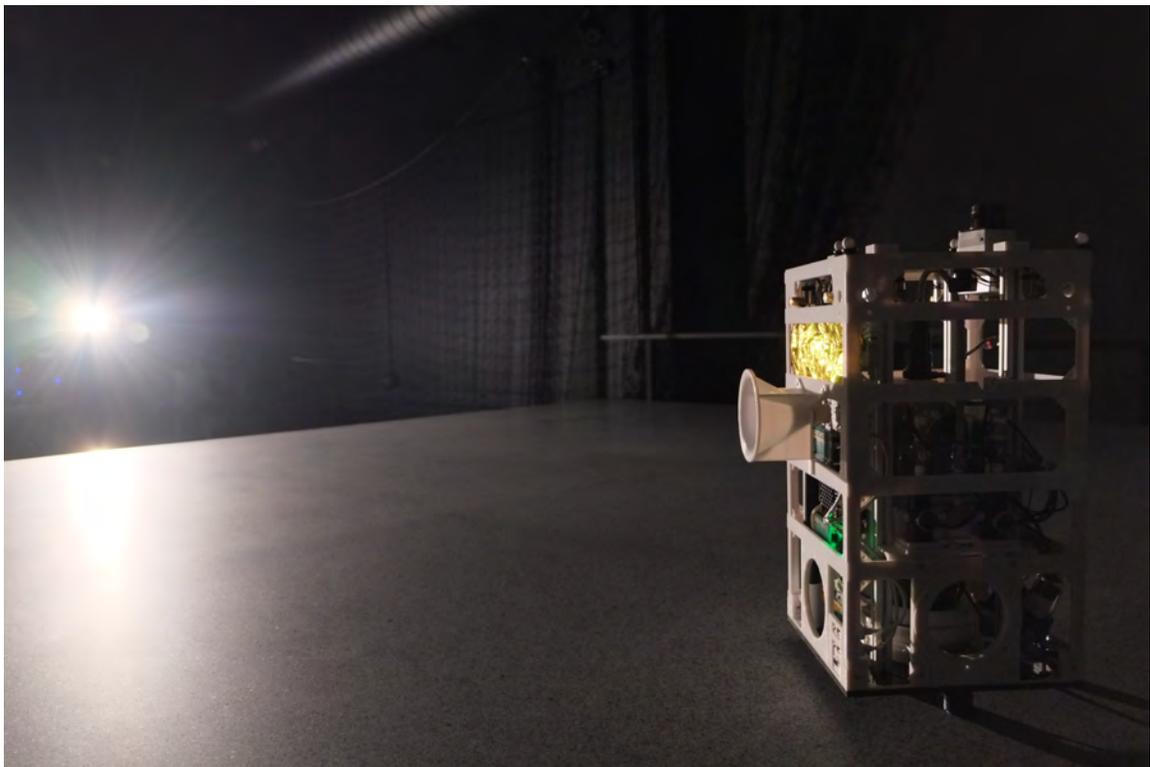


Figure 22. Example of the space-like illumination simulated on the FSS testbed.

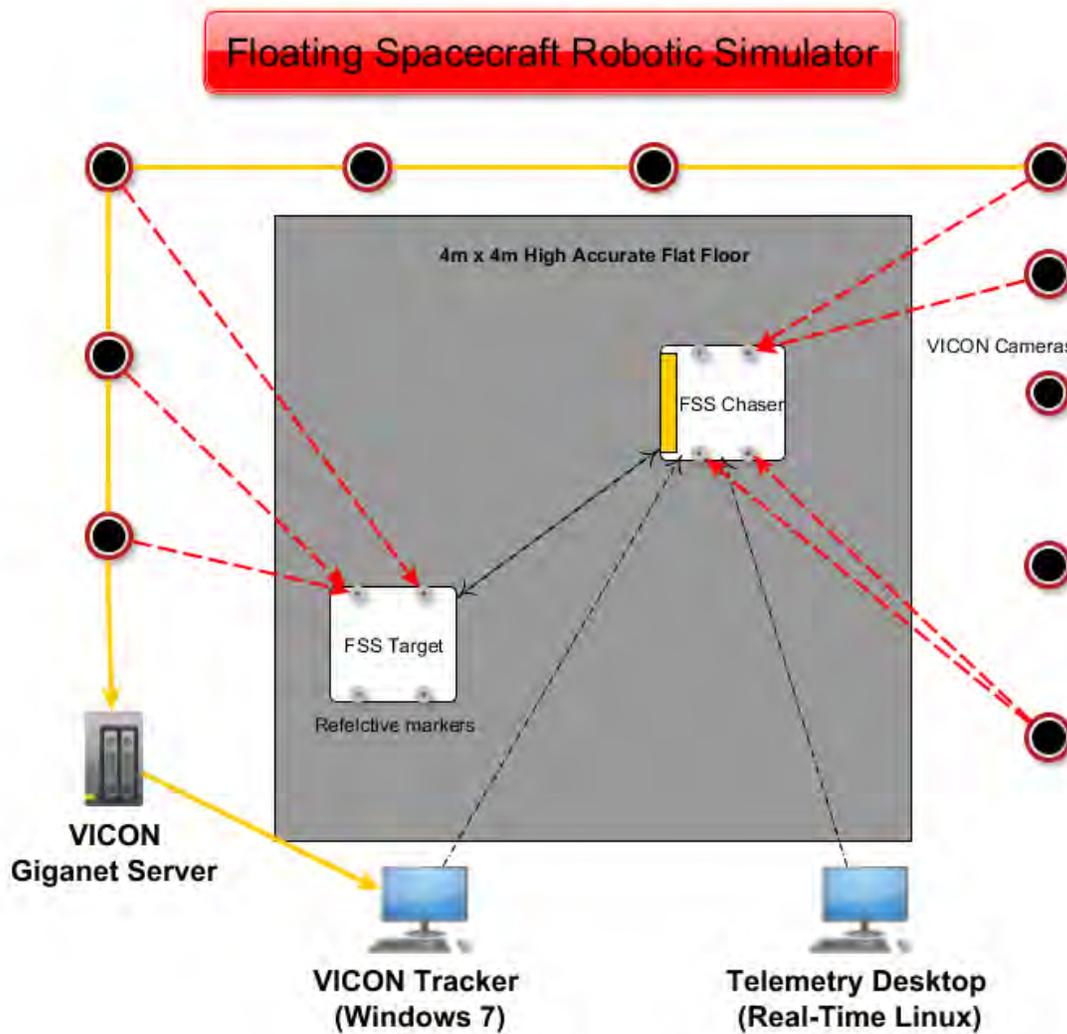


Figure 23. FSS network communication schematic.

3. Telemetry Computer

The telemetry computer is used to compile the algorithms, upload and start the executables, collect telemetry data and visualize results. The RTAI Linux OS is implemented in order to develop algorithms compatible with the real-time libraries installed on the FSS units. A screenshot from the telemetry computer is shown in Figure 24, where it is possible to see two terminals for the SSH (secure shell) wireless link communication with the floating units and a Simulink telemetry model for the collection of the data. The computer is also used to compile the Simulink models in RTAI executables.

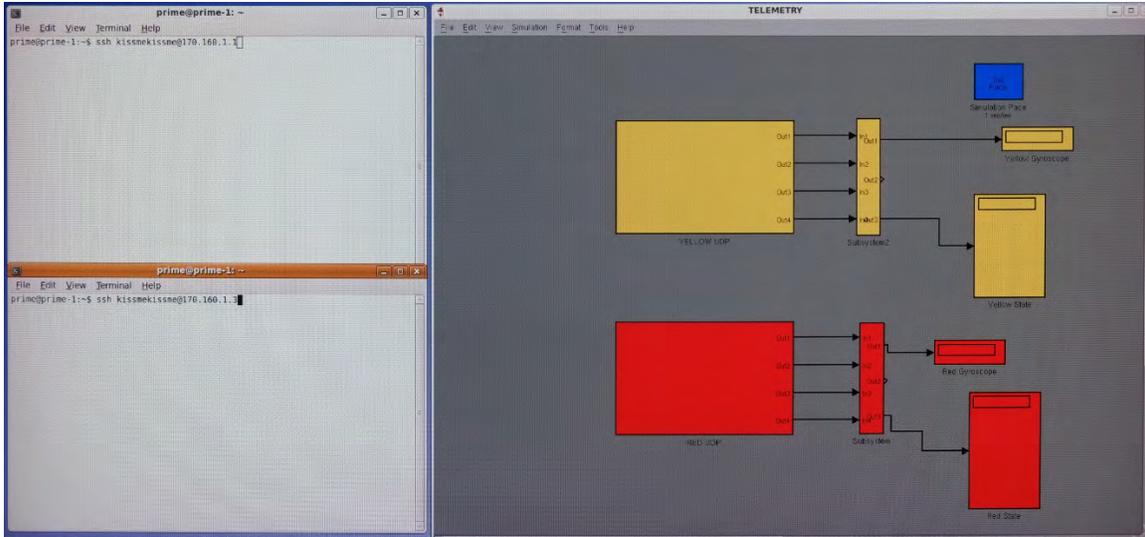


Figure 24. Desktop screenshot of the telemetry software and the SSH terminals.

Ten VICON cameras are installed along the walls of the laboratory to collect and stream high quality, 3D position and attitude information. The VICON system is used to simulate reliable star tracker data or to provide ground truth data in a fixed reference frame. The VICON server is able to provide the position and the attitude of a rigid body with a resolution between 0.001 and 0.01 millimeters. The refresh rate is limited only by the streaming rate of the UDP network, while the resolution depends on the distance and number of passive markers on the tracked body. The VICON cameras can be recognized as red light above the granite table in Figure 25. A closer view of one of the VICON camera is provided in Figure 26. A screenshot of the VICON Tracker software is shown in Figure 27.

4. Floating Units

The FSS test-bed also includes a set of floating units, each provided with a compressed air tank and three flat air-bearings on the bottom. The air-bearings are non-contact interfaces that ensure uniform pressure distribution of the film of compressed air on its surface. The release of compressed air through the bearings generates a small and constant hovering effect that creates a gap of about five microns between the granite flat surface and the pads, drastically reducing the friction. A picture of one of the new generation floating units is provided in Figure 28.



Figure 25. View of the VICON cameras above the granite flat floor of the FSS.



Figure 26. One of the VICON cameras connected to the ceiling of the Spacecraft Robotics Laboratory.

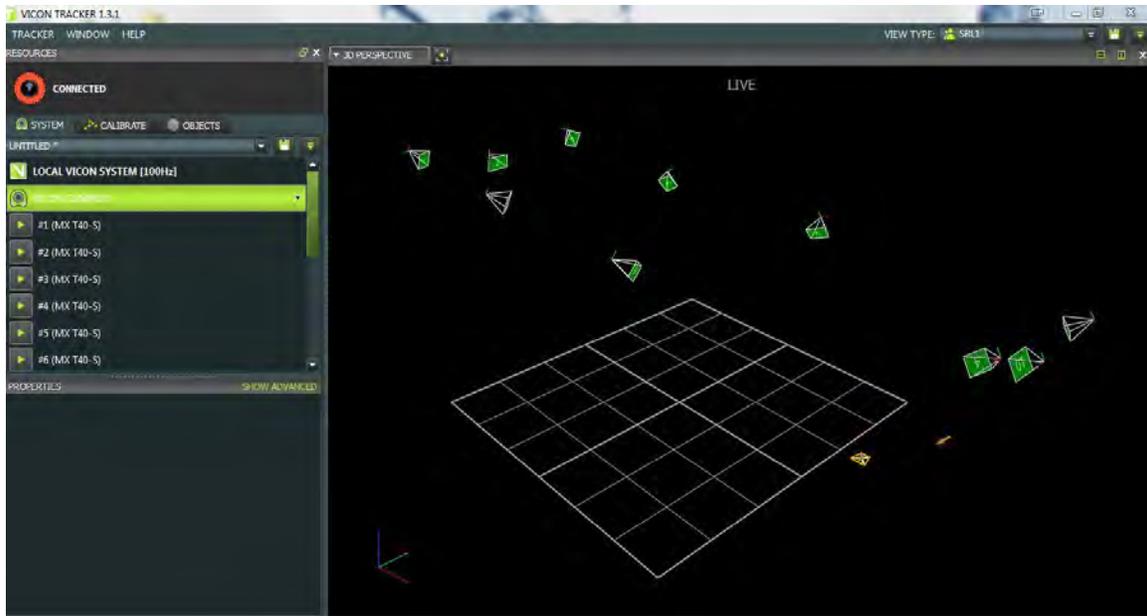


Figure 27. Screenshot of the VICON software tracker. It is possible to recognize (as green squares) the position of the cameras installed along the walls of the laboratory.

The most important components of the floating unit system are highlighted in Figure 29. The external structure was printed using the Fortus 400mc 3D rapid-prototyping printer of the NPS Space Systems Academic Group, while the internal structure is made of aluminum and carbon fiber. The units were built to simulate fully autonomous, small spacecraft and are provided with on-board propulsion, electronics, computer and sensors.

a. Propulsion System

The propulsion of the FSS units is provided by eight supersonic thrusters mounted on each side of the four corners of the external structure. The thrusters release compressed air through custom-made supersonic nozzles mounted on solenoid valves. The air is provided by the same tank that feeds the floating system, while the valves are directly controlled by the PC104 relay board. The compressed air hovering and propulsion system are better described through Figure 30 and the detailed component schematic of Figure 31. Each thruster can produce up to 0.159 N. The combined

activation of the thrusters provides the actuation for the attitude and position control of the unit [63].

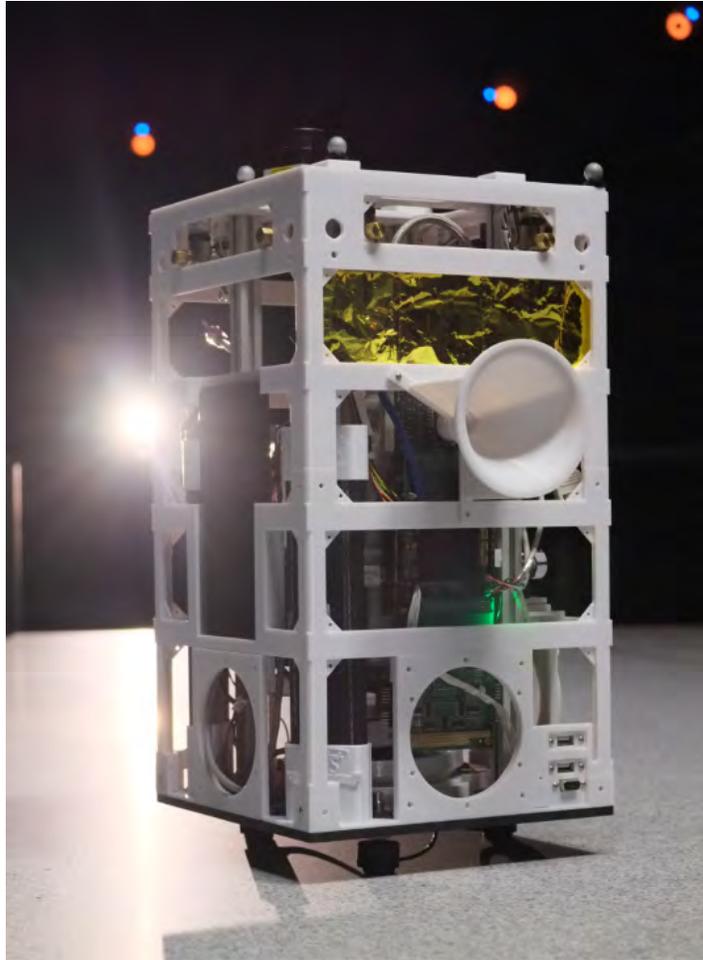


Figure 28. Picture of a fourth generation FSS floating unit.

b. Electronics

A schematic of the electronics mounted on the fourth generation FSS unit is provided in Figure 32.

The power is provided by an Ocean Server Board DC-DC converter. The power board uses two Lithium batteries and provides energy to all the on-board electronics.

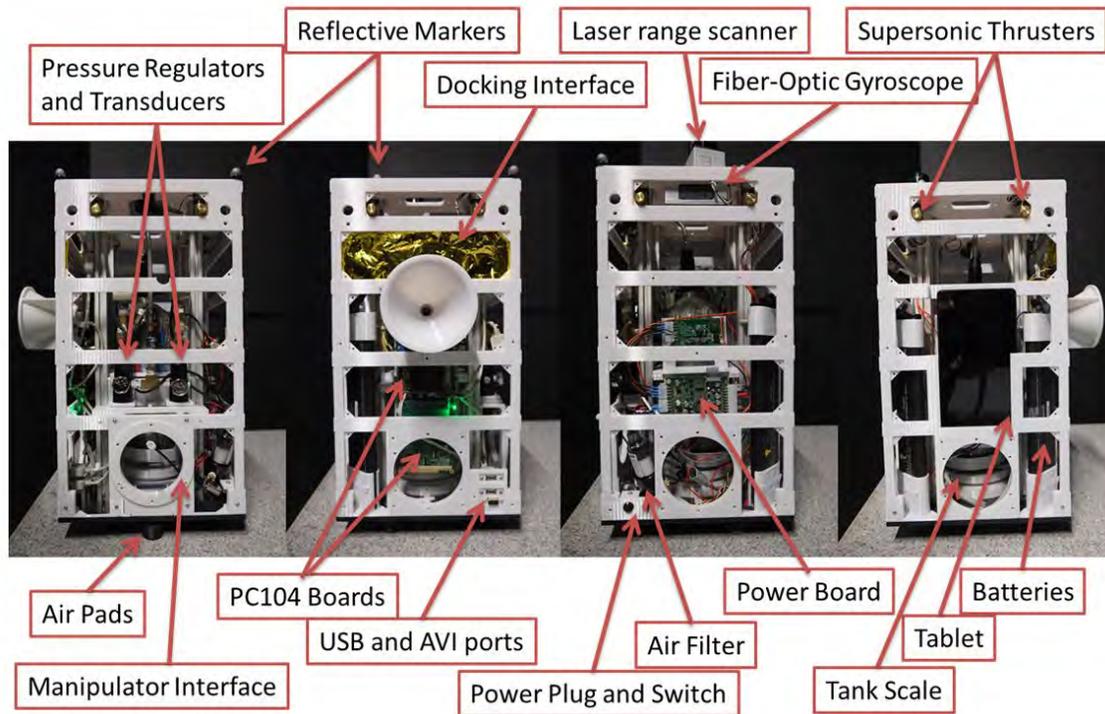


Figure 29. Main components of the FSS units on the four side views.

A stack of PC104 boards is the core of the FSS unit. The main computer is a PC104 ADLS15 PC, Intel® Atom® processor, 1.6 GHz with 2 GB of DDR2-DRAM and a 4 GB On-Board SSD. The computer runs a RTAI Linux compiled version of Ubuntu. This device is used to command and run the executables during the experiments and is connected to all the main actuators and sensors. The PC104 stack includes a serial-port board with nine RS232/485 ports used to connect the on-board PC with several devices such as the fiber-optic gyroscope, the power board and the docking electro-magnets. The solenoid valves of the thrusters and of the air bearings are controlled by the PC through a 20SPST PC104 relay-board.

The stereovision is powered and streams the images through a WDL Systems Fire-wire PC104+ board connected to the PC. The Fire-wire board has two channels and a transfer rate of 400 Mbit/sec.

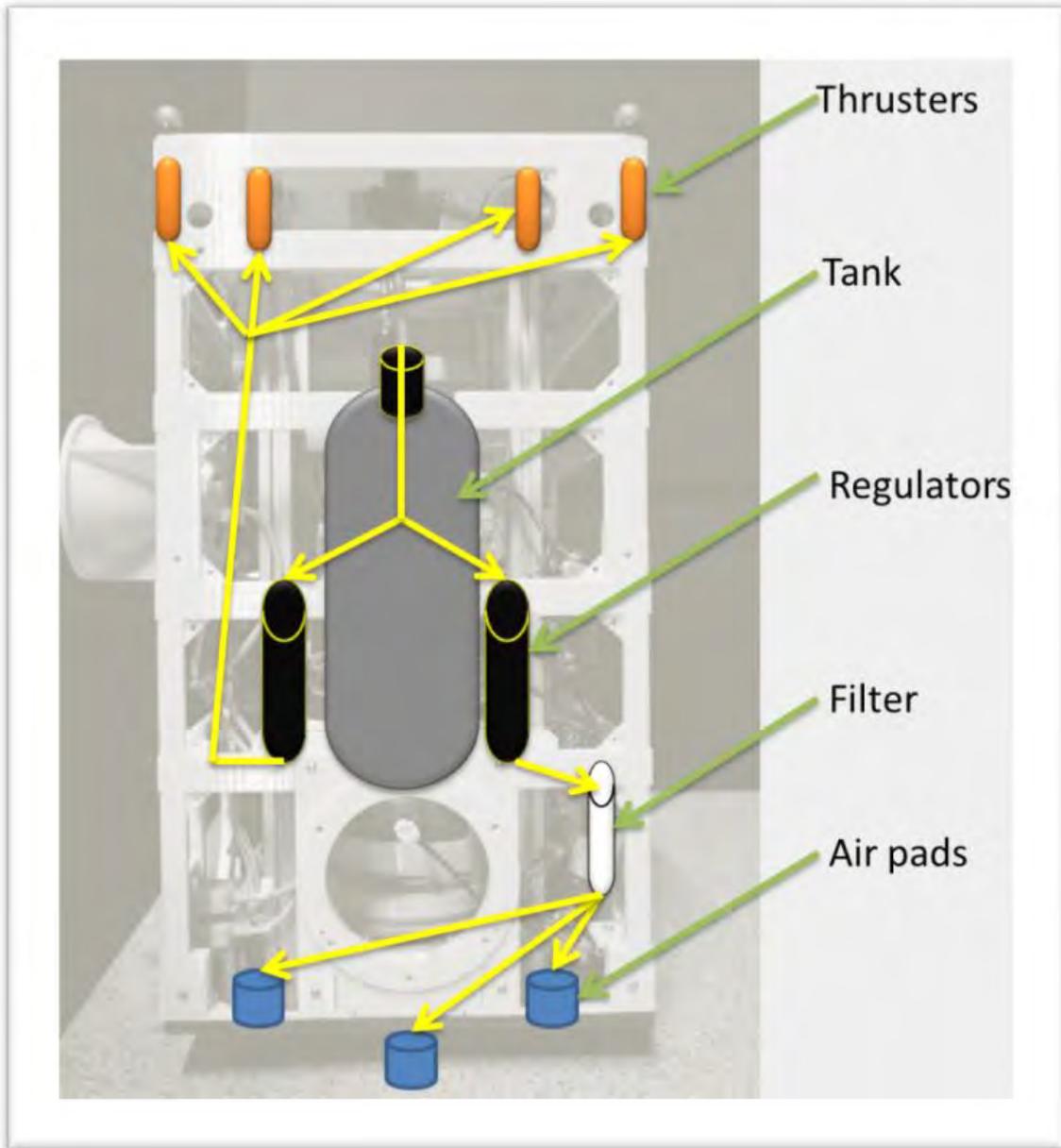


Figure 30. Representation of the hovering and propulsion system. The air flow is represented with yellow arrows.

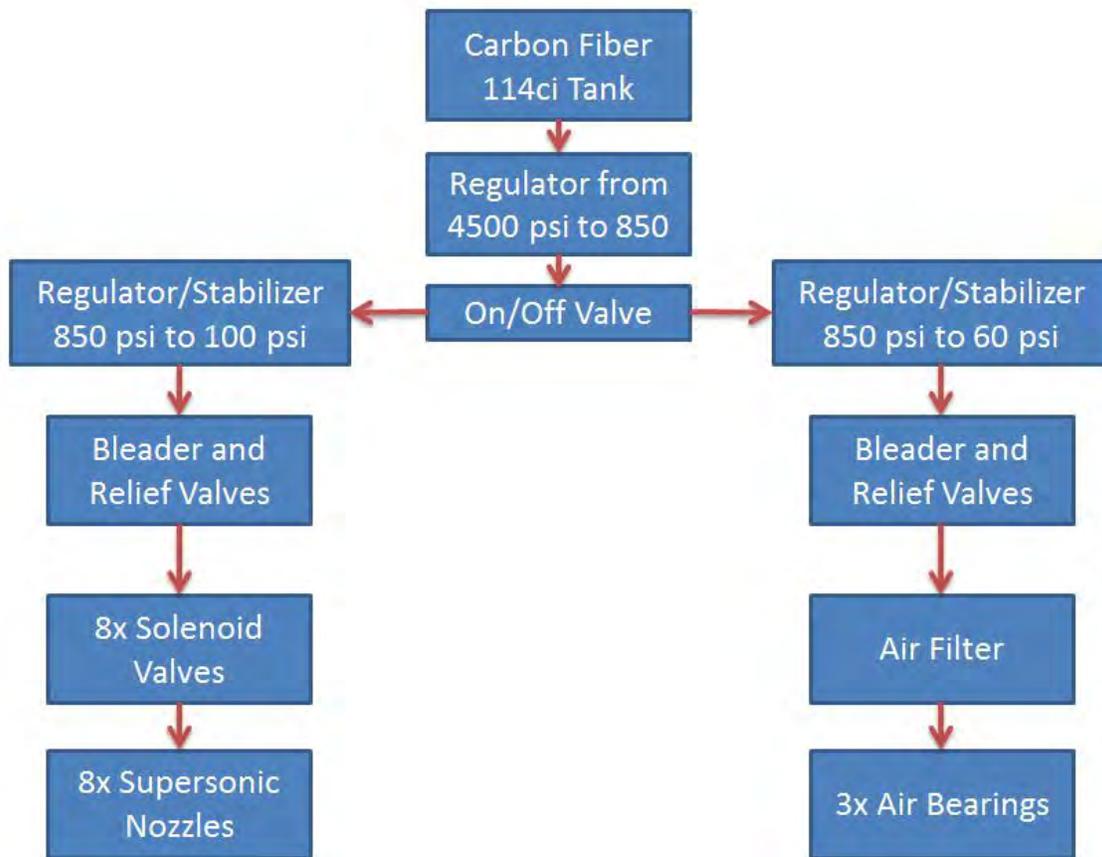


Figure 31. Schematic of all the components of the compressed-air hovering and propulsion system of the FSS floating unit.

Other electronic components are the DLINK wireless routers for the Wi-Fi network connection and the pressure transducers, mounted downstream with respect to the propulsion and floating systems' regulators, to calibrate and control the output pressure. The fourth generation FSS units are also provided with an electronic on-board scale that displays the weight of the high pressure tank, providing an estimate of the consumption of compressed air. An Android tablet mounted on the side of the FSS units is used as secondary wireless control device, used mainly to stream videos and connect with the Go-Pro Cameras. Future use of the tablet includes use as a portable control unit for the Ubuntu terminal and as an additional camera for the docking phase.

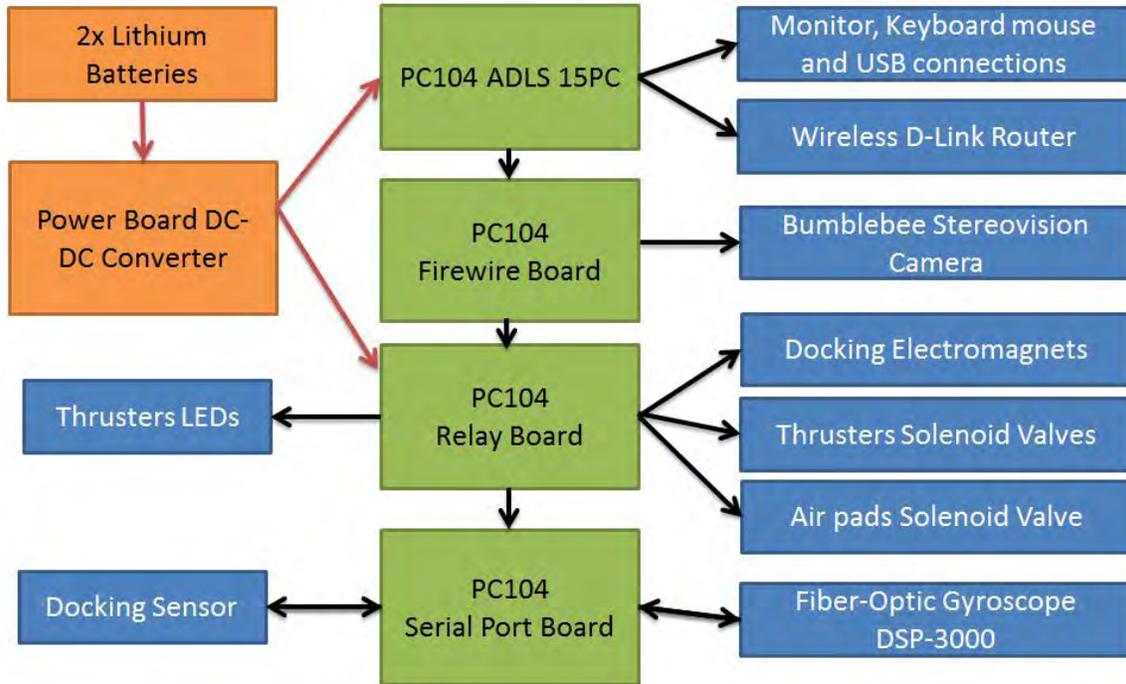


Figure 32. Schematic of the FSS unit electronic system.

c. On-board Sensors

The units are provided with a PointGrey BumbleBee XB3 BBX3 stereovision camera used to retrieve the input images for the vision algorithm. The camera specifications are provided below [64]:

- Color Version: Mono
- Focal Length/FOV: 3.8 mm, 66-deg HFOV
- Resolution: 1.3 Megapixels
- Imaging Sensor: Sony ICX445, 1/3", 3.75 μm
- Imaging Sensor Out: 1280 \times 960 at 16 FPS
- Digital Interface: 2 \times 9-pin IEEE-1394b for camera control and video data transmit
- Transfer Rates: 400 Mbps

An image of the camera is provided in Figure 33.



Figure 33. Point Grey Bumblebee stereovision camera, from [64].

The units are also provided with a DSP-3000 fiber optic gyroscope from KVH. The fiber optic gyroscope provides angular rate information with a bias of 20 degrees per hour and a linearity of 500 ppm (parts per million) [65].

An image of the fiber-optic gyroscope is provided in Figure 34.



Figure 34. Fiber-optic gyroscope DSP-3000 from KVH [65].

Future experiments will include proximity data from the Hokuyo Laser Scanner [66] (shown in Figure 35) and the Leap-motion Infrared Scanner [67] (shown in Figure 36) to improve target range estimation during docking and proximity operations. The integration of these two sensors is still in development stage.



Figure 35. Hokuyo laser scanner, from [66].



Figure 36. Leap Motion, from [67].

5. FSS Software

The GN&C algorithms that control the FSS units are mostly developed and compiled in MATLAB/Simulink. A repository of RTAI Linux compatible Simulink blocks for the actuation, UDP streaming and sensors interface were developed at the Spacecraft Robotics Laboratory [62] and used in all the FSS test-bed experiments and upgrades. The latest version of the general Simulink model used on the FSS to compile the real-time executables are described in this section.

Each Simulink model has at least five main blocks, representing the basic tasks of the executable (Input Sensors, State Estimator, Guidance, Actuator and Telemetry). The blocks are collected into Atomic blocks that isolate the sampling time of each task and provide multithreading capabilities to the executable. With the implementation of Atomic Blocks, the model provides the processor with defined rates and tasks priorities that are used to reallocate computational load. Multithreading solutions were investigated to

overcome the problem of high computational demanding tasks, like optimal guidance and image processing.

a. Main Model

In the standard algorithm developed, the main model connects together a total of six atomic blocks. The model represents the logic connections between Sensing, Kinematics, Estimation and Guidance. All the data is transmitted between the blocks through buses, which allows indexed data on only one line, making the model faster, better organized and easier to read.

An important part of this work includes the research on Atomic blocks implementation. The research was based on a literature review and on hardware experiments to prove the feasibility and the performance of the algorithm with multithreading capabilities.

In order to make the model able to run in multithreading, each block must comply with the following requirements:

- The blocks have to be contained in an Atomic block.
- The Atomic blocks must be function-call generated blocks.
- The function-call generator must specify the sampling rate of the block; larger sample times automatically means lower priority.
- All the inputs of the Atomic blocks must pass through a rate transition block.
- No triggers, clocks, Go-To, From or other Simulink sources can be used in the Atomic blocks.

A screenshot of the full model is provided in Figure 37.

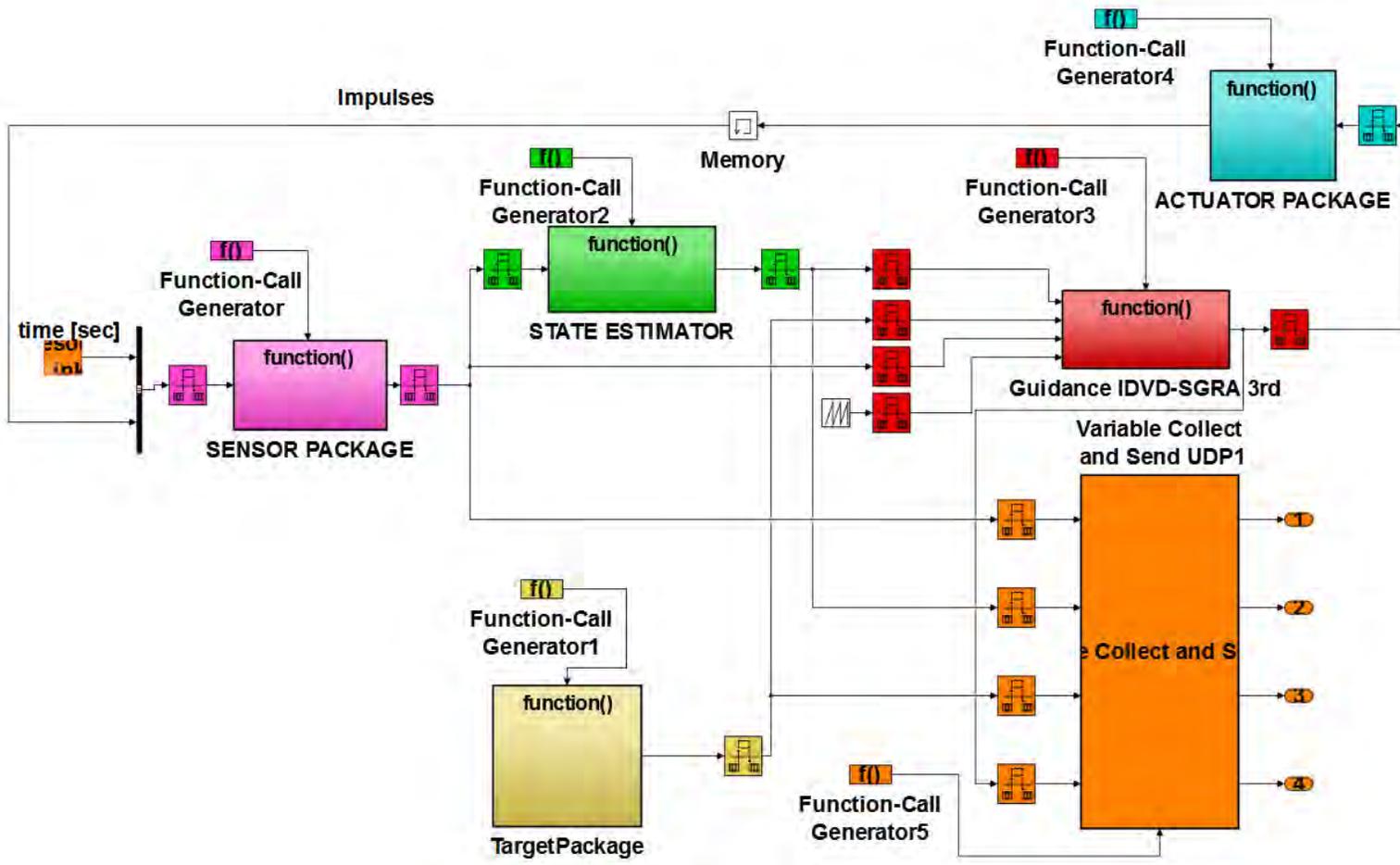


Figure 37. Main Simulink model before the compilation into an executable. Each Atomic Block is identified with a different color.

b. Sensor Package

The sensor package can work in either real mode or simulated mode. The real mode runs two S-function blocks to receive data from the fiber-optic gyroscope and from the VICON camera system. The simulated mode simulates attitude and position data and noise. The simulated data is necessary for the implementation and debugging phase of the guidance algorithm. The data is collected in a bus connection that provides machine time, VICON time and position of the FSS units.

c. State Estimator

The state estimator uses the measurement information and the impulse values from the actuator package to compute the state vector. The computation uses a discrete Kalman filter to compute a state vector robust to sensor measurements losses and errors.

d. Guidance Block

In the guidance block, the information from the state estimator and from the target package are used to compute the forces and torques required in order to accomplish the tasks of the algorithm. A description of specific guidance logic is beyond the scope of this work. Some of the most significant guidance and control logic tested and implemented on the FSS are listed in Table 4.

Table 4. List of the main guidance logic implemented on the FSS.

Guidance Algorithms
Linear quadratic regulator (LQR)
Inverse dynamics (ID)
Inverse dynamics in virtual domain (IDVD)
Proportional-integral-derivative (PID)
Artificial potential function (APF)

e. Actuator Package

The input of the actuator package is the time-history of the forces and torques requested by the guidance block. These values pass through a Schmitt trigger and a pulse-width modulator block to convert the force commanded by the continuous guidance

algorithm into discrete aperture time intervals. The length of the time intervals is a function of the minimum actuator aperture time and the thrust of the propulsion system. The block is also provided with an S-function to communicate with the PC104 relay board. The S-function is used to activate the solenoid valves of the thrusters and of the air pads.

f. Variable Collect and Send

This atomic block saves all the data that is exchanged between the atomic blocks that pass through the buses. The data is also streamed in real time to the telemetry computer and to the other floating units. The UDP connection is obtained through a custom S-function compatible with the RTAI compiler.

g. Target Package

The target package uses a receiver S-function to retrieve the state vector of other FSS units. The state vector is used by the guidance block to compute rendezvous, docking and collision avoidance maneuvers.

B. EXPERIMENTS AND RESULTS

Several experiments have been conducted in order to test and calibrate the algorithm. In this section the experiments are classified into three groups:

5. Test Videos: Test and calibration of the algorithm performed on sample videos.
6. NASA Videos: Calibration of the algorithm on the high quality on-orbit videos provided by NASA
7. Live Target: The inputs of the algorithm are live images of a physical moving object.

All these experiments are described in the following subsections.

1. Test Videos

In the first phase of experiments, a series of tests were performed running the algorithm on a desktop computer and using test videos as inputs. This experimental setup

was necessary to calibrate the algorithm before using live-stream images. The use of computer rendered and recorded videos allowed comparison of the algorithm setup for the same sequence of frames. The test videos experiments were classified according to the group of functions tested.

a. Detection and Tracking Calibration

The first group included a test of the initialization detection and tracking. The algorithm must be able to remove the background, detect the target, initialize a ROI and track the features to update the ROI location and dimensions. Three videos were used to test these functions:

1. A computer rendered spacecraft maneuver with Earth spinning in the background and with artificially simulated trajectories and reflections.
2. Inverted time-lapse of one of the Orbital Express maneuvers.
3. Inverted time-lapse of Cygnus maneuvers in proximity of the ISS.

Specifications of the videos and main initialization setup used to obtain the best performance are provided in Table 5.

Sequences of frames from the abovementioned experiments are provided in Figure 38, Figure 39 and Figure 40, where the green dots indicate Harris features, the red dots indicate KLT tracked features and the yellow box indicates the limits of the ROI.

The main calibration differences are the reduction of the KLT ROI dimensions in video 2, the activation of the background subtraction in video 3, and the different values for the Harry quality threshold in all three experiments.

In video 2 the dimensions of the ROI generated from the KLT and the KLT minimum discarding distance were reduced. This modification was necessary to reduce the probability of detecting background features more like the target than in the other videos.

Table 5. Detection and tracking calibration values.

	Video 1	Video 2	Video 3
Video Name	Computer-Rendered Satellite Maneuver	Orbital Express Docking	Cygnus approach to ISS
Frame Rate	24 fps	10fps	29 fps
Resolution	960×540	318×316	480×480
Number of frames	300	140	179
Compression	avi	avi	avi
Background Subtraction	Not Active	Not Active	Static Background Subtraction
Detector	Harris Corners	Harris Corners	Harris Corners
Harris stronger features	100	100	100
Harris corner quality	0.05	0.48	0.07
ANMS	Not Active	Not Active	Not Active
Blob length	100 pixels	100 pixels	100 pixels
Blob sigma	6	6	6
Blob-ROI base dimension	20 pixels	20 pixels	20 pixels
Tracker	KLT	KLT	KLT
KLT discard distance	20 pixels	15	20 pixels
KLT-ROI base dimension	50 pixels	40	50 pixels

In video 3 the low or null relative motion with the background objects allows automatic removal of most of the background features in the initialization phase, such as the Earth, the ISS Robotic Arm and some visible ISS body features.

From Table 5, we note notice that in video 1, the computer rendered video, the algorithm recognizes corners with a low Harris corner quality threshold, while the real videos require higher quality threshold to recognize the artificial features from the background. In order to demonstrate a correlation between the Harris threshold required and the quality of the image, an additional test was necessary.

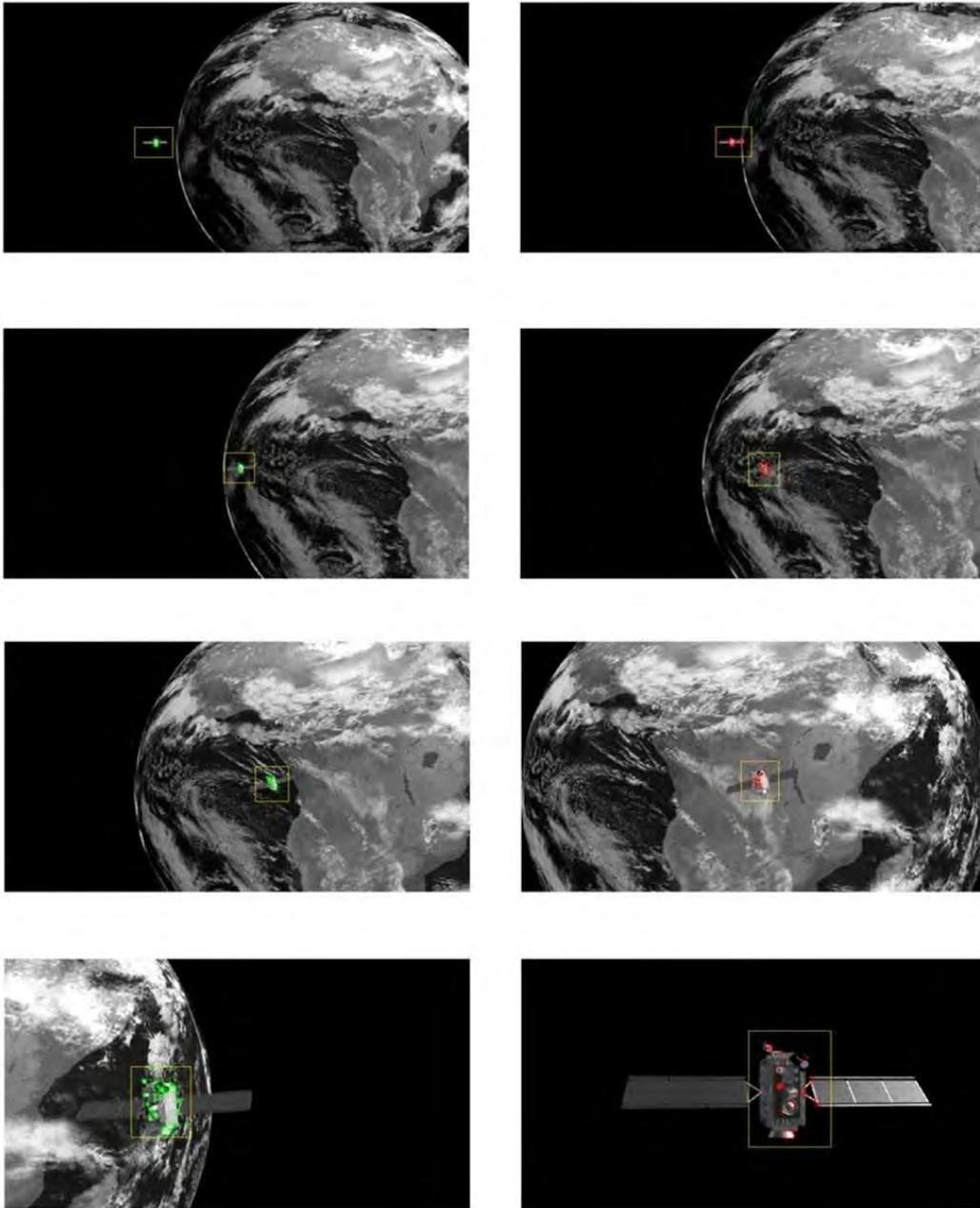


Figure 38. Sequence of frames from the detection and tracking test on video 1. Harris corner features are represented in green, KLT tracked features in red and the ROI is the yellow square.

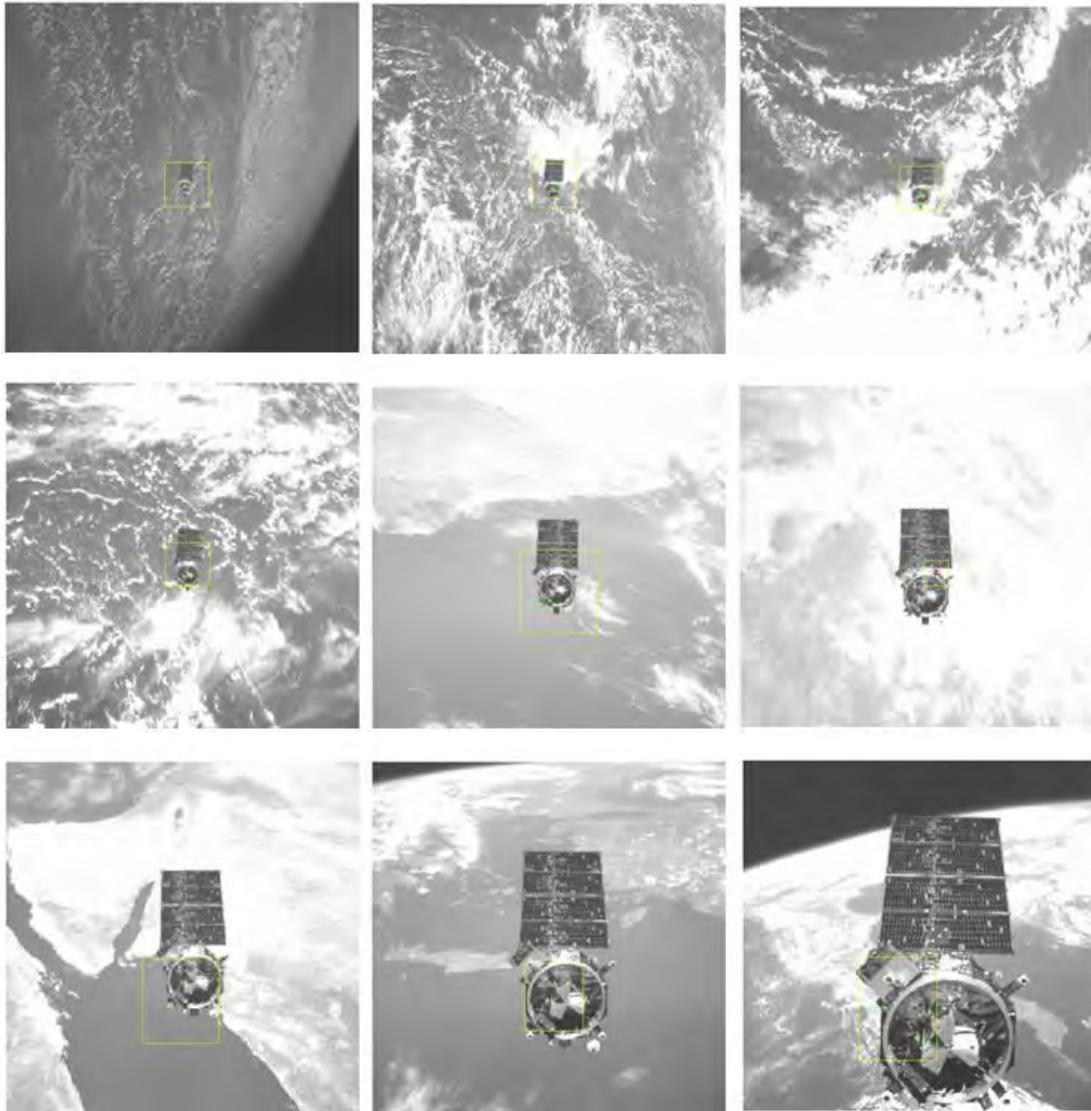


Figure 39. Sequence of frames from the detection and tracking of video 2.
 Harris corner features are represented in green, KLT tracked features in red and the ROI is the yellow square.

Video 1 was degraded to the same resolution and frame rate of Video 2, keeping all other parameters unchanged. The results showed that the reduction in frame rate and resolution do negatively affect the detection as expected, requiring a higher Harris threshold to filter non-artificial features.

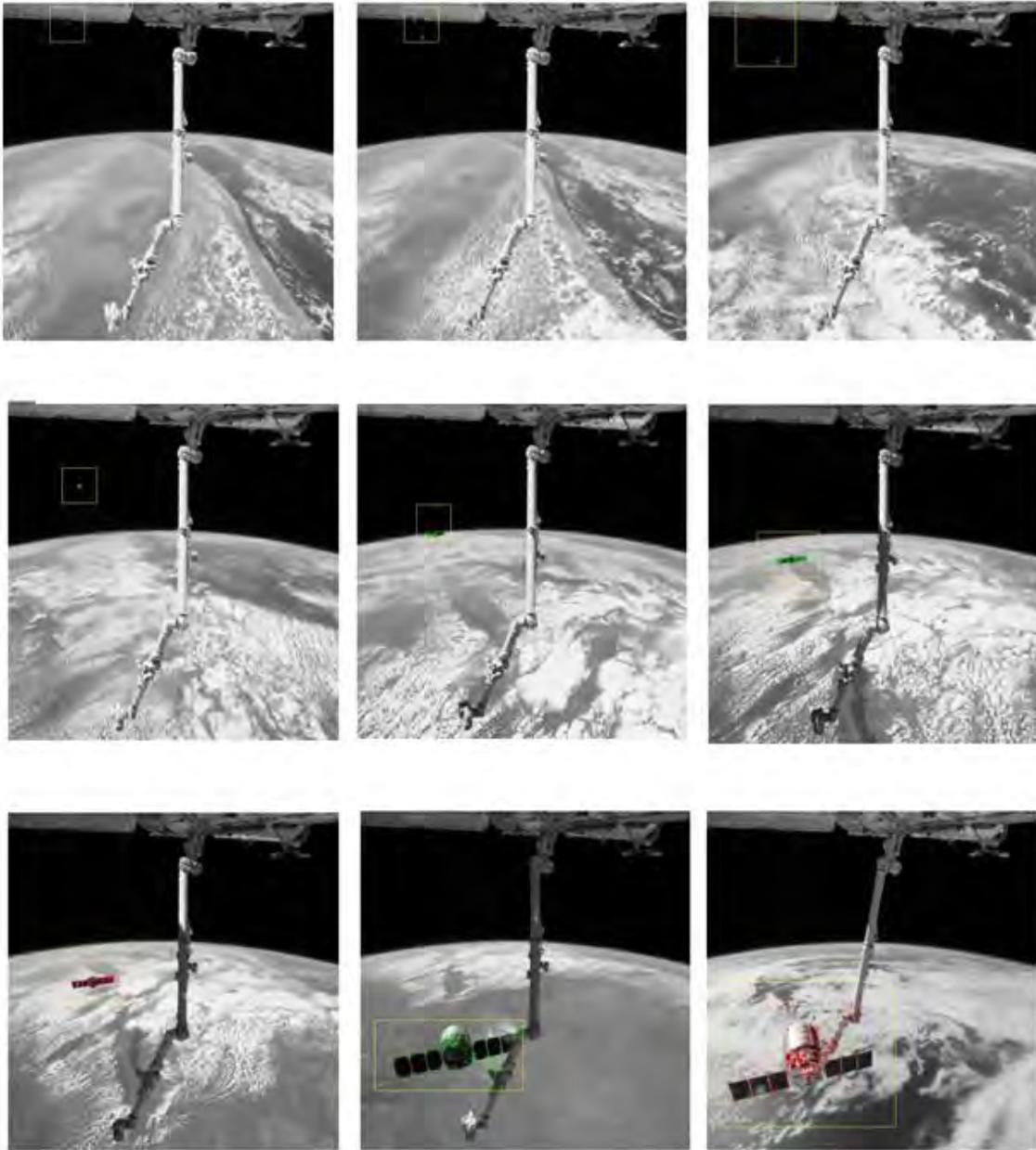


Figure 40. Sequence of frames from the detection and tracking of video 3. Harris corner features are represented in green, KLT tracked features in red and the ROI is the yellow square.

b. Epipolar Transformation Test

The epipolar algorithm was first tested and corrected with the MATLAB computed array of coordinates generated with the code provided in Section B of the Appendix. The code generates a cloud of points rigidly translating and/or rotating

according to the user inputs. The output simulates the array of coordinates generated with a tracker.

A second phase of the test was based on computer rendered videos of a generic 3D satellite used to simulate simple linear motion and rotations along one axis per time. During the tests, this simulated satellite was detected and tracked by the algorithm and the array of coordinates of tracked points sent to the epipolar function. The motion of the six cases tested is indicated with an arrow in Figure 41.

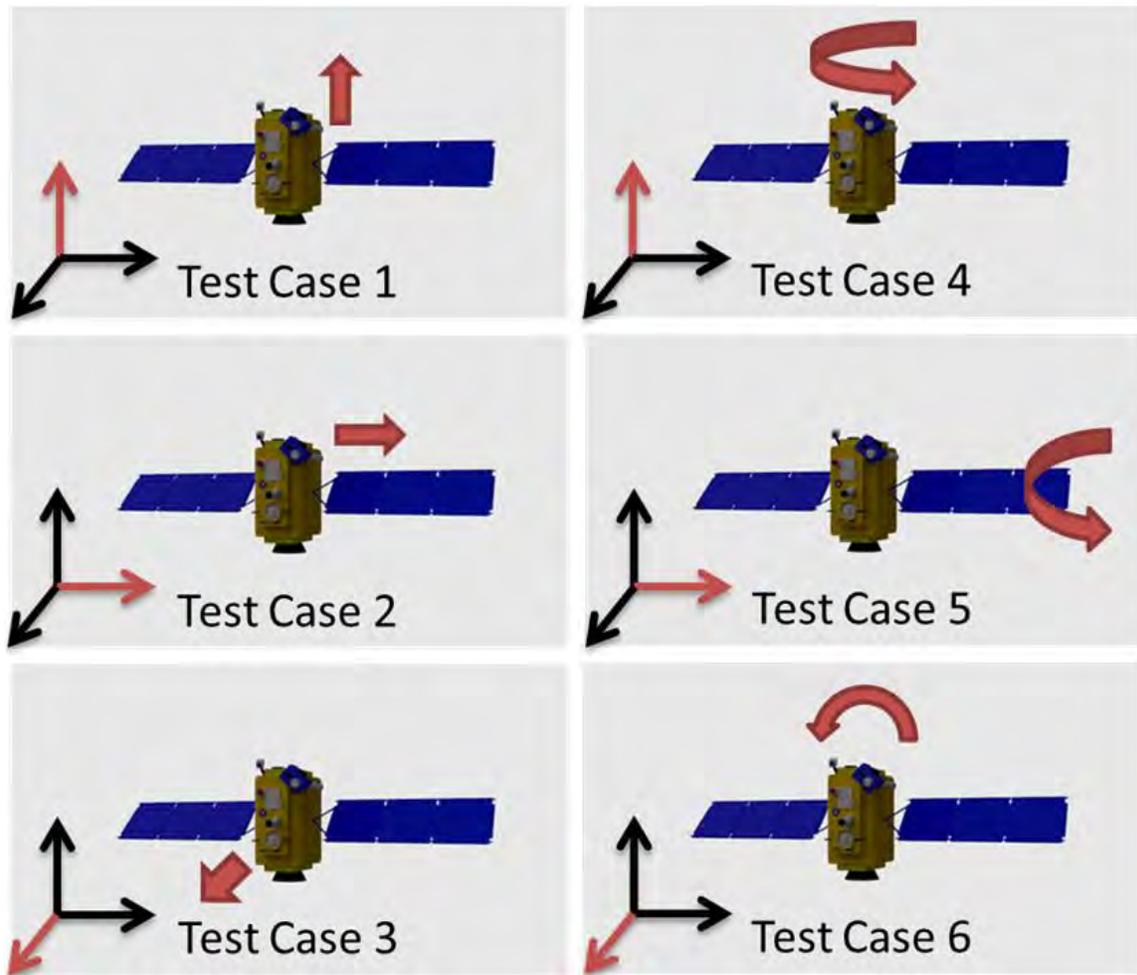


Figure 41. The four test cases to verify the epipolar transformation algorithm.

The first algorithm tested was the linear eight-point algorithm, described in Section IV. This algorithm was unable to provide a unique solution on the computer rendered video. When the features are quasi-planar on the z -axis (axis orthogonal to the

camera plane), the rank of the matrix X is smaller than eight, and the estimation remains undetermined.

The rendered video has a high frame rate compared to the motion. This feature improves the performance of the tracking algorithm but reduces the displacement between matched-features. The linear epipolar algorithms are negatively affected by this, as mentioned in Section IV.

A rendered video was created to test the linear algorithms with the rotation and translation of a group of shapes designed to enhance parallax and features difference in depth. Some frames are provided in Figure 42.

It is safe to assume quasi-planarity and continuous motion for most of the features analyzed in this work. Indeed, most of the features on artificial satellite lay on coordinates with depth dimensions small with respect to the other parameters involved, like the trajectory length and the distance between the camera and the target. Furthermore, most of the space proximity maneuvers are performed at low relative velocities to reduce risks of collision, fuel consumption and other possible docking problems. For these reasons most of the testing and development was focused on the use of the continuous planar four-point algorithm.

The four-points algorithm provides two possible velocity solutions for each time interval. Future work should be dedicated to implementing a Kalman filter that autonomously selects the most valid solution and corrects estimation errors. The time-history of the two solutions computed by the epipolar function for each case are provided in Figure 43, Figure 44, Figure 45, Figure 46, Figure 47 and Figure 48. From these figures, it is possible to see that only cases 1, 2 and 6 provide the expected results, while the curves obtained from the other cases clearly do not represent the dynamics simulated.

The algorithm is able to detect and correctly estimate linear velocities on the axes of the 2D image plane, x and y , and the angular velocity along the perpendicular, z . The angular velocities estimated in the x and y -directions seem to be coupled with each other, while the values of the linear velocity in the z -direction are scaled down and almost completely covered by error noise. A sufficient number of valid features are tracked, and

the optical flow measurements are correct. Nevertheless more work is required to detect the error in the algorithm that causes these effects.

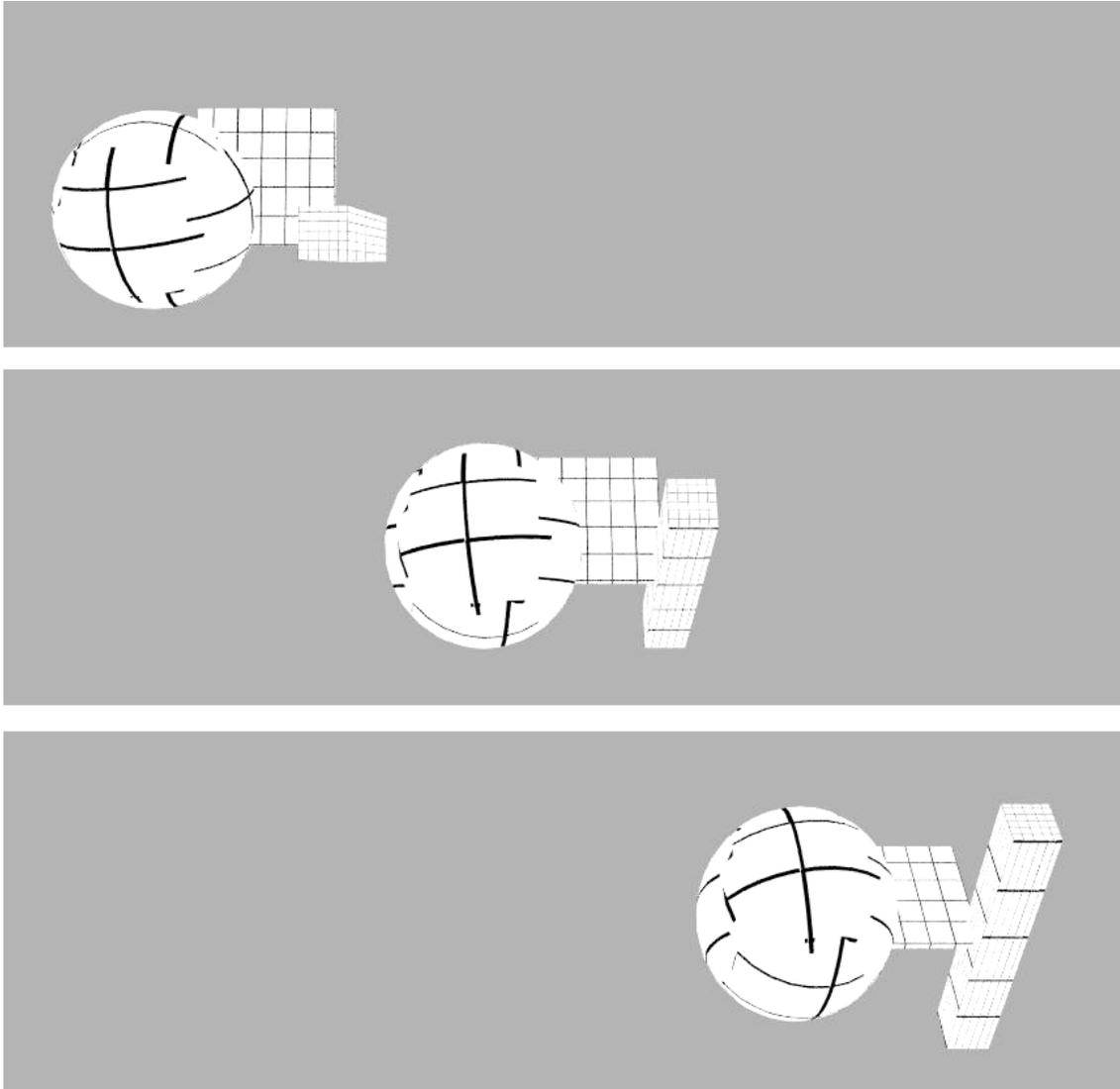


Figure 42. Three frames representing the rotation and translation of a group of computer rendered objects created to test the epipolar transformation reducing planarity and increasing parallax of the features.

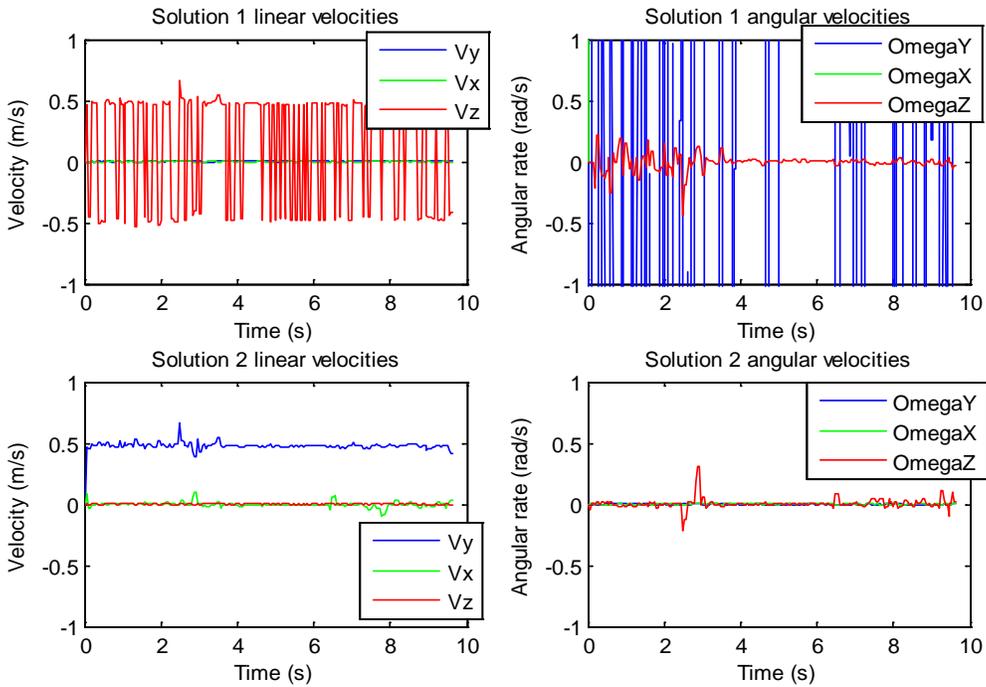


Figure 43. Time-history of linear and angular velocity for the test case 1 where the target is only translating along the y -axis.

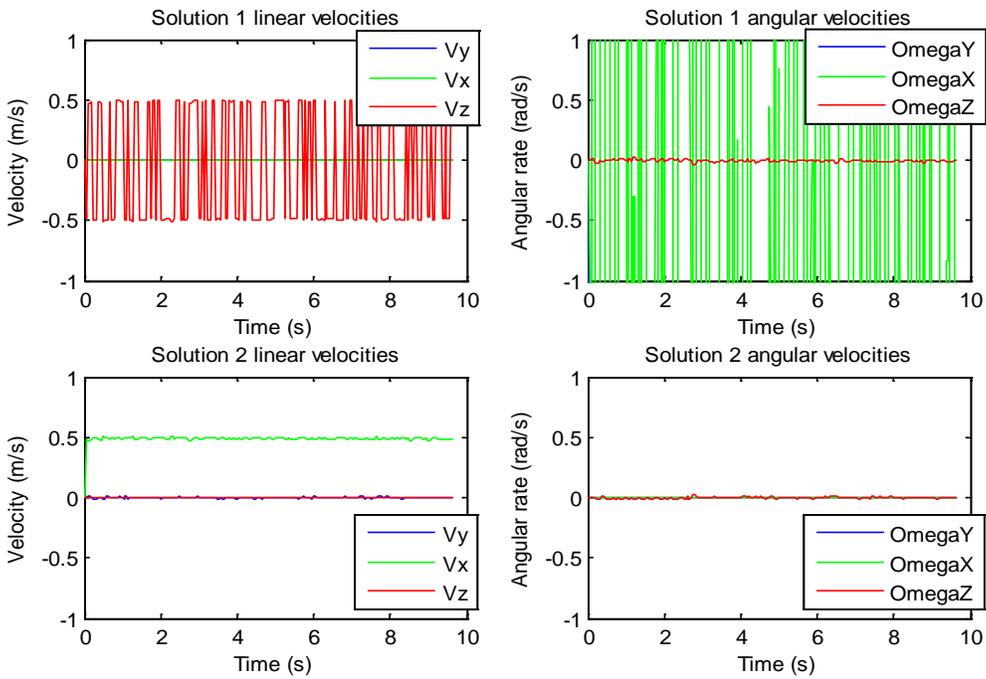


Figure 44. Time-history of linear and angular velocity for the test case 2 where the target is only translating along the x -axis.

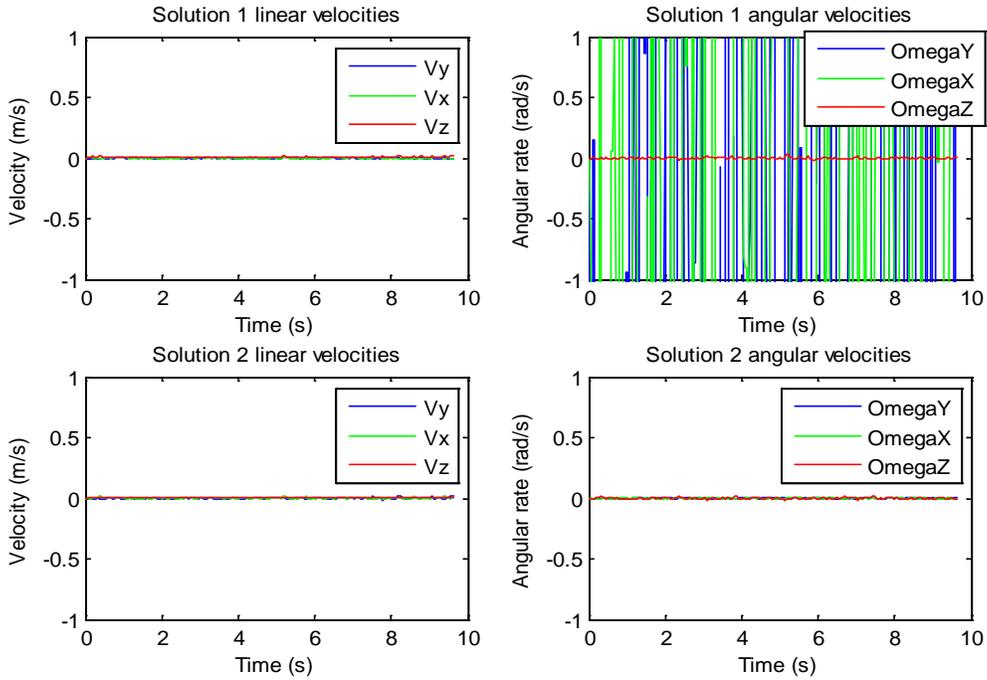


Figure 45. Time-history of linear and angular velocity for the test case 3 where the target is only translating along the z -axis.

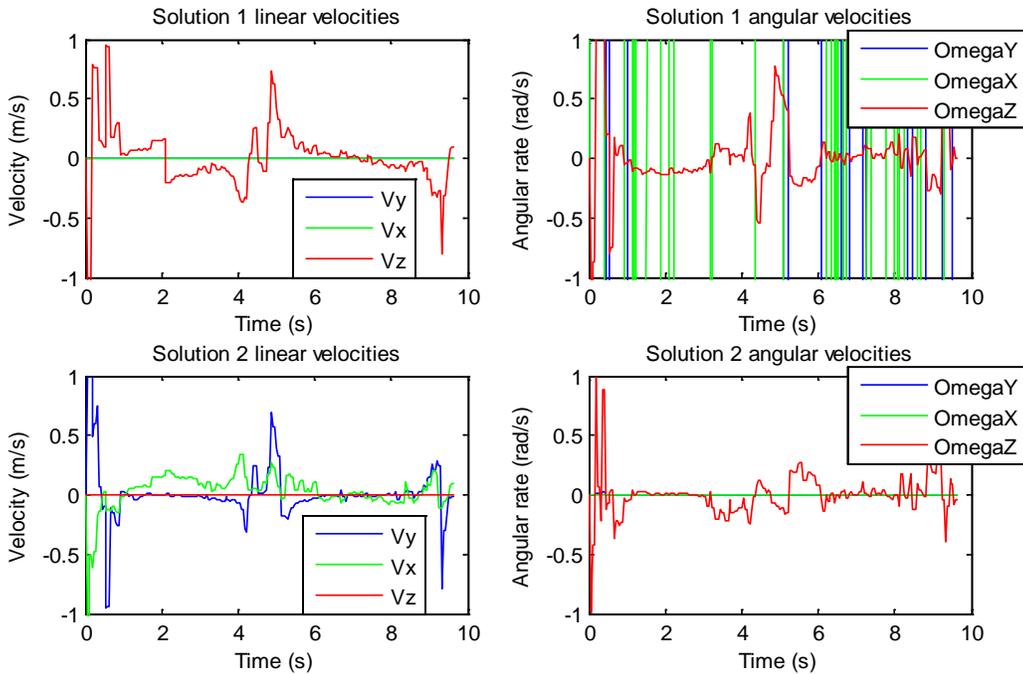


Figure 46. Time-history of linear and angular velocity for the test case 4 where the target is only rotating along the y -axis.

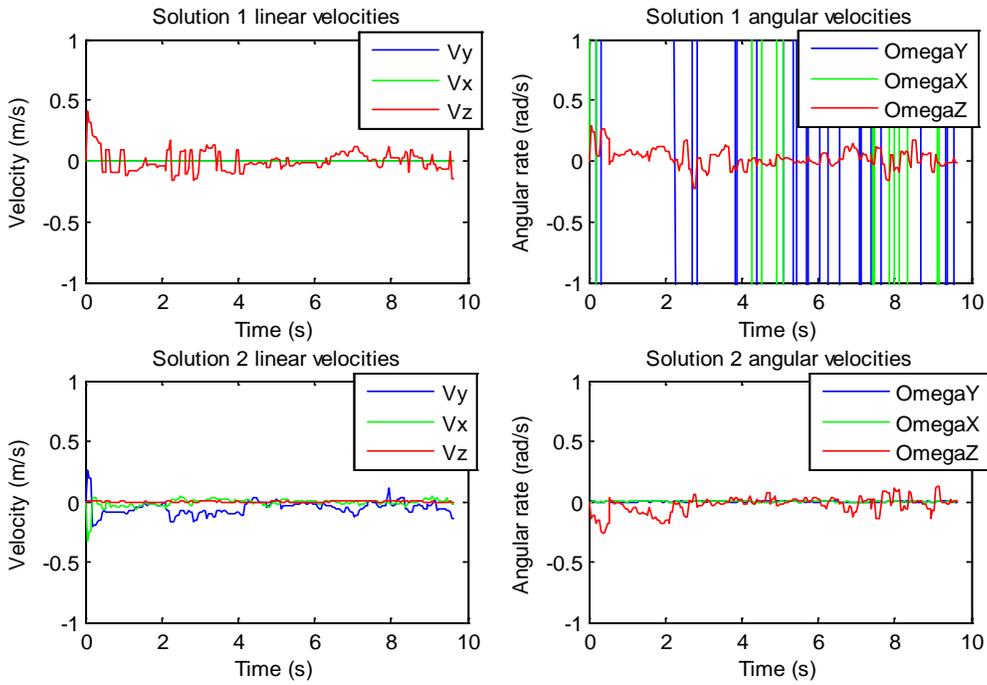


Figure 47. Time-history of linear and angular velocity for the test case 5 where the target is only rotating along the x -axis.

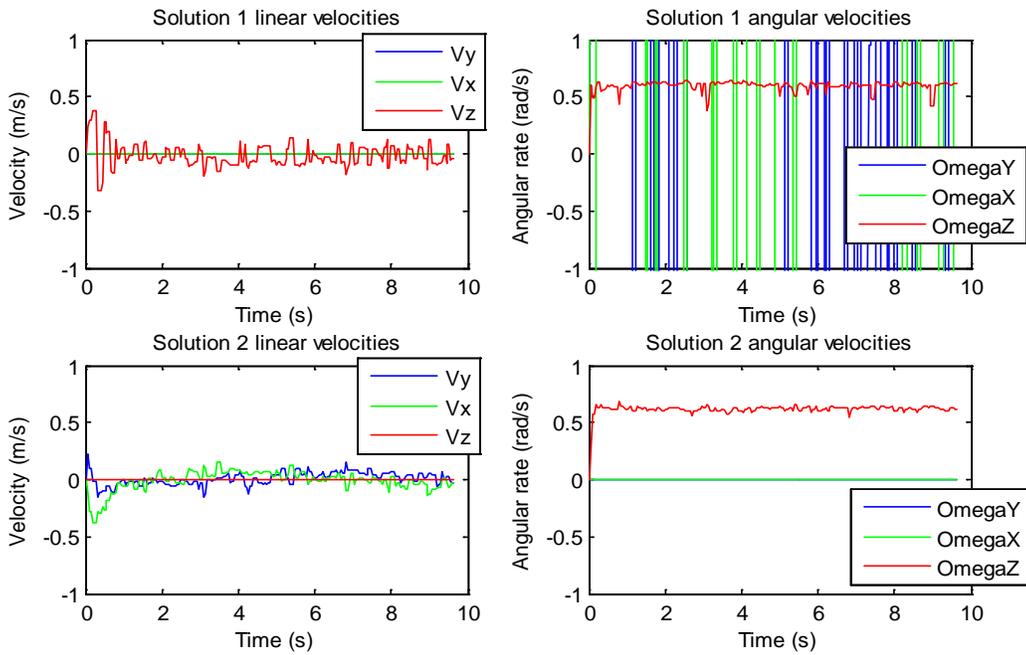


Figure 48. Time-history of linear and angular velocity for the test case 6 where the target is only rotating along the z -axis.

Results and observations for each test are summarized in Table 6.

Table 6. Description of results for the Epipolar analysis on the six cases.

TEST CASE	Observations	Outcome
1	The algorithm is able to estimate the translation along the y -axis. Solution 2 shows some noise, but the results are consistent with the simulated maneuver.	Success
2	The algorithm is able to estimate the translation along the x axis. Solution 2 shows some noise, but the results are consistent with the simulated maneuver.	Success
3	The algorithm does not detect the translation along the z -axis. This behavior may be caused by an error in the code.	Fail
4	The algorithm interprets the rotation along the y -axis as a couple translation along the x and the y -axes. This behavior may be caused by an error in the code.	Fail
5	The algorithm interprets the rotation along the x -axis as a couple translation along the x and the y -axes. This behavior may be caused by an error in the code.	Fail
6	The algorithm is able to estimate the translation along the y -axis. Solution 2 shows some noise, but the results are consistent with the simulated maneuver.	Success

c. Stereovision Algorithm Test

A first test of the stereovision algorithm was implemented using computer rendered stereo images in order to demonstrate the method before using physical targets. The video is identical to video 1, the computer-rendered satellite maneuver, with the addition of another virtual camera in the blender model necessary to have 3D stereo displacement in the images of the target. The graphs of the birds-eye view simulated trajectory, the estimated and simulated distance and the stereovision estimation error are provided in Figure 49.

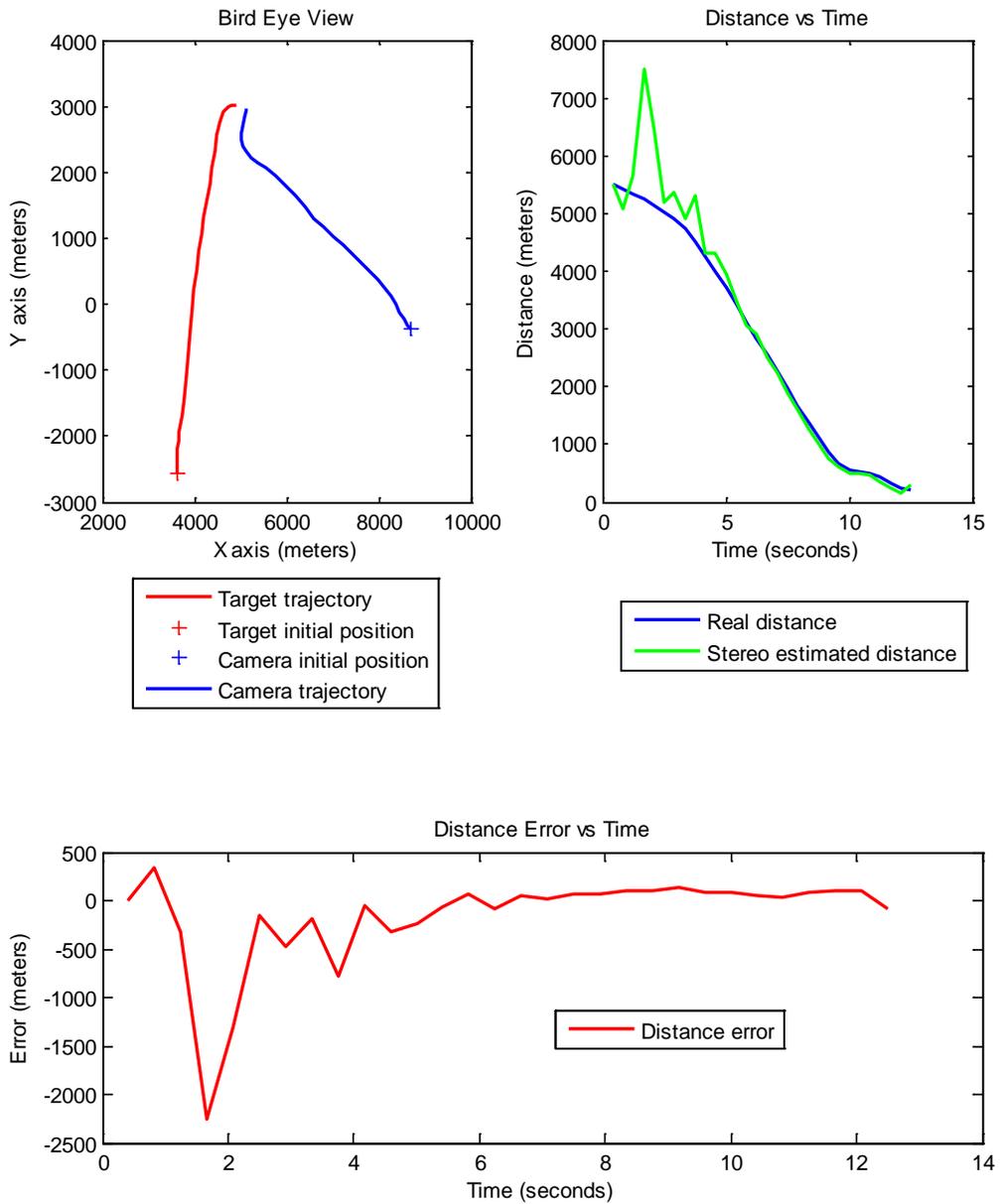


Figure 49. Birds-eye view and stereovision distance estimation results from the test on the computer rendered video.

In the graphs of Figure 49 it is possible to see the correlation between quality in estimation and distance. As expected, estimation errors are larger when the target is still too far away to provide good features to track.

2. NASA Videos

The NASA videos were used to calibrate and validate the detection and the tracking settings of the algorithm. The properties of the videos provided by NASA and the calibration values used in the algorithm are provided in Table 7, Table 8, Table 9, Table 10, Table 11 and Table 12. The information tables are followed by detailed descriptions of the detection and tracking performance.

In all videos the targets were successfully detected and tracked, but different initialization settings were used. One important difference is the frame rate. In all videos it was possible to reduce the frame rate without losing detection or tracking performance, but some videos required higher frame rates than others because of the fast repositioning of the camera and not due to the orbital maneuver velocity.

Another important difference is how the background subtraction is initialized. No subtraction was necessary in videos with only Earth or open space as background. In videos with static features or obstructed areas, the use of the static background subtraction was successful. In video 3 and 4, manually selected ROIs were used. This method reduces the dimensions of the image only for the first detection, excluding regions that are known to be obstructed. Further work can be dedicated to improve this technique creating automatic known-feature recognition extraction and matching in order to detect and recognize which features and regions to exclude.

The results of these experiments are summarized in Table 13, while some significant frames are shown in the images provided from Figure 50 to Figure 64.

Table 7. Video 1 properties and calibration values.

Video Name	Video 1 (757771)
Description	ISS Expedition 24, Soyuz TMA-19 relocation from the Zvezda Service Module (SM) and docking to the Rassvet MRM-1 Module
GMT Day	179, 2010
Frame Rate	30 fps (reduced to 0.3 fps)
Resolution	720×480 pixels
Number of frames	16000 (reduced to 160 frames)
Compression	mp4
Background Subtraction	Not Active
Detector	Harris
Harrys stronger features	100
Harrys corner quality	0.01
Update Period	every 16.6 seconds (equivalent to 500 frames on the original video and 5 frames on the reduced video)
ANMS	Not Active
Blob length	100
Blob sigma	6
Blob-ROI base dimension	20
Tracker	KLT
KLT discard distance	20
KLT-ROI base dimension	50

Table 8. Video 2 properties and calibration values.

Video Name	Video 2 (767109)
Description	STS-135 R-bar Pitch Maneuver (RPM) and rendezvous OPS
GMT Day	191, 2011
Frame Rate	30 fps (reduced to 0.3 fps)
Resolution	720×480 pixels
Number of frames	18000 (reduced to 180 frames)
Compression	mp4
Background Subtraction	Not Active
Detector	Harris
Harrys stronger features	300
Harrys corner quality	0.005
Update Period	every 16.6 seconds (equivalent to 500 frames on the original video and 5 frames on the reduced video)
ANMS	Active
ANMS radius	10
Blob length	100
Blob sigma	6
Blob-ROI base dimension	30
Tracker	KLT
KLT discard distance	40
KLT-ROI base dimension	60

Table 9. Video 3 properties and calibration values.

Video Name	Video 3 (884887)
Description	ISS Expedition 29, Progress 45P docks to the ISS
GMT Day	306, 2011
Frame Rate	30 fps (reduced to 0.3 fps)
Resolution	720×480 pixels
Number of frames	19500 (reduced to 195 frames)
Compression	mp4
Background Subtraction	Initial ROI manually selected
Detector	Harris
Harris stronger features	100
Harris corner quality	0.01
Update Period	every 16.6 seconds (equivalent to 500 frames on the original video and 5 frames on the reduced video)
ANMS	Not Active
ANMS radius	n/n
Blob length	100
Blob sigma	6
Blob-ROI base dimension	20
Tracker	KLT
KLT discard distance	20
KLT-ROI base dimension	50

Table 10. Video 4 properties and calibration values.

Video Name	Video 4 (776046)
Description	ISS Expedition 34, Progress 50P tracking, rendezvous, and docking to the ISS
GMT Day	042, 2013
Frame Rate	30 fps (reduced to 1.2 fps)
Resolution	720×480 pixels
Number of frames	36861 (reduced to 1474 frames)
Compression	mp4
Background Subtraction	Initial ROI manually selected
Detector	Harris
Harris stronger features	100
Harris corner quality	0.01
Update Period	every 4.16 seconds (equivalent to 125 frames on the original video and 5 frames on the reduced video)
ANMS	Not Active
ANMS radius	n\n
Blob length	100
Blob sigma	6
Blob-ROI base dimension	20
Tracker	KLT
KLT discard distance	20
KLT-ROI base dimension	50

Table 11. Video 5 properties and calibration values.

Video Name	Video 5 (757769)
Description	ISS Expedition 24, Soyuz TMA-19 relocation from the Zvezda Service Module (SM) and docking to the Rassvet MRM-1 Module
GMT Day	179, 2010
Frame Rate	30 fps (reduced to 3 fps)
Resolution	720×480 pixels
Number of frames	28000 (reduced to 2800 frames)
Compression	mp4
Background Subtraction	Static Background Subtraction
Detector	Harris
Harrys stronger features	100
Harrys corner quality	0.01
Update Period	every 4.16 seconds (equivalent to 50 frames on the original video and 5 frames on the reduced video)
ANMS	Not Active
ANMS radius	n\n
Blob length	100
Blob sigma	6
Blob-ROI base dimension	15
Tracker	KLT
KLT discard distance	15
KLT-ROI base dimension	30

Table 12. Video 6 properties and calibration values.

Video Name	Video 6 (765734)
Description	View from the CBCS CAM as STS-134 rendezvous and docks with the ISS
GMT Day	179, 2010
Frame Rate	30 fps (reduced to 1.2 fps)
Resolution	720×480 pixels
Number of frames	36861 (reduced to 1474 frames)
Compression	mp4
Background Subtraction	Static Background Subtraction
Detector	Harris
Harris stronger features	100
Harris corner quality	0.03
Update Period	every 4.16 seconds (equivalent to 125 frames on the original video and 5 frames on the reduced video)
ANMS	Not Active
ANMS radius	n\n
Blob length	100
Blob sigma	6
Blob-ROI base dimension	10
Tracker	KLT
KLT discard distance	10
KLT-ROI base dimension	20

Table 13. Description of the performances of the algorithm applied to the NASA videos

	Results description and observations
Video 1	The Algorithm is able to track reliably the Soyuz also at very low frame rates. The target is not well illuminated and most of the features tracked are the external edges of the Soyuz, which could be a challenge for a pose estimator.
Video 2	The Maneuver of the Shuttle is almost constantly tracked by the algorithm. The algorithm loses most of the feature points when the Shuttle shows the bottom part almost feature-less.
Video 3	The Progress is constantly tracked by the algorithm, and the pose estimation measures the translation docking maneuver along the z-axis. An initial ROI had to be manually selected to start the detection excluding known ISS features.
Video 4	The Progress is tracked in different illuminations and background. The algorithm is able to recover the detection when the target is completely out of sight or when to camera is repositioned with fast movement. A higher frame rate was necessary because of the fast repositioning and zooming of the camera. Close to docking two identical spacecraft are visible and the ROI expands to include both.
Video 5	The detection automatically recognizes the target from the ISS features through static background subtraction. The Soyuz is reliably tracked over different backgrounds, in low luminosity conditions and with a fast moving camera. As before higher frame rate was necessary because of the fast repositioning and zooming of the camera. Atmospheric features affect negatively the tracking for a short amount of time and required an increase in Harris corner quality..
Video 6	The main challenge of this experiment is that for the entire time the camera is obstructed by the docking interface. Changes in illumination, aperture and focus cause the loss of the target for few frames, but the algorithm is able to recover and reliably track the target docking interface for almost the entire maneuver.

VIDEO 1 (ISS Expedition 24, Soyuz TMA 19)

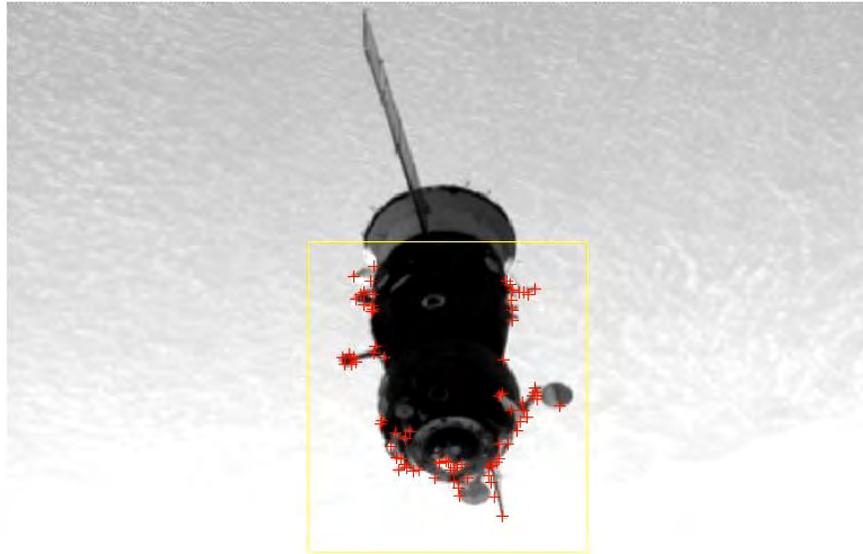


Figure 50. Frame taken from video 1 while the algorithm is tracking the features marked in red.

VIDEO 1 (ISS Expedition 24, Soyuz TMA 19)

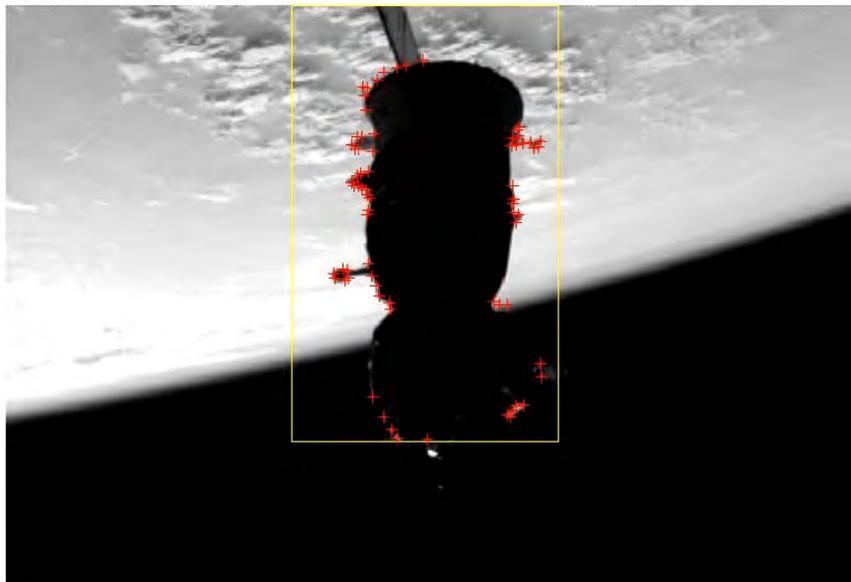


Figure 51. Another frame from video 1. Because of the change in illumination the algorithm tracks only edge features.

VIDEO 1 (ISS Expedition 24, Soyuz TMA 19)

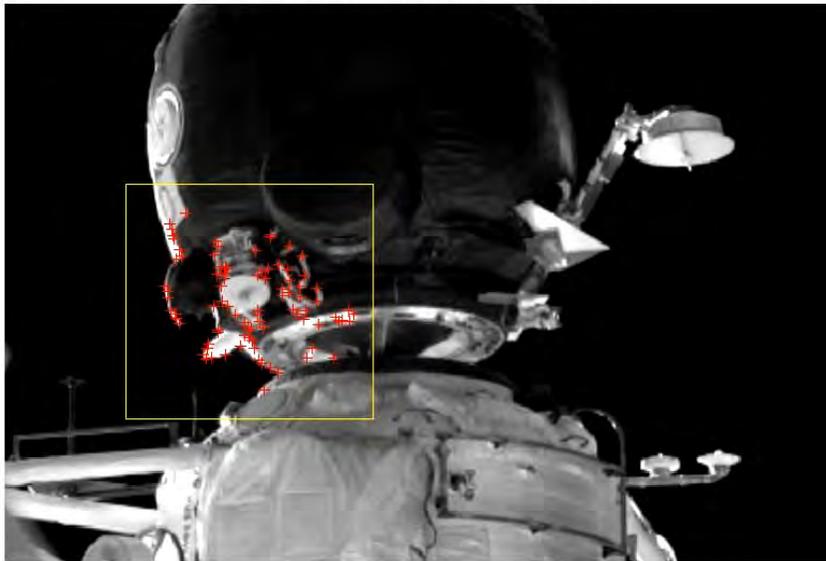


Figure 52. Also during docking the algorithm tracks only features of the Soyuz on the last frames from video 1.

VIDEO 2 (STS-135 RPM and rendezvous OPS)



Figure 53. Tracking of the features of the Shuttle with the Earth as background on a frame from video 2.

VIDEO 2 (STS-135 RPM and rendezvous OPS)

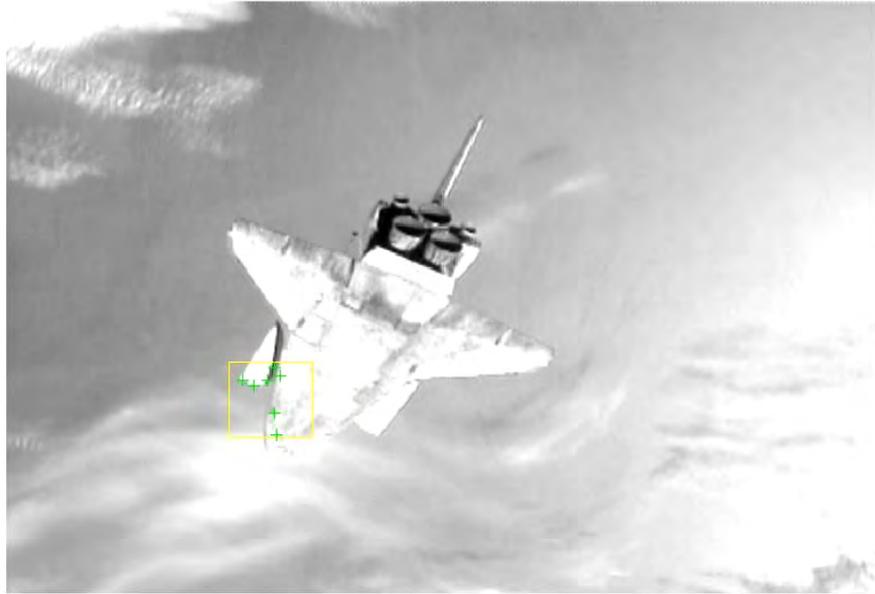


Figure 54. The algorithm tracks only a few features when the camera faces towards the thermal shields of the Space Shuttle.

VIDEO 3 (ISS Expedition 29, Progress 45P docking)

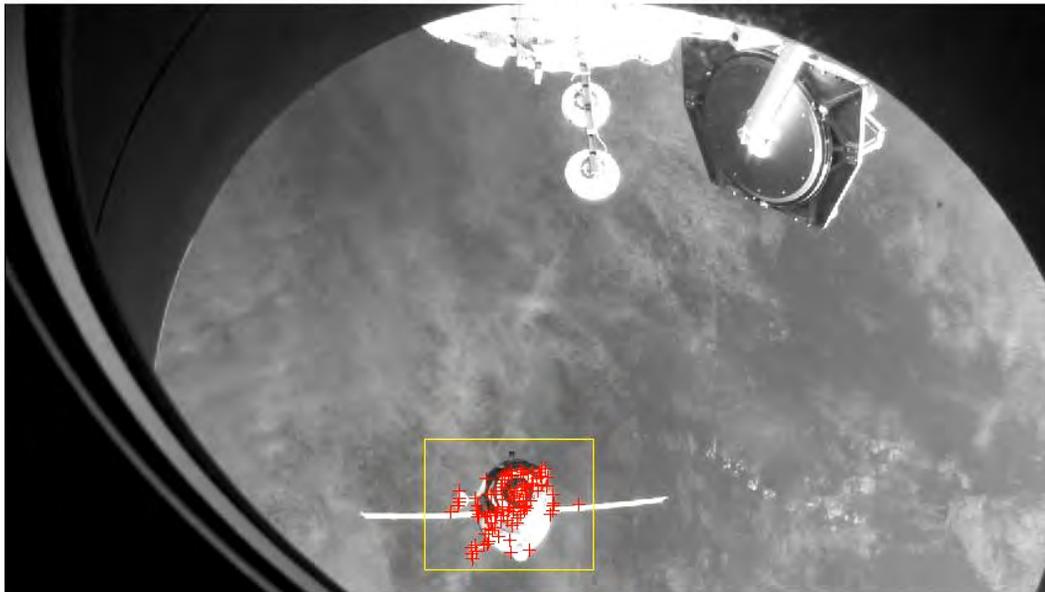


Figure 55. View of cluttered features on the progress and obstructions on the edges of the image from video 3.

VIDEO 4 (ISS Expedition 34, Progress 50P docking)

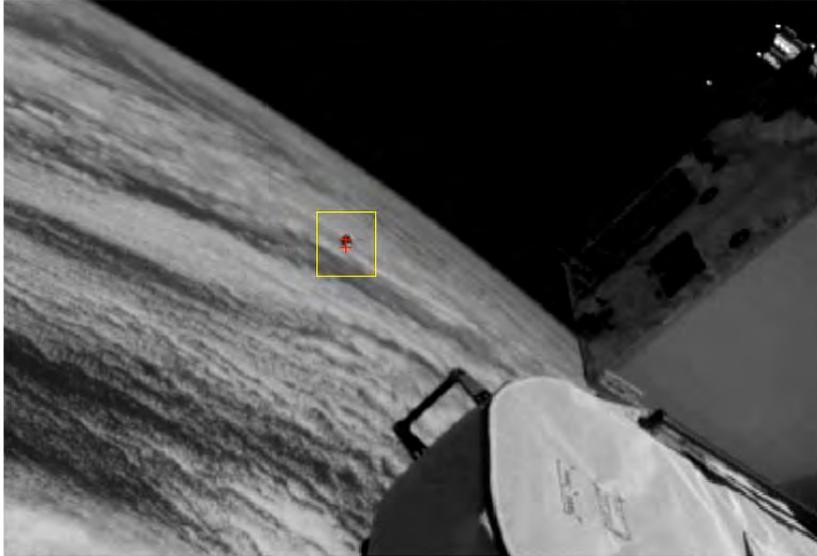


Figure 56. The algorithm is able to automatically detect the Progress also in this challenging frame where the Earth features spin almost at the same speed as the target and have the same luminosity intensity.

VIDEO 4 (ISS Expedition 34, Progress 50P docking)

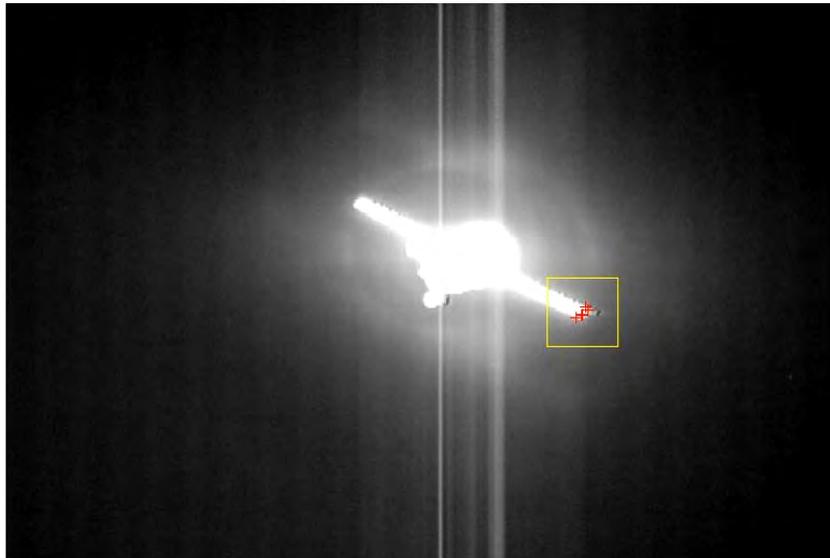


Figure 57. Changes in illumination causes the algorithm to lose most of the features tracked, but it automatically recovers.

VIDEO 4 (ISS Expedition 34, Progress 50P docking)

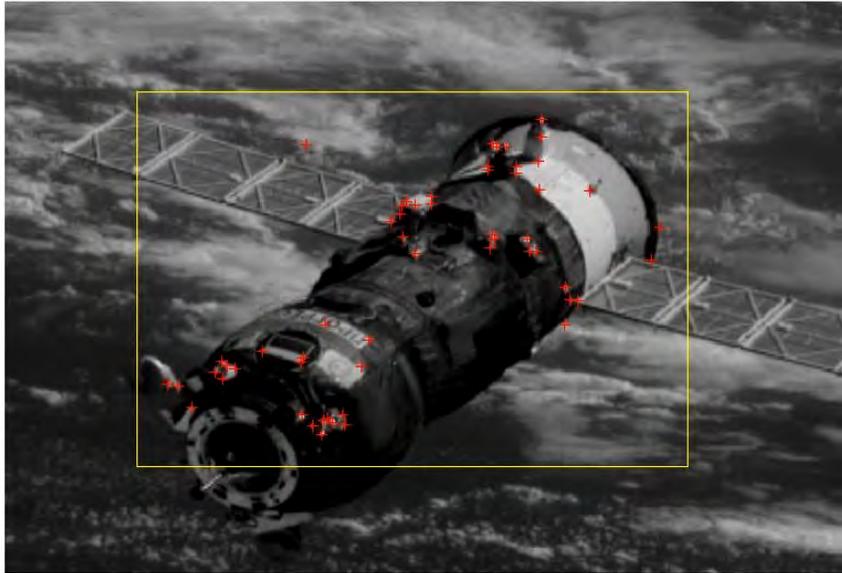


Figure 58. The algorithm fully recovers from illumination changes and provides strong features and a correct ROI of the target.

VIDEO 4 (ISS Expedition 34, Progress 50P docking)

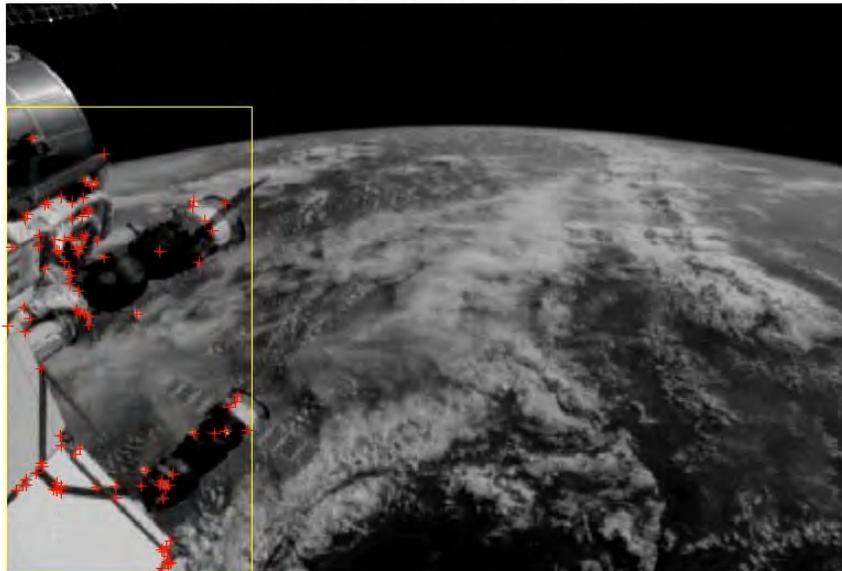


Figure 59. When two spacecraft with identical features are close, the algorithm expands the ROI to include and track both. This causes also other unwanted features to be captured.

VIDEO 5 (ISS Expedition 24, Soyuz TMA 19 relocation)



Figure 60. Detection of a moving target over the static features of the ISS on the initial frames of video 5.

VIDEO 5 (ISS Expedition 24, Soyuz TMA 19 relocation)

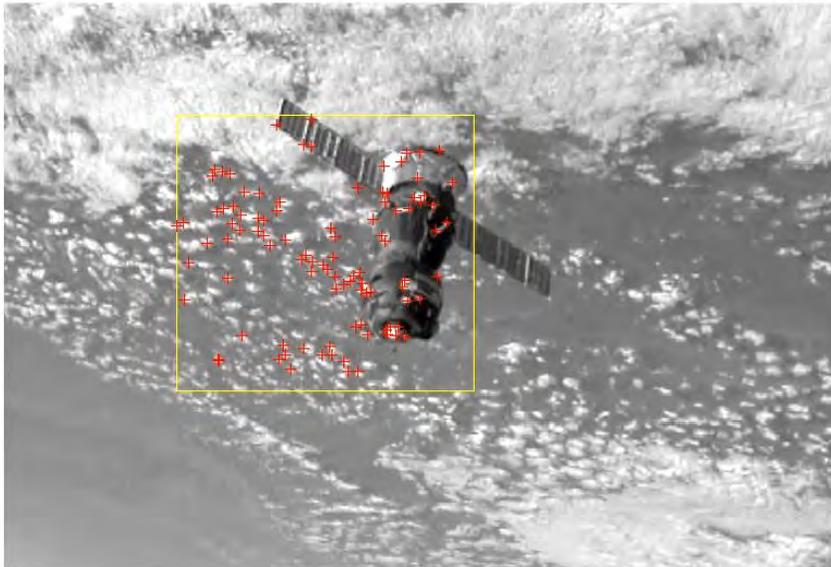


Figure 61. The ROI expands over clouds with high defined edges, confused for target features and tracked.

VIDEO 6 (STS-134 CBCS CAM view during docking with ISS)

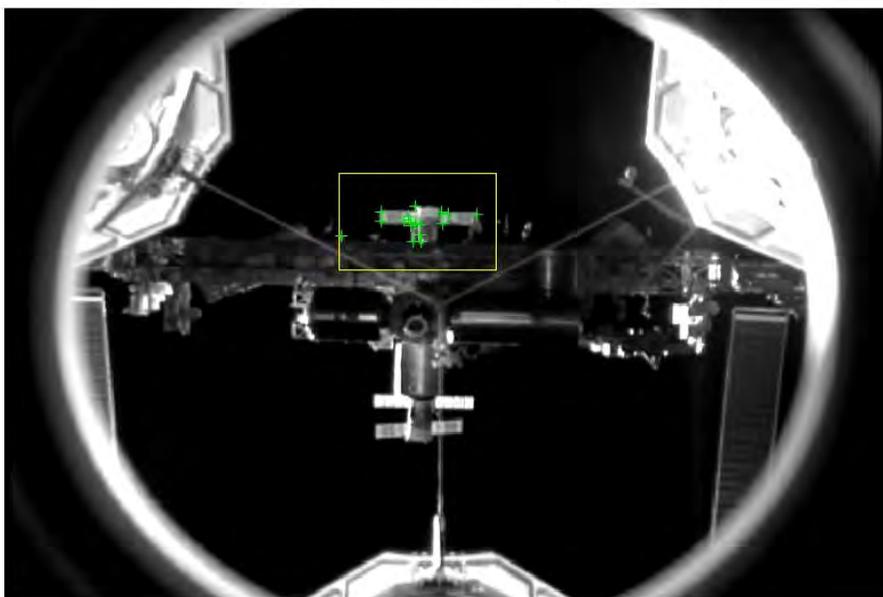


Figure 62. The static background removal method is able to mask the obstructed areas and non-target features.

VIDEO 6 (STS-134 CBCS CAM view during docking with ISS)

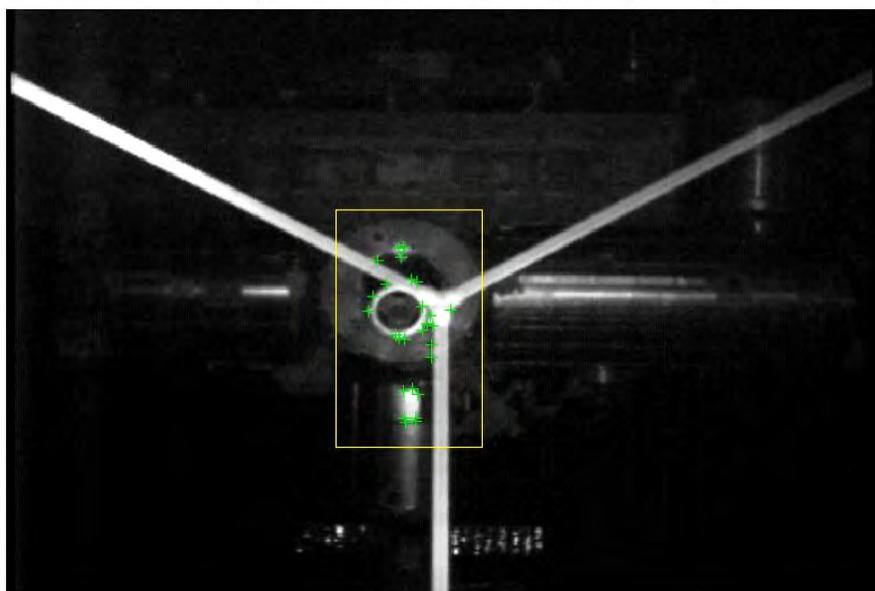


Figure 63. The algorithm is able to track the features of the target behind the docking interface.

VIDEO 6 (STS-134 CBCS CAM view during docking with ISS)

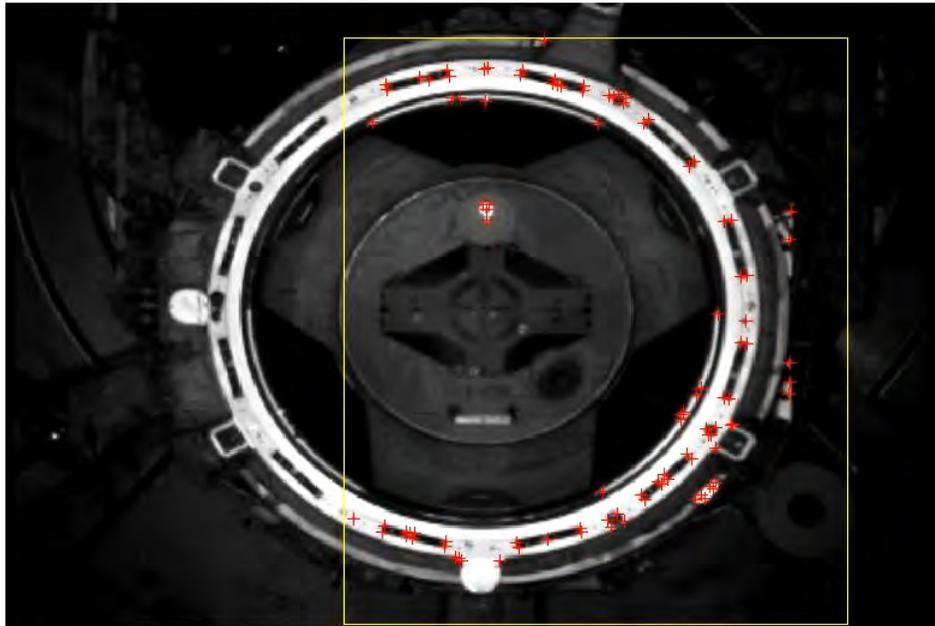


Figure 64. The target docking interface of video 6 tracked by the algorithm.

3. Live Target

One of the main goals of this work was to implement the hardware-in-the-loop testing capabilities on the FSS test-bed. The artificial vision navigation algorithm can be used on the existing guidance models to substitute the attitude and position sensors. The VICON system is then used only as ground truth, and the simulation is more challenging and realistic.

The first phase of the hardware implementation was to test the artificial vision algorithm on a laboratory desktop computer connected with the stereovision camera Bumblebee. This test is required for:

- The validation of the algorithm using real-time images
- The calibration of the stereovision function on real features
- The validation of the detection and tracking functions for different illumination conditions.

The camera was positioned on the side of the FSS flat table at the height of the floating units. An image of the setup is provided in Figure 65, where it is possible to see the stereovision camera in foreground and the FSS floating unit in the center on the

granite floor. In the background, the LED sun simulator and the VICON monitors are also visible.



Figure 65. View of the live desktop + live-target experiment setup and the main components of the test-bed at the Spacecraft Robotics Laboratory of NPS.

Some MATLAB commands had to be added to the original artificial vision algorithm in order to be able to collect the position information from the VICON via UDP and to grab images from the stereovision camera. These updates of the algorithm are included and commented on in the software version provided in Section A of the Appendix.

The maneuver tested is a planar rendezvous translation towards the camera with different spinning velocities in four experiments. The speed is kept almost constant for the entire maneuver. The bird's-eye views of the four maneuvers tested are shown in Figure 66. The data of these plots were streamed from the VICON server during the experiment and used as ground truth.

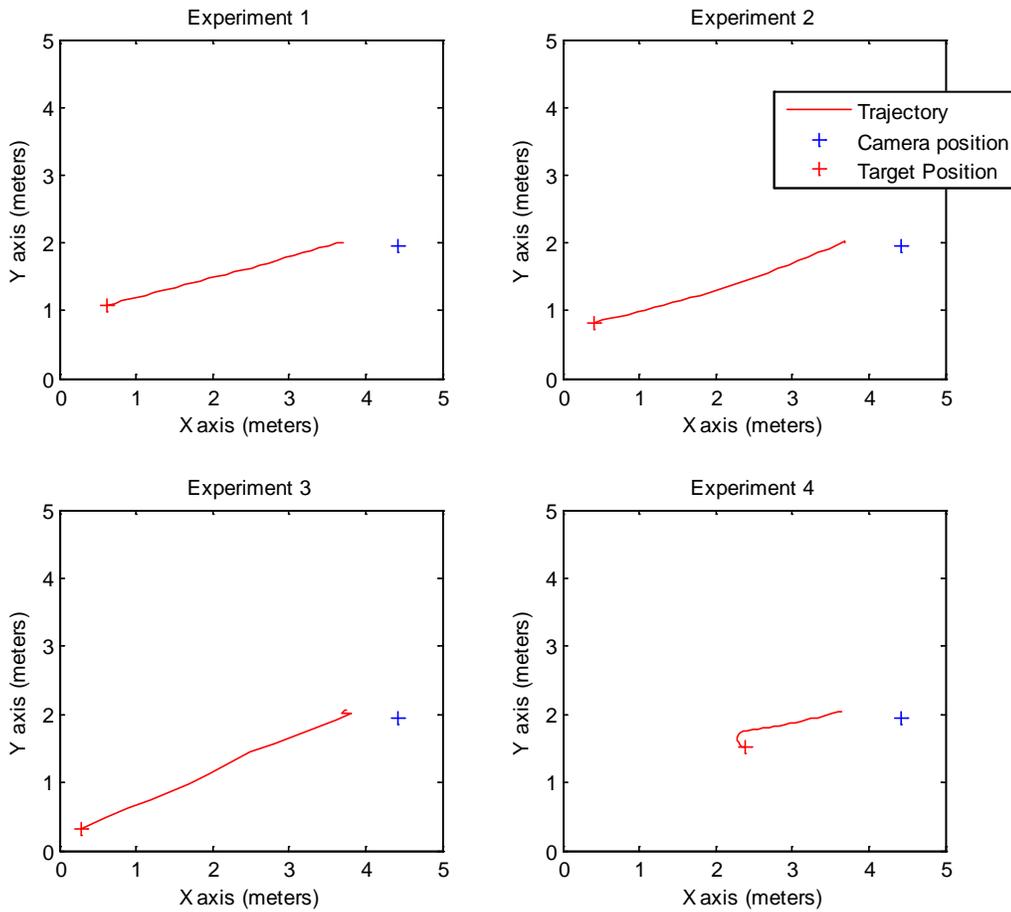


Figure 66. Bird's-eye view of the trajectories of the FSS unit in four experiments.

The test-bed was installed in a low-reflective, black walled laboratory in order to minimize the number of detectable background features. Some examples of frames acquired during tracking are provided in Figure 67. It is possible to see that the only background features that have luminosity intensity comparable to the intensity of the target features are the VICON cameras and the VICON Computer monitor. These background features were manually masked by the algorithm during the detection phase, excluding them from the initial ROI.

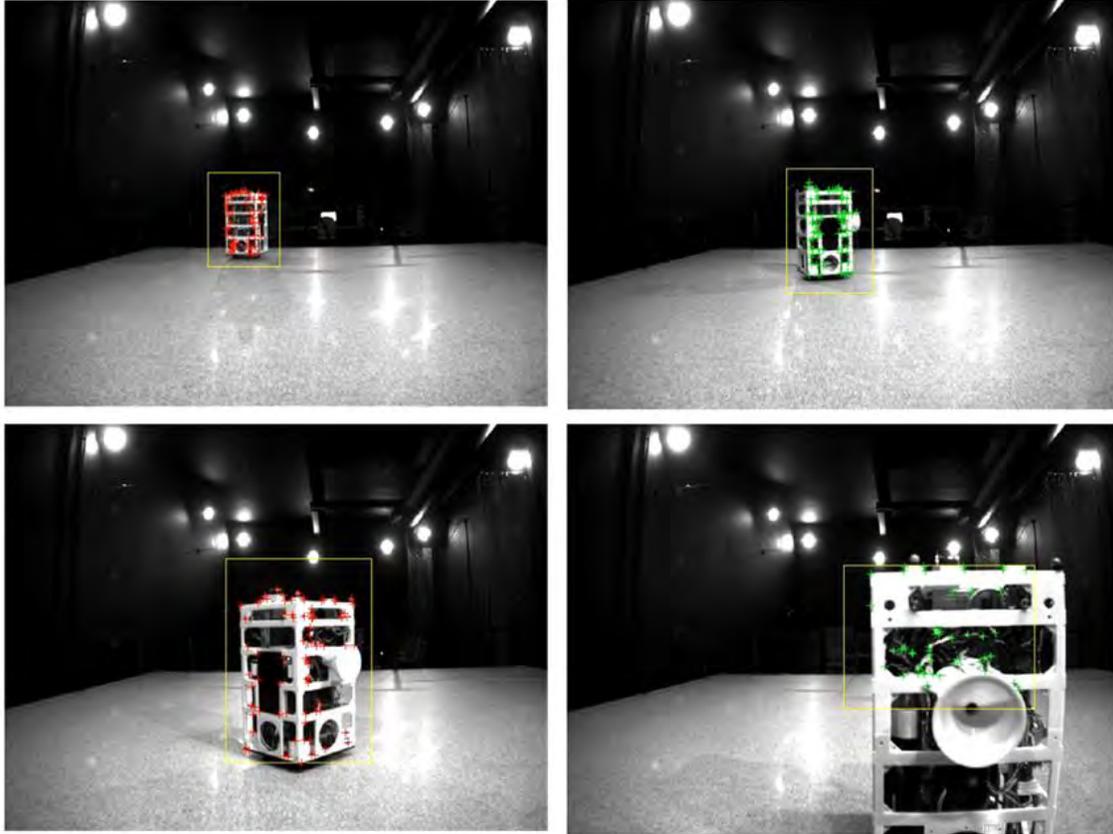


Figure 67. Sequence of frames acquired during one of the experiments on the FSS test-bed. The tracked features are marked in red and the detected features in green.

All the calibration setting used on the FSS experiment are provided in Table 14. The calibration values are almost identical to the parameters used for most of the real and virtual videos. The time history of the distances measured with VICON and estimated with the stereovision and the distance error are provided for the four experiments in Figure 68, Figure 70, Figure 72 and Figure 74.

In the results of experiment 1 and 2 it is possible to see that the estimation error increases drastically at the end of the rendezvous, when the cameras are too close to the target.

In experiment 3 the target reaches the camera more quickly and when the distance is close to zero, the estimation error increases. In Figure 72 the missing values indicate infinite or negative range values due to lack of reliable features to match, the spacecraft, being too close to the camera.

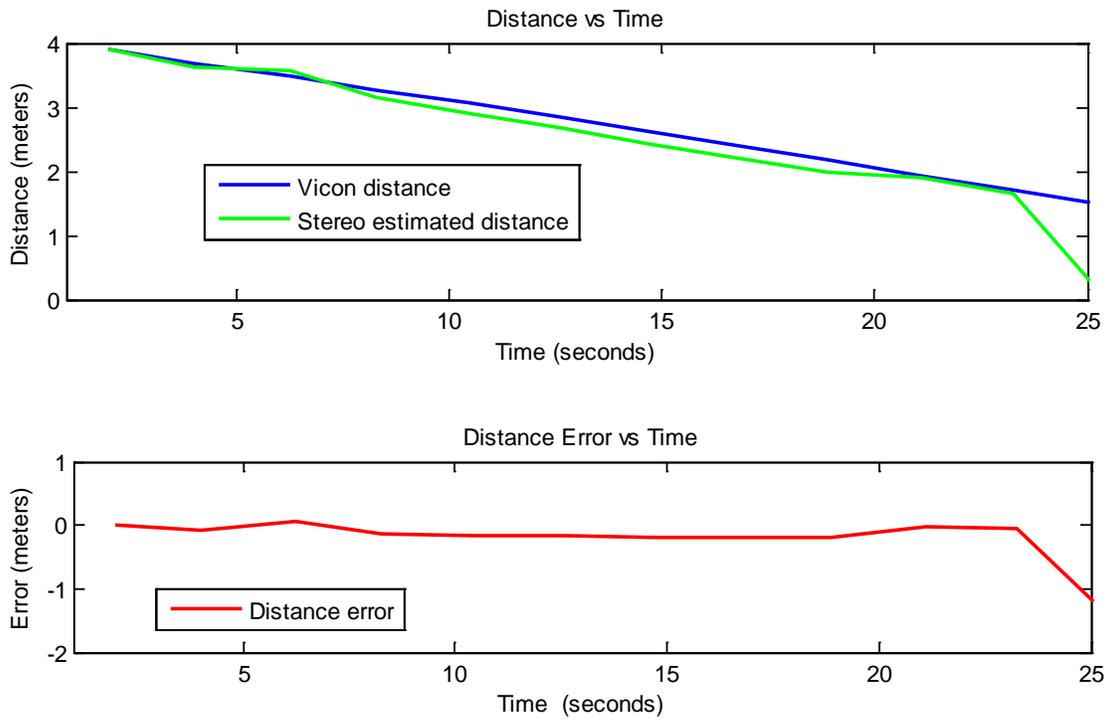


Figure 68. Measured and estimated time-history comparison of the distance between camera and target in experiment 1.

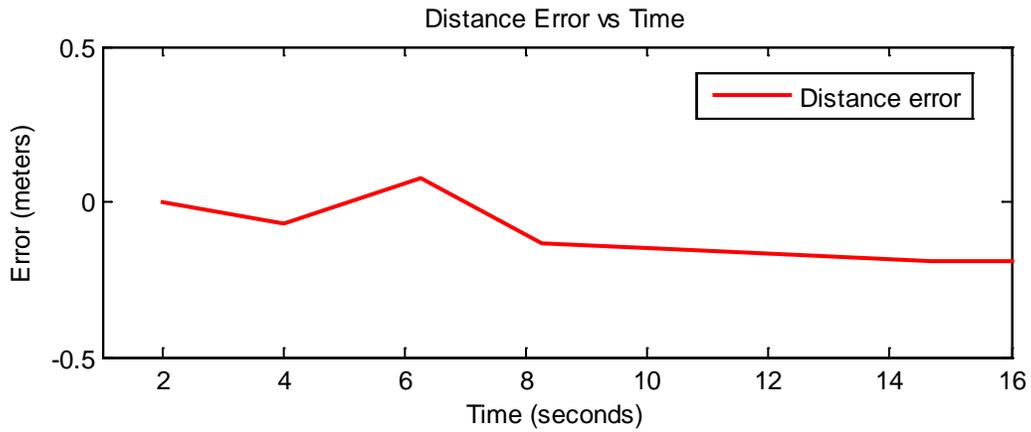


Figure 69. Zoomed view of the distance-error time history in the first 16 seconds of experiment 1.

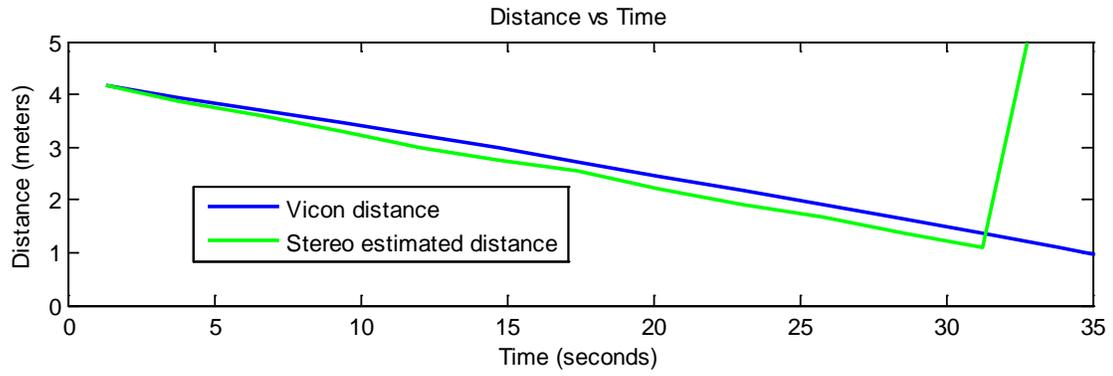


Figure 70. Measured and estimated time-history comparison of the distance between camera and target in experiment 2.

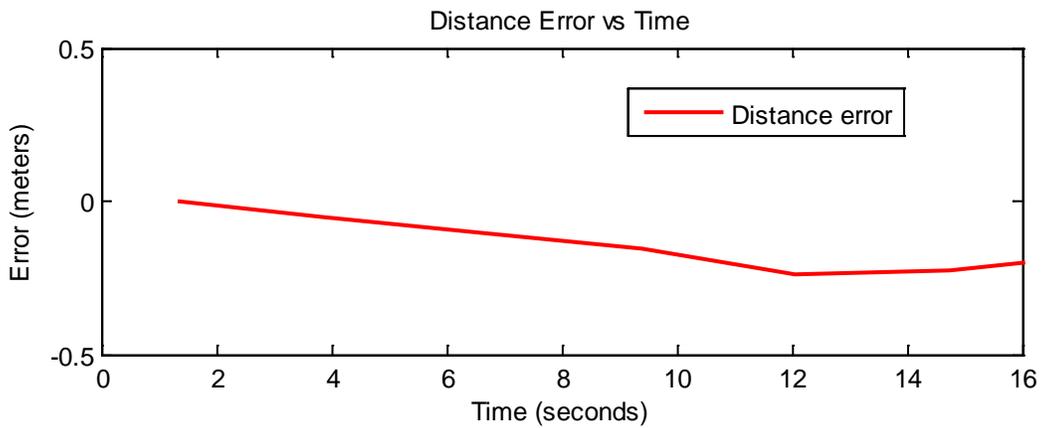


Figure 71. Zoomed view of the distance-error time history in the first 16 seconds of experiment 2.

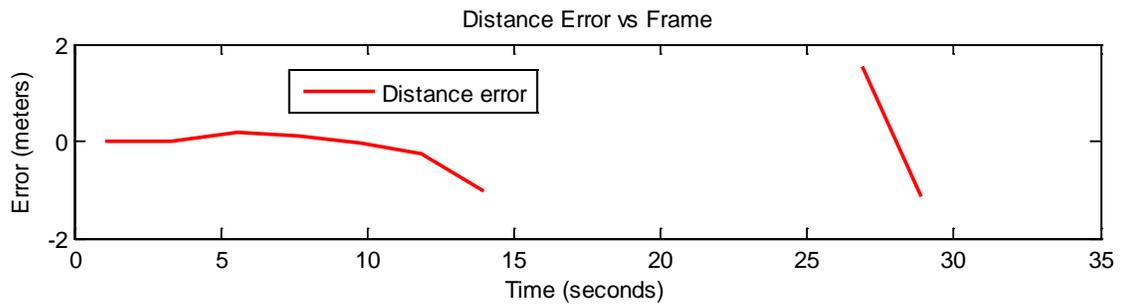
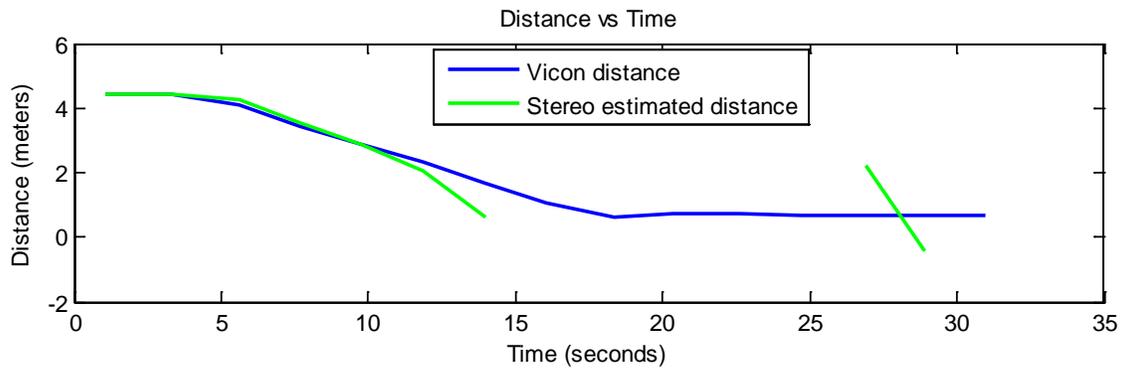


Figure 72. Measured and estimated time history comparison of the distance between camera and target in experiment 3.

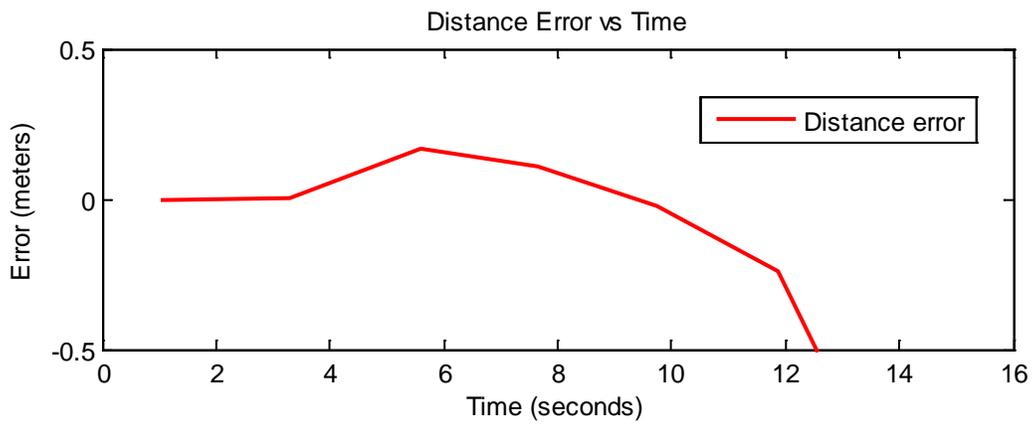


Figure 73. Zoomed view of the distance-error time history in the first 16 seconds of experiment 3.

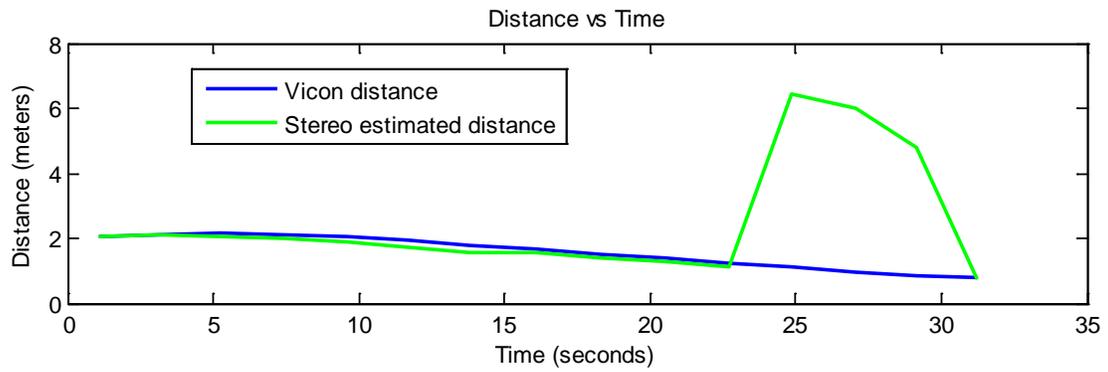


Figure 74. Measured and estimated time history comparison of the distance between camera and target in experiment 4.

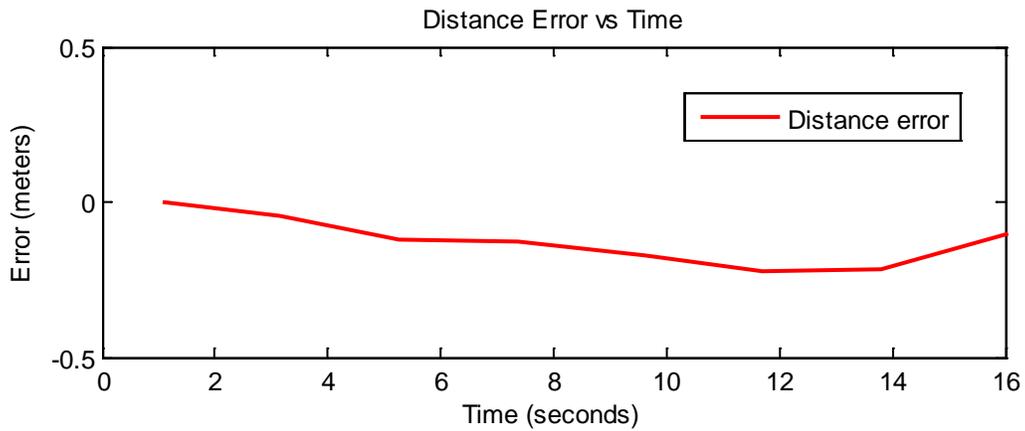


Figure 75. Zoomed view of the distance-error time history in the first 16 seconds of experiment 4.

Table 14. Detection and tracking calibration values.

Video Name	Live Stream from the Bumblebee Camera
Description	The camera acquires live real-time images of the FSS unit floating on the granite floor. Same settings have been used for all experiments.
GMT Day	02/25/2015
Frame Rate	5 fps
Resolution	970×720 pixels
Number of frames	60
Compression	mp4
Background Subtraction	Initial ROI manually selected
Detector	Harris
Harris stronger features	100
Harris corner quality	0.05
Update Period	every 1 second
ANMS	Not Active
ANMS radius	n\n
Blob length	100
Blob sigma	6
Blob-ROI base dimension	10
Tracker	KLT
KLT discard distance	20
KLT-ROI base dimension	50
Stereovision Update Period	every 2 seconds

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS

The tests and the experiments in this thesis were designed to provide the most reliable estimation of the performance of the artificial vision algorithm in a generic on-orbit application in terms of detection, tracking and pose estimation reliability, speed and computational load.

With the videos provided by NASA, the algorithm was shown able to autonomously detect and track real spacecraft features in challenging scenarios with changes in illumination and background. Furthermore, these tests have shown that similar initial parameters can work for a wide variety of on-orbit scenarios, and that acquisition rates of 3.0 fps are sufficient for the algorithm to track the target. Another important observation is that the video is sensitive to the method used for the background subtraction. The implementation of one method versus another drastically improves the performance of the initial detection.

The hardware-in-the-loop experiments for the validation of the artificial vision algorithm demonstrated the capability of a real-time stereovision system to reliably detect and estimate the distance of an unknown target with spacecraft-like features and dynamics in a space-like illumination condition. The target was detected and tracked while hovering over the FSS flat floor in a proximity maneuver. The range estimated in real-time using the stereovision system was compared with the ground with an average error of about 2.5 cm. This average error value was measured from the raw estimation within the stereovision range without using Kalman filters or other methods that could improve the range estimation.

An unresolved bug in the algorithm did not allow testing of the epipolar function on the hardware-in-the-loop experiments. The function provided only valid linear velocities values along the x and y -axis, and angular velocities along the z -axis. Future work is required to detect the error in the algorithm and proceed with the epipolar pose estimation tests and validation.

That passive vision sensing might be an answer for a relatively low cost and reliable relative navigation system for space applications was shown. The ability to autonomously adapt to a wide range of space scenarios, provide consistent information on the target and be implementable in a real-time system was also shown.

The FSS test-bed developed has shown to partially simulate the space proximity maneuvering in terms of dynamics, features and illumination conditions and, therefore, is a valid tool for future hardware-in-the-loop experiments.

A. FUTURE WORK

The algorithm presented an error in the epipolar function. Detection and correction of this error can lead to several experiments on the FSS testbed and on the NASA videos to validate the pose-estimation capability of the code. Also, the use of the Geometric Transformation function provided by MATLAB can be implemented and compared to the epipolar transformation.

The selection of the initial ROI for non-static backgrounds can be automatized and further work is required to improve the optical flow based background subtraction.

In order to obtain further information on the computational load performance of the algorithm on a real-time OS, it would be interesting to compile the algorithm in a RTAI executable and implement it on-board the FSS units with limited processing capabilities. The algorithm can also be combined in a Simulink block with the pre-existent validated guidance models and easily implemented in future FSS experiments as the main sensor.

The implementation of a Kalman filter is also strongly suggested since the results have shown the presence of non-negligible noise in both the stereovision estimation of the range and the epipolar estimation of the velocities.

APPENDIX

A. ARTIFICIAL VISION ALGORITHM

All the MATLAB scripts of the AViATOR algorithm divided by modules are provided in the appendix.

1. Initializer (initializer.m)

```
%% Authors
% Alessio Grompone and Roberto Cristi
% PI: Marcello Romano
% Spacecraft Robotics Laboratory, Naval Postgraduate School 2015

%AViATOR variables and options initializer

close all
clear

global videoname Video VideoR SizeIMG BSmode BSrad Hstrongest ...
    Hquality ANMSSwitch ANMSdistance Blength Bsigma Bnumber Bmode ...
    BroiDim Xroi ROI Ifirst KLTroi KLTvalue KLTroiDim fl pix Dstereo
...
    vpold omegapold Livecam vid

%GENERAL OPTIONS

%these options can be modified to select the input, all the main ...
%options and calibrate the algorithm to achieve better results

CreateVideo=0 %Enables (1) or Desables (0) Video Recording
CreateImage=0 %Enables (1) or Desables (0) Video Recording
Livecam=2 %Selects Video (0) , Webcam (1) or Bumblebee Camera (2)
LIVEframes=300;% number of frames of LIVE acquisition in frames ...
    %(frame rate depends on the camera)
Jump=1; %number of frames to jump to reduce frame rate
Refreshperiod=10*Jump %Number of frames between one SURF Analysis ...
    %and the following one
ReceiveVicon=0 %Enables (1) or Desables (0) Vicon UDP Receiver
%CAMERA OPTIONS
HFOV=66; %[degrees], Camera Horizontal Field of view
fl=0.038; %[meters], Camera focal lenght
Dstereo=0.25; %[meters], distance between stereo cameras
pix=3.75*10^-6; %[micrometers], square pixels dimension
%BACKGROUND STATIC REMOVAL OPTION
BackgroundSub=0; %Enables (1) or Desables (0) Background Subtraction
BSmode=0;%Selects Background Subtraction Mode (0=Static, 1=OpticalFlow)
BSrad=50;%radius of masking circle around unwanted features
%HARRIS OPTIONS
```

```

Hstrongest=100; %number of strongest Harris points
Hquality=0.01; %quality of Harris points
%SURF OPTIONS
SurfSwitch=0; %Enables (1) or Desables (0) SURF detection
Sstrongest=50; %number of strongest SURF points
%ANMS OPTIONS
ANMSSwitch=0; %Enables (1) or Desables (0) ANMS in Detection
ANMSdistance=10;% ANMS radius in pixels
%BLOB OPTIONS
Blength=100;%length added to Blob Gaussian Distribution
Bsigma=6;%standard deviation of the Blob Gaussian Distribution
Bnumber=8;%Number of biggest blobs for ROI detection
Bmode=1; %selects how to create the ROI starting from the BLOB ...
        %(1=from biggest blob maxs)
BroiDim=20;%pixels to add to ROI dimensions (for real videos keep 20)
%EPIPOLAR TRANSFORMATION
vpold=[]; %initializes the linear velocity vector estimation of ...
        %the epipolar
omegapold=[];%initializes the angular velocity vector estimation of ...
        %the epipolar
%KLT TRACKING
KLTroi=1; %Enables (1) or Desables (0) discarding valid points ...
        %too far from the ROI
KLTvalue=20;%[pixels] discards distance (for real videos keep 20)
KLTroiDim=50;%[pixels] lenght to make KLT-ROI bigger ...
        %(for real videos keep 50)
%STEREO OPTIONS
Stereovision=1; %Enables (1) or Desables (0) Stereovision loop
Stereoperiod=5; %Stereovision update period

%% VARIABLES INITIALIZATION (do not modify)

%this is a list of variables that require to be initialized only once

Detect=0;%flag intializer
Nsurf=0;%Number of SURF Features Detected
MODE=0; %Detection=0 KLT=1 STEREO=2 Geometric=3 SURF Check = 4
N=0;
Nroi=[0 0];
Metric=0; %initializes Max Metric value detected
Distance=0; %[scaled value] initialization STEREO distance
STEREOACTIVE=0; %initialization falg
Record=[];%initialization Distance recording matrix
oldpointsK=[];%[pixels] array of valid [x y] points collected
        %in previous frame
v_tot1=[]; %[meters per frame] epipolar linear velocity Solution 1
omega_tot1=[];%[radians per frame] epipolar angular velocity Solution 1
v_tot2=[]; %[meters per frame] epipolar linear velocity Solution 2
omega_tot2=[];%[radians per frame] epipolar angular velocity Solution 2
T0_tot=[];
ALLpoints_totX=[];%[pixels] array of all KLT [x] points collected ...
        %in previous frame
ALLpoints_totY=[];%[pixels] array of all KLT [y] points collected ...
        %in previous frame
Dst=[];%vector to collect the STEREO distance

```

```

Zero=zeros(3,1);
%Background subtraction counter
count=[1];
count2=[1];
for l=2:20
    count=[count,count+1];
    count2=[count2,count2-1];
end
counter=[count2,count];%Counter used to track the number of frames
                        %without detection

%% INPUT INITIALIZATION AND VIDEO CREATION

%This part of the algorithm starts the acquisition and the recording
%functionalities. Name of camera devices and name of the video might
%be changed according to the hardware/software input.

if Livecam==0
%videoname='SatTrasX.avi';%VideoL.avi';%testing Epipolar translation
videoname='SatRotX.avi';%testing Epipolar rotation
%videoname='VideoL.avi';%testing Tracking and Stereovision)
Video = VideoReader(videoname);
Frame = read(Video, 1);%Retrieve and Convert Frame k
Ifirst = rgb2gray(Frame);
SizeIMG=size(Ifirst);
nframes = get(Video, 'NumberOfFrames');
get(Video)
singleFrame = read(Video, 1);
elseif Livecam==1
%cam = webcam('QuickCam Orbit/Sphere MP');%for the LAB Computer
cam = webcam('Logitech QuickCam Pro 5000');%for the OFFICE Computer
Frame = snapshot(cam);
Ifirst = rgb2gray(Frame);
SizeIMG=size(Ifirst);
nframes=LIVEframes;
else
vid = videoinput('pointgrey', 1, 'F7_RGB_1280x960_Mode3');
src = getselectedsource(vid);
vid.FramesPerTrigger = 1;
vid.ReturnedColorspace = 'rgb';
start(vid);
Frame=getdata(vid);
%    Iired = Frame(:, :, 1); %camera 1
%    Iigreen = Frame(:, :, 2); %camera 2
%    Iiblue = Frame(:, :, 3); %camera 3
Ifirst = Frame(:, :, 3);
SizeIMG=size(Ifirst);
nframes=LIVEframes;
end

if CreateVideo==1 && CreateImage==1
    writerObj = VideoWriter('Video.avi','Motion JPEG AVI');
    %writerObj = VideoWriter('Video2.avi','Uncompressed AVI');

```

```

        writerObj.FrameRate=12;%24;
        open(writerObj);
end

InitialFrame=1;%24000;%3000; %Starting frame number of the video
FinalFrame=nframes;          %Final frame number of the video
LOOP=InitialFrame+1; %Detection Loop

%%Stereovision

%this part initializes the acquisition of the right camera for the
%stereovision measurements. The acquisition is active only
periodically.
%the inputs are either the blender video VideoR.avi (requires to be
%used in combination with VideoL.avi), or one of the other cameras of
...
% the bumblebee stereo system.

if Stereovision==1 && Livecam<2
videonameR='VideoR.avi';
VideoR = VideoReader(videonameR);
FrameR = read(VideoR, 1); %Retrieve and Convert Frame k
elseif Stereovision==1 && Livecam==2
FrameR= Frame(:, :, 1);
else
VideoR=Video;
end

%ROI INITIALIZATION

%For some application Background Subtraction cannot be done ...
%automatically, therefore is necessary to select an initial ROI ...
%to mask the regions with unwanted features. In the general case the
%intialized ROI is the entire image.

%ROI=[1,1,368,260];%ROI=[480,560,320,160]; Manually selected ROIs
ROI=[1,1,SizeIMG(2)-2,SizeIMG(1)-2];%Full Image Region of Interest
Xroi=zeros(2,5);%Region of Interest Box Corners Coordinates

```

2. Main script (MAIN_AViATOR.m)

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% AViATOR (ARTIFICIAL VISION ALGORITHM FOR TRACKING ORBITAL ROTATIONS)
%%
%Main Program

```

```

%% Description

% 1) Detects moving objects coming from space (black background area)
using
% Harris detection, Gaussian Blob and a Region of Interest, discarding
% fixed objects and background noise (eg Earth).

% 2) Once Detected the ROI becomes an image, where harris is used again
to
% initialize the KLT tracking of points

% 3) KLT points are used for updating the ROI and (for KLT>N) to define the
the
% geometric transformation for relative frames Camera/Target

% 4) Every 10 steps Surf/Harris points and KLT are taken outside the
ROI
% and the ROI is expanded (or reduced) if necessary

% 5) For Every Nframes checks if the stereo would work and then the
model
% uses stereo vision matching (and epipolar) to define the distance and
% rotations of the reference systems. Also a Geometries Measure is
made.

% 6) For Distance>D2 stereovision is not used and distance is retrieved
% from geometric measurements

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

%% Authors
% Alessio Grompone and Roberto Cristi
% PI: Marcello Romano
% Spacecraft Robotics Laboratory, Naval Postgraduate School 2015

close all
clear
clc
initializer
global Video
if ReceiveVicon==1
    UDPTarget=udp('170.160.1.41', 9090,'LocalPort', 9091);

    UDPCamera=udp('170.160.1.41', 9092,'LocalPort', 9093);

    TargetGround=[];
    CameraGround=[];
end
%% FRAME LOOP (SIMULATES REAL TIME FRAME ACQUISITION)
for k =InitialFrame:Jump:FinalFrame

```

```

k
%VICON ACQUISITION
if ReceiveVicon==1
    fopen(UDPTarget)
    fopen(UDPCamera)
    TargetPosition=str2num(fscanf(UDPTarget));
    CameraPosition=str2num(fscanf(UDPCamera));
    TargetGround=[TargetGround,TargetPosition];
    CameraGround=[CameraGround,CameraPosition];
    fclose(UDPTarget)
    fclose(UDPCamera)

end

if Livecam==0 %Input is a recorder video
    frame = read(Video, k);%Retrieve and Convert Frame k
    I1 = rgb2gray(frame);
elseif Livecam==1 %Input is a webcam
    frame=snapshot(cam);
    I1=rgb2gray(frame);
else %Input is the Bumblebee Camera
    start(vid);
    frame=getdata(vid);
    I1 = frame(:, :, 3);
end
SizeIMG=size(I1);

Periodcheck=(Refreshperiod*round(double(k)/Refreshperiod)==k);
%Periodcheck defines the periods for KLT Update
Stereocheck=(Stereooperiod*round(double(k)/Stereooperiod)==k);
%Stereocheck defines the periods for STEREOVISION Update

%% PHASE 1: PREPARATION and DETECTION
MODE=0;

if k<LOOP && BackgroundSub==1
    %if the Background subtraction option is active this part of
the
    %code runs the Detection on the preprocessed images on all
frames
    %until a target is detected

    framepost = read(Video, k+1);%Retrieve and Convert Frame k
    Ipost = rgb2gray(framepost);
    [I2, Detect, ROI]=FUN_BACKGROUNDSSUB(I1, Ipost);
    if Detect==0
        LOOP=LOOP+1;
        MODE=1;
    else
        MODE=2;
        [points,ROIh,Xroi, blob]=FUN_DETECTION(I2,ROI,MODE);
        tracker = vision.PointTracker('MaxBidirectionalError', 1);

```

```

        initialize(tracker, points.Location, frame);%Initialize KLT
Parameters IF NO SURF Points have been detected (uses Harris)
        LOOP=0;
        oldpointsH=points.Location;
        ROI=ROIh;
        Xroi=Xroi;
        end

    end

    if k==InitialFrame && BackgroundSub==0
        %if the Background subtraction option is not active the
algorithm
        %starts the detection on the entire image until a target is
found.
        %When a target is detected the values of the Detection are used
for
        %the initialization of the KLT tracking.

        MODE=5;
        Detect=1;
        [points,ROIh,Xroi,blob]=FUN_DETECTION(I1,ROI,MODE);
        tracker = vision.PointTracker('MaxBidirectionalError', 1);
        initialize(tracker, points.Location, frame);%Initialize KLT
Parameters IF NO SURF Points have been detected (uses Harris)
        ROI=ROIh;
        Xroi=Xroi;
        oldpointsH=points.Location;

    end

%% PHASE 2: KLT Tracking
if Detect==1
    %when a target is detected the KLT is initialized and run.
Memorization
    %of the points from previous frames are necessary for the optical
flow
    %measurements.

    MODE=3;
    if size(oldpointsK,1)==0
        oldpointsE=oldpointsH;
        oldpointsK=oldpointsH;
    else
        oldpointsK=points;
        oldpointsE=ALLpoints;
    end
    [Vpoints,ROIklt,Xroi,ALLpoints]=FUN_KLT(tracker, frame,ROI,Xroi);
    points=Vpoints;
    %MetricK
    Xroi=Xroi;
    ROI=ROIklt;

%CONTINUOUS EIGHT_POINT ALGORITHM
%runs the Epipolar transformation function. In order to avoid errors
during

```

%the updates, the values during and after the update are discarded with
%copies of the previous values

```
[v_a,omega_a,v_b,omega_b,flag]=FUN_EPIPOLAR(ALLpoints,oldpointsE,ROI);
```

```
%Discarding the values during and after the period update
AfterPeriodcheck=(Refreshperiod*round(double(k)/Refreshperiod)==k-1);
```

```
%AfterPeriodcheck detects the frame following the Harris Update period
```

```
Periodcheck=(Refreshperiod*round(double(k)/Refreshperiod)==k);
%Periodcheck detects the frame of the Harris Update period
```

```
if numel(omega_tot1)==0 || numel(v_tot1)==0 %initialize omega and v
```

```
omega_tot1=[omega_tot1,Zero];
```

```
v_tot1=[v_tot1,Zero];
```

```
omega_tot2=[omega_tot2,Zero];
```

```
v_tot2=[v_tot2,Zero];
```

```
elseif AfterPeriodcheck==1 %|| Periodcheck==1
```

```
%discard the value obtained after the harris update
```

```
omega_tot1=[omega_tot1,omega_tot1(:,end)];
```

```
v_tot1=[v_tot1,v_tot1(:,end)];
```

```
omega_tot2=[omega_tot2,omega_tot2(:,end)];
```

```
v_tot2=[v_tot2,v_tot2(:,end)];
```

```
else
```

```
omega_tot1=[omega_tot1,omega_a];
```

```
v_tot1=[v_tot1,v_a];
```

```
omega_tot2=[omega_tot2,omega_b];
```

```
v_tot2=[v_tot2,v_b];
```

```
end
```

```
%LOST TARGET RECOVERY AND PERIOD RESTART
```

```
%once the KLT loop is completed the Detection is restarted to update the
```

```
%values for the following period. In case the Target is lost during the %period or during the update, the full detection restores the ROI to the
```

```
%entire image.
```

```
if Periodcheck==1 || size(Vpoints,1)==0
```

```
if size(Vpoints,1)==0 %numel(pointsh.Location)==0
```

```
ROI=[1,1,SizeIMG(2)-2,SizeIMG(1)-2];%Region of Interest
```

```
%Xroi=[2 2 SizeIMG(2)-2 SizeIMG(2)-2 2;2 SizeIMG(1)-2
```

```
SizeIMG(1)-2 2 2];
```

```
MODE=5;
```

```
Detect=1;
```

```
[pointsh,ROIh,Xroi,blob]=FUN_DETECTION(I1,ROI,MODE);
```

```
Metric=max(pointsh.Metric);
```

```
ROI=ROIh;
```

```
Xroi=Xroi;
```

```

else
    [pointsh,ROIh,Xroi,blob]=FUN_DETECTION(I1,ROI,MODE);
    Metric=max(pointsh.Metric);
end %
if Metric>1*10^(-8)
    MODE=4;
    tracker = vision.PointTracker('MaxBidirectionalError', 1);
    initialize(tracker, pointsh.Location, frame);%Initialize
KLT Parameters IF HARRIS PARAMETERS HAVE BEEN FOUND
    oldpointsK=points;
    points=pointsh;
end
end
end

%% PHASE 3: Stereovision

%this part of the code runs the stereovision function and saves a
record of
%distances and frame for each stereovision update

if Stereovision==1 && k>InitialFrame
    if Stereocheck==1
        [Distance]=FUN_STEREO(ROI,I1,k);
        Dst=[Distance;k];
        Record=[Record,Dst];
    end
end

%% PHASE 4: Geometric Estimation
%   if D>Dgeom
        %Match features to recognize geometries
        %GeometricDistance estimation
%   end

%% PHASE 5: Plotting and recording

%here the Algorithm creates images from the analyzed frames adding in
Red
%the KLT tracked points, in Green the Harris Updated corners and in
Yellow
%the ROI. The Images are used also for the creation of a video.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if CreateImage==1
h=figure;
imshow(I1), %title('SURF(green)/KLT(red)');
hold on;
title(['AVIATOR Video']);
plot(Xroi(1,:),Xroi(2,:), 'y');
if MODE==2
    plot(oldpointsH(:,1),oldpointsH(:,2), 'b+');
elseif MODE==3

```

```

        if size(Vpoints,1)>=1
            plot(ALLpoints(:,1),ALLpoints(:,2),'r+');
        end
    elseif MODE==4
        if size(points.Location,1)>=1
            plot(points.Location(:,1),points.Location(:,2),'g+');
        end
    end
end

print(h, '-r120', '-dbmp', '1.bmp');

%in order to save frames as images uncomment this part
%     Filename=['Frame',num2str(k),'.bmp'];
%     print(h, '-r120', '-dbmp',Filename);

img =imread('1.bmp');
if CreateVideo==1
    writeVideo(writerObj,img);
end
close all
end
end

save('record.mat')

```

3. Background Subtraction (FUN_BACKGROUNDSUB.m)

```

function [I2,Detect,ROIout]=FUN_BACKGROUNDSUB(I1,Ipost)

%% Background Subtraction Function

%% Authors
% Alessio Grompone and Roberto Cristi
% PI: Marcello Romano
% Spacecraft Robotics Laboratory, Naval Postgraduate School 2015

%% INPUTS
%I1 = (gray scale image) current frame
%Ipost = (gray scale image) previous frame
%% OUTPUTS
%I2 = (gray scale image) preprocessed image (with masked background)
%Detect = detection status flag (0 no target detected, 1 target
detected)
%ROIout = [corner x, corner y, width, length] (4x1) (pixels)
        %Region of interest based on unwanted features

%%
global SizeIMG Ifirst BSmode BSrad Hquality ROI

if BSmode==0
%% Static Background subtraction

```

```

(mask in following frames all regions with features in first frames)

I2=I1;
Detect=0;
% identifies not black pixels
for i=1:SizeIMG(1)
    for j=1:SizeIMG(2)
        if Ifirst(i,j)> 2
            I2(i,j)=1;
        end
    end
end

if max(max(I2))>20
    Detect=1;
end

ROIout=ROI;
else

%% Optical Background subtraction (mask features that do not move)
I2=Ipost;

BSHquality=0.05;
%detects unwanted features
points1=detectHarrisFeatures(Ifirst,'MinQuality',BSHquality);
points2=detectHarrisFeatures(Ipost,'MinQuality',Hquality);
pointsLocation1=points1.Location;
pointsLocation2=points2.Location;
pointsMetric1=points1.Metric;
pointsMetric2=points2.Metric;
%discard points that are in the neighborhood of previously detected
points

%% static points location filter
SizePoints1=size(pointsLocation1,1);
SizePoints2=size(pointsLocation2,1);
Location=[];%static points
Metric=[];

Eliminated=0;
for m=1:SizePoints2
    for n=1:SizePoints1
        Eliminated_n=find(Eliminated==m);
        if isempty(Eliminated_n) %se non eliminato i
            if abs(pointsLocation1(n,1)-
pointsLocation2(m,1))+abs(pointsLocation1(n,2)-
pointsLocation2(m,2))<BSrad
                Eliminated=[Eliminated;m];
            end
        end
    end
end
if isempty(Eliminated_n)
    Location=[Location;pointsLocation2(m,1),pointsLocation2(m,2)];
end

```

```

        Metric=[Metric;pointsMetric2(m,:)];
    end
    Eliminated=0;
end
SpointsB=struct('Location',Location,'Metric',Metric);

if numel(SpointsB.Location)==0
    ROIout=ROI;
    Detect=0;
else
    [blob,ROIh,Xroi]=FUN_BLOB(SpointsB);
    [ Xroiout,ROIout] =
FUN_ROILIMITER(ROIh(1),ROIh(2),ROIh(3),ROIh(4));
    MODE=2;
    [points,ROIh,Xroi,blob]=FUN_DETECTION(I2,ROIout,MODE);
    if numel(points.Location)==0
        Detect=0;
    else
        Detect=1;
    end
end
end
end
end

```

4. HARRIS Detection (FUN_DETECTION.m)

```

function [points,ROIh,Xroi,blob]=FUN_DETECTION(I1,ROI,MODE)

global Hstrongest Hquality ANMSSwitch ANMSdistance
%% Authors
% Alessio Grompone and Roberto Cristi
% PI: Marcello Romano
% Spacecraft Robotics Laboratory, Naval Postgraduate School 2015

%Module for Harris Features Detection

if MODE==1 || (ROI(3)<1 && ROI(4)<1)
points=detectHarrisFeatures(I1,'MinQuality',Hquality);
else
points=detectHarrisFeatures(I1,'MinQuality',Hquality,'ROI',ROI);
end
if MODE==2 %%was MODE<=2
points=points.selectStrongest(1);
else
points=points.selectStrongest(Hstrongest);
end

pointsLocation=points.Location;
pointsMetric=points.Metric;

```

```

if ANMSSwitch==1
[points]= anms_fun(pointsMetric, pointsLocation(:,1),
pointsLocation(:,2), ANMSdistance);
pointsH=points;
end

if MODE<=5
[blob,ROIh,XroiH]=FUN_BLOB(points); %blobs coordinates - points,
gaussian parameters, image size
else
blob=[];
ROIh=ROI;
end

%[ XroiH,ROIh] = FUN_ROILIMITER(ROIh(1),ROIh(2),ROIh(3),ROIh(4));
end

function [points]= anms_fun(x, x_i, x_j, D)
%% ANMS FUNCTION
%reduces the number of Harris detected features in cluttered areas
%Author: Roberto Cristi, modified by Alessio Grompone

% [y, y_i, y_j]= anms(x_strength, x_i, x_j, D)

% x, y input and output vectors of "strength"

% x_i, x_j, y_i, y_j , input and ouput vectors of 2D coordinates (i,j)
for
% associated to each point

% D min distance between points we keep. There is at most one point of
% strength in any square which is 2D x 2D

A=[x,x_i,x_j]; % create a matrix of [ metric, x position, y position]

%sort the matrix A in function of the metric (strongest first)
[Z, K]=sort(A(:,1), 'descend');
Z=A(K,:);

z_x=Z(:,2); z_y=Z(:,3);
z_metric=Z(:,1);
Nsizei=size(K);
Eliminated=0;
zx=[];
zx_i=[];
zx_j=[];
for i=1:Nsizei(1)
Eliminated_i=find(Eliminated==i);
if isempty(Eliminated_i) %se non eliminato i
for j=1:Nsizei(1)
Eliminated_j=find(Eliminated==j);
if isempty(Eliminated_j) %se non eliminato j

```

```

        if abs(z_x(j)-z_x(i))+abs((z_y(j))-z_y(i))>D ...
            || abs(z_x(j)-z_x(i))+abs((z_y(j))-z_y(i))==0
            Eliminated=[Eliminated];
        else
            Eliminated=[Eliminated;j];
        end
    end
end
end
end
for i=1:Nsizei(1)
    Eliminated_i=find(Eliminated==i);
    if isempty(Eliminated_i) %se non eliminato i
        zx=[zx;z_metric(i)];
        zx_i=[zx_i;z_x(i)];
        zx_j=[zx_j;z_y(i)];
    end
end
Location=[zx_i, zx_j];
elements=size(zx);
elem=elements(1);
points = cornerPoints(Location, 'Metric', zx);
end

```

5. BLOB Selection (FUN_BLOB.m)

```

function [blob,ROIh,Xroi]=FUN_BLOB(points)
%% BLOB FUNCTION

%% Authors
% Alessio Grompone and Roberto Cristi
% PI: Marcello Romano
% Spacecraft Robotics Laboratory, Naval Postgraduate School 2015

%% INPUTS
%points = (pixel) [x y] (1,2) Pixel coordinates of the points detected

%% OUTPUTS
%blob = Matrix of points (Image size) with white blobs
%ROIh = (pixels) (4x1) Region of interest[corner x, corner y,
width,length]
%Xroi = (pixels) [x,y](5x2) Region of Interest Box 4 Corners
Coordinates

%%
global Blength Bsigma SizeIMG Xroi
Xrois=Xroi;
SIZE_IMG=[SizeIMG(2),SizeIMG(1)];
L=Blength;
sigma=Bsigma;

```

```

M=zeros(SIZE_IMG+(2*L+1));
% Gaussian Blobs Creation and Filtering
for n0=1:length(points)
n=-L:L;
h1=(1/(sqrt(2*pi)*sigma))*exp(-(n.^2)/(2*sigma^2));
hgauss=h1'*h1;
IJ=round(points.Location(n0,:));
K=points.Metric(n0,:);
%save('blob.mat')
M(IJ(1):IJ(1)+2*L, IJ(2):IJ(2)+2*L)=M(IJ(1):IJ(1)+2*L,
IJ(2):IJ(2)+2*L)...
+K*hgauss;
end
M=M(L+1:SIZE_IMG(1)+L, L+1:SIZE_IMG(2)+L);
M=sign(M-0.5*max(max(M)));

%Convert to binary (0 and 1 only)
for i=1:SIZE_IMG(1)
for j=1:SIZE_IMG(2)
if M(i,j)==-1
M(i,j)=0;
end
end
end

blob=M; %Matrix image of blobs

[ROIh,Xroi]=roiblob_fun(blob,Xrois);% build ROI from blobs+HARRIS

end

%%ROI from BLOB FLUNCTION
function [ROI,Xroi]=roiblob_fun(blob,Xrois)
global Bnumber Bmode SizeIMG Broidim

Xroi=Xrois;
SIZE_IMG=SizeIMG;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Detects and selects Blob and transforms the index in coordinates
%detect connected areas (blobs)
CC = bwconncomp(blob',Bnumber);

index=[]; %index vector of the blobs that we want to include
Xcoll=[];
Ycoll=[];
if CC.NumObjects>0
if Bmode==1 %First frame chose the biggest Blob and Build the ROI
D=0;
for i=1:(CC.NumObjects)
%Choose only the biggest perimeter component
A=size(CC.PixelIdxList{1,i});
if D<A(1)
D=A(1); %Number of pixels in connected

```

```

index=i; %Index of the collection
end
end
%linear pixel index to XY converter
pase=10;
[X,Y]=indextolinear(CC,D,index,pase);
%Average Center for Region of Interest ROI
Xmean=floor(mean(X));
Ymean=floor(mean(Y));
width=floor(max(X)-min(X))+ BroiDim ;
height=floor(max(Y)-min(Y))+ BroiDim ;

if width<1
    width=1;
end
if height<1
    height=1;
end

Xroi(:,1)=[Xmean-(width/2);Ymean-(height/2)];
Xroi(:,2)=[Xmean-(width/2);Ymean+(height/2)];
Xroi(:,3)=[Xmean+(width/2);Ymean+(height/2)];
Xroi(:,4)=[Xmean+(width/2);Ymean-(height/2)];
Xroi(:,5)=[Xmean-(width/2);Ymean-(height/2)];

Xcoll=X;
Ycoll=Y;
else
for i=1:(CC.NumObjects)
A=size(CC.PixelIdxList{1,i});
pase=10;
[X,Y]=indextolinear(CC,A,i,pase);
Xmean=floor(mean(X));
Ymean=floor(mean(Y));

%Choose only the blobs within the old ROI
if Xmean>Xroi(1,1) && Xmean<Xroi(1,3) && Ymean>Xroi(2,1) &&
Ymean<Xroi(2,2)
Xcoll=[Xcoll,X];
Ycoll=[Ycoll,Y];
end
end

%Average Center for Region of Interest ROI
Xmean=floor(mean(Xcoll));
Ymean=floor(mean(Ycoll));
width=floor(max(Xcoll)-min(Xcoll))+ BroiDim ;
height=floor(max(Ycoll)-min(Ycoll))+ BroiDim ;

if width<1
    width=1;
end
if height<1
    height=1;
end

```

```

end
%save('blob.mat')
Xroi(:,1)=[Xmean-(width/2);Ymean-(height/2)];
Xroi(:,2)=[Xmean-(width/2);Ymean+(height/2)];
Xroi(:,3)=[Xmean+(width/2);Ymean+(height/2)];
Xroi(:,4)=[Xmean+(width/2);Ymean-(height/2)];
Xroi(:,5)=[Xmean-(width/2);Ymean-(height/2)];
end

Xroi(1,5)=Xroi(1,1);
Xroi(2,5)=Xroi(2,1);

width=Xroi(1,3)-Xroi(1,1);
height=Xroi(2,3)-Xroi(2,1);

ROI=[Xroi(1,1),Xroi(2,1),width,height];
else
%Use entire IImage as Region of interest in case of loss
ROI=[2,2,SizeIMG(2)-2,SizeIMG(1)-2];
Xroi=[2 2 SizeIMG(2)-2 SizeIMG(2)-2 2;
      2 SizeIMG(1)-2 SizeIMG(1)-2 2 2];
end
end

```

6. KLT Tracking (FUN_KLT.m)

```

function [Vpoints,ROIo,Xroio,ALLpoints]=FUN_KLT(tracker,frame,ROI,Xroi)

%% KLT tracker function

%% Authors
% Alessio Grompone and Roberto Cristi
% PI: Marcello Romano
% Spacecraft Robotics Laboratory, Naval Postgraduate School 2015

%% INPUTS
%tracker = initialization structure
%frame = current frame image (gray image)
%ROI= (pixels) (4x1) Region of interest [corner x, corner y,
width,length]
%% OUTPUTS
%Vpoints = (nx2) array of valid tracked points
%ROIo =(pixels) (4x1) Region of interest [corner x, corner y,
width,length]
%Xroio =(pixels) [x,y](5x2) Region of Interest Box 4 Corners
Coordinates
%ALLpoints = (mx2) Valid and lost points in matching order

%%
global SizeIMG KLTroi KLTvalue KLTroiDim

```

```

VpointsROI=[];
VpointsOUT=[];
KLTmagnitudesROI=[];
KLTdirectionsROI=[];
Nvpoints2=[0,0];
MODE=1;

SIZE_IMG=SizeIMG;
%provides al KLT tracked points and validity vector
[KLTpoints, validity] = step(tracker, frame); %
Vpoints=KLTpoints(validity,:); %filter only valid points
ALLpoints=KLTpoints;

for i=1:size(ALLpoints,1)
    if validity(i)==0
        ALLpoints(i,:)= [0,0];
    end
end

%Remove "Valid" Points that are too far from ROI
N=size(Vpoints,1);
if N>0 && KLTroi==1
    for i=1:N %If inside the input ROI
        if Vpoints(i,1)>=Xroi(1,1)-KLTvalue &&
Vpoints(i,1)<=Xroi(1,3)+...
            KLTvalue && Vpoints(i,2)>=Xroi(2,1)-KLTvalue && ...
            Vpoints(i,2)<=Xroi(2,2)+KLTvalue
            VpointsROI=[VpointsROI;Vpoints(i,:)];
        end
    end
end

Vpoints=VpointsROI;

%% KLT ROI UPDATE

%uses the mean KLT valid points to expand, shrink or translate the ROI

if size(Vpoints,1)>=1
    %Calculate mean of valid KLT points to shift the old ROI
    KLTmeanX=mean(Vpoints(:,1));
    KLTmeanY=mean(Vpoints(:,2));
    %Estimate the dimensions of the new ROI
    KLTwidth=(max(Vpoints(:,1))-min(Vpoints(:,1)))+KLTroiDim; %For
ROI(3)
    KLTlenght=(max(Vpoints(:,2))-min(Vpoints(:,2)))+KLTroiDim;%For
ROI(4)
    ROI1=KLTmeanX-(KLTwidth/2);
    ROI2=KLTmeanY-(KLTlenght/2);
    [ Xroiout,ROIout] = FUN_ROILIMITER(ROI1,ROI2,KLTwidth,KLTlenght);
    Xroi=Xroiout;
    ROI=ROIout;
else
    Vpoints=[];
end

```

```

Xroi0=Xroi;
ROI0=ROI;

end

```

7. Epipolar Transformation (FUN_EPIPOLAR.m)

```

function
[v_a,omega_a,v_b,omega_b,flag]=FUN_EPIPOLAR(Vpoints,VpointsOld,ROI)
%% Authors
% Alessio Grompone and Roberto Cristi
% PI: Marcello Romano
% Spacecraft Robotics Laboratory, Naval Postgraduate School 2015

% From the "Continuous eight-point algorithm"
% Ref "An Invitation to 3D Vision" Page 151, Algorithm 5.3

%% INPUTS
%Vpoints = (pixels) (nx2) Tracked Ponto coordinates [x, y] in the
current frame
%VpointsOld = (pixels) (nx2) Tracked Ponto coordinates [x, y] from
previous frame
%ROI= (pixels) (4x1) Region of interest [corner x, corner y,
width,length]

%% OUTPUTS
%v_a = [vx, vy, vz] (meters/frame) (3x1) Estimated Body Linear
Velocities
    %for the Solution1
%omega_a = [wx,wy,wz] (radians/frame)(3x1) Estimated Body Angular
Velocities
    %for the Solution1
%v_b = [vx, vy, vz] (meters/frame) (3x1)Estimated Body Linear
Velocities
    %for the Solution2
%omega_b = [wx,wy,wz] (radians/frame) (3x1) Estimated Body Angular
Velocities
    %for the Solution2
%flag = (1) debugging flag (1 if are using the epipolar function)

%%
global fl vpold omegapold

focallenght=f1;
n=size(Vpoints,1);
X_2Dcamera1=[];
X_2Dcamera2=[];
%reinitializes just in case KLT doesn't have the same number of matched

```

```

%points(like when it updates HARRIS).
vs=zeros(3,3,4);
omegas=zeros(3,3,4);
v_a=zeros(3,1);
omega_a=zeros(3,1);
v_b=zeros(3,1);
omega_b=zeros(3,1);
flag=0;%checks if this function is running or not

%% Check if we are in the same KLT loop form the size of the Valid
Points
if n>0 && size(Vpoints,1)==size(VpointsOld,1) &&
((max(abs(Vpoints(:,1))-VpointsOld(:,1)))>0) || (max(abs(Vpoints(:,2))-
VpointsOld(:,2)))>0))
    flag=1;
    %Makes the vector of points xj in the coordinates of page 141
    for i=1:n
        if Vpoints(i,1)==0 || VpointsOld(i,1)==0 || Vpoints(i,2)==0 ||
VpointsOld(i,2)==0
            %discard points that are zero (not valid KLT points)
            n=n-1;%counter to reduce total number of points
        else
            %sets the coordinates frame in the center of the ROI
            x1=[VpointsOld(i,2)-(ROI(2)+(ROI(4)/2));-
VpointsOld(i,1)+(ROI(1)+(ROI(3)/2));focallenght];
            x2=[Vpoints(i,2)-(ROI(2)+(ROI(4)/2));-
Vpoints(i,1)+(ROI(1)+(ROI(3)/2));focallenght];
            %collects the coordinates in a (nx2) array
            X_2Dcamera1=[X_2Dcamera1,x1];%Points on 2D plane Camera 1
(x z f)
            X_2Dcamera2=[X_2Dcamera2,x2];%Points on 2D plane Camera 2
(x z f)
        end
    end

    %% Optical flow function (measures the velocities projected on the
image)

    [u]=FUN_OPTFLOW(X_2Dcamera1,X_2Dcamera2);

    %% Estimate essential vector
    [vp1,omegap1,vp2,omegap2]=epipolar3(X_2Dcamera1,u,n);
    %use epipolar1 for the Eight-Point linear Algorithm
    %use epipolar2 for the Eight-Point continuous Algorithm
    %use epipolar3 for the Four-Point continuous Algorithm

    %% Collectes the two solutions in two arrays
    %This parts regroupes the solutions based on the proximity with the
    %previous value
    if numel(vpold)==0 %need to initialize vpold with the first
solution
        vpold=vp1;
        omegapold=omegap1;
    end

```

```

Dv1=mean(abs(vp1-vpold)+abs(omegap1-omegapold));
Dv2=mean(abs(vp2-vpold)+abs(omegap2-omegapold));
if Dv1<=Dv2
    v_a=vp1;
    omega_a=omegap1;
    v_b=vp2;
    omega_b=omegap2;
else
    v_a=vp2;
    omega_a=omegap2;
    v_b=vp1;
    omega_b=omegap1;
end
vpold=(vpold+v_a)/2;
omegapold=(omegapold+omega_a)/2;
end
end

```

8. Continuous Eight-Points Algorithm (epipolar1.m)

```

function [vp1,omegap1,vp2,omegap2]=epipolar1(X_2Dcamera1,u,n)

% Continuous eight-point algorithm
% Ref "An Invitation to 3D Vision" Page 151, Algorithm 5.3
%% Authors
% Alessio Grompone and Roberto Cristi
% PI: Marcello Romano
% Spacecraft Robotics Laboratory, Naval Postgraduate School 2015

%%INPUTS
%X_2Dcamera1 = [x,y] (pixels) (nx2) Coordinates of tracked valid points
%u = [Vx,Vy] (pixel/frame) (nx2) velocities from optical flow
%n = number of valid points

%%OUTPUTS
%vp1 = [vx, vy, vz] (meters/frame) (3x1) Estimated Body Linear
Velocities
    %for the Solution1
%omegap1 = [wx,wy,wz] (radians/frame) (3x1) Estimated Body Angular
Velocities
    %for the Solution1
%vp2 = [vx, vy, vz] (meters/frame) (3x1) Estimated Body Linear
Velocities
    %for the Solution2
%omegap2 = [wx,wy,wz] (radians/frame) (3x1) Estimated Body Angular
Velocities
    %for the Solution2
%flag = debugging flag (1 if are using the epipolar function)

global fl SizeIMG

```

```

vp1=zeros(3,1);
omegap1=zeros(3,1);
vp2=zeros(3,1);
omegap2=zeros(3,1);
%% Estimate essential vector
for i=1:n
    x=X_2Dcamera1(1,i);
    y=X_2Dcamera1(2,i);
    z=X_2Dcamera1(3,i);
    a(:,i)=[u(3,i)*y-u(2,i)*z,u(1,i)*z-u(3,i)*x,u(2,i)*x-
u(1,i)*y,x^2,...
    2*x*y,2*x*z,y^2,2*y*z,z^2]';
    X(:,i)=[a(:,i)];
end
X=X';
if rank(X)>=8
    %%For not noisy measurements we want to minimize X*Es=0
    [Ux,Sx,Vx]=svd(X);
    Es1=Vx(:,9);
    %%For noisy measurements we want to minimize ||X*Es||^2=0
    [Vxn,Dxn]=eig(X'*X,'vector');
    Dxmin=min(Dxn);
    for i=1:size(Dxn,1)
        if Dxn(i)==Dxmin
            Es=Vxn(:,i); %Stacked Epipolar Matrix
        end
    end
    %
    % Es3=lsqlin(X,zeros(n,1));
    %
    % Es2=null(X);

    vo=[Es(1);Es(2);Es(3)];
    %Es=Es/norm(vo);
    vo=[Es(1);Es(2);Es(3)];
    s_e=[Es(4) Es(5) Es(6) Es(7) Es(8) Es(9)];
    s=[s_e(1) s_e(2) s_e(3);s_e(2) s_e(4) s_e(5);s_e(3) s_e(5) s_e(6)];
    %%Multiply Es with a scalar such that the vector vo becomes unit
norm

    %% Recover the symmetric epipolar component
    %s might not be symmetric due to noise, therefore we project it in
the space
    %of symmetric epipolar components

    [V1,D]=eig(s,'vector');
    [lamb,index]=sort(D,'descend');
    Vvect=V1(:,index);

    sigma=[(2*lamb(1)+lamb(2)-
lamb(3))/3;(lamb(1)+2*lamb(2)+lamb(3))/3;...
    (2*lamb(3)+lamb(2)-lamb(1))/3];
    s=Vvect*diag(sigma)*Vvect';%Symmetrized s

```

```

%% Recover the velocity from the symmetric epipolar component
lambda1=sigma(1)-sigma(3);
theta=acos((-sigma(2)/lambda1));
theta2=(theta/2)-(pi/2);

Ry1=[cos(theta) 0 sin(theta);
      0 1 0 ;
      -sin(theta) 0 cos(theta)];%@(theta)
Ry2=[cos(theta2) 0 sin(theta2);
      0 1 0 ;
      -sin(theta2) 0 cos(theta2)];%@(theta/2)-(pi/2)
Rz1=[cos(pi/2) -sin(pi/2) 0;
      sin(pi/2) cos(pi/2) 0;
      0 0 1];%@(+pi/2)
Rz2=[cos(-pi/2) -sin(-pi/2) 0;
      sin(-pi/2) cos(-pi/2) 0;
      0 0 1];%@(-pi/2)

V=Vvect*Ry2';%As the book
U=-V*Ry1;

Siglam=diag([lambda1,lambda1,0]);
Sigl=diag([1,1,0]);

omegas(:,:,1)=U*Rz1*Siglam*U'; vs(:,:,1)=V*Rz1*Sigl*V';
omegas(:,:,2)=U*Rz2*Siglam*U'; vs(:,:,2)=V*Rz2*Sigl*V';
omegas(:,:,3)=V*Rz1*Siglam*V'; vs(:,:,3)=U*Rz1*Sigl*U';
omegas(:,:,4)=V*Rz2*Siglam*V'; vs(:,:,4)=U*Rz2*Sigl*U';

%% Recover the velocity from continuous essential matrix
vtempold=0;
vsolutions=[];
for g=1:4
    vsolutions=[vsolutions,[vs(3,2,g);vs(1,3,g);vs(2,1,g)]];
    vtemp=[vs(3,2,g);vs(1,3,g);vs(2,1,g)]'*vo;
    if vtemp>vtempold
        v=[vs(3,2,g);vs(1,3,g);vs(2,1,g)];
        omega=[omegas(3,2,g);omegas(1,3,g);omegas(2,1,g)];
        vtempold=vtemp;
    end
end
vp1=v;
omegap1=omega;
vp2=[vs(3,2,2);vs(1,3,2);vs(2,1,2)];
omegap2=[omegas(3,2,2);omegas(1,3,2);omegas(2,1,2)];

end
%save('epipolar1.mat')
end

```

9. Continuous Four-Points Algorithm (epipolar3.m)

```
function [vp1,omegap1,vp2,omegap2]=epipolar3(X_2Dcamera1,u,n)

%% Estimate Epipolar from 4 planar or more values

%% Authors
% Alessio Grompone and Roberto Cristi
% PI: Marcello Romano
% Spacecraft Robotics Laboratory, Naval Postgraduate School 2015

%%INPUTS
%X_2Dcamera1 = [x,y] (pixels) (nx2) Coordinates of tracked valid points
%u = [Vx,Vy] (pixel/frame) (nx2) velocities from optical flow
%n = number of valid points

%OUTPUTS
%vp1 = [vx, vy, vz] (meters/frame) (3x1) Estimated Body Linear
Velocities
    %for the Solution1
%omegap1 = [wx,wy,wz] (radians/frame) (3x1) Estimated Body Angular
Velocities
    %for the Solution1
%vp2 = [vx, vy, vz] (meters/frame) (3x1) Estimated Body Linear
Velocities
    %for the Solution2
%omegap2 = [wx,wy,wz] (radians/frame) (3x1) Estimated Body Angular
Velocities
    %for the Solution2
%flag = debugging flag (1 if are using the epipolar function)

    X=[];
    B=[];
%% Compute first approximation of the continuous homography matrix
    for i=1:n
        % skewsymmetric matrix build
        x=X_2Dcamera1(1,i);
        y=X_2Dcamera1(2,i);
        z=X_2Dcamera1(3,i);
        skew=[0 -z y/z 0 -x;-y x 0];
        % X matrix build
        a(:,:)=kron(X_2Dcamera1(:,i),skew); %kronecher
        X=[X,a(:,:)]';
        % B matrix build
        b=skew*u(:,i);
        B=[B,b']';
    end
    X=X';
    B=B';

    Hls=pinv(X)*B;%Stacked homography matrix not in essential space

    if numel(Hls)==0
        %initialize the homography matrix not in essential space
```

```

        Hl=zeros(3,3);
    else
        %unstack the homography matrix not in essential space
Hl=[Hls(1),Hls(2),Hls(3);Hls(4),Hls(5),Hls(6);Hls(7),Hls(8),Hls(9)]';
    end
    %% Normalization of the continuous homography matrix
    [Vl,Dl]=eig(Hl'+Hl);%measure eigenvalues and eigenvectors
    H=Hl-0.5*Dl(2,2)*eye(3);%Homography Matrix

    %% Decomposition of the continuous homography matrix
    [V,D]=eig(H'+H);
    Di=[D(1,1);D(2,2);D(3,3)];
    % reorder Eigenvalues and vectors from max to min eigenvalue
    [lamb,index]=sort(Di,'descend');
    V=V(:,index);

    alpha=0.5*(lamb(1)-lamb(3));

    vlh=0.5*(sqrt(2*lamb(1))*V(:,1)+sqrt(-2*lamb(3))*V(:,3));
    N1h=0.5*(sqrt(2*lamb(1))*V(:,1)-sqrt(-2*lamb(3))*V(:,3));
    v2h=N1h;
    N2h=vlh;

    e3=[0,0,1]'; %optical axis
    Ncheck=zeros(3,1); %initialize Depth constraint check

    %%Compute Solution 1
    Vd1=sqrt(alpha)*vlh;%Linear Velocity
    N1=(1/sqrt(alpha))*N1h;
    omega1=0.5*((H-vlh*N1h')-(H-vlh*N1h'))';%Angular Velocity
Skewsymmetric
    Ncheck(1)=N1'*e3;%Depth constraint check
    %%Compute Solution 2
    Vd2=sqrt(alpha)*v2h;%Linear Velocity
    N2=(1/sqrt(alpha))*N2h;
    omega2=0.5*((H-v2h*N2h')-(H-v2h*N2h'))';%Angular Velocity
Skewsymmetric
    Ncheck(2)=N2'*e3;%Depth constraint check
    %%Compute Solution 3
    Vd3=-Vd1;%Linear Velocity
    N3=-N1;
    omega3=omega1;%Angular Velocity Skewsymmetric
    %%Compute Solution 4
    Vd4=-Vd2;%Linear Velocity
    N4=-N2;
    omega4=omega2;%Angular Velocity Skewsymmetric

    %Select Solutions

    if D(1)==0 && D(2)==0 && D(3)==0
        %if all eigenvalues are zero only one solution exist
        Vsoll=zeros(3,1);
        Nsoll=zeros(3,1);

```

```

    Osol1=H;
    Vsol2=zeros(3,1);
    Nsol2=zeros(3,1);
    Osol2=H;
    %elseif Vd1==zeros(3,1) || Vd2==zeros(3,1) %|| cross(Vd1,N1)==0
|| e3'*v==0 %There is a unique solutions
    elseif Vd1(1)==0 && Vd1(2)==0 && Vd1(3)==0 %|| cross(Vd1,N1)==0 ||
e3'*v==0 %There is a unique solutions
    %if all linear velocities are zero only one solution exist
    Vsol1=zeros(3,1);
    Nsol1=N1; %?
    Osol1=H;
    Vsol2=Vsol1;
    Nsol2=Nsol1;
    Osol2=Osol2;
else
    %if 4 solutions exist select only the 2 valid solution N'e3>0
    if Ncheck(1)>0
        Vsol1=Vd1;
        Nsol1=N1;
        Osol1=omega1;
    else
        Vsol1=Vd3;
        Nsol1=N3;
        Osol1=omega3;
    end
    if Ncheck(2)>0
        Vsol2=Vd2;
        Nsol2=N2;
        Osol2=omega2;
    else
        Vsol2=Vd4;
        Nsol2=N4;
        Osol2=omega4;
    end
end

vp1=Vsol1;
omegap1=[Osol1(3,2);Osol1(1,3);Osol1(2,1)];
vp2=Vsol2;
omegap2=[Osol2(3,2);Osol2(1,3);Osol2(2,1)];

% save('epipolar3.mat')
end

```

10. Stereovision Range Estimation (FUN_STEREO_RANGE.m)

```

function [Distance]=FUN_STEREO(ROI,I1,k)
%% Features and Stereo

```

```

%Subfunction of the AViATOR algorithm, for the detection and matching
of
%features on stereo images and estimation of the distance from a
tracked
%target.

%% Authors
% Alessio Grompone and Roberto Cristi
% PI: Marcello Romano
% Spacecraft Robotics Laboratory, Naval Postgraduate School 2015

%% INPUT
%ROI = [corner x, corner y, width, legth] (4x1) (pixels) Region of
interest
%I1 = (gray scale image) current frame
%k = (scalar) current frame number
%% OUTPUT
%Distance = (scalar) scaled Distance
%%
global fl pix Dstereo Livecam vid experiment %VideoR
%Cameras Relative Properties
    MODER=6;
    if Livecam==0 %Input is a recorder video
        %frameR = read(VideoR, k);%Retrieve and Convert Frame k
        %ks=30-k+1;
        ks=k;

folderR=('C:\Users\Grompone\Desktop\AViATOR\TEST5_LIVEepipolarEstere\'
,experiment,'\FrameR',num2str(ks),'.bmp']);
        frameR = imread(folderR); %read(Video, 1);%Retrieve and Convert
Frame k
        I1R = rgb2gray(frameR);
        elseif Livecam==2 %Input is a webcam
            start(vid);
            frameR=getdata(vid);
            I1R = frameR(:, :, 1);
            end

        I1L=I1;
        % Find SURF matched points for STEREO COMPARISON
        points1 = detectSURFFeatures(I1L,'ROI',ROI);
        points2 = detectSURFFeatures(I1R,'ROI',ROI);
        [f1, vpts1] = extractFeatures(I1L, points1);
        [f2, vpts2] = extractFeatures(I1R, points2);
        indexPairs = matchFeatures(f1, f2, 'Prenormalized', true) ;
        matchedPoints1 = vpts1(indexPairs(:, 1));
        matchedPoints2 = vpts2(indexPairs(:, 2));
        PointsL=matchedPoints1.Location;
        PointsR=matchedPoints2.Location;

        %Cameras Relative Properties
        pointsL=PointsL;
        pointsR=PointsR;
        %fl=0.0038;%Cameras Focal lenght (m)

```

```

%we assume camera 1 in the Inertia Reference frame
%we assume only camera rotations along the Z axis

Xc1=[0;0;0;0;0;0];%Camera position and attitude 1 (X Y Z theta
phi psi) Inertia Frame
Xc2=[Dsterео;0;0;0;0;0];%Camera position and attitude 2 (X Y Z
theta phi psi) Inertia Frame

%rotation between camera position 1 and Inertia frame
Rc=[1 0 0;0 cos(Xc1(4)) sin(Xc1(4)); 0 -sin(Xc1(4))
cos(Xc1(4))] * [cos(Xc1(5)) 0 sin(Xc1(5));0 1 0;sin(Xc1(5)) 0
cos(Xc1(5))] * [cos(Xc1(6)) -sin(Xc1(6)) 0;sin(Xc1(6)) cos(Xc1(6)) 0; 0
0 1];

%translation between camera position 1 and Inertia frame
Tc=[0,0,0,1];

%rotation between camera position 1 and 2
R21=[1 0 0;0 cos(Xc2(4)-Xc1(4)) sin(Xc2(4)-Xc1(4)); 0 -
sin(Xc2(4)-Xc1(4)) cos(Xc2(4)-Xc1(4))] * [cos(Xc2(5)-Xc1(5)) 0
sin(Xc2(5)-Xc1(5));0 1 0;sin(Xc2(5)-Xc1(5)) 0 cos(Xc2(5)-Xc1(5))] *
[cos(Xc2(6)-Xc1(6)) -sin(Xc2(6)-Xc1(6)) 0;sin(Xc2(6)-Xc1(6))
cos(Xc2(6)-Xc1(6)) 0; 0 0 1];
R12=R21';
%translation between camera position 1 and 2
T12=[Xc1(1)-Xc2(1);
Xc1(2)-Xc2(2);
Xc1(3)-Xc2(3)];
Nsp= size(pointsR,1);
LL=[];
C=0;
d=0;
if Nsp>0
    for i=1:Nsp
        i=1;
        % %in meters
        x1=pointsL(i,:)*pix;
        x2=pointsR(i,:)*pix;

        x1=[x1(1);x1(2);f1]; %Point on object (on 2D plane
Camera 1) (x y f)
        x2=[x2(1);x2(2);f1]; %Point on object (on 2D plane
Camera 2) (x y f)

        x2cross=[0 -x2(3) x2(2);x2(3) 0 -x2(1); -x2(2) x2(1)
0];

        %l1*(x2cross*T21)+(x2cross*R21*x1)=0;
        C=double(x2cross*R21*x1);
        d=double(x2cross*T12); %or T21 ?

        %least squares for Lambda1 determination
        l1=lsqlin(C,d);
        LL=[LL,l1];
    end
end

```

```

end
l1=mean(LL);
%Point distance from reference camera 1
Z=l1*f1;
%Distance=-Z;%in
Distance=-Z;%*(80/27.5)/100;%Calibrated Value (in m)

```

11. Optical Flow Estimation (FUN_OPTFLOW.m)

```

function [u]=FUN_OPTFLOW(X_2Dcamera1,X_2Dcamera2)

%% Optical Flow measurement

%% Authors
% Alessio Grompone and Roberto Cristi
% PI: Marcello Romano
% Spacecraft Robotics Laboratory, Naval Postgraduate School 2015

%%INPUT
% X_2Dcamera1 = [x y f] (3x1) 2D camera position of points in frame 1
% X_2Dcamera2 = [x y f] (3x1) 2D camera position of points in frame 2

%%OUTPUT
% u = [xdot ydot 0] %velocity vector (pixels/frame)

n = size(X_2Dcamera1,2);
u = zeros(3,n);

for i = 1:n
u(:,i) = [X_2Dcamera2(1,i)-X_2Dcamera1(1,i);X_2Dcamera2(2,i)-
X_2Dcamera1(2,i); 0];
end

```

12. Computed ROI Limits Validation (FUN_ROILIMITER.m)

```

function [ Xroiout,ROIout] = FUN_ROILIMITER(ROI1,ROI2,ROI3,ROI4)

global SizeIMG
%ROILIMITER_FUN Summary of this function goes here
% Limits the ROI within the image
%used for the blob, the klt and for the surf functions

SIZE_IMG=SizeIMG;
%Move ROI with KLT tracked mean
width=ROI3;
height=ROI4;
Xmean=ROI1+(width/2);
Ymean=ROI2+(height/2);

```

```

Xroi(:,1)=round([Xmean-(width/2);Ymean-(height/2)]);
Xroi(:,2)=round([Xmean-(width/2);Ymean+(height/2)]);
Xroi(:,3)=round([Xmean+(width/2);Ymean+(height/2)]);
Xroi(:,4)=round([Xmean+(width/2);Ymean-(height/2)]);
Xroi(:,5)=round([Xmean-(width/2);Ymean-(height/2)]);
%limit the ROI within the image
if Xroi(1,3)>SIZE_IMG(2)-1
    Xroi(1,3)=SIZE_IMG(2)-1;
    Xroi(1,4)=SIZE_IMG(2)-1;
end
if Xroi(2,2)>SIZE_IMG(1)-1
    Xroi(2,2)=SIZE_IMG(1)-1;
    Xroi(2,3)=SIZE_IMG(1)-1;
end
if Xroi(1,1)>SIZE_IMG(2)-2
    Xroi(1,1)=SIZE_IMG(2)-2;
    Xroi(1,2)=SIZE_IMG(2)-2;
end
if Xroi(2,1)>SIZE_IMG(1)-2
    Xroi(2,1)=SIZE_IMG(1)-2;
    Xroi(2,4)=SIZE_IMG(1)-2;
end
if Xroi(1,1)<=1
    Xroi(1,1)=1;
    Xroi(1,2)=1;
end
if Xroi(2,1)<=1
    Xroi(2,1)=1;
    Xroi(2,4)=1;
end
if Xroi(1,3)<=2
    Xroi(1,3)=2;
    Xroi(1,4)=2;
end
if Xroi(2,2)<=2
    Xroi(2,2)=2;
    Xroi(2,3)=2;
end
Xroi(1,5)=Xroi(1,1);
Xroi(2,5)=Xroi(2,1);

Xroiout=Xroi;

width=Xroi(1,3)-Xroi(1,1);
height=Xroi(2,3)-Xroi(2,1);

ROIout=[Xroi(1,1),Xroi(2,1),width,height];

end

```

13. Image Indexing Transformation (indextolinear.m)

```
function[X,Y]=indextolinear(CC,A,i,pase)
Y=[];
X=[];
for j=1:pase:A %I am analyzing only 1/50 of the pixels involved
index1=CC.PixelIdxList{1,i}(j,1);
x=floor(index1/CC.ImageSize(1));
y=floor(index1-x*CC.ImageSize(1));
Y=[Y,y];
X=[X,x];
end
```

B. MATLAB RIGID CLOUD

Below is provided the code used to create a 3D rigid cloud of points rotating and translating according to the user inputs. The points generated have been used to test the Epipolar function during the development phase

```
%Grompone Alessio 07/09/2014
function
[CameraPoints1, CameraPoints2, f1, XwCamera1, XwCamera2]=PointsCreatorBox(B
oxState1, BoxState2, f1)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Initialization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

rad=-45*(pi/180);%degrees to radians
%BoxState1=[0,0,20,0,0,0];%[X,Y,Z,phi,theta,psi] Initial condition
%BoxState2=[30,0,20,0,rad,0];%[X,Y,Z,phi,theta,psi] Final condition
SizeIMG=[800,600];
%f1=1; %Camera Focal correction 0.0038;(m)

FOVx=2*atan2(SizeIMG(1)/2,f1);
FOVy=2*atan2(SizeIMG(2)/2,f1);
FOV=[FOVx,FOVy];

%intrinsic parameters
S=[SizeIMG(1) 0 SizeIMG(1)/2;
  0 SizeIMG(2) SizeIMG(2)/2;
  0 0 1];
FI=[f1 0 0;
  0 f1 0;
  0 0 1];
PI=[1 0 0 0;
  0 1 0 0;
  0 0 1 0];
Xw1=zeros(4,1);
```

```

CameraPoints1=[];
CameraPoints2=[];
Xw3D=[];
X3D=[];
XwCamera1=[];
XwCamera2=[];
X_2Dcamera1=[];
X_2Dcamera2=[];

%The reference system has X along the camera, Z towards the camera view
and
%y towards down in the image plane

CameraState1=[0,0,0,0,0,0];%[X,Y,Z,phi,theta,psi]
CameraState2=[0,0,0,0,0,0];

CameraOnOff=1; %shows the relative 3D camera position in the plots
(remove=0)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Projection of 20 points of an object on two cameras planes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Body Points in the inertial frame
%the order of the box points is important for plotting purposes 1234 on
first
%plane 5678 on second plane

%Boxes
% Box1=[-3,3,3;3,3,3;3,-3,3;-3,-3,3;-3,3,-3;3,3,-3;3,-3,-3;-3,-3,-3];
% Box2=[-3,3,3;3,3,3;3,-3,3;-3,-3,3;-3,3,-3;3,3,-3;3,-3,-3;-3,-3,-
3]./2.5;
%Irregular shapes
% Box1=[-2,3,3;3,1,3;5,-3,2;-3,-3,3;-3,3,-3;4,3,-3;2,-3,-3;-1,-3,-3];
% Box2=[-3,4,3;3,3,2;3,-3,3;-3,-3,3;-3,3,-3;3,5,-3;3,-2,-3;-3,-4,-
3]./2.5;

%Three Boxes
Box1=[-2,5,3;5,5,3;5,-2,3;-2,-2,3;-2,5,-3;5,5,-3;5,-2,-3;-2,-2,-3];
Box2=[-3,3,3;3,3,3;3,-3,3;-3,-3,3;-3,3,-3;3,3,-3;3,-3,-3;-3,-3,-
3]./2.5;
Box3=[-1,4,5;5,5,2;5,-5,5;-5,-5,6;-6,6,-6;6,5,-6;6,-2,-6;-4,-4,-
4]./2.5;

%planar grid
z=10;
Box=zeros(z,3);
a = 1;
b = 1.2;
a1=-1;
b1=1;

```

```

for i=1:z
    r1 = (b1-a1).*rand + a1;
    Box(i,1)=r1;
    r2 = (b1-a1).*rand + a1;
    Box(i,2)=r2;
    r = (b-a).*rand + a;
    Box(i,3)=r;
end

%Box=[Box1;Box2;Box3];
%Box=rand(300,3)*10;
%Box Attitude and Translation in frame 1
phil_body=BoxState1(4);
thetal_body=BoxState1(5);
psil_body=BoxState1(6);
%Body Rotation and Translation
RcBZ1=[cos(psil_body) -sin(psil_body) 0;sin(psil_body) cos(psil_body)
0; 0 0 1];
RcBY1=[cos(thetal_body) 0 sin(thetal_body); 0 1 0; -sin(thetal_body) 0
cos(thetal_body)];
RcBX1=[1 0 0; 0 cos(phil_body) -sin(phil_body);0 sin(phil_body)
cos(phil_body); ];
Rbody1=RcBX1*RcBY1*RcBZ1;
Tbody1=[BoxState1(1);BoxState1(2);BoxState1(3)];
%Box Attitude and Translation in frame 2
phi2_body=BoxState2(4);
theta2_body=BoxState2(5);
psi2_body=BoxState2(6);
%Body Rotation and Translation
RcBZ2=[cos(psi2_body) -sin(psi2_body) 0;sin(psi2_body) cos(psi2_body)
0; 0 0 1];
RcBY2=[cos(theta2_body) 0 sin(theta2_body); 0 1 0; -sin(theta2_body) 0
cos(theta2_body)];
RcBX2=[1 0 0; 0 cos(phi2_body) -sin(phi2_body);0 sin(phi2_body)
cos(phi2_body); ];
Rbody2=RcBX2*RcBY2*RcBZ2;
Tbody2=[BoxState2(1);BoxState2(2);BoxState2(3)];
m=size(Box,1);
for i=1:m
    Box1_frame1(:,i)=Rbody1*Box(i,:)' +Tbody1;
    Box1_frame2(:,i)=Rbody2*Box(i,:)' +Tbody2;
end

%% Frame 1

PointsMatrix1=Box1_frame1;
PointsMatrix2=Box1_frame2;
n=size(PointsMatrix1,2);
n=n(1);%Number of Body Points

phil=CameraState1(4);
thetal=CameraState1(5);
psil=CameraState1(6);

```

```

phi2=CameraState2(4);
theta2=CameraState2(5);
psi2=CameraState2(6);

%Camera Rotations

RcZ1=[cos(psi1) -sin(psi1) 0;sin(psi1) cos(psi1) 0; 0 0 1];
RcY1=[cos(theta1) 0 sin(theta1); 0 1 0; -sin(theta1) 0 cos(theta1)];
RcX1=[1 0 0; 0 cos(phi1) -sin(phi1);0 sin(phi1) cos(phi1)];
R1=RcX1*RcY1*RcZ1;

RcZ2=[cos(psi2) -sin(psi2) 0;sin(psi2) cos(psi2) 0; 0 0 1];
RcY2=[cos(theta2) 0 sin(theta2); 0 1 0; -sin(theta2) 0 cos(theta2)];
RcX2=[1 0 0; 0 cos(phi2) -sin(phi2);0 sin(phi2) cos(phi2)];
R2=RcX2*RcY2*RcZ2;

T1=CameraState1(1:3)';
T2=CameraState2(1:3)';%need to correct because the rotation is around
the f

for i=1:n %n
Xw1=[PointsMatrix1(:,i);1]; %homogeneous position vector
Xw2=[PointsMatrix2(:,i);1]; %homogeneous position vector
ginv1=[R1' -R1'*T1; 0 0 0 1];
Xc1=(FI*PI*ginv1*Xw1); %Point in Camera1 Reference (X,Y,Z)
ginv2=[R2' -R2'*T2; 0 0 0 1];
Xc2=(FI*PI*ginv2*Xw2); %Point in Camera2 Reference (X,Y,Z)

%Xw3D=[Xw3D,Xw1];
XwCamera1=[XwCamera1,Xc1]; %Point 3D Position in Camera 1 frame
XwCamera2=[XwCamera2,Xc2]; %Point 3D Position in Camera 2 frame
X1=[Xc1(1);Xc1(2)]*(f1/Xc1(3));
X2=[Xc2(1);Xc2(2)]*(f1/Xc2(3));
CameraPoints1=[CameraPoints1,X1]; %2D image points Z Scaled
CameraPoints2=[CameraPoints2,X2]; %2D image points Z Scaled
end

CameraDirection1=[0,0,0;0,0,f1];
CameraDirection2=[0,0,0;0,0,f1];

%% Plotting box1

[ x ] = boxplots3D( Box1_frame1,CameraDirection1,0);
[ x ] = boxplots3D( Box1_frame2,CameraDirection2,0);

% %plot 3D points frame 1
[ x ] = boxplots3D( XwCamera1,CameraDirection1,1);

% %plot 3D points frame 2
[ x ] = boxplots3D( XwCamera2,CameraDirection2,1);

% %plot 2D Image frame 1

```

```
[ x ] = boxplots2D( CameraPoints1,FOV,f1);  
  
% %plot 2D Image frame 2  
[ x ] = boxplots2D( CameraPoints2,FOV,f1);  
save('box.mat')  
  
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- [1] NASA, “On-orbit satellite servicing study project report,” October 2010. [Online]. Available: http://ssco.gsfc.nasa.gov/servicing_study.html
- [2] J. Liang and O. Ma, “Angular velocity tracking for satellite rendezvous and docking,” *Acta Astronautica*, vol. 69, no. 11–12, pp. 1019–1028, 2011.
- [3] A. M. Long, M. G. Richards, and D. E. Hasting, “On-orbit servicing: a new value proposition for satellite design and operation,” *Journal of Spacecraft and Rockets*, vol. 44, no. 4, pp. 964–976, 2007.
- [4] S. Dubowsky, “Advanced methods for the dynamic control of high performance robotic devices and manipulators with potential for application in space”, NASA-CR-181061, 1987.
- [5] S. Segal, P. Gurfil, and K. Shahid, “In-orbit tracking of resident space objects: a comparison of monocular and stereoscopic vision,” *IEEE Transactions Aerospace and Electronic Systems*, vol. 50, no. 1, pp. 676,688, January 2014.
- [6] A. Flores-Abad, O. Ma, K. Pham, and S. Ulrich, “A review of space robotics technologies for on-orbit servicing,” *Progress in Aerospace Science*, vol. 68, pp. 1–26, 2014.
- [7] D. Zimpfer, P. Kachmar, and S. Tuohy, “Autonomous rendezvous, capture and in-space assembly: past, present and future,” in *Proc. 1st Space Exploration Conference: Continuing the Voyage of Discovery AIAA*, Orlando, Florida, 2005.
- [8] T. Kasai, M. Oda, and T. Suzuki, “Results of the ETS-7 mission - rendezvous docking and space robotics experiments,” NASDA, ETS-VII project team, Sengen, Tsukuba-shi, Ibaraki-ken, Japan, 1999.
- [9] R. Howard, A. Heaton, R. Pinson, and C. Carrington, “Orbital Express advanced video guidance sensor,” in *Proc. IEEE Aerospace Conference, Inst. of Electrical and Electronics Engineers*, Piscataway, NJ, 2008.
- [10] P. Jasiobedzki, M. Greenspan, and G. Roth, “Pose determination and tracking for autonomous satellite capture,” in *Proc. 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space*, Canadian Space Agency, St-Hubert, Quebec, Canada, 2001.

- [11] D. Fourie, B. E. Tweddle, S. Ulrich, and A. Saenz-Otero, "Flight results of vision-based navigation for autonomous spacecraft inspection of unknown objects," *Journal of Spacecraft and Rockets*, vol. 51, no. 6, pp. 2016–2026, 2014.
- [12] L. Song, Z. Li, and X. Ma, "Autonomous rendezvous and docking of an unknown tumbling space target with a monocular camera," in *Proc. IEEE Chinese Guidance, Navigation and Control Conference*, Yantai, China, 2014.
- [13] S. Segal, A. Carmi, and P. Gurfil, "Vision-based relative state estimation of non-cooperative spacecraft under modeling uncertainty," in *Proc. IEEE Aerospace Conference*, 2011.
- [14] J. P. Jumper, "Counterspace operations," Rep. 2–2.1, United State Air Force, 2004.
- [15] N. Li, Vision based trajectory tracking of space debris in close proximity via integrated estimation and control, M.S thesis, University of Central Florida, Orlando, Florida, 2011.
- [16] M. P. Wilkins, A. Pfeffer, P. W. Schumacher, and M. K. Jah, "Towards an artificial space object taxonomy," Applied Defense Solutions, Columbia, MD, 2013.
- [17] N. W. Oumer and G. Panin, "3D point tracking and pose estimation of a space object using stereo images," in *Proc. 21st International Conference on Pattern Recognition*, Tsukuba, Japan, 2012.
- [18] N. W. Oumer and G. Panin, "Tracking and pose estimation of non-cooperative satellite for on-orbit servicing," in *Proc. International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Turin, Italy, 2012.
- [19] M. W. Walker and L. Shao, "Estimating 3-d location parameters using dual number quaternions," *CVGIP: Image Understanding*, vol. 54, no. 3, pp. 358–367, 1991.
- [20] H. Benninghoff, F. Rems, and T. Boge, "Development and hardware-in-the-loop test of a guidance, navigation and control system for on-orbit servicing," *Acta Astronautica*, vol. 102, pp. 67–80, 2014.
- [21] F. Yu, Z. He, B. Qiao, and X. Yu, "Stereo-vision-based relative pose estimation for the rendezvous and docking of noncooperative satellites," *Mathematical Problems in Engineering*, vol. 2014, p. 12 pages, 2014.

- [22] J. C. Russ, *Image Processing Handbook*, Boca Raton, FL: CRC Taylor & Francis Group, 2006.
- [23] J. R. Jensen and L. Kalmesh, "Introductory digital image processing: a remote sensing perspective," *Geocarto International*, vol. 2, no. 1, pp. 65–65, 1987.
- [24] A. Robertson, T. Corazzini, and J. P. How, "Formation sensing and control technologies for a separated spacecraft interferometer," in *Proceedings of the American Control Conference*, Philadelphia, PA, 1998.
- [25] JAXA, "Engineering test satellite VII "KIKU-7" (ETS-VII)," Japan Aerospace Exploration Agency, [Online]. Available: <http://global.jaxa.jp>. [Accessed 10/05/2014].
- [26] C. J. Dennehy, "Relative navigation light detection and ranging (LIDAR) sensor development test objective performance verification," National Aeronautics and Space Administration (NASA), Hampton, Virginia, 2013.
- [27] X. Cao, F. Su, H. Sun and G. Xu, "Space debris observation via space-based ISAR," in *Proc. International Conference on Microwave and Millimeter Wave Technology*, 2007.
- [28] X. Fu, G. Liu and M. Gao, "Overview of orbital debris detection using spaceborne radar," in *Proc. IEEE Conference on Industrial Electronics and Applications*, 2008.
- [29] Y. Arimoto, J. Uchida and A. Semerok, "Space debris detection using laser communications demonstration equipment," in *IEEE Aerospace Conference Proceedings*, 2000.
- [30] M. Mokuno, I. Kawano and T. Suzuki, "In orbit demonstration of rendezvous laser radar for unmanned autonomous rendezvous docking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 2, pp. 617–626 , 2004.
- [31] J. A. Christian and S. Cryan, "A survey of LIDAR technology and its use in spacecraft relative navigation," in *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2008.
- [32] D. Pinard, S. Reynaud, P. Delpy, and S. E. Strandmoe, "Accurate and autonomous navigation for the ATV," *Aerospace Science and Technology*, vol. 11, pp. 490–498, 2007.

- [33] K. Shahid and G. Okouneva, "Intelligent LIDAR scanning region selection for satellite pose estimation," *Computer Vision and Image Understanding*, vol. 107, pp. 203–209, 2007.
- [34] M. Nimelman, J. Tripp, G. Bailak, and J. Bolger, "Spaceborne scanning lidar system (SSLS)," *Proc. SPIE 5798, Spaceborne Sensors II*, vol. 73, pp. 73–82, 2005.
- [35] S. Ruel, T. Luu, and A. Berube, "Space shuttle testing of the TriDAR 3D Rendezvous and Docking Sensor," *Journal of Field Robotics*, vol. 29, no. 4, pp. 535–553, 2012.
- [36] C. English, S. Zhu, C. Smith, S. Ruel, and I. Christie, "TriDAR: a hybrid sensor for exploring the complementary nature of triangulation and LIDAR technologies," in *Proc. 8th International Symposium Artificial Intelligence, Robotics and Automation in Space*, Munich, Germany, 2005.
- [37] B. E. Tweddle, "Computer vision based navigation for spacecraft proximity operations," M.S. thesis, Massachusetts Institute of Technology, 2010.
- [38] G. Zhang, Z. Wang, J. Du, T. Wang, and Z. Jiang, "A generalized visual aid system for teleoperation applied to satellite servicing," *International Journal of Advanced Robotic Systems*, vol. 11, no. 28, pp. 1–7, 2013.
- [39] R. T. Haward, A. F. Heaton, R. M. Pinson, C. L. Carrington, J. E. Lee, T. C. Bryan, B. A. Robertson, S. H. Spencer, and J. E. Johnson, "The advanced video guidance sensor: Orbital Express and the next generation," *Proc. Of Space Technology and Applications International Forum*, Albuquerque, New Mexico, 2008
- [40] A. Deslauriers, C. English, C. Bennett, P. Iles, R. Taylor, and A. Montpool, "3rd inspection for the shuttle return flight," *SPIE, Spaceborne Sensors III*, vol. 6220, p. 62200H, 2006.
- [41] Y. Gao, C. Spiteri, M.-T. Pham, and S. Al-Milli, "A survey on recent object detection techniques useful for monocular vision-based planetary terrain classification," *Robotics and Autonomous Systems*, vol. 62, pp. 151–167, 2014.
- [42] A. Shaukat, C. Spiteri, Y. Gao, S. Al-Milli, and A. Bajpai, "Quasi-thematic feature detection and tracking for future rover long-distance autonomous navigation," in *Proc. ESA Conference on Advanced Space Technologies in Robotics and Automation*, ASTRA, Noordwijk, Netherlands, 20–13.

- [43] K. Duncan and S. Sarkar, "Saliency in images and video: a brief survey," *IET Computer Vision*, vol. 6, no. 6, pp. 514–523, 2012.
- [44] F. Kennedy, "Orbital Express space operation architecture," DARPA Tactical Tecnology Office, [Online]. Available: archive.darpa.mil/orbitalexpress/index.html. [Accessed 01/15/2015].
- [45] D. M. Camp, "Evaluation of object detection algorithms for ship detection in the visible spectrum," M.S. thesis Naval Postgraduate School, Monterey, California, 2013.
- [46] "MATLAB Documentation," MathWorks, [Online]. Available: <http://www.mathworks.com/help/>. [Accessed 01/15/2015].
- [47] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [48] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.
- [49] Y. Liu, B. Tian, S. Chen, F. Zhu, and K. Wang, "A survey of vision-based vehicle detection and tracking techniques in ITS," in *IEEE International Conference on Vehicular Electronics and Safety*, Dongguan, China, 2013.
- [50] S. Y. Chen, "Kalman filter for robot vision: a survey," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4409–4420, 2012.
- [51] E. Wan, R. Van Der Merwe, and A. T. Nelson, "Dual estimation and the unscented transformation," *NIPS*, pp. 666–672, 1999.
- [52] R. Wiseman, "ISS Cygnus time-lapse," 2014. [Online]. Available: <https://vine.co/u/977976778226286592>. [Accessed 10/05/2014].
- [53] Y. Ma, S. Soatto, J. Kosecka and S. S. Sastry, *An Invitation To 3-D Vision: From Images To Geometric Models*, New York, NY: Springer-Verlag, 2003.
- [54] M. Brown, R. Szeliski, and S. Winder, "Multi image matching using multiscale oriented patches," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, CA, 2005.

- [55] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, CA, USA, 2005.
- [56] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, CVPR1994, IEEE Computer Society Conference on*, Seattle, WA, 1994.
- [57] C. Tomasi and T. Kanade, "Shape and motion from image streams: a factorization method—part 3, detection and tracking of point features, technical report CMU-CS-91-132," Carnegie Mellon University, Pittsburgh, PA, 1991.
- [58] "Blender, 2.73a release," Blender Foundation, 2015. [Online]. Available: <http://www.blender.org/>. [Accessed 01/15/2015].
- [59] R. Bevilacqua, A. P. Caprari, J. Hall, and M. Romano, "Laboratory experimentation of multiple spacecraft autonomous assembly," in *Proc. AIAA Guidance, Navigation and Control Conference*, Chicago, Illinois, 2009.
- [60] M. Ciarcià, A. Grompone and M. Romano, "A near-optical guidance for cooperative docking maneuvers," *Acta Astronautica*, vol. 102, pp. 367–377, 2014.
- [61] M. Romano, D. A. Friedman, and T. J. Shay, "Laboratory experimentation of autonomous spacecraft approach and docking to a collaborative target," *Journal of Spacecraft and Rockets*, vol. 44, no. 1, pp. 164–173, 2007.
- [62] R. Bevilacqua, J. S. Hall, J. Horning, and M. Romano, "Ad-hoc wireless networking and shared computation for autonomous multirobot systems," *Journal of Aerospace Computing, Information, and Communication*, vol. 6, pp. 328–352, 2009.
- [63] C. Luigini and M. Romano, "A ballistic-pendulum test stand to characterize small cold-gas thruster nozzles," *Acta Astronautica*, vol. 64, no. 5, pp. 615–625, 2009.
- [64] "PointGrey, Bumblebee XB3 1394b," Point Grey Research, 2015. [Online]. Available: <http://www.ptgrey.com>. [Accessed 01/15/2015].
- [65] "KVH, DSP-3000 Fiber Optic Gyro," KVH Industries, 2015. [Online]. Available: <http://www.kvh.com>. [Accessed 01/15/2015].
- [66] "Hokuyo laser scanner," Hokuyo, 2015. [Online]. Available: <https://www.hokuyo-aut.jp>. [Accessed 01/15/2015].

- [67] “Leap Motion controller,” Leap Motion, 2015. [Online]. Available: <https://www.leapmotion.com/>. [Accessed 01/15/2015].

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California