2008

Evaluation of machine vision techniques for use within flight control systems

Marco Mammarella
West Virginia University

EVALUATION OF MACHINE VISION TECHNIQUES FOR USE WITHIN FLIGHT CONTROL SYSTEMS

Marco Mammarella

Dissertation submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in Aerospace Engineering

Marcello Napolitano, Ph.D., Chair
Giampiero Campa, Ph.D.
Jacky C. Prucz, Ph.D.
Bojan Cukic, Ph.D.
Xin Li, Ph.D.

Department of Mechanical and Aerospace Engineering
Morgantown, West Virginia
2008

Keyword: Machine Vision, Aerial Refueling, Collision Identification, Point Matching, Pose Estimation, Sensor Fusion, Extended Kalman Filter, Optical Flow, Ideal Flow.

ABSTRACT

Evaluation of Machine Vision Techniques for use within Flight Control Systems

Marco Mammarella

In this thesis, two of the main technical limitations for a massive deployment of Unmanned Aerial Vehicle (UAV) have been considered.

The Aerial Refueling problem is analyzed in the first section. A solution based on the integration of 'conventional' GPS/INS and Machine Vision sensor is proposed with the purpose of measuring the relative distance between a refueling tanker and UAV. In this effort, comparisons between Point Matching (PM) algorithms and Pose Estimation (PE) algorithms have been developed in order to improve the performance of the Machine Vision sensor. A method of integration based on Extended Kalman Filter (EKF) between GPS/INS and Machine Vision system is also developed with the goal of reducing the tracking error in the 'pre-contact' to contact and refueling phases.

In the second section of the thesis the issue of Collision Identification (CI) is addressed. A proposed solution consists on the use of Optical Flow (OF) algorithms for the detection of possible collisions in the range of vision of a single camera. The effort includes a study of the performance of different Optical Flow algorithms in different scenarios as well as a method to compute the ideal optical flow with the aim of evaluating the algorithms. An analysis on the suitability for a future real time implementation is also performed for all the analyzed algorithms.

Results of the tests show that the Machine Vision technology can be used to improve the performance in the Aerial Refueling problem. In the Collision Identification problem, the Machine Vision has to be integrated with standard sensors in order to be used inside the Flight Control System.

ACKNOWLEDGEMENTS

I want to express my gratitude to Prof. Napolitano for giving me the opportunity of studying in this graduate program always trusting in me.

Special thanks go to Dr. Campa, which during these years worked with me every day with patience and enthusiasm. It was a pleasure for me learns from the best researcher I ever seen and it is inestimable what he taught to me.

Appreciations go to the committee members Dr. Prucz, Dr. Cukic and Dr. Li for their help in the writing of this effort.

One gigantic thank goes to Sofia for staying with me and loving me in these years that I consider the best period of my life. Finally now, we can rejoin and start a new section of our life, I was waiting for this moment for a long time.

A special acknowlegment for my family that even if they were far, they tried to stay as close as possible when I need them.

Other acknowlegments go to Segio, Jason and Josh for making the working environment more pleasant and Ale, Fabio and Michelangelo for taking care of me when Sofia went back to Spain these last months.

TABLE OF CONTENTS

1 Introduction 1
2 The Machine Vision based Aerial Refueling Problem and Simulation 11
2.1 The MV-based AR Problem 14
2.1.1 Reference frames and Notation 14
2.1.2 Geometric Formulation of the AR Problem 15
2.1.3 Receptacle-3DW-center vector 16
2.2 Aircraft, Boom and Turbulence modeling 17
2.2.1 Modeling of the tanker and UAV aircraft 17
2.2.2 Modeling of the boom 19
2.2.3 Modeling of the atmospheric turbulence and wake effects 21
2.3 Virtual Reality Scenery and Image Acquisition 21
2.4 The Feature Extraction algorithm 23
2.5 Point Matching Problem 24
2.5.1 Point Matching Algorithm # 1 - Mutual Nearest Point (MNP) 25
2.5.2 Point Matching Algorithm # 2 - Maximum Clique Detection (MCD) 28
2.5.2.1 Graph definition 29
2.5.2.2 Maximum Clique Detection Algorithm 30
2.6 Pose estimation algorithms 31
2.6.1 The GLSDC algorithm 31
2.6.2 The LHM algorithm 33
2.7 The Sensor Fusion system 35
2.8 UAV Docking Control Laws 37
2.9 The Reference Path generation system 39
2.10 Comparative Study between Point Matching Algorithms 41
2.10.1 Computational effort 41
2.10.2 Virtual Image analysis 43
2.10.3 Real Image Analysis 46
2.10.4 LHM Estimation Errors 48
2.11 Comparative Study between Pose Estimation Algorithms 50
2.11.1 Speed performance 51
2.11.2 Accuracy (Difference between true and estimated ${}^{C}T_{T}$ values) 51
2.11.3 Robustness 54
2.11.3.1 Noise addition in the corners position with correct point matching 54
2.11.3.2 Robustness to point matching errors 57
2.11.3.3 Robustness to errors in initial conditions 57
2.11.3.4 Error propagation analysis 58
2.11.4 Tracking error analysis 66
2.12 Sensor Fusion system based on Extended Kalman Filter 67
2.12.1 Sensor modeling 67
2.12.1.1 Modeling of the MV Sensor 67
2.12.1.2 Modeling of the INS Sensor 67
2.12.1.3 Modeling of the Pressure, Nose probe, Gyro, and Heading Sensors 69
2.12.1.4 Modeling of the GPS Position Sensor 70
2.12.2 Sensors Fusion Using EKF 71

2.12.2.1 EKF Background Theory 71
2.12.2.2 Sensors fusion using EKF 74
2.12.3 Performance Analysis 77
2.12.4 Robustness Analysis 82
3 The Collision Identification Problem and Simulation 84
3.1 Collision Identification Simulation 86
3.1.1 The Virtual Reality Environment 87
3.2 The Optical Flow 88
3.2.1 "Gradient" Methods 90
3.2.2 Phase Methods 92
3.2.3 Matching techniques 92
3.2.4 Feature-based methods 93
3.3 Derivation of the Ideal Optical Flow 94
3.3.1 Calculation of the points relative to the ground 96
3.4 Optical Flow Comparison in Simple Motion 97
3.4.1 Rotating Disk experiments 98
3.4.1.1 Overall Angular Velocity Error: 98
3.4.1.2 Angular and Magnitude Errors w.r.t the Ideal Flow: 100
3.4.1.3 Results of the test 100
3.4.2 Sliding Cart Experiments 103
3.4.2.1 Overall Velocity Error: 104
3.4.2.2 Angular and Magnitude Errors w.r.t the Ideal Flow: 104
3.4.2.3 Results of the test 105
3.4.3 Forward Translation Experiments 107
3.4.3.1 Overall Velocity Error: 108
3.4.3.2 Angular and Magnitude Errors w.r.t the Ideal Flow: 108
3.4.3.3 Results of the test 109
3.5 Optical Flow Comparison in Complex Motion 112
3.5.1 Virtual Images Analysis 112
3.5.1.1 Overall Velocity Error 112
3.5.1.2 Angular and Magnitude Errors w.r.t the Ideal Flow 114
3.5.1.3 Results of the test 115
3.5.2 Real Images Analysis with Data from the Sensors 119
3.5.2.1 Overall Velocity Error 121
3.5.2.2 Angular and Magnitude Errors w.r.t the Ideal Flow 121
3.5.2.3 Results of the test 121
3.6 Computational Requirements Analysis 126
3.7 Adaptation of the SIFT algorithm for Real-Time purpose 127
3.7.1 Comparison between the two SIFT implementations 128
4 Conclusion 131
5 Appendix A 134
6 References 140

ACRONYMS

AAR: Autonomous Aerial Refueling

AFRL: Air Force Research Laboratory

AGV: Automated Ground Vehicle

AI: Artificial Intelligence

AMT: All Moving Tip

AR: Aerial Refueling

BYU: Brigham Young University

CG: Center of Gravity

CI: Collision Identification

CMOS: Complementary Metal-Oxide Semiconductor

CNIC: Correctly Not Identified Corner

CPU: Central Processing Unit

CRF: Camera Reference Frame

DC: Direct Current

DGPS: Differential Global Positioning System

DLL: Dynamic Linked Library

DOD: Department Of Defense

DOG: Difference of Gaussian

EKF: Extended Kalman Filter

ERF: Earth Reference Frame

ERR: Error

FAA: Federal Aviation Agency

FCS: Flight Control System

FE: Feature Extraction

Gb: Giga Byte

GHz: Giga Hertz

GLSDC: Gaussian Least Square Differantial Correction

GPS: Global Positioning System

GUI: Graphic User Interface

HSU: Hue Saturation and Value

IMU: Inertial Measurement Unit

INIC: Incorrectly Not Identified Corners

INS: Inertial Navigation System

IR: Infrared

IUAS: Individual Unmanned Air Scouts

LEF: Leading Edge Flap

LHM: Lu, Hager and Mjolsness

LKF: Linear Kalman Filter

LQR: Linear Quadratic Regulator

MAE: Mechanical and Aerospace

MAV: Micro Aerial Vehicle

MCD: Maximum Clique Detection

MNP: Mutual Nearest Point

MV: Machine Vision

NASA: National Aeronautic and Space Administration

NP: Non-deterministic Polynomial time

OEM: Original Equipment Manufacturer

OF: Optical Flow

Pc: Personal Computer

PE: Pose Estimation

PF: Pitch Flap

PGPS: Precision Global Positioning System

PM: Point Matching

PRF: ERF having the *x*-axis aligned with the planar component of the tanker velocity

PSD: Power Spectral Density

RAM: Random Access Memory

RF: Reference Frame

RGB: Red Green Blue

RPV: Remotely Piloted Vehicle

SAD: Sum of Absolute Difference

SIFT: Scale Invariant Feature Transform

SSD: Spoiler Slot Deflector

STD: Standard Deviation

TCAS: Traffic Collision Avoidance System

TEE: Trailing Edge Elevon

TRF: Tanker Reference Frame

UAV: Unmanned Aerial Vehicle

URF: UAV Reference Frame

VR: Virtual Reality

VRE: Virtual Reality Environment

VRML: Virtual Reality Modeling Language

VRT: Virtual Reality Toolbox

WGN: White Gaussian Noise

WV: West Virginia

WVU: West Virginia University

LIST OF SYMBOLS

ω: Angular velocity

α: Attack angle (rad/sec)

δ: Displacement

μ: Mean value

θ: Pitch angle (rad)

φ: Roll angle (rad)

β: Side Slip angle (rad/sec)

σ: Standard deviation

ψ: Yaw angle (rad)

a: area of a corner

B: Boom point

C_D: Drag coefficient

C_L: Lift coefficient

C_l: Yawing moment coefficient

C_m: Rolling moment coefficient

C_n: Side Force coefficient

C_Y: Pitching moment coefficient

f: Focal length

h: hue value of a corner

I: Grey level intensity of image

L: Lagrangian

O: Complexity of an algorithm

O: Origin of reference frame

P: Point

p: Roll rate (rad/sec)

q: Pitch rate (rad/sec)

R: Receptacle point

r: Yaw rate (rad/sec)

T: Homogeneous transformation matrix

u: Horizontal component in an image

V: Linear Velocity

v: Vertical component in an image

x: First coordinate of a point

y: Second first coordinate of a point

z: Third coordinate of a point

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have been referred to in many ways: RPVs (Remotely Piloted Vehicles), drones, robot planes, and "pilot-less" aircraft are a few such names. Most often called UAVs, they are defined by the Department of Defense (DOD) as powered, aerial vehicles that do not carry a human operator, use aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expendable or recoverable, and can carry a lethal or non-lethal payload [1]. Ballistic or semi-ballistic vehicles, cruise missiles, and artillery projectiles are not considered UAVs by the DOD definition. UAVs differ from RPVs in that some UAVs can fly autonomously. UAVs are either described as a single air vehicle (with associated surveillance sensors) or a UAV system [2], which usually consists of three to six air vehicles, a ground control station, and support equipment. UAVs are thought to offer two main advantages over manned aircraft; they are arguably cheaper to produce, and they eliminate the risk to a pilot's life. Furthermore, for those certain missions that require a very small aircraft, only a UAV can be deployed because there is no equivalent manned system small enough for the job [1]. Recently, UAVs have achieved a wide consideration thanks to recent technology improvements. For example advanced video surveillance and sensing systems can now be placed on UAVs, something that was unthinkable instead just a few years ago.

UAVs range from the size of an insect to that of a commercial airliner. DOD currently possesses five major UAVs: the Air Force's Predator and Global Hawk [3] the Navy and Marine Corps's Pioneer [4], and the Army's Hunter and Shadow [5].

Figure 1: Predator UAV (left) and Pioneer UAV (right)

The non-military use of UAVs is expected to increase in the future as technologies evolve that allow the safe, reliable flight of UAVs over populated areas. One emerging application is the use of less sophisticated UAVs as aerial camera platforms for the movie making and entertainment industries [6]. A similar market is growing rapidly in the television news reporting. As demand in these markets grows, aircraft such as the Individual Unmanned Air Scouts (IUAS) [7] will become a more desirable aerial platform than less-capable hobbyist aircraft, as safety, reliability, ease-of-use, and rapid deployment become important priorities. Additional roles for UAVs in the near future will include homeland security and medical re-supply. For Example the Coast Guard and Border Patrol – parts of the newly formed Department of Homeland Security – already have plans to deploy UAVs to watch coastal waters, patrol the nation's borders, and protect major oil and gas pipelines. Congressional support, currently, exists for using UAVs for border security [8]. During a Senate Armed Services Committee hearing on homeland defense, it was stated that although it would not be appropriate or constitutional for the military to patrol the border, domestic agencies using UAVs could carry out this mission. On the medical side, UAVs such as the Army's Shadow have been studied as delivery vehicles for critical medical supplies needed on the battlefield [5]. Not

all of these new applications have been approved. In fact, UAV advocates state that in order for UAVs to take an active role in homeland security, Federal Aviation Administration (FAA) regulations concerning the use of UAVs will have to change. The Coast Guard will most likely take the lead in resolving UAV airspace issues with the FAA. The National Aeronautics and Space Administration (NASA) and the UAV industry will also be working with the FAA on the issue, as they are joining forces in an initiative to achieve routine UAV operations in the national airspace within a few years [9].

Despite the increasing exploits of UAVs in specific missions, the use of these particular systems is limited to low-medium range operations and to "collisions – free" scenarios. Normally, UAVs are smaller then manned aircraft and this characteristic limits the fuel capacity and payload. In addiction, UAVs needs more sensors and complicated on-board computers in order to perform a specific task. For these reasons, detailed studies are needed to improve the capabilities of the systems[10][11] [13][15][33][43].

In this thesis, two of the major technological limitations preventing a larger scale UAV deployment have been addressed. In order to solve the Aerial Refueling problem for UAVs a highly accurate sensor is needed when the refueling tanker and the UAV are in proximity. The solution proposed by Junkins and Schaub [69] called VisNav is a sensor able to generate highly accurate measurements with an update rate of up to 100 Hz makes it a sensor that can be used for autonomous refueling operations. VisNav is capable of producing six degree of freedom relative navigation information. VisNav calculates line of sight vectors to each beacon using voltage measurements from a light sensitive diode. A controller on the receiver coordinates the sequence and timing of the

active beacon array through a wireless data link. This guarantees correspondence between each measurement and the known position of the beacon on the target. The VisNav sensor in combination with Inertial Navigation System (INS) and Differential Global Position System (DGPS) were used in the research proposed by Valasek and his research group [11], [70], [71] in order to achieve the solution of the Aerial Refueling problem.

The U.S Air Force Research Laboratory (AFRL) is actually starting the second phase of its Automated Aerial Refueling (AAR) program to demonstrate the capability to refuel unmanned aircraft in flight [81]. AAR Phase II is scheduled to run from late fiscal 2008-2012, at an estimated cost of $49 million, the program will build on Phase I, which demonstrated a proof-of-concept AAR capability using a single-channel relative navigation system based on the Precision Global Positioning System (PGPS). In Phase I, a Calspan-operated Learjet acting as a surrogate UAV demonstrated the entire refueling operation, but did not make contact with the boom of the KC-135 tanker. Phase II will culminate in a wet contact between the tanker and a manned surrogate. Phase II will involve two spirals. In Spiral 1, according to AFRL pre-solicitation documents [3], the selected integrator will take the government-furnished, single-channel AAR system from Phase I and develop a multi-channel PGPS relative navigation system and automated flight control system (FCS). The Spiral 1 system will use dual-redundant Tactical Targeting Network Technology wideband data-links to connect the PGPS systems in the tanker and receiver. Using outputs from the relative navigation system, the redundant FCS will control the receiver aircraft during aerial refueling, including rendezvous with the tanker, station-keeping, repositioning, separation and contingencies. A version of the Spiral 1 system will be installed in a receptacle-equipped military aircraft, likely an F-16,

that will act as the surrogate UAV for flight-tests in fiscal 2010 and 2011. Under Spiral 2, meanwhile, the Phase II integrator will study a "sensor-augmented" AAR relative navigation system able to operate in environments where GPS is degraded or denied [3] [81].

In this thesis, the Aerial Refueling problem will be analyzed in the first section. A solution based on the integration of 'conventional' GPS/INS and Machine Vision sensor is proposed with the purpose of measuring the relative distance between a refueling tanker and UAV [12][26][32][40][55]. A set of control laws for the guidance between the "pre-contact" to "contact" and refueling phases was previously developed at WVU.

A first contribution of this section of the thesis is achieve desirable accuracy for the Machine Vision sensor through comparisons between two different Point Matching (PM) algorithms and two different Pose Estimation (PE). A second specific contribution is the development of new method of integration based on Extended Kalman Filter (EKF) [18] between GPS/INS and Machine Vision system. The EKF is introduced with the goal of reducing the tracking error in the 'pre-contact' to contact and refueling phases.

The Collision Avoidance problem is instead a more common problem that can be applied to all the type of vehicles and robotic applications. In the available literature many different scenarios have been analyzed. [72] proposed a scheme for collision avoidance for robotic arms based on infrared proximity sensor. In this case the slow relative motion and the possibility to arrive very close – compared with the distance between aerial vehicles - to the object that the robotic arm has to avoid encourage the development of many type of sensors able to recognize the proximity of the object. Examples of sensors that can be used in robotic applications for collision avoidance

purpose are ultrasonic sensors with a range up to 2.5 m, radar proximity sensors with a range from 4 cm to 5.5 m, inductive proximity sensors with a range up to 35 mm, and capacitive proximity sensors with a range up to 10 mm.

Many applications of collision avoidance have been developed for Automatic Ground Vehicle (AGV). [73] proposed an embedded multi sensors collision avoidance system for automotive application. The system includes sensors like video camera, ultrasonic sensors, a PC hardware computer, a CAN network and a dedicated software for signal and image processing, data fusion and AI expert system. In [74] the author tries to summarize the sensors used in automotive industries (Figure 2) and sensors used in aeronautic applications.

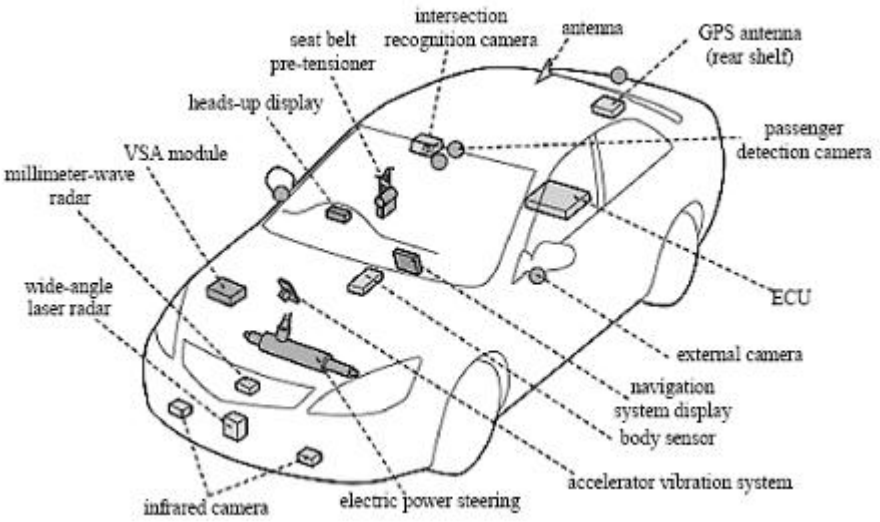


Figure 2: The Honda ASV-3 uses cameras and radar to detect obstacles and approaching vehicles and assists in steering and braking

The sensors analyzed are: SkyWatch HP and Proteus. SkyWatch HP is an active surveillance traffic advisory system. It interrogates the transponders of aircraft in the vicinity, calculates their distance, bearing, relative altitude and speed of closure. Proteus developed at NASA Dryden Flight Research Center where the autopilot and satellite

communications systems on the aircraft allowed ground-based staff to control it “over the horizon”.

In 2002, a plane was equipped with Skywatch HP to detect other aircraft with transponders so that the ground pilot could change the aircraft’s direction or altitude to avoid collision, and in 2003 a small 35 Ghz radar system and an infrared optical sensor were added to detect aircraft without transponders. The optical system detects a potential colliding object and the radar measures its range and closing speed. [75] presented a new radar sensor for collision avoidance purpose. The receiver front-end module is based on a six-port phase/frequency discriminator and it can measure the relative velocity and position of the identified object. The radar is able to identify possible collision up to 75 m. In [76] a second radar with lower frequency and bigger rang (6.4 km) is presented and tested for low altitude UAVs.

[78] presented a collision avoidance method based on the Traffic alert and Collision Avoidance System II (TCAS II) for UAVs. The TCAS is a method based on a transponder used in civil aviation in order to avoid midair collision. Anyway, the system does not have any information about terrain or building and it presents high costs ($25,000 – $150,000). Being based on transponder, the TCAS only works with cooperative aircraft; the authors propose the augmentation of the sensors with radar, lidar (laser radar) and optical sensors for non-cooperative aircraft.

[45],[77] presented a collision avoidance method based on a combination of sensors. The implementation required a Ka-band pulsed radar as the main sensor, and four electro-optical cameras – two in the visible spectrum and two in the infrared spectrum- as aiding sensors. Real time sensors fusion merges all the information provided

by the sensors. The cameras implement Optical Flow techniques in order to find possible collisions. Specifically, in [77] the Lucas-Kanade [48] and Horn and Schunck's [49] methods were tested. The Lucas-Kanade method was considered more suitable for implementation but after some analysis was observed that the limitations of an Optical Flow-only based approach made it not adequate to replace the main system architecture of the anti-collision system that includes a radar.

[79] described a Miniature Aerial Vehicle (MAV) equipped with obstacle and terrain avoidance system. Figure 3 shows the MAV and the sensors used, the round hole on the right and the large hole on the belly are the optic-flow sensors. The square hole in the center is the laser range, and the other two round holes are for electro-optical cameras.

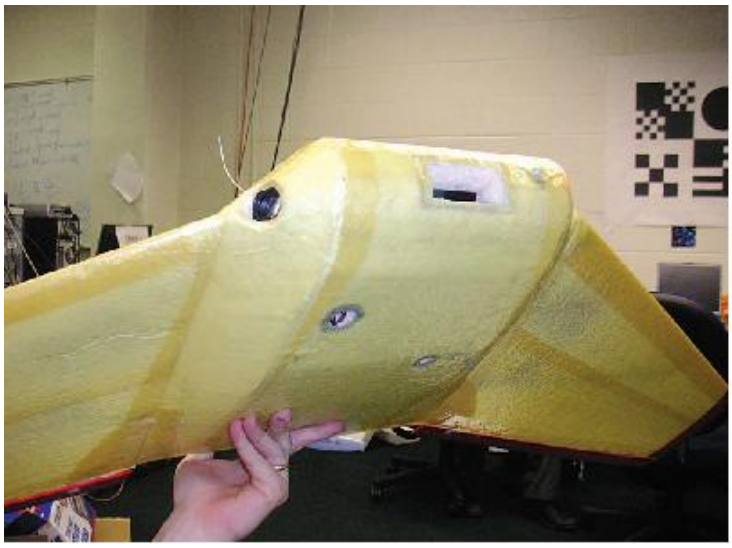


Figure 3: MAV developed at BYU with collision avoidance system

The optical flow sensors are implemented through a lens to an Agilent ADNS-2610 optical mouse sensor. The ADNS-2610 has a small form factor, measuring only 10 mm by 12.5 mm and runs at 1,500 frames/s. It requires a light intensity of at least 80

mW/m^2 at a wavelength of 639 nm or 100 mW/m^2 at a wavelength of 875 nm. The ADNS-2610 measures the flow of features across an 18 × 18 pixel complementary metal-oxide semiconductor (CMOS) imager. The range on which the sensor is able to detect a motion is not provided but the velocity - on which the aircraft was tested (V=13 m/s) - permits to perform an anti-collision maneuver even if the object is close.

In the second section of this thesis the problem of Collision Identification (CI) is analyzed. The proposed solution consists on the use of Optical Flow (OF) [47], [48], [49], [50], [51], [52], [53], [57] algorithms for the detection of possible collisions in the range of vision of a single camera.

The main contribution of this section is the development of performance metrics for the comparison of the Optical Flow algorithms. Particularly, a formula able to compute the Ideal Flow in simple and complex scenarios permits to define a standard approach for the direct comparison of the velocity vector field. On the other hand, the inversion of the formula permits to extract linear and angular velocities in order to compare the Optical Flow algorithms using an indirect approach.

Another original contribution of this work is a quantitative comparison 9 different OF algorithms using image sequences from different real-world experiments involving composed (i.e. translational and rotational) 3D motion of rigid objects. In particular the last experiment relied on a flight test experiment involving an autonomous F-22 aircraft model designed, built and instrumented by the flight control group of the Mechanical and Aerospace Department at West Virginia University (WVU).

The thesis reports a complete study of the performance of the different Optical Flow algorithms in different scenarios as well as a method to compute the ideal optical

flow useful in order to test the Optical Flow algorithms. An analysis on the suitability for a future real time implementation is also performed for all the analyzed algorithms. The problem of extracting the information from the Optical Flow algorithms, the problem of estimating of the “no-flight zones” and the relative commands that have to be provided to the flight control system will not be considered in this effort.

2 THE MACHINE VISION BASED AERIAL REFUELING PROBLEM AND SIMULATION

One of the biggest current limitations of Unmanned Aerial Vehicles (UAVs) is their lack of aerial refueling (AR) capabilities. There are currently two hardware configurations used for aerial refueling for manned aircraft. The first configuration is used by the US Air Force and features a refueling boom maneuvered by a boom operator to connect with the fuel receptacle of the aircraft to be refueled. The second configuration is used by the US Navy and features a flexible hose with an aerodynamically stabilized perforated cone - known as the 'Probe and Drogue' system. The effort described in this thesis is relative to the US Air Force refueling boom system with the general goal of extending the use of this system to the refueling of UAV's. For this purpose, a key issue is represented by the need of accurate measurement of the 'tanker-UAV' relative position and orientation from the 'pre-contact' to the 'contact' position and during the refueling. Although sensors based on laser, infrared radar, and GPS technologies are suitable for autonomous docking [10], there might be limitations associated with their use. For example, the use of UAV GPS signals might not always be possible since the GPS signals may be distorted by the tanker airframe. Therefore, the use of Machine Vision (MV) technology has been proposed in addition - or as an alternative - to these technologies [11][12][33]. A MV-based system has been investigated for close proximity operations of aerospace vehicles [13] and for the navigation of UAV's [14].

The control objective is to guide the UAV within a defined 3D Window (3DW) below the tanker where the boom operator can then manually proceed to the docking of

the refueling boom followed by the refueling phase. A MV approach assumes the availability of a digital camera installed on the UAV providing the images of the target (that is, the refueling tanker), which are then processed to solve a pose estimation problem, leading to the real-time estimates of the relative position and orientation vectors. These vectors are used for the purpose of guiding the UAV from a "pre-contact" to a "contact" position. Once the UAV reaches the contact position, the boom operator takes over and manually proceeds to the refueling operation.

A simulation environment for the UAV Aerial Refueling has been developed and will be summarized in this effort. This environment features detailed mathematical models for the tanker, the UAV, the refueling boom, the wake effects, the atmospheric turbulence, and the sensors noise. The simulation interacts with a Virtual Reality (VR) environment by moving visual 3D models of the aircraft in a virtual world and by acquiring a stream of images from the environment. Images are then processed by a MV sensor block, which includes algorithms for Feature Extraction (FE), Point Matching (PM), and Pose Estimation (PE). The position and orientation information coming from the MV and GPS sensors are then fused in order to be used by the UAV control laws to guide the aircraft during the docking maneuver and to maintain the UAV within the 3D window during the refueling phase. The general block diagram of the MV scheme is shown in Figure 4.

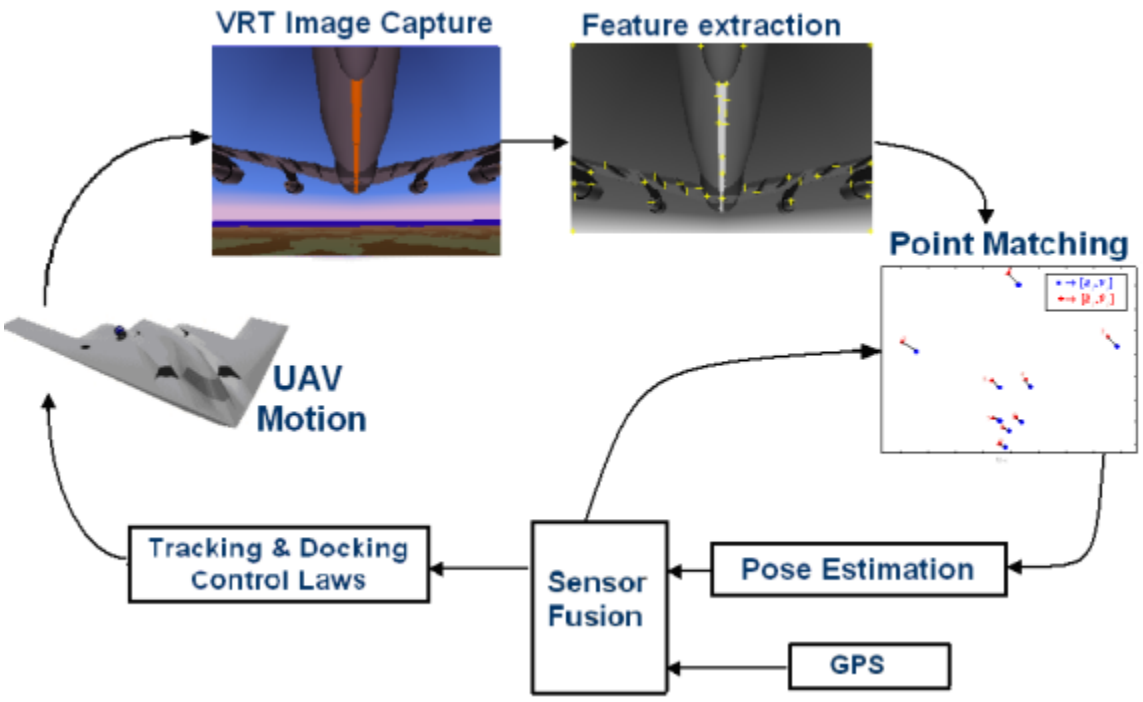


Figure 4– Block diagram of the proposed MV scheme for UAV AR

The AR problem is formally described in the next section. Next, the modeling of the tanker, UAV, boom, wake effects and turbulence are summarized. The following sections are dedicated to the description of the main components of the Machine Vision system, respectively the Feature Extraction (FE), the Point Matching (PM), and the Pose Estimation (PE) algorithms. A basic sensor fusion and the path generation methods, as well as the tracking and docking control laws are then considered in the subsequent sections. Finally, the results are presented with the purpose of improving the existing simulation. Particularly, detailed analyses of two PM and two PE algorithms are provided. The analyses are useful in order to find a final arrangement in order to reduce the error for the Machine Vision system A new sensor fusion that combine GPS/INS and Machine Vision system based on EKF has been developed and compared with the previous sensor fusion technique.

2.1 The MV-based AR Problem

A block diagram of the MV-based AR problem is shown in Figure 5

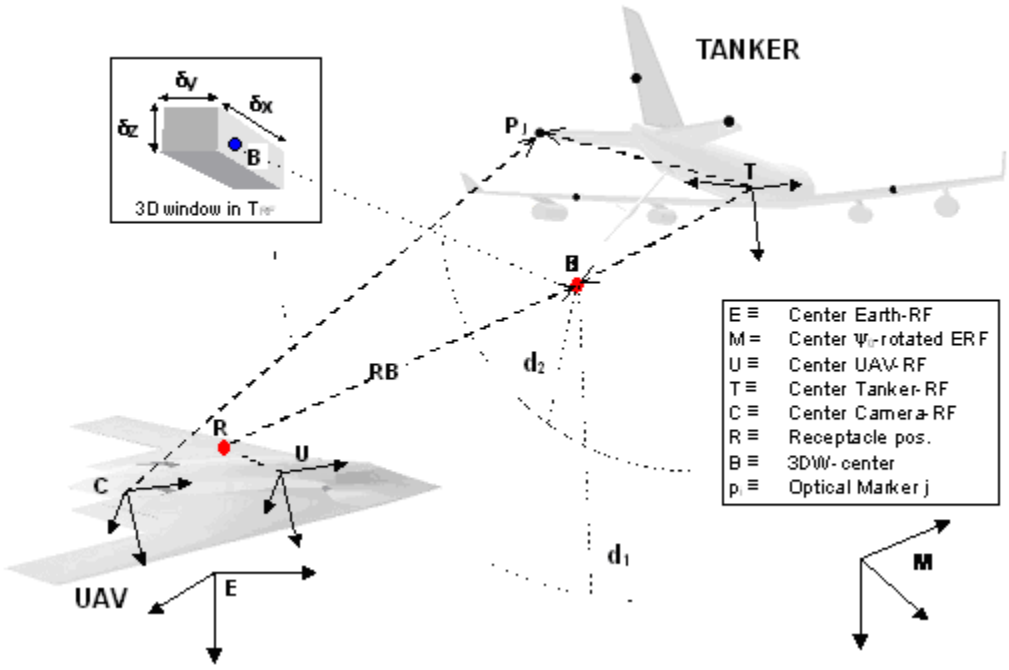


Figure 5– Reference Frames for the AR Problem

2.1.1 Reference frames and Notation

The study of the AR problem requires the definition of the following Reference Frames (RFs):

- ERF, or *E*: earth-fixed reference frame.
- PRF, or *P*: earth-fixed reference frame having the *x* axis aligned with the planar component of the tanker velocity vector.
- TRF or *T*: body-fixed tanker reference frame located at the tanker center of gravity (CG).
- URF or *U*: body-fixed UAV reference frame located at the UAV CG.

- CRF or *C*: body-fixed UAV camera reference frame.

Within this study geometric points are expressed using homogeneous (4D) coordinates and are indicated with a capital letter and a left superscript denoting the associated reference frame. For example, a point *P* expressed in the *F* reference frame has coordinates ${}^{F}P = [x,y,z,1]^{T}$, where the right '*T*' superscript indicates transposition. Vectors are defined as difference between points; therefore their 4th coordinate is always '0'. Also, vectors are denoted by two uppercase letters, indicating the two points at the extremes of the vector. For example, ${}^{E}BR = {}^{E}B - {}^{E}R$ is the vector from the point R to the point B expressed in the Earth Reference Frame. The transformation matrices are (4 x 4) matrices relating points and vectors expressed in an initial reference frame to points and vectors expressed in a final reference frame. They are denoted with a capital *T* with a right subscript indicating the "initial" reference frame and a left superscript indicating the "final" reference frame. For example, the matrix ${}^{E}T_{T}$ represents the homogeneous transformation matrix that transforms a vector/point expressed in TRF to a vector/point expressed in ERF.

2.1.2 Geometric Formulation of the AR Problem

The objective is to guide the UAV such that its fuel receptacle (point *R* in Figure 5) is "transferred" to the center of a 3-dimensional window (3DW, also called "Refueling Box") under the tanker (point *B* in Figure 5). It is assumed that the boom operator can take control of the refueling operations once the UAV fuel receptacle reaches and remains within this 3DW. It should be emphasized that point *B* is fixed within the TRF; also, the dimensions of the 3DW $\delta x, \delta y, \delta z$ are known design parameters. It is

additionally assumed that the tanker and the UAV can share a short-range data communication link during the docking maneuver. Furthermore, the UAV is assumed to be equipped with a digital camera along with an on-board computer hosting the MV algorithms acquiring the images of the tanker. Finally, the 2-D image plane of the MV is defined as the '*y-z*' plane of the CRF.

2.1.3 Receptacle-3DW-center vector

The reliability of the AR docking maneuver is strongly dependent on the accuracy of the measurement of the vector ${}^{P}BR$, that is the distance vector between the UAV fuel receptacle and the center of the 3D refueling window, expressed within the PRF:

$$ {}^{P}BR = {}^{P}T_{T}\,{}^{T}B - {}^{P}T_{U}\,{}^{U}R = {}^{P}T_{T}\,{}^{T}B - {}^{P}T_{T}\,{}^{T}T_{C}\,{}^{C}T_{U}\,{}^{U}R \quad (1) $$

In the above relationships both ${}^{U}R$ and ${}^{T}B$ are known and constant parameters since the fuel receptacle (point *R*) and the 3DW center (point *B*) are located at fixed and known positions with respect to the UAV and tanker frames respectively. The transformation matrix ${}^{C}T_{U}$ represents the position and orientation of the CRF with respect to the URF; therefore, it is also known and assumed to be constant. The transformation matrix ${}^{P}T_{T}$ represents the position and orientation of the tanker respect to PRF, which are measured on the tanker and broadcasted to the UAV through the data communication link. In particular, if the sideslip angle β of the tanker is negligible then ${}^{P}T_{T}$ only depends on the tanker roll and pitch angles. Finally, ${}^{T}T_{C}$, is the inverse of ${}^{C}T_{T}$, which can be evaluated either "directly"- that is using the relative position and orientation information provided by the MV system- or "indirectly"- that is by using the formula ${}^{C}T_{T} = {}^{C}T_{U}\left({}^{E}T_{U}\right)^{-1}\,{}^{E}T_{T}$, where the matrices ${}^{E}T_{U}$ and ${}^{E}T_{T}$ can be evaluated using

information from the position (GPS) and orientation (gyros) sensors of the tanker and UAV respectively.

2.2 Aircraft, Boom and Turbulence modeling

2.2.1 Modeling of the tanker and UAV aircraft

The non-linear aircraft models of the UAV and tanker have been developed using the conventional modeling procedures and conventions outlined in [15][16]. Specifically, a non-linear model of a Boeing 747 aircraft with linearized aerodynamics was used for the modeling of the tanker. A similar non-linear model was used for the modeling of the UAV. The selected UAV dynamics is relative to a concept aircraft – represented in Figure 6 – known as "ICE-101" [17]. A conventional state variable modeling procedure was used for both aircraft, leading to the state vector:

$$\left[V,\alpha,\beta,p,q,r,\psi,\theta,\varphi,x,y,z\right]^{T} \tag{2}$$

where V,α,β represent the aircraft velocity in the stability axes; p,q,r are the components of the angular velocity in the body reference frame while $\psi,\theta,\varphi,x,y,z$ represent the aircraft orientation and position with respect to ERF [16].

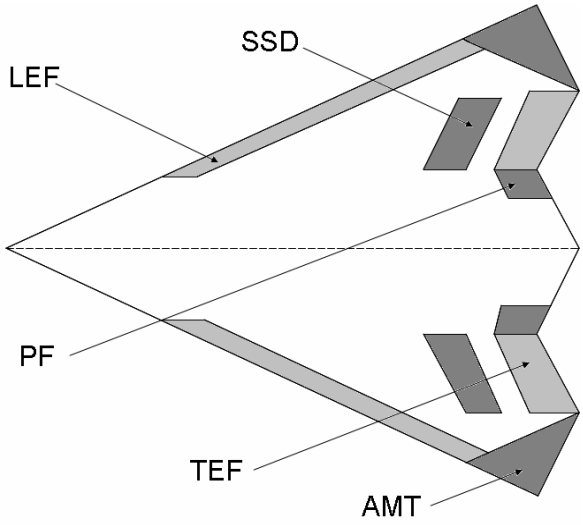


Figure 6: ICE 101 model

First order responses have been used for the modeling of the actuators of the different control surfaces. The ICE101 features 10 control surfaces [17]:

$$U_{1-5} = \left[\delta_{\text{THROTTLE}}, \delta_{\text{AMT\_R}}, \delta_{\text{AMT\_L}}, \delta_{\text{TEE\_R}}, \delta_{\text{TEE\_L}} \right]^T \tag{3}$$

$$U_{6-10} = \left[\delta_{\text{LEF\_L}}, \delta_{\text{LEF\_R}}, \delta_{\text{PF}}, \delta_{\text{SSD\_L}}, \delta_{\text{SSD\_R}} \right]^T \tag{4}$$

where AMT stands for All Moving Tip, LEF for Leading Edge Flap, PF for Pitch Flap, SSD for Spoiler Slot Deflector and TEE for Trailing Edge Elevon, and they are represented in Figure 6. Steady state rectilinear conditions (Mach = 0.65, H = 6,000 m) are assumed for the refueling. The tanker autopilot system is designed using LQR-based control laws [18]. The design of the UAV control laws is outlined in one of the following sections.

2.2.2 Modeling of the boom

A detailed mathematical model of the boom was developed to provide a realistic simulation from the boom operator point of view. A joystick block for boom maneuvering was also added to the simulation environment.

The dynamic model of the boom has been derived using the Lagrange method [19],[20]:

$$\frac{d}{dt}\frac{\partial L(q,\dot{q})}{\partial \dot{q}_i} - \frac{\partial L(q,\dot{q})}{\partial q_i} = F_i, \quad i = 1,...,n \tag{5}$$

where $L(q,\dot{q}) = T(q,\dot{q}) - U(q)$ is the Lagrangian, that is the difference between the boom kinetic and potential energy, and q is the vector of Lagrangian coordinates, defining the position and orientation of the boom elements. Since the inertial and gravitational forces are included in the left-hand side of (5), F_i only represents the active forces (wind and control forces) acting on the boom. The boom was modeled as a system consisting of two rigid elements, as shown in Figure 7. The first element is connected to the tanker point ${}^{E}P$ by two revolute joints allowing vertical and lateral relative rotations (θ_4 and θ_5). The second element is connected to the first by a prismatic allowing the extension d_6. Thus, the boom has 6 degrees of freedom, that is, the first three components of ${}^{E}P = \left[d_1, d_2, d_3, 1\right]^T$, the rotations θ_4 and θ_5, and the extension d_6, leading to $q = \left[d_1, d_2, d_3, \theta_4, \theta_5, d_6\right]^T$. Note that the point ${}^{E}P$ can be expressed as ${}^{E}P = {}^{E}T_T \, {}^{T}P$ where ${}^{T}P$ is known and constant. The consequent Denavit - Hartenberg parameter table for the boom system is obtained and shown in Table 1. The Denavit – Hartenberg table permits to easily define the transformation matrix from the tanker to the end-effector of the boom going through all the joints of the robotic arm. Knowing the height a_i, the azimuth angle

α_i, the distance d_i and the twist θ_i of all the joints, it is possible find the transformation matrix from the joint *n* to the joint *n-1*. Multiplying the rotation matrix on the axis *z* with variable θ_n:, the translation matrix on the axis –z in the variable a_n, the translation matrix on the axis – x in the variable d_n and the rotation matrix on the axis – x on the variable α_n it can be obtained the matrix ${}^{n}T_{n-1.}$ Multiplying all the matrix for each joint of the robotic arm it is possible to extract the kinematics equations of the arm.

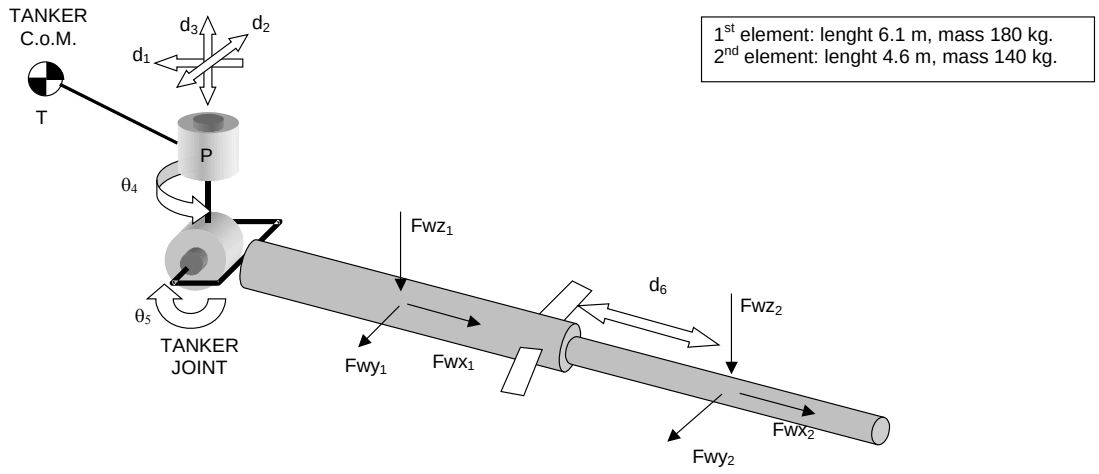


Figure 7–Model of the "Refueling Boom"

	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	d_1	0
2	0	$\pi/2$	d_2	$\pi/2$
3	0	$\pi/2$	d_3	0
4	0	$-\pi/2$	0	θ_4
5	0	$-\pi/2$	0	θ_5
6	0	$\pi/2$	d_6	$\pi/2$

Table 1: Denavit-Hartenberg parameter for the Boom system

2.2.3 Modeling of the atmospheric turbulence and wake effects

The atmospheric turbulence on the probe system and on both tanker and the UAV aircraft has been modeled using the Dryden wind turbulence model [21] at light/moderate conditions. The wake effects of the tanker on the UAV have been modeled through the interpolation from a large amount of experimental data [22],[23] as perturbations to the aerodynamic coefficients $C_D, C_L, C_m, C_Y, C_l, C_n$ for the UAV aerodynamic forces and moments.

2.3 Virtual Reality Scenery and Image Acquisition

The simulation outputs were linked to a Virtual Reality Toolbox® (VRT) [24] interface to provide typical scenarios associated with the refueling maneuvers. Such interface allows the positions of the UAV, tanker, and boom within the simulation to drive the position and orientation of the associated objects in the Virtual World (VW). The VW consisted in a VRML file [25] including visual models of the landscape, tanker, UAV, and boom. Several objects including the tanker, the landscape and different parts of the boom were originally modeled using 3D Studio and later exported to VRML. Every object was scaled according to its real dimensions. A B747 model was re-scaled to match the size of a KC-135 tanker – as shown in Figure 8 - while a B2 model was rescaled to match the size of the ICE 101 aircraft. Eight different viewpoints were made available to the user, including the view from the UAV camera and the view from the boom operator. The latter allows the simulator to be used as a boom station simulator if so desired. The simulation main scheme also features a number of graphic user interface (GUI) menus allowing the user to set a number of simulation parameters including:

- initial conditions of the AR maneuver;
- level of atmospheric turbulence;
- location of the camera on the UAV and its orientation within the UAV body frame;
- location of the fuel receptacle on the UAV;

From the VW, images of the tanker as seen from the UAV camera are continuously acquired and processed during the simulation. Specifically, after the images are acquired, they are scaled and processed by a corner detection algorithm. The corner detection algorithm finds the 2D coordinates on the image plane of the points associated with specific physical corners and/or features of the tanker.

Figure 8– Graphic Model of the tanker

2.4 The Feature Extraction algorithm

The performances of two specific feature extraction algorithms for the detection of corners in the image were compared in a previous effort [26]. The Harris corner detector [27],[28] was selected for this study. This method is based on the analysis of the matrix of the intensity derivatives, also known as "Plessey operator" [27], which is defined as follows:

$$M = \begin{bmatrix} I_X{}^2 & I_{XY} \\ I_{YX} & I_Y{}^2 \end{bmatrix} \tag{6}$$

where I is the gray level intensity of the image while I_X, I_Y, I_{XY}, I_{YX} are its directional derivatives. The directional derivatives are determined by convolving the image by a kernel of the correspondent derivative of a Gaussian. If at a certain point the eigenvalues of the matrix M take on large values, then a small change in any direction will cause a substantial change in the gray level. This indicates that the point is a corner. A "cornerness" value C for each pixel of the image is calculated using:

$$C = \det(M) - k * Tr(M) \tag{7}$$

If the value of C exceeds a certain threshold, the associated pixel is declared a corner. The sensitiveness of the detector is proportional to the value of k. The generally used value of k is 0.04 [27]. The main drawback of the method is that the parameter k needs to be tuned manually. This drawback was overcome by a modified version of the Harris "cornerness" function proposed by Noble [28]:

$$C = \frac{\det(M)}{Tr(M) + \varepsilon} \tag{8}$$

The small constant ε is used to avoid a singular denominator in case of a rank zero auto-correlation matrix (*M*). In both Harris detector method [27] and its variation by Noble [19] a local maxima search is performed as a final step of the algorithm with the goal of maximizing the value of C for the selected corners.

2.5 Point Matching Problem

The subset $[\hat{u}_j, \hat{v}_j]$ is the 2D projection in the camera plane of the corners *P(j)* using the standard "pin-hole" projection model [29]. Once the subset $[\hat{u}_j, \hat{v}_j]$ is available, the problem of relating the points extracted from the camera measurements to the actual features on the tanker can be formalized in terms of matching the set of points $\{p_1, p_2, ..., p_m\}$ - where $p_j = [u_j, v_j]$ is the generic 'to be matched' point from the camera - to the set of points $\{\hat{p}_1, \hat{p}_2, ..., \hat{p}_n\}$, where $\hat{p}_j = [\hat{u}_j, \hat{v}_j]$ is the generic point obtained through projecting the known nominal corners in the camera plane. Since the two data sets represents the 2D projections of the same points on the same plane - at the previous time instant - a high degree of correlation between the two sets is expected. However, due to the relative motion between camera and tanker, as well as to the presence of different sources of system and measurement noise, a certain level of difference between the two point sets is always observed. Thus, a matching problem has to be defined and solved. In fact, simulation studies show that when the previous position estimation is used as an approximation of the current relative position, and the sampling rate of the MV system is set to 10 Hz. The relative movement between camera and tanker during one image and the next - together with the effect of measurement and system noise -

normally causes a 2D corner displacement on the image plane within 3% of the horizontal image range. While the spacing among the projected corners that have to be matched is usually greater than 10% of the horizontal image range. This minimizes the possibility that the PM algorithm could mistakenly assign a certain projected corner to the wrong detected corner.

A detailed technical literature describes a number of robust matching techniques between point sets [30]. The degree of similarity between two data sets is typically defined in terms of a cost function or a distance function derived on general principles as geometric proximity, rigidity, and exclusion [34]. The best matching is then evaluated as the result of on optimization process exploring the space of the potential solutions. Often, the problem can be set as a classical assignment problem, and therefore is solved using standard polynomial Network Flow algorithms. A definition of the point-matching problem as an assignment problem along with an extensive analysis of different matching algorithms was performed in [31]. In the current effort, two different matching algorithms are implemented; both algorithms solve the matching problem using a heuristic procedure [32]. The algorithms are reviewed in the sections below.

2.5.1 Point Matching Algorithm # 1 - Mutual Nearest Point (MNP)

This algorithm features a "mutual nearest point" technique to perform a point matching and then arranges the vector of matched corner coordinates in the format $G_{PM} = \left[u_1, v_1 \ldots u_n, v_n\right]$. If the k^{th} corner is not matched then an overflow value is entered in the position *2*k* and *2*k+1*. Let $\hat{P} = \left\{\hat{p}_1, \hat{p}_2, ..., \hat{p}_n\right\}$ denote the set of the *n* projected

corners, and let $P = \{p_1, p_2, ..., p_m\}$ denote the set of *m* detected corners. Each point p_j has the 4 coordinates $p_j = [u_j \quad v_j \quad a_j \quad h_j]$, where u_j and v_j are the coordinates of the corner j as described above, a_j is the "area" of the corner j, and h_j is the mean hue value [35] of the surrounding of the corner *j*. The hue value was calculated using the specific Matlab function 'rgb2hsv' which converts an image from the RGB format to the HSV (Hue, Saturation, and Value) format. Similarly, for the point $\hat{p}_j$ the 4 coordinates $\hat{p}_j = [\hat{u}_j \quad \hat{v}_j \quad \hat{a}_j \quad \hat{h}_j]$ are defined; however, in this case the values $\hat{a}_j$ and $\hat{h}_j$ – representing the area and the hue value – are constants.

The area is essentially an intrinsic geometric property of the object, and it is evaluated by analyzing a 5 x 5 matrix of pixels around the corner. In this matrix, the maximum and the minimum values are selected. The matrix is then converted to a logic form (zeros and ones) using as a threshold the half of the average distance between maximum and minimum. The area of the corner is then defined as the number of "ones". The hue value, that is the mean value of the hue in a 5 x 5 matrix around the corner, is instead a "color" information about the object which does not vary with the lightness. A detailed study showed that the range of variation of both the "area" and the hue value for a specified corner is limited and only dependent on the specific types of corner and image.

The point matching function creates a matrix *Err* (Table 2) of dimension *n* x *m*, whose entries are all the Euclidian distance between $\hat{P}$ and P. The function allows the definition of a maximum range of variation for each dimension; these ranges define a hypercube around each corner of the set $\hat{P}$. The distance actually is computed only if the

four-dimensional point pj lies into one of the hyper-cubes defined around each point of the set $\hat{P}$; otherwise, it is automatically set to infinity. A normalization is then performed for comparing different values among the four dimensions during the evaluation of the distance.

$$
\begin{aligned}
Err &= \begin{bmatrix} d(\hat{p}_1, p_1) & d(\hat{p}_1, p_2) & d(\hat{p}_1, p_3) & d(\hat{p}_1, p_4) \\ d(\hat{p}_2, p_1) & d(\hat{p}_2, p_2) & d(\hat{p}_2, p_3) & d(\hat{p}_2, p_4) \\ d(\hat{p}_3, p_1) & d(\hat{p}_3, p_2) & d(\hat{p}_3, p_3) & d(\hat{p}_3, p_4) \end{bmatrix} \\
MinC &= \begin{bmatrix} \min_x d(\hat{p}_x, p_1) & \min_x d(\hat{p}_x, p_2) & \min_x d(\hat{p}_x, p_2) & \min_x d(\hat{p}_x, p_2) \end{bmatrix} \\
Index &= \begin{bmatrix} idx(MinC_1) & idx(MinC_2) & idx(MinC_3) & idx(MinC_4) \end{bmatrix} \\
MinR &= \begin{bmatrix} \min_x d(\hat{p}_1, p_x) \\ \min_x d(\hat{p}_1, p_x) \\ \min_x d(\hat{p}_1, p_x) \end{bmatrix}
\end{aligned}
$$

Table 2 - Data structure used in the matching function

Next, the four dimensions have to be weighted before calculating the Euclidian distance between $\hat{P}$ and P. A detailed study showed that the best results are obtained when only the u and v components are equally weighted, while the weights for the area and hue parameter are set to zero. In other words, the area and hue dimensions are only used to define the hypercube around the corners of the set $\hat{P}$ but do not explicitly influence the distances in Table 2. In the table, the three vectors *MinR, MinC* and *Index* - with dimensions n, m and m respectively - are created (as shown in Table 2).

The minimum element of every column of *Err* is stored in the row vector *MinC* while the index of the row in which the function finds the minimum is stored in another row vector *Index*. The minimum element of every row of *Err* is instead stored in the column vector *MinR*. The position of the detected corner '*j*' in P is deemed "valid" if:

$$MinC[j] == MinR[Index[j]] \quad (9)$$

The detected corners satisfying the validity condition are assigned to their nearest projected corners. On the other side, the detected corners that do not satisfy the validity condition are discarded. In other words, the validity condition ensures that only one detected corner - among the set of detected corners that are closer to a certain projected corner than to other projected corners - is assigned to that projected corner. The other detected corners in the same set are not assigned to any other projected corners. The resulting algorithm has a computational complexity proportional of O(m*n); the method avoids the typical problems associated with a matching function that simply assigns the detected corners P to the nearest corners in $\hat{P}$ [32].

2.5.2 Point Matching Algorithm # 2 - Maximum Clique Detection (MCD)

In this approach, the matching problem is based on the criteria of 'Exclusion, Proximity, and Rigidity' [34]. In other words, the distance between corresponding points of the same set is considered in addition to the information derived from the distances between the corresponding points of the two matching sets. These concepts can be applied using a graph in which vertexes represent couples of potential matching and edges represent the compatibility between the couples of potential matches. The matching problem is then reduced to a Maximum Clique Detection (MCD) algorithm. The algorithm is explained with more details below.

2.5.2.1 Graph definition

The first step of the algorithm consists in the construction of the graph $G=\{V,E\}$ associated to the point matching problem, where:

V is the set of vertexes of G and represents the set of potential matching between one element P and one element of $\hat{P}$.

E is the set of edges of G and represents the compatibility between the couple of potential matching.

Proximity Principle: The vertexes V are determined based on the proximity principle. The association $\alpha_{ij} = (p_i, \hat{p}_j)$ is a vertex of G if and only if the distance between p_i e $\hat{p}_j$ is less than a defined threshold T_P.

Exclusion and Rigidity Principle: The set of edges E is determined based on the exclusion and rigidity principle. According to the exclusion principle in the graph G, couples of edges $e_{ij,ik} = (\alpha_{ij}, \alpha_{ik})$ and $e_{ji,ki} = (\alpha_{ji}, \alpha_{ki})$ cannot exist. Namely, one element of P cannot be associated to more than one of the elements of $\hat{P}$ and vice versa. According to the rigidity principle $e_{ij,hk} = (\alpha_{ij,}\alpha_{hk})$ is an edge of G if and only if:

$$\left| d(p_i, p_h) - d(\hat{p}_j, \hat{p}_k) \right| < T_R \tag{10}$$

where T_R is pre-defined threshold. Namely, the distances between points in P have to be similar to the distances between the corresponding points of $\hat{P}$.

The computational complexity for the construction of the graph G connected to the point matching problem is $O(m^2 \cdot n^2)$ where m is the number of points in the set P and n is the number of points of the set $\hat{P}$.

2.5.2.2 Maximum Clique Detection Algorithm

After the construction of the graph *G* has been performed, the feature matching is determined through the evaluation of the "*maximum clique*" of the graph [34]. The MC of a graph *G* is defined as the largest sub-graph where all the vertexes are connected with a single edge of *G*. In this study, the MC of *G* represents the maximum set of compatible associations between elements of *P* and elements of $\hat{P}$. Unfortunately, from the theory of computational complexity, the determination of the MC of *G* is known to be an NP-complete (Non-deterministic Polynomial time) problem. This means that its solution can be verified in polynomial time. Therefore, for large graphs, the algorithm is likely to be incompatible with the real-time constraint imposed by the AR problem. As generally happens, NP-complete problems are addressed by using heuristic algorithms in practice. The heuristic rule outlined in Table 3 has been implemented for the identification of a sub-optimal problem. This algorithm provides desirable computational performance when the graph is sparse.

$S = V$
$Q = \varnothing$
for each $\alpha \in V$ find his degree in G
while $S \neq \varnothing$
    choose a vertex α in S with maximum degree in G
    $N_g(\alpha)$ =set of vertexes adjacent to α in V
    $Q = Q \cup \{\alpha\}$
    $S = S \cap N_g(\alpha)$
return Q

Table 3 - Heuristic for determining a sub-optimal solution in the Maximum Clique Detection Algorithm

2.6 Pose estimation algorithms

Following the solution of the PM problem, the information in the set of points $\{p_1, p_2, ..., p_m\}$ must be used to derive the rigid transformation relating the CRF to the TRF. The literature offers a wide range of alternatives to solve this typical 2D to 3D correspondence problem. Many of the available methods are sensitive to parameters like the number of detected point or the length of the image sequence. Algorithms that rely on the estimation of the kinematics and structure of the rigid object for solving a recursive pose estimation problem have been presented in [36] and [37]. Within this study, two algorithms to solve the more specific problem of estimating relative position and orientation from a set of 2D to 3D point correspondences within a single image were implemented and compared.

2.6.1 The GLSDC algorithm

The Gaussian Least Squares Differential Correlation (GLSDC) algorithm [38] is based on the application of the Gauss-Newton method for the minimization of a non-linear cost function expressing the difference between estimated and detected corners positions. This algorithm was selected because it represents a class of algorithms that is still very widely used in photogrammetry. The GLSDC algorithm has a simple structure; additionally, it has already been used to solve pose estimation problems within simulation study for an autonomous aerial refueling setting, as described in [40]. Furthermore, its simple structure allows a straightforward real time implementation. Within the GLSDC

algorithm, at every sample time *k*, the matrix ${}^{C}T_{T}$ is expressed as a function of an *estimate* $\overline{X}(k)$ of the *unknown* vector *X(k):*

$$\overline{X}(k) = [{}^{c}\overline{x}_{T}, {}^{c}\overline{y}_{T}, {}^{c}\overline{z}_{T}, {}^{c}\overline{\psi}_{T}, {}^{c}\overline{\theta}_{T}, {}^{c}\overline{\varphi}_{T}]^{T} \tag{11}$$

Using $\overline{X}(k)$ to project of the corner '*j*' in the camera plane yields the following 2D coordinates:

$$\begin{bmatrix} \overline{u}_{j} \\ \overline{v}_{j} \end{bmatrix} = \frac{f}{{}^{c}\overline{x}_{p,j}} \begin{bmatrix} {}^{c}\overline{y}_{p,j} \\ {}^{c}\overline{z}_{p,j} \end{bmatrix} = g\left(f, {}^{C}T_{T}(\overline{X}(k)) \cdot {}^{T}P_{(j)}\right) \tag{12}$$

By rearranging the coordinates of all the projected corners, the following vector is obtained

$$G(\overline{X}(k)) = \left[\overline{u}_{1}, \overline{v}_{1}, ..., \overline{u}_{m}, \overline{v}_{m}\right] \tag{13}$$

At this point, the following MV estimation error can be defined at the time *k*:

$$\Delta G(k) = G_{PM}(k) - G(\overline{X}(k)) \tag{14}$$

where $G_{PM}(k)$ contains the coordinates of the points provided by the PM algorithm extracted from the camera:

$$G_{PM}(k) = \left[u_{1}, v_{1}, ..., u_{m}, v_{m}\right] \tag{15}$$

the GLSDC algorithm iteratively refines the initial value of $\overline{X}(k)$ by repeating the following steps for a number of iterations (with index *i*):

$$\overline{X}_{i+1}(k) = \overline{X}_{i}(k) + R_{i}^{-1}(k) A_{i}^{T}(k) W(k) \Delta G_{i}(k) \tag{16}$$

where

$$R_{i}(k) = A_{i}^{T}(k) W(k) A_{i}(k) \tag{17}$$

$$A_i(k) = \left. \frac{\partial G_i(k)}{\partial X} \right|_{X = \bar{X}_i(k)} \tag{18}$$

and *W(k)* is usually set to the (2m x 2m) covariance matrix of the estimation error.

The initial guess $\bar{X}_0(k)$ at *k* is the final estimation at the previous sample time *k*-1. The basic algorithm outlined in Eqs. (16)-(18) was designed to work with a fixed number of *m* corners. Simple modifications have been introduced for dealing with a time-varying number of corners. Specifically, at each time step, the nominal corners that are not visible are removed from the estimation process. This implies that at each time step, (13) is modified such that it includes only the appropriate number of rows. Thus, the dimensions and the values of *A* and *W* in (16)-(18) are adjusted accordingly.

2.6.2 The LHM algorithm

Lu, Hager and Mjolsness (LHM) [39] proposed an algorithm based on the minimization of an object-space collinearity error. Specifically, the 'observed, detected, and matched' point '*j*' $[u_j, v_j]$ corresponds to ${}^{C}P_{(j)} = {}^{C}T_T\left(X(k)\right){}^{T}P_{(j)}$. This is the known point ${}^{T}P_{(j)}$ reported in CRF using the homogeneous transformation matrix ${}^{C}T_T\left(X(k)\right)$ at the time instant *k*.

Let $h_j(k)$ be:

$$h_j(k) = \begin{bmatrix} u_j & v_j & 1 \end{bmatrix}^T \tag{19}$$

Similarly, let $\hat{h}_j(k) = \begin{bmatrix} \hat{u}_j & \hat{v}_j & 1 \end{bmatrix}^T$ represent the projection on the image plane of the point ${}^C\hat{P}_{(j)} = {}^C T_T(\bar{X}(k))^T P_{(j)}$ where $\bar{X}(k)$ is the estimation of $X(k)$ internal to the Pose Estimation algorithm – as defined in (11).

Then, the 'object-space collinearity error' vector e_j, at the time instant k, can be defined as follows:

$$e_j(k) = \left(I - V_j(k)\right) {}^C T_T(\bar{X}(k))^T P_{(j)} \tag{20}$$

where

$$V_j(k) = \begin{bmatrix} \dfrac{h_j(k)h_j^T(k)}{h_j^T(k)h_j(k)} & 0 \\ 0 & 1 \end{bmatrix} \tag{21}$$

which is the line of sight projection matrix that, when applied to a scene point, projects the point orthogonally to the line of sight defined by the image point $h_j(k)$ [39]. In other words, the object space collinearity error represents the difference between the point ${}^C\hat{P}_{(j)}$ and its projection on the line joining the origin of the camera frame and the point $h_j(k)$.

The pose estimation problem is then formulated as the problem of minimizing the sum of the squared errors:

$$E(\bar{X}(k)) = \sum_{j=1}^{m} \left\| e_j(k) \right\|^2 \tag{22}$$

The algorithm operates by iteratively improving an estimate of the rotation portion of the pose. Next, the algorithm estimates the associated translation only when a satisfactory estimate of the rotation is found. [39] shows that the LHM algorithm is globally convergent in the sense that it always converges to the rotation matrix that minimizes the collinearity error for any set of observed point and any initial rotation matrix. Furthermore, empirical results suggest that the algorithm is also very efficient and usually converges within 5 to 10 iterations starting from any range of initial conditions.

2.7 The Sensor Fusion system

Due to typical limitations of cameras performance, the MV system can provide reliable results only within a certain limited distance form the tanker. On the other hand, the GPS signal received by the UAV may be shadowed or distorted by the tanker airframe when the UAV is near or below the tanker. This could lead to losses in accuracy and reliability of the GPS-based UAV position measurement.

The purpose of the fusion filter is to ensure that at large UAV-tanker distances only the GPSs-based distance measurement $[{}^{C}x_{T}, {}^{C}y_{T}, {}^{C}z_{T}]_{(GPS)}$ is used for navigation and control purposes. On the other hand, as the UAV-tanker distance decreases, the GPS-based measurements are gradually replaced by the more accurate and reliable $[{}^{C}x_{T}, {}^{C}y_{T}, {}^{C}z_{T}]_{(MV)}$ distance measurements. Specifically, the distance measurement vector $[{}^{C}x_{T}, {}^{C}y_{T}, {}^{C}z_{T}]_{(F)}$ that is used at any time for tracking and docking purposes is a bounded linear combination of $[{}^{C}x_{T}, {}^{C}y_{T}, {}^{C}z_{T}]_{(GPS)}$ and $[{}^{C}x_{T}, {}^{C}y_{T}, {}^{C}z_{T}]_{(MV)}$ using the following relationships in terms of the relative distance d between the UAV and the tanker:

Large distances (GPS only):

$$if \quad d > d_1 \quad , \quad \begin{bmatrix} {}^{C}x_T \\ {}^{C}y_T \\ {}^{C}z_T \end{bmatrix}_{(F)} = \begin{bmatrix} {}^{C}x_T \\ {}^{C}y_T \\ {}^{C}z_T \end{bmatrix}_{(GPS)} \tag{23}$$

where d_1 is a user-defined constant. In the simulation d_1 has been set to 40m.

Note that the MV system should start yielding accurate measurements whenever the UAV–tanker relative distance is somewhat greater than d_1.

Intermediate distance (GPS to MV transition):

$$if \ d_1 \geq d > d_2 \ , \quad \begin{bmatrix} {}^{C}x_T \\ {}^{C}y_T \\ {}^{C}z_T \end{bmatrix}_{(F)} = \left(1 - \frac{d - d_1}{d_2 - d_1}\right) \begin{bmatrix} {}^{C}x_T \\ {}^{C}y_T \\ {}^{C}z_T \end{bmatrix}_{(GPS)} + \left(\frac{d - d_1}{d_2 - d_1}\right) \begin{bmatrix} {}^{C}x_T \\ {}^{C}y_T \\ {}^{C}z_T \end{bmatrix}_{(MV)} \tag{24}$$

where d_2 ($d_2<d_1$) is another user-defined constant. In the simulation d_2 has been set to 23m.

Small distance (MV only):

$$if \quad d \leq d_2 \quad , \quad \begin{bmatrix} {}^{C}x_T \\ {}^{C}y_T \\ {}^{C}z_T \end{bmatrix}_{(F)} = \begin{bmatrix} {}^{C}x_T \\ {}^{C}y_T \\ {}^{C}z_T \end{bmatrix}_{(MV)} \tag{25}$$

For simplicity purposes, and without any loss of generality, the relative distance d governing the transition has been considered coincident with the relative distance calculated from the GPS measurements:

$$d = norm([{}^{C}x_T, {}^{C}y_T, {}^{C}z_T]_{(GPS)}) \tag{26}$$

This corresponds to the assumption that the accuracy of the GPS measurement may not be enough to allow a smooth refueling maneuver; it is however, appropriate to determine whether the UAV is sufficiently near to the tanker for the MV measurements to be used. The resulting vector $[{}^{C}x_T, {}^{C}y_T, {}^{C}z_T]_{(F)}$ is then used along with the relative

orientation information from the attitude sensors of the tanker and UAV respectively to calculate the relative distance ${}^{P}BR$ between the fuel receptacle on the UAV and the center of the 3D window, as shown in eq.(1).

2.8 UAV Docking Control Laws

The receptacle position in PRF, that is ${}^{P}R$, and the UAV center of mass in ERF, that is ${}^{E}U$ are two equivalent ways to represent the UAV position information, since the following relationship applies:

$$ {}^{P}R = {}^{P}T_{E}(\psi_0)\,{}^{E}T_{U}(\psi,\theta,\varphi,{}^{E}U)\,{}^{U}R \tag{27} $$

and since ${}^{U}R$, the UAV Euler angles, and the tanker heading angle Ψ_0 are all known.

To include this additional information, an augmented state space model - with respect to the model outlined in (2) - was selected for the UAV:

$$ Z = \left[V,\alpha,\beta,p,q,r,\psi,\theta,\varphi,{}^{P}R,\ \int_{t_0}^{t} {}^{P}R\,dt \right]^{T} \tag{28} $$

In the above vector the last six states represent respectively the three components of the ${}^{P}R$ point (that is the UAV receptacle) in PRF, and their integral over time. The last 3 states were added to facilitate the synthesis of a controller capable of zero steady state tracking error.

As outlined in previous section, the ICE101 features 10 control surfaces [17]. Assuming that the tanker is flying at a straight and level flight conditions with a known velocity V_0 and heading angle Ψ_0, the center of the refueling window ${}^{P}B(t)$ is subjected to a rectilinear uniform motion, described by:

$$ {}^{P}B(t) = \begin{bmatrix} {}^{P}B_1(t_0) + V_0(t - t_0) & 0 & 0 \end{bmatrix}^T \tag{29} $$

where ${}^{P}B_1(t_0)$ is a known initial condition.

The following trajectory in the UAV state space:

$$ Z_{ref}(t) = \begin{bmatrix} V_0, \alpha_0, 0, 0, 0, 0, \psi_0, \theta_0, 0, {}^{P}B(t), \int_{t_0}^{t} {}^{P}B(t)\ dt \end{bmatrix}^T \tag{30} $$

represents a trim point for the first 9 UAV states. The reference input U_{ref} corresponding to the above reference trajectory was calculated using a Simulink trim utility. Since the objective of the UAV control laws is to guide the UAV so that ${}^{P}R$ (the fuel receptacle) is eventually "transferred" to the point ${}^{P}B$, it is reasonable to assume small perturbations from the flight condition in (30) during the refueling maneuver. Under this assumption, the UAV dynamics can be modeled as the linear system resulting from the linearization of the UAV equations about the reference trajectory in (30):

$$ \begin{aligned} \dot{\tilde{Z}} &= A\tilde{Z} + B\tilde{U} \\ \tilde{Y} &= C\tilde{Z} = \begin{bmatrix} {}^{P}RB, \int_{t_0}^{t} {}^{P}RB\ dt \end{bmatrix}^T \end{aligned} \tag{31} $$

where the "~" denotes deviation from the reference trajectory, the state space matrices A and B describe the dynamics of the resulting linear system, and C defines a "performance" output vector containing ${}^{P}RB$ and its integral over time.

The design of the UAV docking control laws was then performed using a Linear Quadratic Regulator (LQR) approach [18]. The resulting cost function is expressed as:

$$ J = \int_0^{\infty} \left(\tilde{Y}^T Q \tilde{Y} + \tilde{U}^T R \tilde{U} \right) dt \tag{32} $$

where the selected weighting matrices are given by:

$$ \begin{aligned} Q &= diag([10, 10, 10, 0.1, 0.001, 0.1]) \\ R &= diag([0.1, 1000, 1000, 1, 1000, 1000, 1000, 1000, 0.1, 0.1]) \end{aligned} \tag{33} $$

Accordingly, the LQR control law is given by:

$$\tilde{U} = -K \cdot \tilde{Z} \tag{34}$$

where the LQR matrix K is obtained by solving an Algebraic Riccati Equation [18]. Following the structure of the state vector, equation (34) can be decomposed into the following terms:

$$\tilde{U} = -K_d \cdot \tilde{Z}_{1-9} + K_p \, {}^{P}BR + K_i \int_{t_0}^{t} {}^{P}BR \, dt \tag{35}$$

where the derivative term K_d is applied to the first 9 element of the state, and the proportional and integral terms K_p and K_i are applied respectively to ${}^{P}BR$ (which is obtained as discussed in the previous sections) and its integral over time.

2.9 The Reference Path generation system

Once the AR "tracking & docking" scheme is activated, the UAV control system is tasked to generate a suitable sequence of feasible commands leading to a precise docking within a defined time. This cannot be achieved by directly using the control law in (35), which can only be used under the assumption of small deviations from the reference trajectory. In fact, when the ${}^{P}BR$ vector takes on large values (as it happens when the UAV is at the pre-contact position and the control system is activated) the proportional term in (35) will typically generate a large command, which can potentially drive the system outside the validity range of the small perturbation assumption.

To avoid the above problem, the control law in (35) was modified to include a desired trajectory ${}^{P}BR_{des(t)}$:

$$\tilde{U} = -K_d \cdot \tilde{Z}_{1-9} + K_p ({}^{P}BR - {}^{P}BR_{des}) + K_i \int_{t_0}^{t} ({}^{P}BR - {}^{P}BR_{des}) \, dt \tag{36}$$

where ${}^{P}BR_{des}(t)$, is generated when the tracking and docking control system is activated.

Let t_f be the desired duration of the docking phase and let ${}^{P}BR(0)$ denote the distance between the 3DW and the UAV receptacle at the pre-contact position. The relative velocity between the UAV and the tanker is designed to start from zero, to reach its maximum value at $t_f/2$, and to return to zero at $t = t_f$. That is when the UAV reaches the contact position at the center of the 3D refueling window. Thus, the desired trajectory can be defined through the following relationship:

$$ {}^{P}BR_{des,i}(t) = a_i t^3 + b_i t^2 + c_i t + d_i, \quad i = x, y, z \tag{37} $$

The coefficients in the above polynomial are evaluated through imposing the boundary conditions on the initial and final positions:

$$ {}^{P}BR_{des,i}(0) = {}^{P}BR_{i}(0), \quad {}^{P}BR_{des,i}(t_f) = 0, \quad i = x, y, z \tag{38} $$

and the initial and final velocities:

$$ {}^{T}\dot{B}R_{des,i}(0) = 0, \quad {}^{T}\dot{B}R_{des,i}(t_f) = 0, \quad i = x, y, z \tag{39} $$

The resulting reference trajectory is defined by:

$$ {}^{P}BR_{des,i}(t) = {}^{P}BR_{des,i}(0)\left[2\left(t/t_f\right)^3 - 3\left(t/t_f\right)^2 + 1\right], \quad 0 \le t \le t_f \tag{40} $$

Dividing the vertical and lateral components of the reference trajectory by the longitudinal component will yield the following constants:

$$ \frac{{}^{P}BR_{des,z}(t)}{{}^{P}BR_{des,x}(t)} = \frac{{}^{P}BR_{des,z}(0)}{{}^{P}BR_{des,x}(0)}; \quad \frac{{}^{P}BR_{des,y}(t)}{{}^{P}BR_{des,x}(t)} = \frac{{}^{P}BR_{des,y}(0)}{{}^{P}BR_{des,x}(0)} \tag{41} $$

which in turn means that the reference trajectory is a straight line, because the lateral and vertical components are a linear function of the longitudinal one.

Finally, the maximum values of the velocity and acceleration along the trajectory are found to be:

$$V_{\max,i} = -\frac{3 \cdot {}^{P}BR_{des,i}(0)}{2t_f}, \quad Acc_{\max,i} = \pm\frac{6 \cdot {}^{P}BR_{des,i}(0)}{t_f^2}, \quad i = x, y, z \tag{42}$$

As required, the UAV docking from the "pre-contact" to the "contact" position is performed with the UAV perfectly aligned with the tanker longitudinal axis, resulting in the initial condition ${}^{P}BR_{des,y}(0) = 0$.

2.10 Comparative Study between Point Matching Algorithms

This section discusses the results of a study focused on evaluating the performance of the matching algorithms through four specific tests. These tests were developed with the goal of evaluating the suitability of the algorithms.

2.10.1 Computational effort

The computational effort test has the purpose of evaluating the CPU time required by each algorithm for providing the correct point matching result. This study aims at providing some empirical results about the computational requirements and the total complexity of the implementation of the proposed algorithms.

The study includes two parts. The first part consists in measuring the computational effort of the two matching functions when the number of the corners *m* in the set P (detected) is varied while the number of the corners *n* in the set $\hat{P}$ (projected) is kept constant to 10. The second part is instead relative to the case when *n* is varied and *m*

is kept constant to 100. A Pentium 4, 3.2 GHz of clock speed and 1 Gbyte of RAM was used for this analysis. The speed performance was measured with the Simulink® "profiler" tool, which provides the running time in seconds for each called function and sub-function. The simulation lasted 35 seconds with the MV system featuring a sampling time of 0.1 sec. Table 4 shows clearly that MNP has a linear trend, while MCD has a parabolic trend in both sets of data. This trend was expected since the computational complexity of MNP is $O(m*n)$ while the computational complexity for the construction of the graph of MCD is $O(m^2*n^2)$. Therefore, the sub-optimal solution of the Maximum Clique Detection Algorithm does not seem to provide specific problems increasing the level of complexity in MCD, which seems mostly due to its "graph construction" part. Figure 9 shows that the MNP algorithm is more than three orders of magnitude faster than MCD.

m	MNP(sec)	MCD(sec)	n	MNP(sec)	MCD(sec)
100	$4.55*10^{-5}$	0.01	10	$1.33*10^{-4}$	0.011
200	$9.09*10^{-5}$	0.043	20	$8.81*10^{-5}$	0.047
300	$1.45*10^{-4}$	0.14	30	$1.31*10^{-4}$	0.15
400	$3.10*10^{-4}$	0.38	40	$2.22*10^{-4}$	0.42
500	$5.74*10^{-4}$	0.98	50	$4.43*10^{-4}$	0.88

Table 4: Computational effort varying the number of *m* and *n* corners

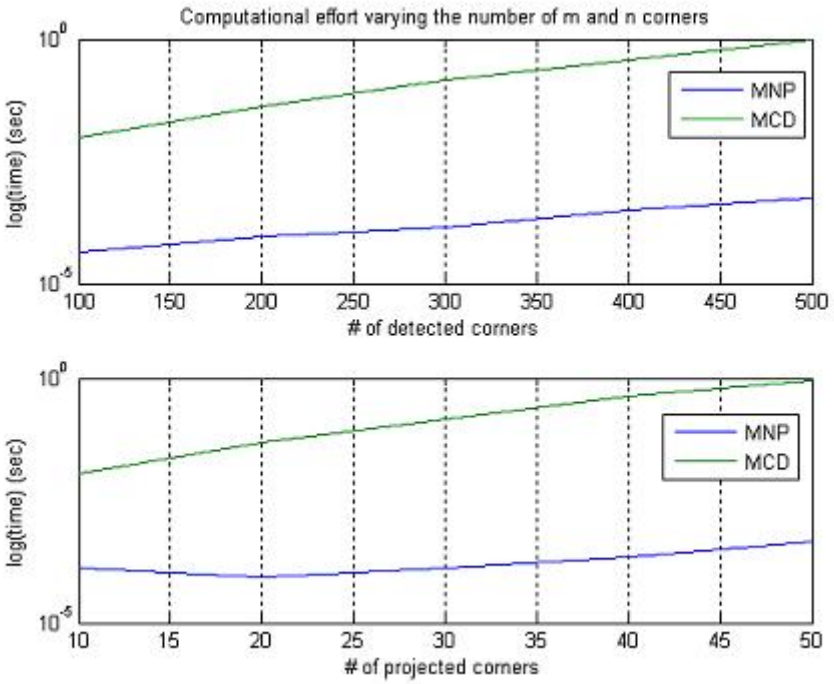


Figure 9: Computational effort varying the number of detected and projected corners in a logarithmic scale

2.10.2 Virtual Image analysis

The images from the closed loop WVU VRT–based simulation were used for the purpose of comparing the two matching algorithms. The test consisted on the analysis of the allocation of the points of the set *P* when the points of the set $\hat{P}$ change. These points were selected using different images that gradually diverge from the analyzed image. Specifically, the set of the points $\hat{P}$ was provided extracting the corners from the image at time (*t**-*i*) while the points of the set *P* are always extracted from the image at time *t**. Therefore, the points in $\hat{P}$ gradually became more distant from the points in *P*; thus, the performance of both algorithms was expected to decrease as the number increased. The parameters of the two algorithms were empirically tuned so that they could provide their

best performance. MNP was set to create a hypercube around each corner of the set $\hat{P}$ with sizes equal to 5% of the screen size for the (u – v) dimensions, 5 for the area dimension and 0.1 for the hue dimension. All the points outside these hyper-cubes were discarded and did not play a role in the evaluation of the distance. MCD features instead two thresholds; T_P is defined as the Proximity Principle parameter and it was set to 5% of the norm of the screen size; T_R is defined as the Exclusion and Rigidity Principle threshold and it was set to be $0.5*10^{-3}$. In the test, the number of points in the set $\hat{P}$ (projected points) was selected to be 10 while the number of points in set P (detected points) was less than 150. Ten different images, with i ranging from 0 to 1 sec at 0.1 sec, intervals were supplied to the corner detection and point matching algorithms.

Additionally, to better evaluate the performance of the algorithms within real-world situations, one point of the set $\hat{P}$ (the uppermost point in Figure 10) was purposely placed in a position that did not exactly correspond to any physical corner, although being close to some points that the FE algorithm recognized as corners. Ideally, this corner should not be assigned to any detected corner. In fact, Figure 10 shows that both algorithms did not recognize this corner in this particular image. However, as also shown in Table 4 in the column CNIC (Correctly Not Identified Corners) MNP statistically performed better than MCD. The results of this analysis are summarized in Table 5. The column INIC (Incorrectly Not Identified Corners) counts the number of detected corners that are either not identified or matched to the wrong projected corner. Figure 11, instead, shows the behavior of the two algorithms when the points of the set $\hat{P}$ are at the maximum distance from the points of the set P. Summarizing, the average of the column

CNIC should be as closed as possible to the value 1 and the average of the column INIC should be as small as possible.

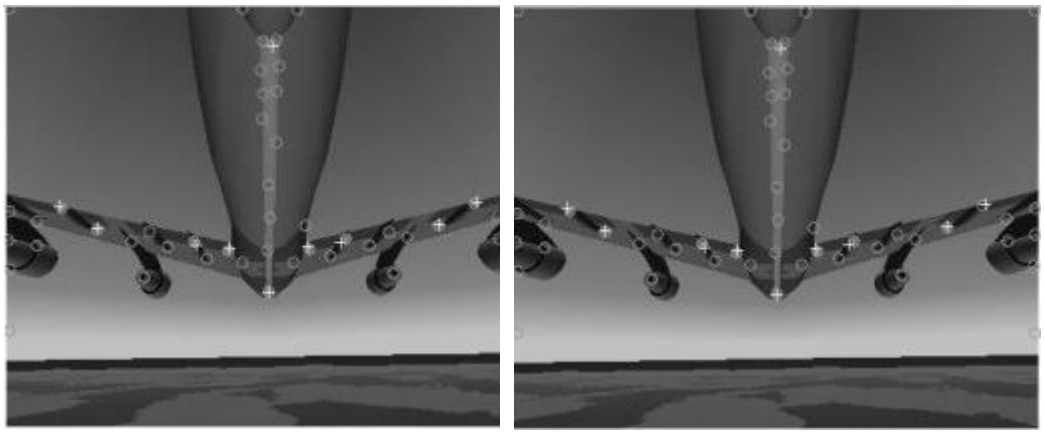


Figure 10: Matched points in the "Virtual Image Analysis" from MNP (left) and MCD (right) with distance between the image *i*=0.1. The points of the set $\hat{P}$ are represented using the symbol +, the points of the set *P* are represented using the symbol O, the points selected by the matching algorithm are represented using the symbol *.

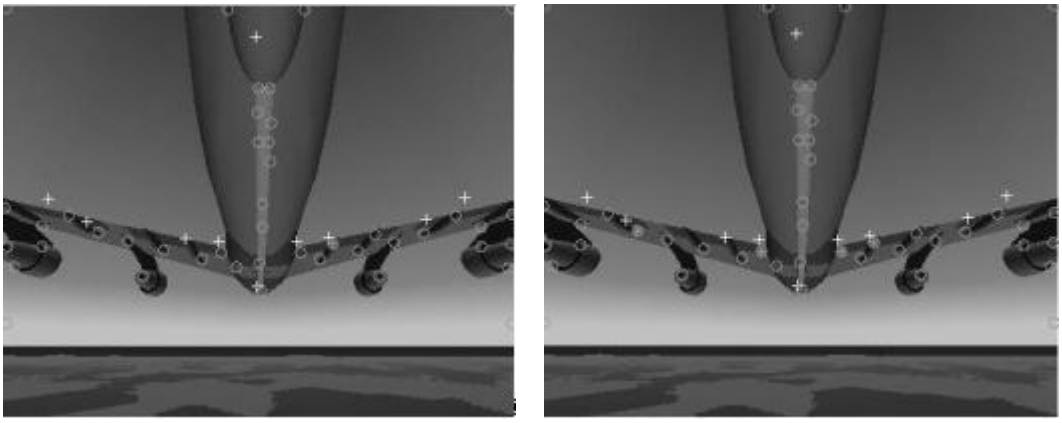


Figure 11: Matched points in the "Virtual Image Analysis" from MNP (left) and MCD (right) with distance between the image *i*=1.0. The points of the set $\hat{P}$ are represented using the symbol +, the points of the set *P* are represented using the symbol O, the points selected by the matching algorithm are represented using the symbol *

	MNP		MCD	
i (sec)	CNIC	INIC	CNIC	INIC
0.1	1	0	1	1
0.2	1	0	1	1
0.3	0	1	0	2
0.4	0	1	0	3
0.5	1	0	0	2
0.6	1	0	0	3
0.7	1	0	0	3
0.8	1	0	0	2
0.9	0	3	0	2
1	1	8	1	5
Avg	0.7	1.3	0.3	2.4

Table 5: Summary of the virtual image analysis data

2.10.3 Real Image Analysis

A similar analysis was performed using images from a video acquired with a digital camera in lieu of virtual images. The video featured a static Boeing 747-400 model – with a 25" wingspan – hanging from the ceiling of our MAE laboratory. The digital camera recorded the image simulating an approach maneuver from pre-contact to contact. The Harris Corner Detection method – used for feature extraction purposes – was tuned through selection of the thresholds, to provide a reasonable number of corners. The set $\hat{P}$ contains 11 corners that the matching algorithms were supposed to recognize.

The parameters of the two algorithms were set as follows. MNP was set to create its hypercube around each corner of the set $\hat{P}$ with dimensions $\begin{bmatrix}10\% & 10\% & 10 & 0.3\end{bmatrix}$, where 10% are related of the screen size for u (horizontal) and v (vertical) dimensions, while 10 and 0.3 are related to the area and hue dimensions. MCD had the parameter T_P = 10% of the screen diagonal, and $T_R = 5*10^{-3}$. The associated results are shown in Figure 12 - Figure 14 and Table 6. An analysis of the results reveals that the MNP performs

better when the set of the points $\hat{P}$ is at larger distances from the set of point P while MCD recognizes more corners when points in $\hat{P}$ are closer to the points in P.

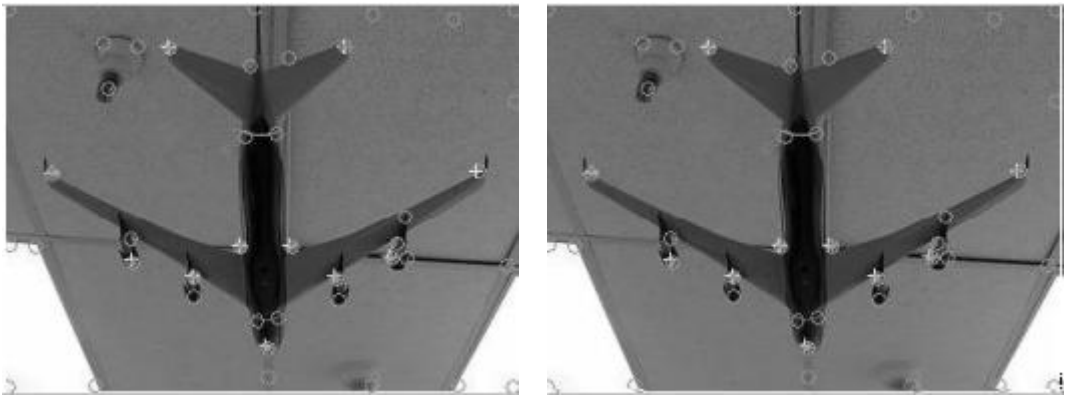


Figure 12: Matched points in the "Real Image Analysis" from MNP (left) and MCD (right) with distance between the image i=0.1. The points of the set $\hat{P}$ are represented using the symbol +, the points of the set *P* are represented using the symbol O, the points selected by the matching algorithm are represented using the symbol * .

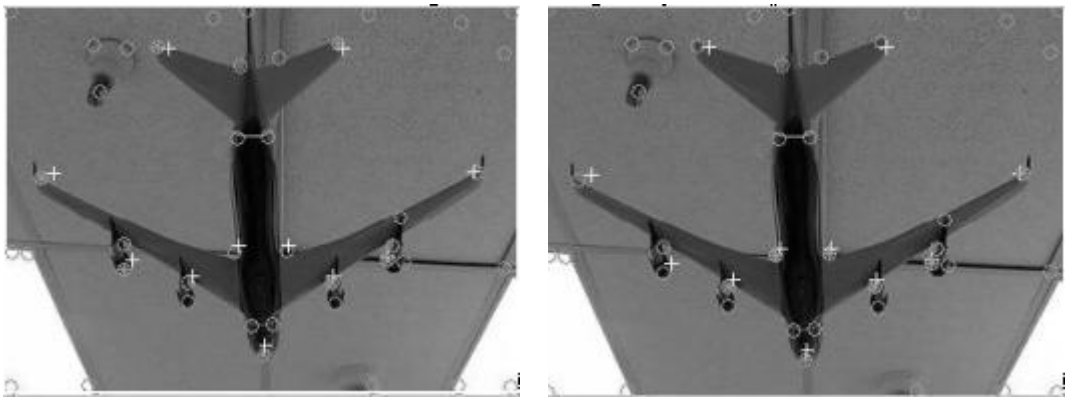


Figure 13: Matched points in the "Real Image Analysis" from MNP (left) and MCD (right) with distance between the image i=0.4. The points of the set $\hat{P}$ are represented using the symbol +, the points of the set *P* are represented using the symbol O, the points selected by the matching algorithm are represented using the symbol * .

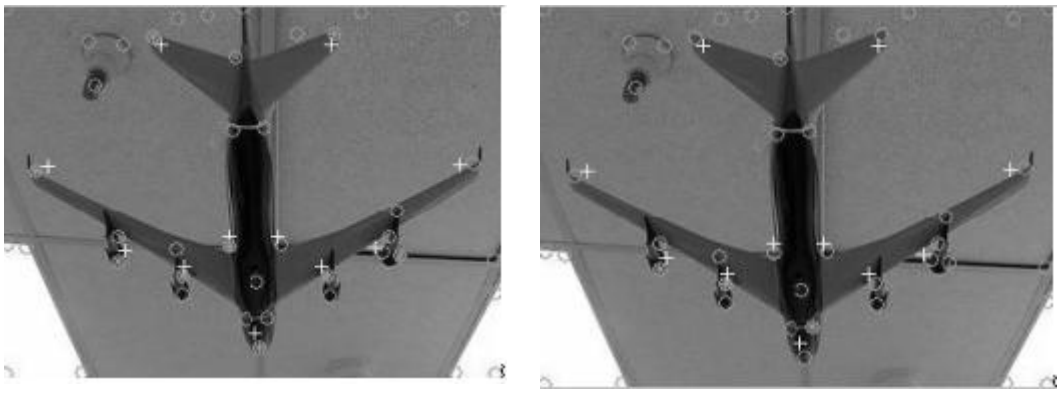


Figure 14: Matched points in the "Real Image Analysis" from MNP (left) and MCD (right) with distance between the image i=0.9. The points of the set $\hat{P}$ are represented using the symbol +, the points of the set *P* are represented using the symbol O, the points selected by the matching algorithm are represented using the symbol * .

	MNP	MCD
i (sec)	INIC	INIC
0.1	1	0
0.2	2	0
0.3	2	1
0.4	4	4
0.5	3	4
0.6	3	5
0.7	4	5
0.8	3	9
0.9	3	9
1	4	9
Avg	2.9	4.6

Table 6: Real Image analysis data varying the points of the set $\hat{P}$ using different images

2.10.4 LHM Estimation Errors

This study was conducted with the objective of evaluating the performance of the LHM algorithm when each matching algorithm is used. Since the two matching algorithms will provide different points to the LHM, it is expected that the PE algorithm will provide different estimates on the tanker-UAV relative position and orientation. Furthermore, it is important to emphasize that a minimum of 5 'matched' points have to

be supplied by the PM algorithm to the LHM algorithm in order for the latter to produce an updated estimate; otherwise, the estimate at the previous time step is returned by the LHM.

The estimation error is defined as the difference between the position and orientation values provided by the LHM and the "real" position and orientation values (which are of course available only in simulation). The results are summarized in Table 7 and Figure 15. Specifically, Figure 15 shows the norm of the position estimation error. It can be seen from the x axis of Figure 15 that the LHM is executed about 450 times and 530 times depending on whether it is interfaced with MNP or MCD respectively.

This implies that MNP supplies the minimum required 5 matched points less often than MCD. On the other hand, MNP provides a more accurate overall matching in terms of the final estimation error. This means that MNP is less likely to mismatch corners than MCD.

Table 6 shows the statistical results of the different components of the relative position errors in terms of mean and standard deviation. The results indicate that the MCD outperforms MNP for the estimation of the relative angles while the results are very similar for the estimation of the relative position.

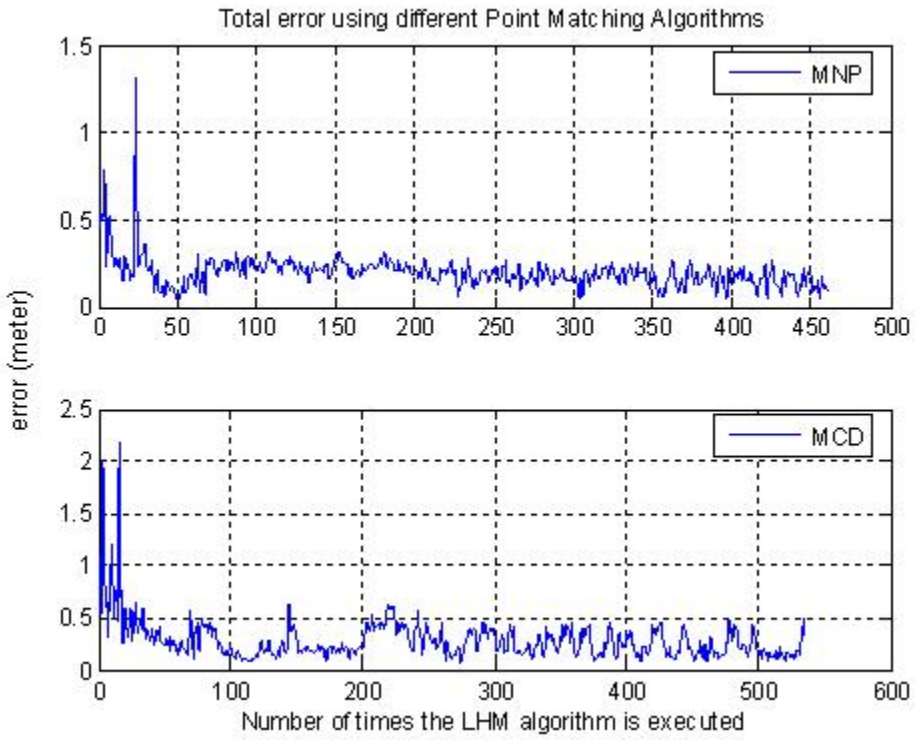


Figure 15: Position estimation error vs. number of times the LHM is executed

	e_x	σ_x	e_y	σ_y	e_z	σ_z	e_{dist}	σ_{dist}
MNP	0.1033	0.1039	-0.0413	0.0567	0.1067	0.0912	0.1939	0.0919
MCD	0.0844	0.2817	-0.0433	0.0480	0.1318	0.0761	0.2805	0.1871
	e_{Roll}	σ_{Roll}	e_{Pitch}	σ_{Pitch}	e_{Yaw}	σ_{Yaw}	e_{ang}	σ_{ang}
MNP	0.0080	0.1459	0.0095	0.0378	$8.5*10^{-4}$	0.0129	0.0224	0.1501
MCD	0.0016	0.0059	0.0113	0.0086	0.0034	0.0089	0.0163	0.0080

Table 7: Estimation error between LHM and exact measurement

2.11 Comparative Study between Pose Estimation Algorithms

The following performance criteria were introduced for a detailed comparison between the GLSDC and LHM pose estimation algorithms:

1. Speed
2. Accuracy

3. Robustness
4. Tracking error

2.11.1 Speed performance

A Pentium 4, 2.53 GHz laptop with 448 Mbytes of RAM was again used for this analysis. The speed performance was measured with the Simulink® "profiler" tool, which provides the running time in seconds for each called function and sub-function. The simulation lasted 40 seconds. The MV system was used with a sampling time of 0.1 sec. On average, the GLSDC and LHM algorithms required $5.3*10^{-3}$ sec and $20.1*10^{-3}$ sec per simulation step, respectively. Thus, the LHM algorithm is approximately 4 times slower than the GLSDC.

2.11.2 Accuracy (Difference between true and estimated ${}^{C}\boldsymbol{T}_{\boldsymbol{T}}$ values)

For the purpose of this analysis, "true values" are defined as the distance and orientation of the tanker in camera frame read at each simulation step from the linear and angular position (simulated) sensors. The "estimated values" are instead the distances and orientations of the tanker in camera frame provided directly by the two pose estimation algorithms. . It should be emphasized that both the true and estimated values are slightly dependent on the particular UAV trajectory, which is in turn dependent on the linear and angular position estimations provided by the particular PE algorithm that is being used within the control loop.

An analysis of the results highlights that the LHM needs a minimum of 5 corners while the GLSDC requires at least 4 corners. However, the GLSDC performance strongly

depends on the accuracy of the initial conditions. Table 8 shows the results from t_1 = 15 sec until t_2 = 50 sec. Overall, it appears that the GLSDC and LHM algorithms provide similar levels of accuracy. Figure 16 to Figure 19 show the linear and angular position estimations from the two algorithms.

	X	Y	Z	Roll	Pitch	Yaw
GLSDC	0.3254	0.0962	0.1679	0.0090	0.0150	0.0098
LHM	0.3364	0.0963	0.1367	0.0109	0.0125	0.0104

Table 8: RMS values of the error for the GLSDC and LHM algorithms between t1=15 sec and t2=50 sec

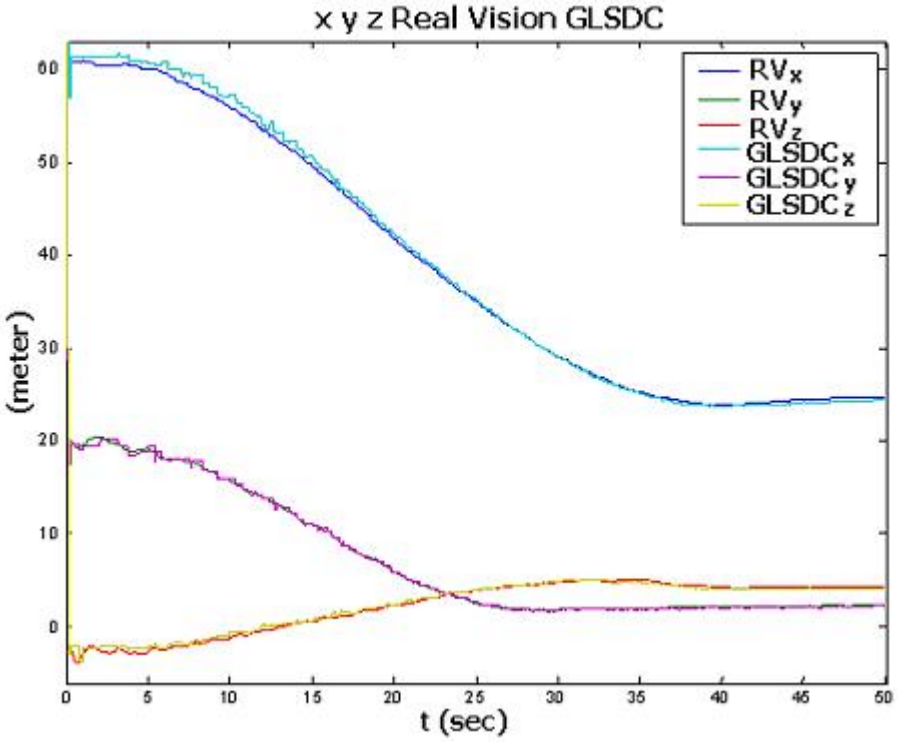


Figure 16: 'Real' x y z vs. x y z estimates from the GLSDC algorithm

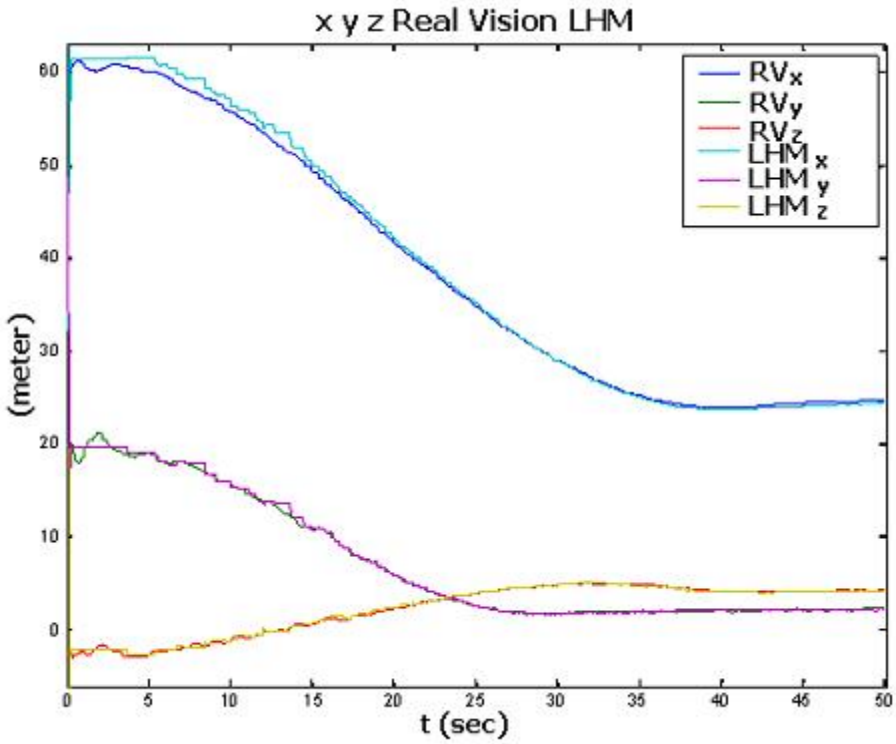


Figure 17 - 'Real' x y z vs. x y z estimates from the LHM algorithm

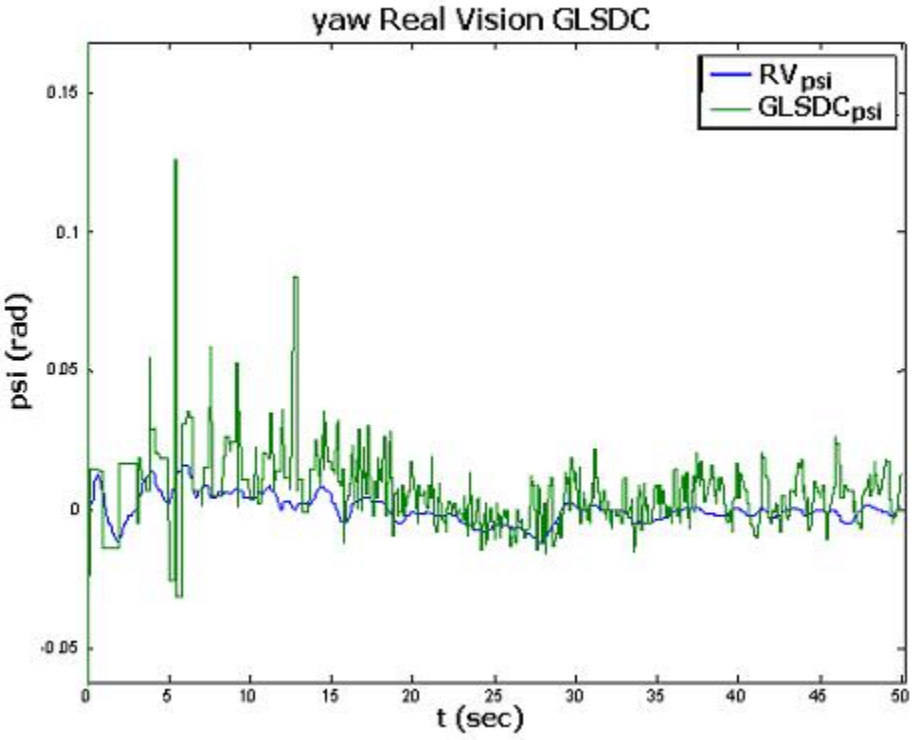


Figure 18: 'Real' yaw angle vs. estimate from the GLSDC algorithm

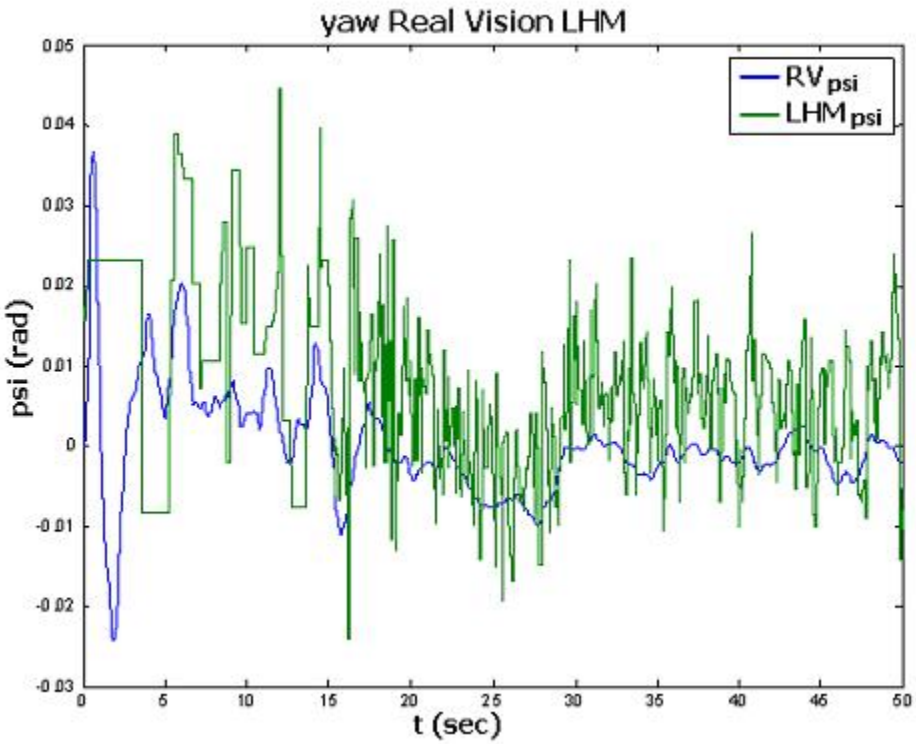


Figure 19: 'Real' yaw angle vs. estimate from the LHM algorithm

2.11.3 Robustness

This analysis was performed in terms of robustness with respect to the following parameters:

- noise addition in the corners position (with correct point matching).
- incorrect performance of the point matching algorithm.
- errors in initial conditions.
- input noise.

2.11.3.1 Noise addition in the corners position with correct point matching

This analysis was performed with the MV algorithms in open-loop mode, that is, the docking control laws used values from sensors and GPS to perform the docking

maneuver. Different levels of noise were added to the correctly point matched 2D corners positions. Specifically, the selected noise was a band limited white noise with correlation time t_c=0.05 sec, the value of the Power Spectral Density (PSD) in the bandwidth of interest was entered trough the "Noise Power" parameter. Different values of the PSD were evaluated, starting form 0 to $5*10^{-9}$ m^2 with an interval of $1*10^{-9}$ m^2. Up to a value of $4*10^{-9}$ m^2, both algorithms performed reasonably well. For a PSD of $5*10^{-9}$ m^2 the LHM performance started to deteriorate, as shown in Figure 20. However, in order for this performance degradation to take place, the input noise has to act in a situation in which the number of detected corners is limited. In fact, better performance can promptly be recovered whenever the number of detected corners increases and/or the noise power levels decrease. Conversely, it was observed that for a PSD of $5*10^{-9}$ m^2 the GLSDC performance tend to degrade abruptly, leading the algorithm outside of its stability region, as shown in Figure 21. In this event, the algorithm is not able to recover an acceptable performance, independently on the number of detected corners. On the other hand, whenever the number of detected corners is greater than 5, both algorithms provide similar desirable performance.

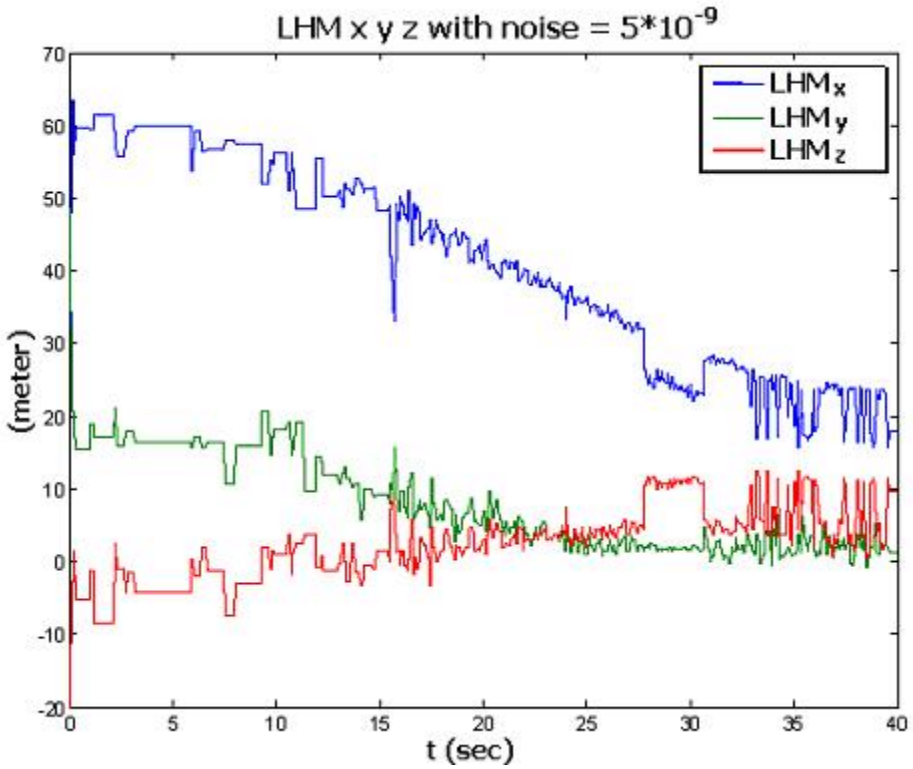


Figure 20: LHM behavior with 5*10-9 noise power

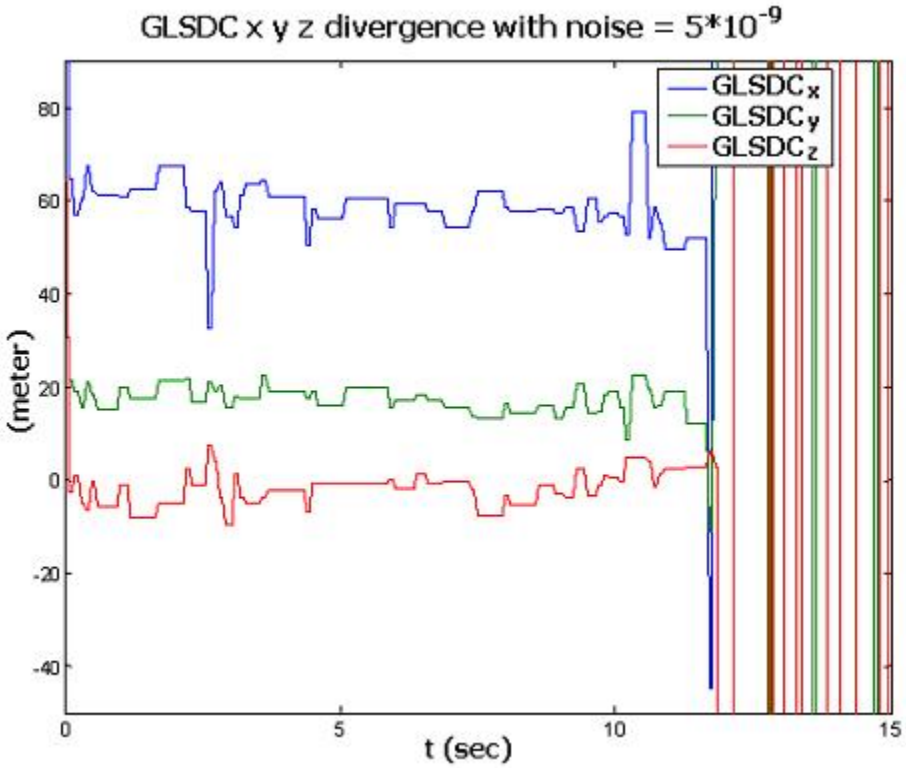


Figure 21: GLDSC behavior with 5*10^{-9} noise power

2.11.3.2 Robustness to point matching errors

In this study, incorrect performance of the point matching algorithms were simulated by inverting the point matching of two corners at two simulation steps, specifically for t = 20 sec and t = 20.1 sec. During these time instants, the corners #8 and #7 were exchanged and provided as inputs to the PE algorithms. The analysis shows that the GLSDC is substantially more sensitive to point matching errors than the LHM algorithm. For example, the x estimated by the GLSDC drops from 42.8 m to 12.6 m against the 33.5 m provided by the LHM; a similar behavior was observed for the other estimated variables. An even more important conclusion is that the GLSDC algorithm requires a considerably longer time to return to the "nominal" performance than the LHM algorithm. This conclusion is consistent with the previous results on the noise robustness of the two algorithms.

2.11.3.3 Robustness to errors in initial conditions

The purpose of this study was to evaluate the performance of the GLSDC and LHM algorithms under a broad range of errors in the initial linear and angular position of the tanker in camera frame. To perform the test, the number of detected corners was held constant at 5 for both algorithms; a larger convergence area is of course expected whenever additional visible corners are detected. The exact initial conditions were set to be $x_o = [x, y, z, \psi, \theta, \varphi] = [60.5\ 20\ -2.5\ 0\ 0.467\ 0]$. The results, which are summarized in Table 9, show that the GLSDC algorithm has a limited convergence area while the LHM algorithm performs very well for any range of erroneous initial conditions. Furthermore, within its convergence area, the GLSDC shows a larger settling time before providing

reasonably accurate estimates. On the other side, while outside its convergence area, the GLSDC algorithm either diverges or provides inaccurate estimates.

Translation vector		**Yaw angle (*psi*)**	
GLSDC interval from exact initial condition	LHM interval from exact initial condition	GLSDC interval from exact initial condition	LHM interval from exact initial condition
[-44.7 65.4]	[-∞ +∞]	[-1.74 3.06] rad = [-100 175] °	[0 2π]

Table 9: Convergence region for the initial condition

2.11.3.4 Error propagation analysis

In this study, the noise propagation behavior of the two algorithms was investigated. Specifically, a white Gaussian noise (WGN) was added to the position (*x, y*) of one single corner. This noise propagates through the pose estimation algorithm, resulting in a noisy output. At this point it is interesting to investigate in whether the MV system acts as a linear system as far as noise propagation, between an input (on the corner position) and the output (on the translation vector), is concerned. The system was considered composed by one input and three outputs as if they were three systems with one input and one output. The sub-systems were called $GLSDC_X$, $GLSDC_Y$ and $GLSDC_Z$ for the GLSDC algorithm and LHM_X, LHM_Y and LHM_Z for the LHM algorithm. In addition, the estimated PSD with the periodogram method is defined:

$$PSD(f) = \frac{2\pi}{n}\left|\sum_{l=1}^{n} x_l e^{-j\omega l}\right|^2 \tag{43}$$

where n is the number of element of the noise data and x_l is the position l of the vector. From the signal theory we know:

$$\frac{PSD_{out}(f)}{PSD_{in}(f)} = \left|H(f)\right|^2 \tag{44}$$

where PSD_{out} represent the output noise PSD, PSD_{in} represent the input noise PSD, $H(f)$ is the frequency response of the system. If the ratio between PSD_{out} and PSD_{in} is almost equal among the various powers of input noise, then for what concerns noise propagation, we can approximate the systems with a linear one. The linearity is a sufficient condition for the preservation of the Gaussian distribution. Figure 22 - Figure 24 represent the square of the frequency response for the sub-systems called $GLSDC_X$, $GLSDC_Y$ and $GLSDC_Z$. Figure 25 - Figure 27 show the square of the frequency response of the sub-systems called LHM_X, LHM_Y and LHM_Z. It is clearly visible that the behavior of 6 systems is exactly linear since the line for different power noise exactly overlap in all the 6 plots. Once the linearity is verified, it is possible to state that the output noise is Gaussian with mean equal to 0 since the input noise has mean value equal to 0 and the following relationship is valid:

$$\eta_{out} = H(0)\eta_{in} \tag{45}$$

where η_{in} is the input mean value, η_{out} the output mean value, and $H(0)$ is the static gain of the system. The output error has variance equal to the second order moment because the error has mean equal to 0, and the second order moment is provided by the relationship:

$$E\left\{X^2(t)\right\} = 2\int_0^\infty PSD_{out}(f)df \tag{46}$$

where PSD_{out} is the power spectral density of the output noise.

From the above considerations is it possible model the noise; if the input noise is white and gaussian, the output noise will also be white and gaussian. Additionally, the

Machine Vision system behaves as a linear system if it is considered the relation between the position of the detected corners and the measurement of the relative distance between camera and tanker.

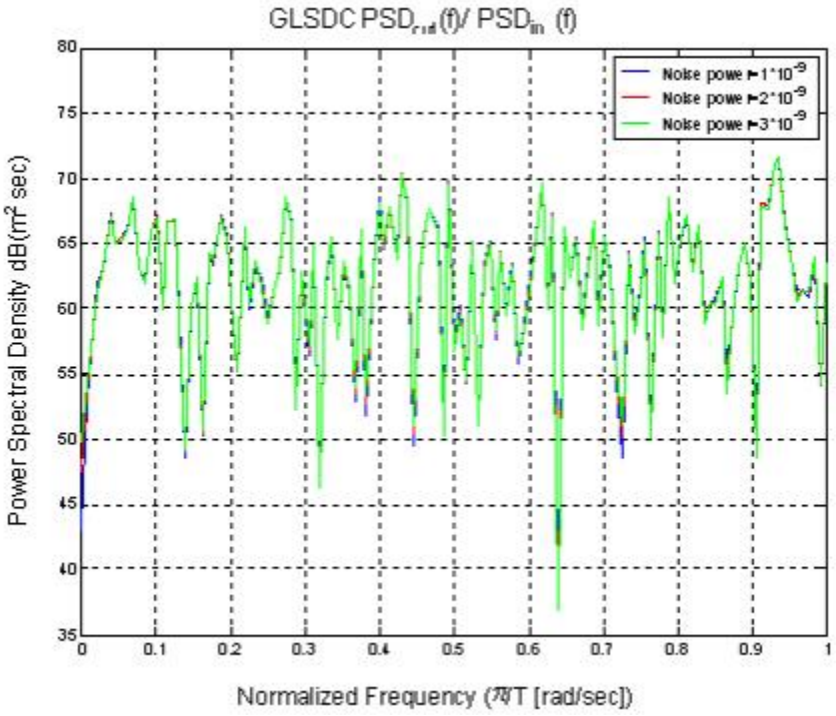


Figure 22: verification of linearity propriety for $GLSDC_X$ system

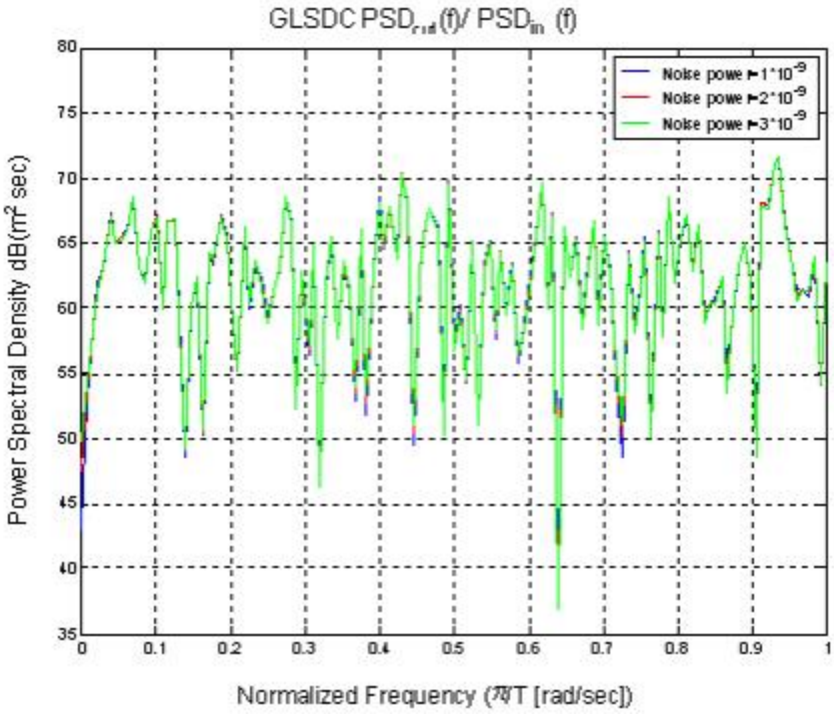


Figure 23: verification of linearity propriety for GLSDC$_Y$ system

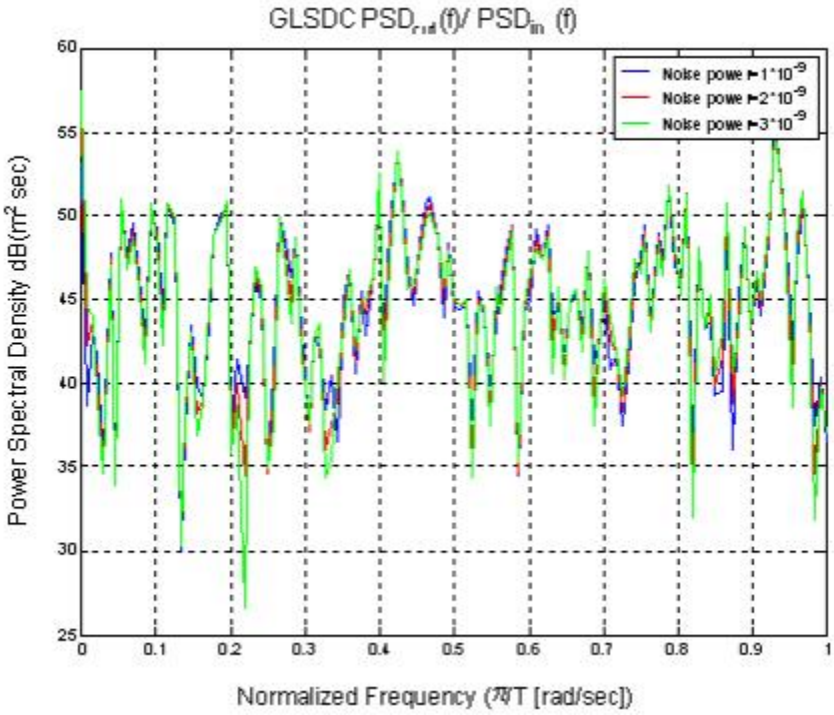


Figure 24: verification of linearity propriety for GLSDC$_Z$ system

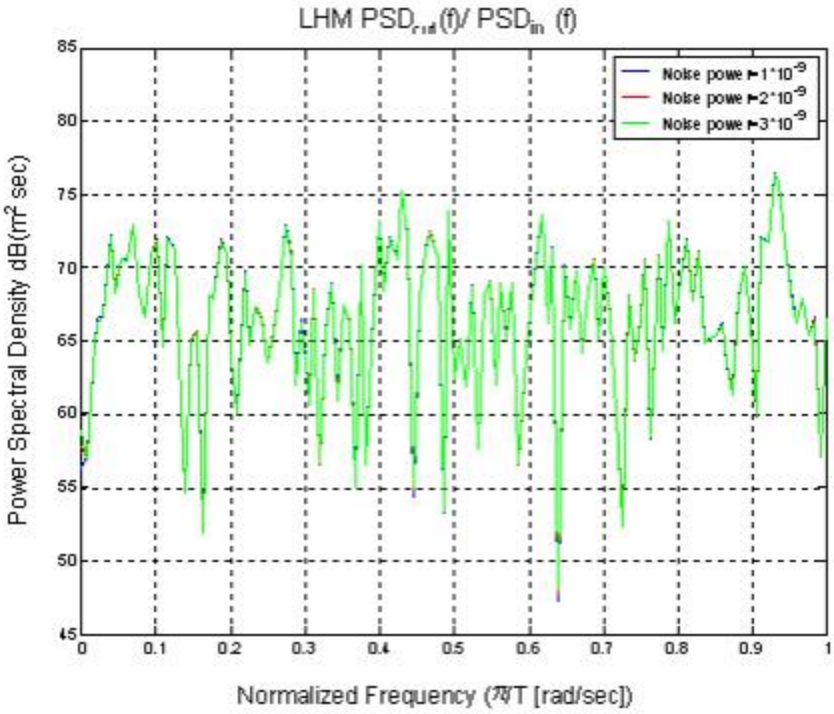


Figure 25: verification of linearity propriety for LHM_X system

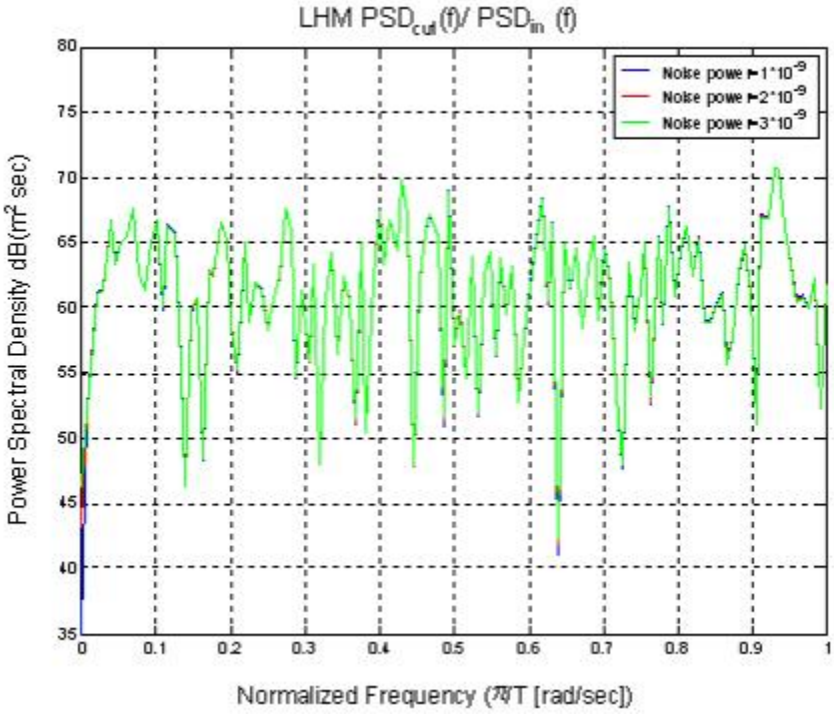


Figure 26: verification of linearity propriety for LHM_Y system

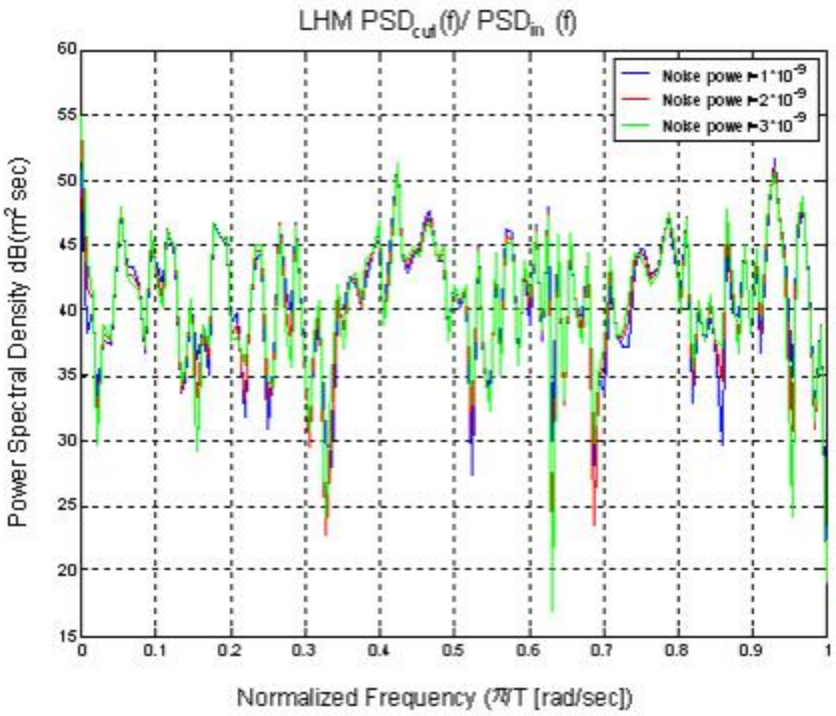


Figure 27: verification of linearity propriety for LHM_Z system

At this point, it is possible compare the output noise between the three sub-systems that compose the GLSDC algorithm and the three sub-systems that compose the LHM algorithm in order to understand which algorithm amplifies more the noise. Figure 28 - Figure 30 show a direct comparison between the PSD of the systems $GLSDC_X$ and LHM_X, $GLSDC_Y$ and LHM_Y, $GLSDC_Z$ and LHM_Z. In Figure 28 and Figure 29, the lines are overlapped which means that the GLSDC and LHM algorithms propagate the errors in the same way. In Figure 30, the GLSDC algorithm amplifies the noise more than LHM algorithm as it concerns the variable *z*.

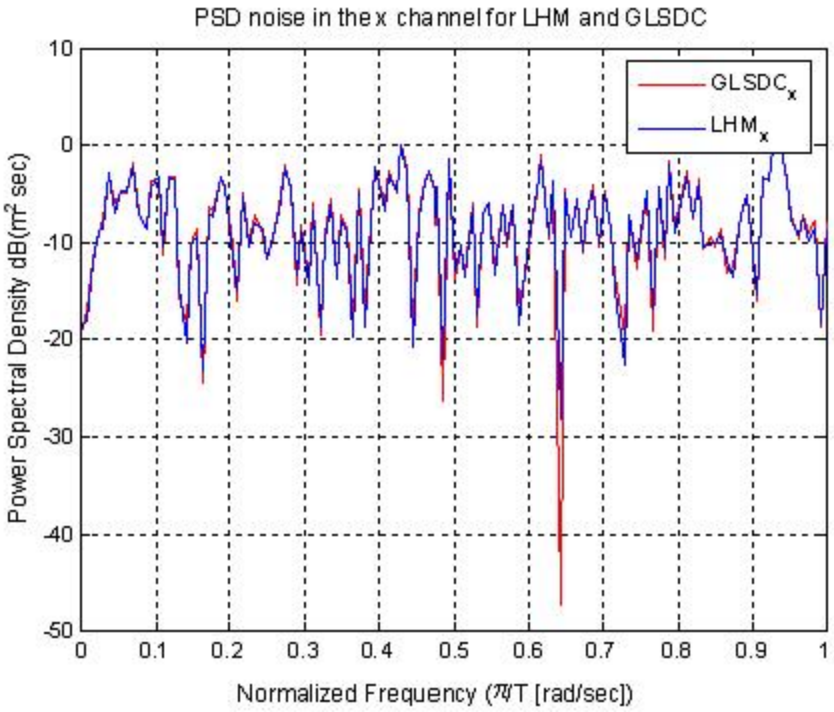


Figure 28: PSD of $GLSDC_X$ and LHM_X with noise $1*10^{-9}$

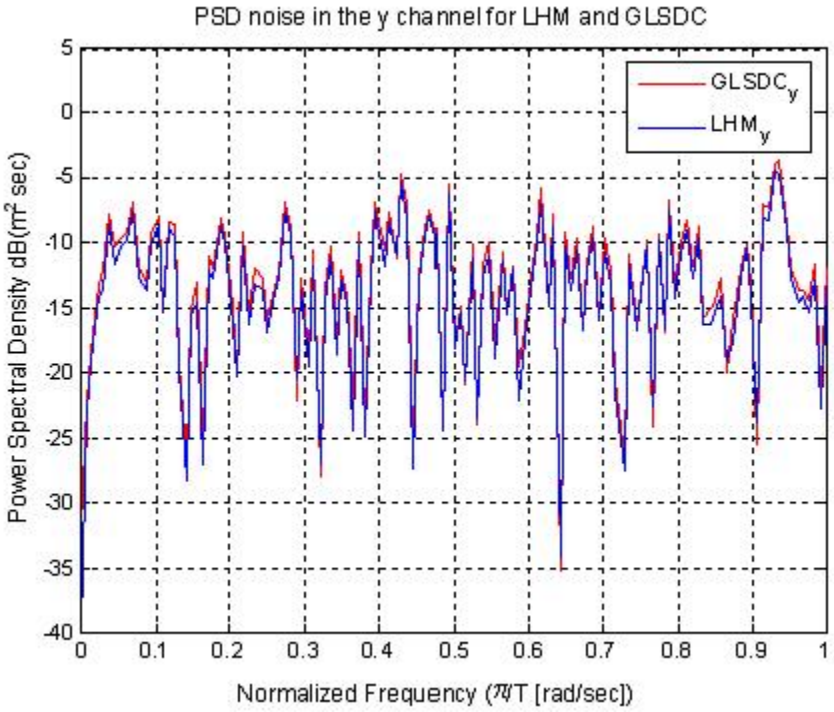


Figure 29: PSD of $GLSDC_Y$ and LHM_Y with noise $1*10^{-9}$

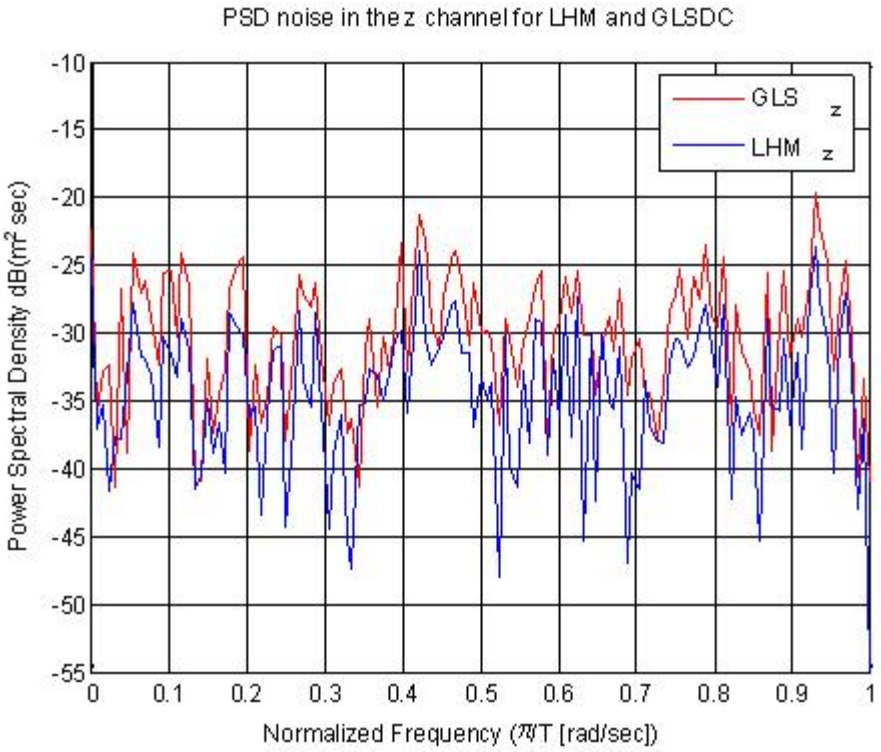


Figure 30: PSD of $GLSDC_Z$ and LHM_Z with noise $1*10^{-9}$

Summarizing, it was observed that the output noise retains the same properties of the input noise, that is, Gaussian with zero mean, and with a PSD proportional to the power of the input noise. It is interesting to note that, while the GLSDC and LHM algorithms propagated the errors in a similar manner for the variables *x* and *y*, the GLSDC algorithm amplified the input noise more than the LHM algorithm along the *z* channel, as shown in Figure 30. The properties of the output noise suggest that a Kalman-Filter based approach could be highly effective in handling the measurements provided by the MV system.

2.11.4 Tracking error analysis

As a final study, the closed-loop tracking error (that is the difference between the actual docking path and the nominal docking path) has been evaluated for both LHM and GLSDC algorithms. Since the level of accuracy for the two algorithms is comparable, it was not expected for the tracking error to be influenced by the performance of the pose estimation algorithm. Figure 31 shows the tracking error along the 3 axes (in the LHM case) while Table 10shows the mean of the tracking error for the two algorithms.

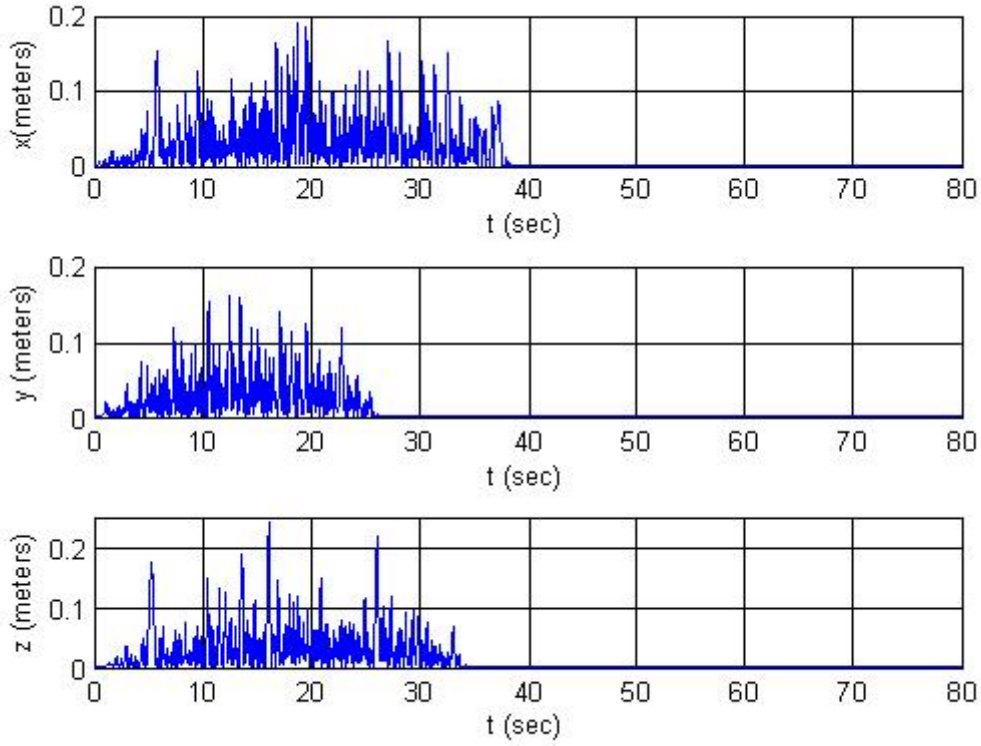


Figure 31: Tracking Error (Using the LHM algorithm)

	GLSDC (meter)	LHM (meter)
x	$1.3831*10^{-2}$	$1.4180*10^{-2}$
y	$1.0154*10^{-2}$	$0.9996*10^{-2}$
z	$1.2835*10^{-2}$	$1.3674*10^{-2}$

Table 10: Mean tracking error with GLSDC and LHM

2.12 Sensor Fusion system based on Extended Kalman Filter

2.12.1 Sensor modeling

2.12.1.1 Modeling of the MV Sensor

The MV system can be considered as a smart sensor providing the relative distance between a known object and the camera. Therefore, a detailed description of the characteristics of its output signals is critical for the use of this sensor. Since a zero mean Gaussian noise can be only considered in theory for infinite number of samples the measurements provided by the MV will be affected by a Gaussian White Noise with non-zero mean [80]. A summary of the output characteristics is provided in Table 11. Being the noises white and Gaussian, only the means (μ) and the standard deviations (σ) of the errors in the CRF directions (*x, y, z*) are required for their complete statistical descriptions.

	x *(meter)*	**y** *(meter)*	**z** *(meter)*
μ	-0.090	0.015	-0.069
σ	0.056	0.060	0.065

Table 11: Statistical Parameters of the MV-Based Position Sensor

2.12.1.2 Modeling of the INS Sensor

Both aircraft are assumed to be equipped with Inertial Navigation Systems (INS), which are capable of providing the velocities and attitudes of the aircraft by measuring its

linear accelerations and angular rates. Within the developed simulation environment, 'realistic' INS outputs are simulated by adding a white gaussian noise (WGN) to the corresponding entries of the aircraft state vector. To validate this type of modeling, the noise within the signals acquired by the INS has been analyzed using the normal probability analysis and the Power Spectral Density (PSD). This allowed assessing whether such noise could be modeled as white and Gaussian.

The flight data used to validate the modeling of the INS noise were taken from a recent experimental project involving the flight testing of multiple YF-22 research aircraft models [56]. The analysis was performed with a sampling time of 10 Hz for all the aircraft sensors. The results for the pitch rate q are shown Figure 32.

The upper portion of Figure 32 shows the normal probability plot – plotted using the Matlab "*normplot*" command - of the simulated noise and of the noise provided by the real sensor. The purpose of this plot is to assess whether the data could come from a normal distribution. In such a case, the plot is perfectly linear. For the noise related to the pitch rate channel, the part of the noise close to zero follows a linear trend, implying a normal distribution. Note that due some outliers, the tails of the curve corresponding to the real sensor do not follow this trend. However, the fact that the trend is followed within the central part of the plot – which represents the majority of the data - validates that this noise can be modeled as a Gaussian process in a certain neighborhood of zero.

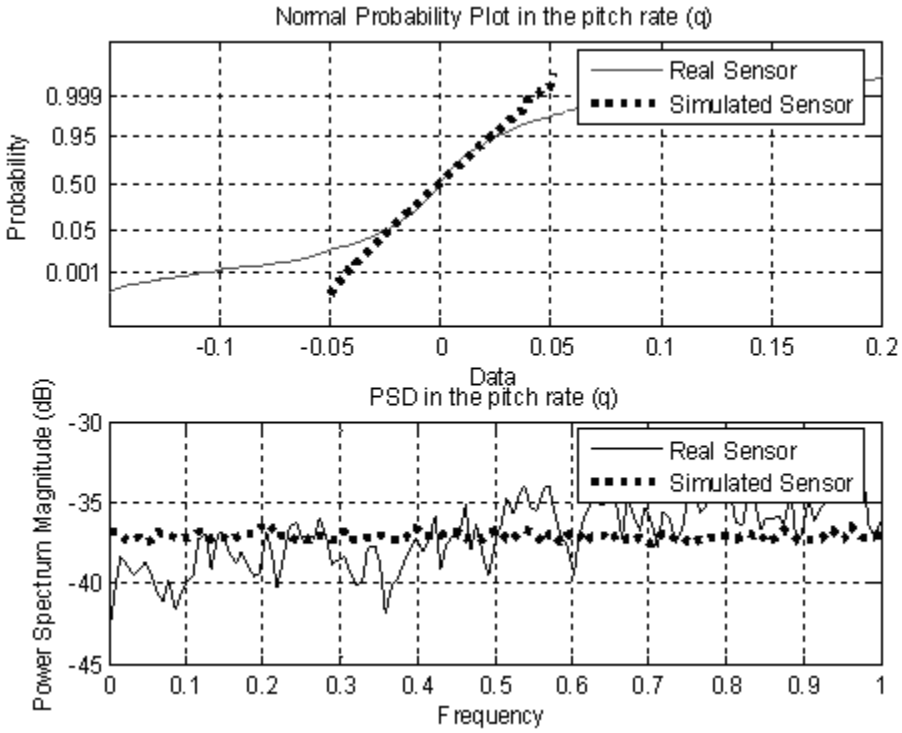


Figure 32 - The normal probability and PSD in the pitch rate (*q*) in Real and Simulated INS

A PSD analysis also confirms the hypothesis of white noise. In fact, the lower portion of Figure 32 shows that the spectrum of the noise from the real sensor, although not as flat as the spectrum of the simulated noise (shown as a dotted line), is still fairly well distributed throughout the frequency range. Thus, both the normal probability and PSD analysis confirm that the noise on the IMU *q* channel measurement can be modeled as a white Gaussian random vector. Similar conclusions can be achieved for the *p* and *r* IMU channels.

2.12.1.3 Modeling of the Pressure, Nose probe, Gyro, and Heading Sensors

An air-data nose probe - for measuring flow angles and pressure data - was installed on the UAV. This sensor provides the measurements of the velocity (V), the angle-of-attack (α), and the sideslip angle (β), while the vertical gyro provides measurements for

the aircraft pitch and roll angles (θ and φ). Within this analysis the heading was approximated with the angle of the planar velocity in ERF, that is $\psi = atan2(V_y, V_x)$, where *atan2* is the 4 quadrant arctangent function and the velocity are supplied by the GPS unit and are based on carrier-phase wave information. However, the heading can also be calculated by gyros, magnetic sensors, or by a filtered combination of all the above methods.

Following a similar analysis to the one performed for the INS, the noise on the measurements from the above sensors was modeled as white and gaussian noise (WGN). Table 12 summarizes the results in terms of noise variances for the different aircraft dynamic variables.

	V $(m/s)^2$	α $(rad)^2$	β $(rad)^2$	p $(rad/s)^2$	q $(rad/s)^2$	r $(rad/s)^2$	ψ $(rad)^2$	θ $(rad)^2$	φ $(rad)^2$
σ^2	2e-1	2e-3	2e-3	2e-2	2e-2	2e-2	2e-3	2e-3	2e-3

Table 12: Variance of the Noise of the Sensors

2.12.1.4 Modeling of the GPS Position Sensor

The GPS sensor provides its position (x, y, z) with respect to the ERF. A composition of four different Band Limited White Noises was used to simulate the GPS noise. Specifically, the four noises have different power and sample times. Three of these noise signals are added and filtered with a low-pass filter and the resulting signal is added to the fourth noise and sampled with a zero-order-hold. In fact, GPS measurements –in case that more than 4 satellite signals are received - normally exhibit a "short term" noise with amplitude within 2 to 3 meters, as well as "long term" trend deviations and "jumps" due to satellites motion and occlusions. Therefore, while the first "short term" noise has

been modeled as a White Gaussian Noise, the trend deviations and jumps have been modeled using the other 3 lower-frequency, filtered, noises.

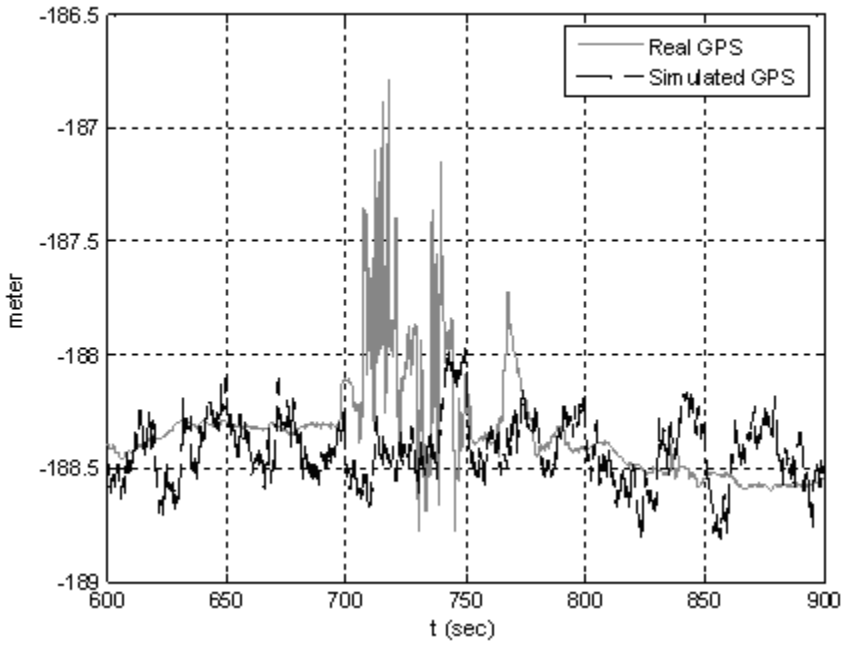


Figure 33 - Comparison between Real and Simulated GPS signals

Figure 33 shows both the signal from a real GPS receiver (Novatel-OEM4), and the simulated GPS signal.

2.12.2 Sensors Fusion Using EKF

2.12.2.1 EKF Background Theory

The main purpose of the Kalman filter algorithm [18] is to provide optimal estimates of the system dynamics through available measurements assuming 'a priori' known statistical models for the system and measurement noises.

The Discrete-Time Kalman Filter [18] involves two basic steps. The first step consists in using the system dynamic model to predict the evolution of the state between

consecutive measurements instances. The second step consists in the use of the measurements along with the system dynamic model for evaluating the optimal (Newton-like) correction of the estimated values at the time of the measurements. The filter characterizes the stochastic disturbance input through its spectral density matrix and through the measurement error by its covariance.

In many applications the measurement model, the system dynamics, or both are potentially non-linear. In these cases, the KF may not be an optimal estimator. Non-linear estimation methods are discussed in [61][62]. The Extended Kalman Filter retains the KF calculations of the covariance and gain matrices, and it updates the state estimate using a linear function of the filter residual [18]. However, it uses the original non linear equations of the system dynamics for state propagation and output vector calculation. The EKF equations are briefly reviewed below.

Given a generic discrete dynamic system:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, w_k) \\ y_k &= h(x_k, v_k) \end{aligned} \tag{47}$$

where u_k, x_k, and y_k are respectively the input, state, and output vectors of the dynamic system. w_k and v_k are white and Gaussian noises with the following statistical properties:

$$\begin{aligned} &E\left[w_k\right] = 0 \\ &E\left[w_k w_k^T\right] = W_k \\ &E\left[v_k\right] = 0 \\ &E\left[v_k v_k^T\right] = V_k \\ &E\left[w_j v_k^T\right] = 0 \end{aligned} \tag{48}$$

Furthermore, the initial state x_0 is a random variable with the following mean value and covariance matrix:

$$E[x_0] = \hat{x}_0$$
$$E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T] = P_0 \tag{49}$$

Assuming that *f* and *h* are locally differentiable, the following Jacobian matrices are calculated:

$$F_k = \frac{\partial f(\bullet)}{\partial x_k}, \qquad H_k = \frac{\partial h(\bullet)}{\partial x_k} \tag{50}$$

Under these assumptions, an EKF can be implemented using the following equations:

State Estimate Propagation

$$\hat{x}_{k+1}^- = f(\hat{x}_k, u_k, 0) \tag{51}$$

Covariance Estimate Propagation

$$P_{k+1}^- = F_k P_{k-1} F_k^T + W_{k-1} \tag{52}$$

Filter Gain Computation

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k)^{-1} \tag{53}$$

State Estimate Update

$$\hat{x}_k = \hat{x}_k^- + K_k (y_k - h(\hat{x}_k^-, 0)) \tag{54}$$

Covariance Estimate Update

$$P_k = (I - K_k H_k) P_k^- \tag{55}$$

Note that the EKF does not preserve the optimality properties of the Linear Kalman Filter (LKF). However, its simplicity and robustness are very appealing.

2.12.2.2 Sensors fusion using EKF

The use of EKF for sensor fusion is well documented in robotics applications for the fusion of inertial, GPS, and odometer sensors as described in [63][64][65][66]. Within this effort, emphasis was placed on the fusion between data from a MV-based sensor system and data from the INS/GPS system. In general, sensor fusion applications require the output function $y_k = h(x_k, v_k)$ of the dynamic system to be adapted to the number of sensors that the filter has to combine.

In this case, the output function contains the following variables:

$$y_k = \begin{bmatrix} V & \alpha & \beta & p & q & r & \psi & \theta & \varphi & x_{GPS} & y_{GPS} & z_{GPS} & x_{MV} & y_{MV} & z_{MV} \end{bmatrix} \quad (56)$$

where the subscript GPS indicates measurements from the GPS system while the subscript MV indicates measurements from the MV system.

The EKF formulation assumes that the measurements are affected by a white and Gaussian noise (16). Therefore, the noise affecting the variables x_{GPS}, y_{GPS}, and z_{GPS}, were considered to be white and gaussian with variances of 0.014 m^2, 0.013 m^2, and 0.022 m^2 respectively. These values were calculated using the MATLAB "*var*" command on a large set of data from the GPS sensor, simulated as described in previous section. The MATLAB "*mean*" command applied on the same set of data, provided results under 2% of the range, which validated the zero-mean assumption. Similarly, for the MV-based position sensor, Table 11 indicates that the mean values of the MV position measurements can be approximated to be zero.

The EKF scheme requires 3 specific inputs. The first input is the UAV command vector u_k containing the throttle level and the deflections of the control surfaces. The

second input is the complete system output vector defined in (24), which includes data from the INS/GPS and the MV sensors. The third and last input is the number of corners used by the PE algorithm, which is critical since the MV system provides reliable estimates of the relative position vector only if a sufficient number of corners (greater than 6) are properly detected by the 'Mutual Nearest Point' algorithm. Specifically, the entries of V_k relative to the MV position measurements are multiplied by a factor of 1000 when the number of detected corners is lower than the required amount. Essentially this causes the exclusion of the MV information from the sensor fusion process.

The output of the EKF is the estimate $\hat{x}_k$ of the system's state vector x_k, which contains the 12 aircraft state variables. Specifically, the last 3 variables of the EKF output are the estimates of the aircraft position in the ERF. The values of these variables are the results of the sensor fusion between the data supplied by the two different position sensor systems.

According to the selected state and output variables, the matrix H_k in (18) becomes a matrix with dimension 15×12 containing the derivatives of the outputs with respect to the states. Similarly, the matrix V_{kx} is a matrix of dimensions 15×15, containing all the noise covariances, including the ones from the GPS and the MV systems.

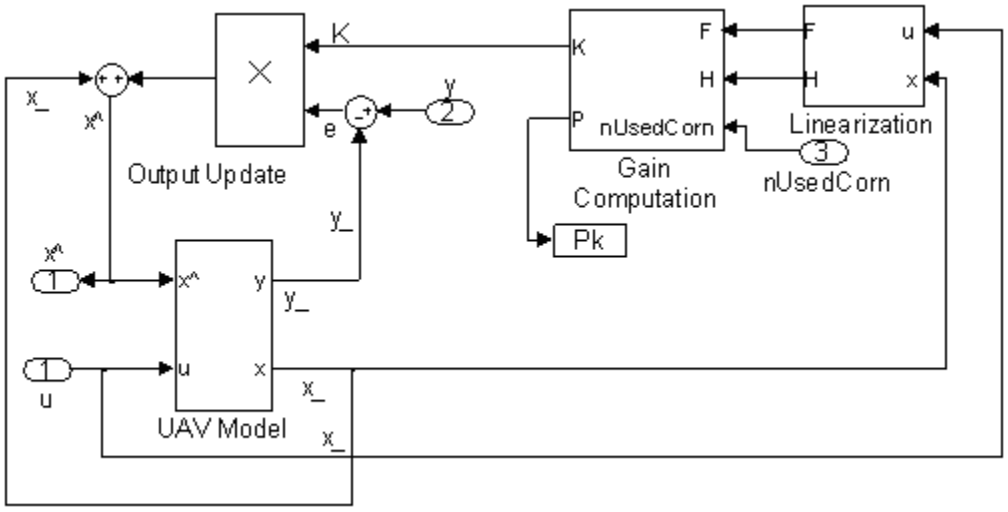


Figure 34 - Scheme of EKF for sensors fusion

Figure 34 shows the general Simulink Scheme of the EKF, including its different components blocks such as the "Linearization" block - which performs the calculations in Eq..(18) - the "Gain Computation" block - which calculates Eq. (20), (21), and (23) - and the "Output Update", which calculates Eq. (22). The tuning of the EKF is performed as follows. First, the initial state of the filter is set equal to the state of the UAV system at the time instant when the EKF is switched on (Table 13 shows typical values of such initial state). The matrix P_0 is then set to zero. Next, the matrix W_k is kept constant and equal to the identity matrix with dimensions 12*12. As previously mentioned, the matrix V_k, varies as a function of the corners detected by the MV system. Specifically, if the number of corners is greater than 6 the matrix V_k, is a diagonal matrix containing the 9 values provided in Table 2, the 3 variances of the GPS measurements: 0.014 m^2, 0.013 m^2, and 0.022 m^2 for *x y* and *z* directions respectively, and the 3 variances related to the distances measured by the MV system, provided in Table 11. Whenever the number of detected corners is less than 6 then the 3 variances related to the MV system are multiplied by 1000 so that these measurements are practically discarded.

Variable	V	α	β	p	q	r	ψ	θ	φ	xe	ye	H
Value	205	0.077	0	0	0	0	0	0.077	0	-58.8	0	6068

Table 13: Typical Initial State Vector

2.12.3 Performance Analysis

The analysis of the closed loop simulations was performed to validate the performance of the EKF. In this study, the UAV acquires data from all its onboard sensors (modeled as described in previous sections) and receives data from the tanker, which are pre-filtered for noise reduction purposes. The EKF output - that is the result of the sensor fusion between the MV and GPS data - is used in the docking control laws for guiding the UAV from the 'pre-contact' position to the 'contact' position and for holding position in the defined 3DW once the contact position has been reached.

Without any loss of generality, the 'pre-contact' position was assumed to be located 50 *m* behind and 10 *m* below the tanker aircraft, while the 'contact' position, i.e. the 3DW position, was assumed to be right below the tanker, within the reach of the telescopic portion of the refueling boom.

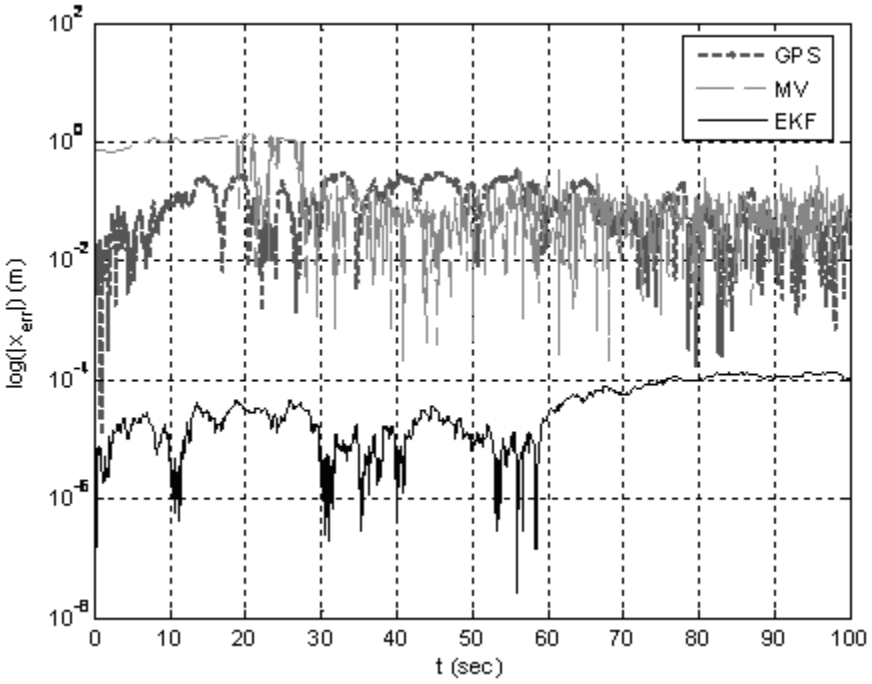


Figure 35 - Comparison of errors along the x-axis between EKF, GPS and MV system

Due to finite camera resolution and due to the fact that objects appear smaller at larger distances, a MV-based system cannot provide reliable results when the tanker-UAV distance is too large [12][59][29]. Thus, MV-based results are fairly inaccurate until approx. 30 sec. in the simulation.

In Figure 35 - Figure 37 the logarithm of the EKF error - defined as the absolute value of the difference between the actual position and the EKF-based position - is compared with the MV and GPS noises, for the x, y, and z axes respectively. It can be noted that the error of the EKF output is approximately two orders of magnitude smaller than the noises of both the GPS and MV systems.

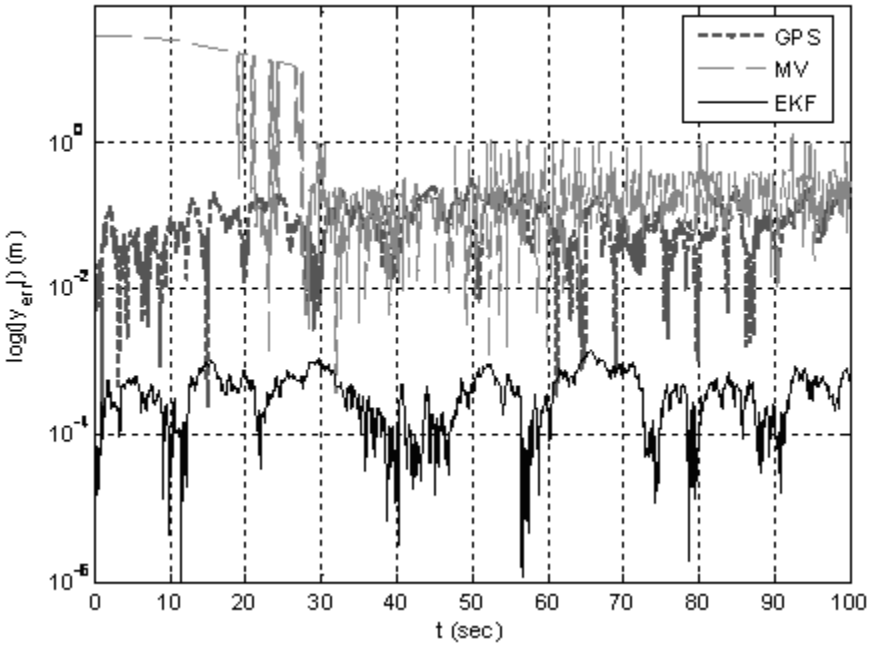


Figure 36 - Comparison of errors along the y-axis between EKF, GPS and MV system

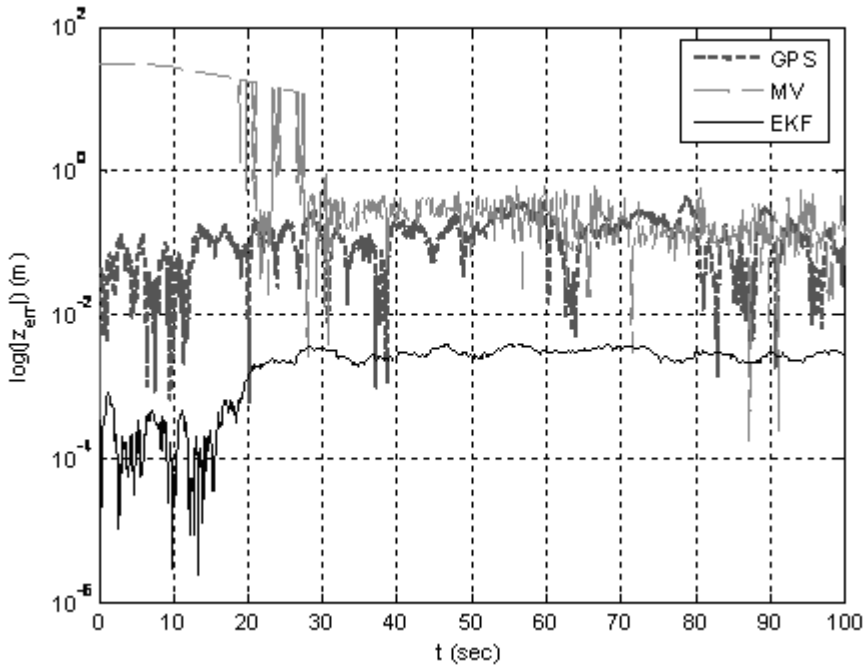


Figure 37 - Comparison of errors along the z-axis between EKF, GPS and MV system

The accuracy of the EKF-based estimations is particularly evident in Figure 38, which shows the components of the EKF error along the 3 axes.

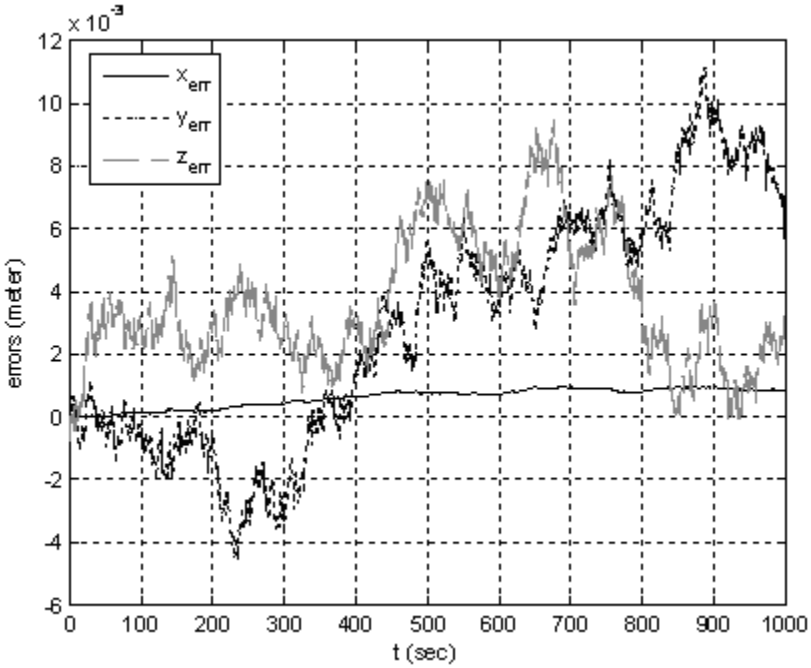


Figure 38 - Errors in the data using EKF

In the following figures, the performance of the EKF based sensor fusion scheme is compared with the performance of another 'baseline' sensor fusion scheme described in previous section and in [60], [31], [55]. Specifically, the old 'baseline' sensor fusion scheme consisted in using a linear interpolation between the distances supplied by the GPS and the MV systems when the relative aircraft distance was between two pre-defined values d_2 and d_1. Particularly, within the baseline scheme the GPS measurements were used when the distance was greater than d_2, while the MV distance estimation was used when the distance was lower than d_1.

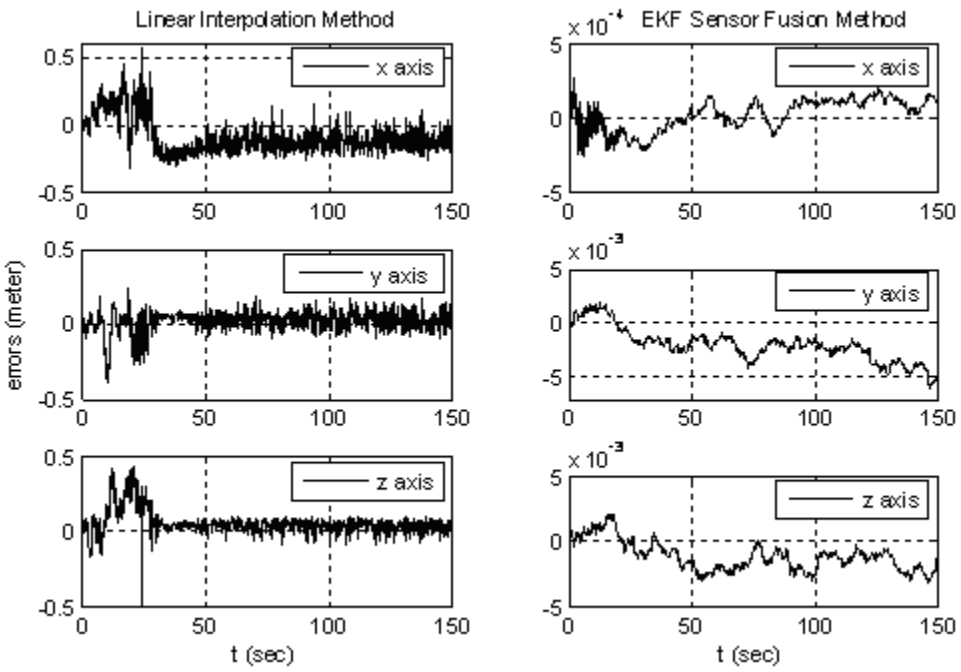


Figure 39 - Comparison in the tracking error between EKF sensors fusion method and sensors linear interpolation method

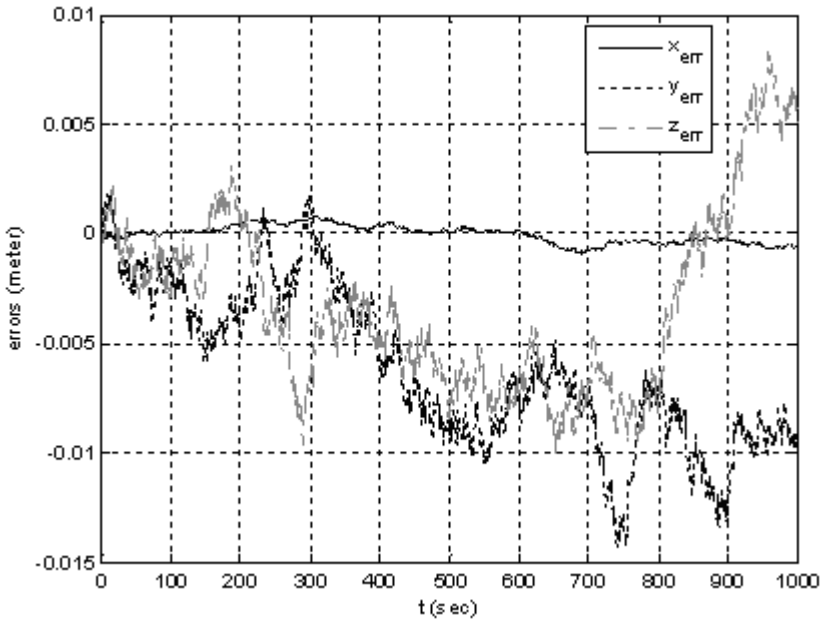


Figure 40 - Errors in the components of the tracking error

The motivation behind the 'baseline' sensor fusion scheme is that for distances larger than d_1 the MV system does not provide the necessary level of accuracy in the estimations; therefore, only GPS data were used. On the other side, for distances lower than d_2, the tanker itself could act a shield leading to potentially inaccurate GPS measurements; thus, only data from the MV system could be used. Without any loss of generality the values of 40 *m* and 23 *m* were used for the distances d_1 and d_2 respectively in the previous studies.

Figure 39 and Figure 40 show the UAV tracking error during the approach and docking phases. It can be seen that the EKF based sensor fusion scheme provides a substantial improvement in terms of tracking performance during the UAV docking phase.

2.12.4 Robustness Analysis

A robustness study was performed for assessing the robustness of the filter to perturbations such as biases in the θ and φ signals from the gyro, variations of the initial UAV position *x0*, *y0*, *z0*, and variations in the entries of the V matrix associated with the MV and GPS sensors systems. Specifically, for each of the above conditions, a simulation was performed in which the parameter was changed, while all the other parameters retained their values.

Parameter	**Variation**	**Avg x axis Error (*m*)**	**Avg y axis Error (*m*)**	**Avg z axis Error (*m*)**	**Avg Mag. Error (*m*)**
None	None	$-2.93*10^{-5}$	$-1.67*10^{-4}$	$-2.28*10^{-3}$	$2.42*10^{-3}$
θ	$1.77*10^{-4}$ rad	$-2.91*10^{-3}$	$-1.58*10^{-4}$	1.82	1.82
φ	$2.1*10^{-3}$ rad	$-5.10*10^{-6}$	-1.65	$-2.51*10^{-2}$	1.65
x0	2 *m*	$5.6*10^{-2}$	$1.31*10^{-2}$	$1.24*10^{-1}$	$2.21*10^{-1}$
y0	2 *m*	$4.11*10^{-2}$	$3.51*10^{-2}$	$1.24*10^{-1}$	$2.19*10^{-1}$
z0	2 *m*	$4.11*10^{-2}$	$1.31*10^{-2}$	$1.46*10^{-1}$	$2.25*10^{-1}$
Var (GPS)	100%	$3.06*10^{-5}$	$-1.71*10^{-4}$	$-2.30*10^{-3}$	$2.44*10^{-3}$
Var (MV)	100%	$3.24*10^{-5}$	$-1.75*10^{-4}$	$2.32*10^{-3}$	$2.47*10^{-3}$

Table 14: Robustness Results

The average errors for the EKF position output – for each coordinate as well as in magnitude - are reported in Table 14 for each simulation. Note that the first row represents the baseline case in which no parameter was changed, while the last two rows represent the cases in which the 3 tuning variances for the MV and GPS sensors were doubled.

Overall, the table indicates that the EKF features desirable robustness characteristics with respect to both limited biases and variation in some of the tuning parameters.

3 THE COLLISION IDENTIFICATION PROBLEM AND SIMULATION

The second biggest current limitation of UAVs is their lack of collision identification (CI) and avoidance capabilities. Clearly, this problem does not have a unique solution, since in-flight collisions could be infinite and they can be avoided with a large set of escaping maneuvers. For example, other aircraft or birds, high mountains or large buildings are possible cause of collision.

A system able to identify collision is becoming a central research issue not only for the scientific relevance of the problem but also for the importance of the application. Current FAA regulations for UAVs require:

"... a method that provides an equivalent level of safety, comparable to the see-and-avoid requirements of manned aircraft"

[U.S. FAA Order 7106.4 Chapter 12, Section 9]

Much research is underway to address this issue [41][42]. From the first analysis, scientists understood that technologies as Traffic alert and Collision Avoidance System (TCAS) could not be used in UAVs because of the necessity to be "invisible" during most of the missions. Many developed Collision Avoidance system uses active systems like Laser [43] or Ka-band radar [44] or a fusion between active and passive system, cameras with different spectrum (IR, panchromatic and color), and radar [45]. These sensors are usually detectable and too heavy to be installed in small UAVs. For example, in [45] an apparatus consisting of four cameras, one radar, and two computers was

required; size and weight of this equipment clearly exceeds the limit payload for small UAVs.

The lack of a sensor able to register all the possible collision risks led scientists to imitate the most actually efficient sensor, that is the human vision system. Being impossible –with the current technology- to reproduce such a complicate system, scientists have observed that many animals –such as bugs – with simpler vision system and brain are still able to avoid collisions even within complex scenarios. Bugs are been analyzed for many years from biologist; one of the most efficient approaches used by bugs to avoid collisions is Optical Flow. The Optical Flow (OF) [46], [47] is the velocity vector field on the image plane generated by the relative motion with respect to the objects in the field of view. Using similar techniques some animals can perform quick and highly accurate navigation maneuvers.

In this effort, the goal is use a single camera and a single computer for the development of a Collision Identification system able of recognizing risks derived from other aircraft or object with a dynamic slower than UAVs. Considering the limitations of the problem, the possible risk of collision could only be identified only in the direction on which the camera is pointing and in a space that depends on the field of view of the camera. The collision identification system will use Optical Flow algorithms in order to identify a possible collision; this information will then be used into the aircraft controller for the generation of the "collision free" trajectory for the navigation of the UAV. The problem of extracting collision information from the Optical Flow algorithms; the estimation of the "no-flight zones" and the generation of commands to avoid such areas are not considered in this thesis.

A general flow chart of the Collision Identification problem is shown in Figure 41. In order to test the results an 'ad hoc' simulation was developed using the Matlab Virtual Reality Toolbox (VRT). In the document a general description of the simulation is provided, other details are addressed to the description of the Optical Flow algorithms and the methodology used to identify the possible collisions.

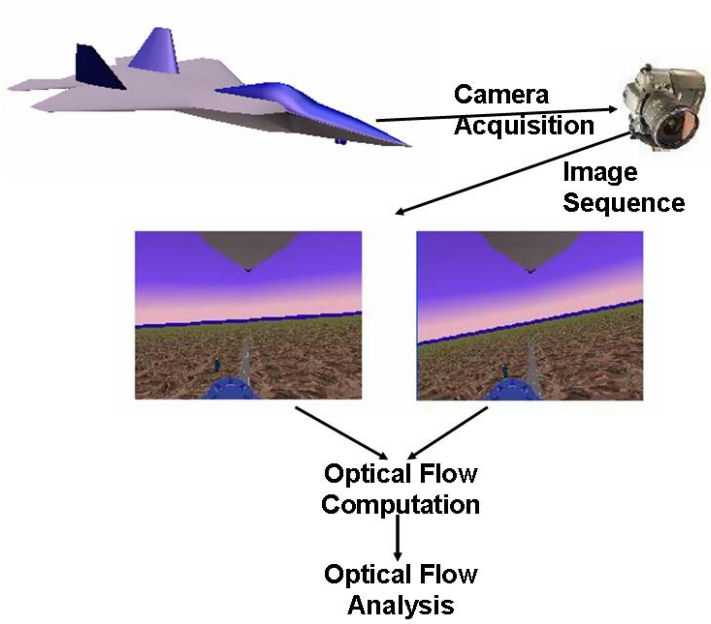


Figure 41: steps for Collision Identification

3.1 Collision Identification Simulation

A Collision Identification simulation was used as test-bed for the problem, considering the delicateness of the problem an accurate work of simulation is needed before a real hardware implementation. The problem requires that simulation was as much as possible accurate and close to the reality.

Into the VR environment, two UAVs performing different and pre-defined trajectories are implemented. Into the simulation, the UAVs are implemented with a mathematical model of an "ICE-101" which was described in previous section. The

trajectory of the UAVs can be predefined using 4D waypoints. A waypoint is composed by a 3D coordinate (*x, y, z*) and the crossing time (*t*). The reach time of a given waypoint can be respected only if the dynamic constraints due to the saturation of the throttle can be satisfied. The predefinition of the trajectory allows the UAVs trajectories to cross each other and consequently a UAV can enter into the camera field of view of the second UAV.

3.1.1 The Virtual Reality Environment

The Virtual Reality Environment is defined using the Matlab Virtual Reality Toolbox. Many objects were added with the intention of providing a realistic simulation.

An image from Google Earth was captured to create a realistic representation of the ground. The image selected was relative to a location close to the WVU Jackson's Mill Airstrip. Highways with static cars were added to the environment and some buildings were placed on the ground. In order to simulate obstacles during the flight, two aerostatic balloons can be placed by the user in predefined positions. Generally, into the simulation, the balloons and the other UAV represent the obstacles that a selected UAV has to identify; the buildings do not represent obstacles because of the UAV's altitude.

The trajectory of the UAVs can be defined by the user using a GUI. The user has to select the number of waypoints representing the trajectory of the UAVs; each waypoint is composed by a 4D coordinate, that is position *x*, *y* and altitude and the time that the waypoint has to be crossed. Figure 42 shows the GUI that permits to select the 4D coordinate for the waypoint #2 and the UAV #1 in the left and the corresponding trajectory in the right, the line in blue is the trajectory of UAV #1 and the line in green is the trajectory of UAV #2.

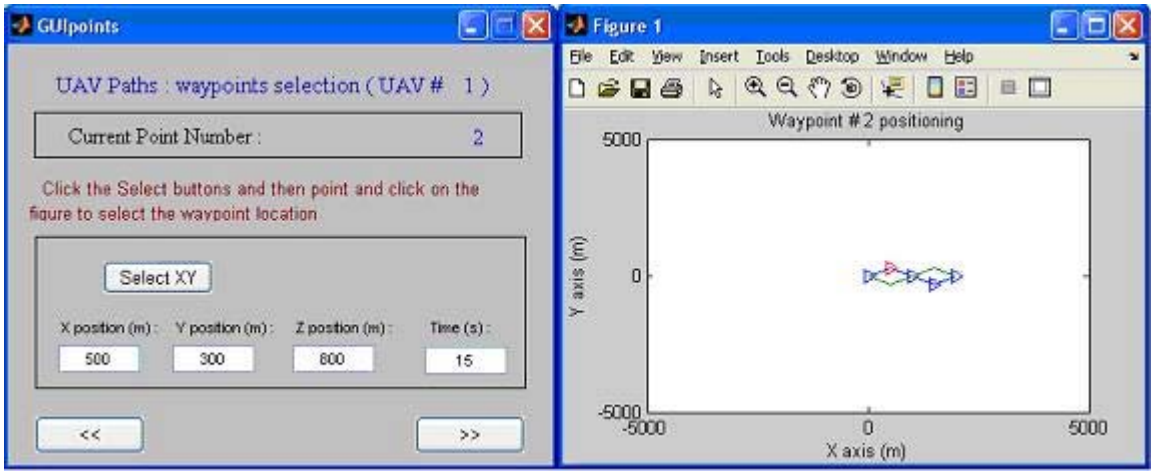


Figure 42: GUI used to choose the trajectory and the corresponding trajectory plotted

Finally, the GUI allows the user to select the position of the static balloons. Figure 43 shows typical scenarios of the VRE, Particularly, in the left part it is visible the static balloon and one building, the right part shows the UAV #2 crossing the trajectory of the UAV #1 and a highway.

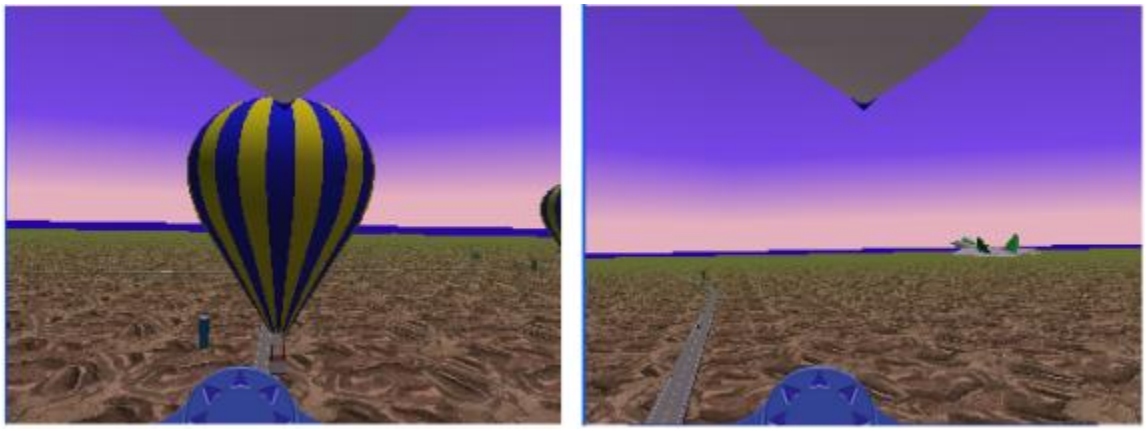


Figure 43: examples of the VRE, in the left balloons and building are visible, in the right the UAV #2 cross the trajectory of UAV #1.

3.2 The Optical Flow

The approach used for identifying the potential collisions is based on the use of the Optical Flow. According to Horn and Schunck [49]: “The optical flow is a velocity

field in the image that transforms one image into the next image in a sequence. As such it is not uniquely determined; the motion field, on the other hand, is a purely geometric concept, without any ambiguity – it is the projection into the image of three-dimensional motion vectors". In general, the OF calculation relies on three fundamental assumptions:

Brightness Constancy: the local changes in image intensity are caused only by the motion of a certain object with respect to the camera.

Spatial Coherence: the motion is uniform over a small patch of pixels

Temporal Persistence: the image motion of a surface patch changes gradually over time.

Analytically, if $I(u,v,t)$ is the intensity (i.e. brightness) of a pixel – having horizontal and vertical image plane coordinates u,v – representing a feature that moves of δu, δv during the time δt, then, under the three assumptions mentioned above, $I(u,v,t) = I(u+\delta u, v+\delta v, t+\delta t)$. A derivation with respect to time leads to the following conservation equation:

$$I_u \dot{u} + I_v \dot{v} + I_t = 0 \tag{57}$$

where I_u and I_v are the spatial derivatives of the image along the u and v image dimensions, calculated at a given pixel location, I_t is the temporal derivative of the image at that location, and the terms $\dot{u}$ and $\dot{v}$ represent the "optical flow" vector at that point and time.

In general, OF algorithms can be classified within the following broad classes:

- "Gradient" methods;
- "Phase" methods;
- "Matching" methods;

- "Feature-based" methods.

3.2.1 "Gradient" Methods

Generally speaking, gradient methods rely on the solution of equation (57) for calculating the unknowns $\dot{u}$ and $\dot{v}$. However, it can be noticed that for each pixel there is *one* corresponding scalar equation (with known values of I_u, I_v and I_t) while there are *two* scalar unknowns $\dot{u}$ and $\dot{v}$. This leads to an analytically under-determined algebraic system, also known as the "aperture problem". The methods belonging to the "Gradient" class typically tackle this problem by including some constraints – usually based on some form of spatial or temporal coherence - in the algebraic system of equations to be solved. Within this effort, 4 algorithms belonging to this category have been analyzed:

- "Gradient" method [47],
- "Lucas-Kanade" method [48],
- "Horn and Shunck" [49] method,
- "Proesmans" [50] method.

The "Gradient" algorithm – which was developed as a Matlab function at WVU - calculates the OF for each pixel belonging to a predefined grid assuming that $\dot{u}$ and $\dot{v}$ are constant within a certain spatial and temporal neighborhood of the pixel. Therefore, an over-determined system of equations is assembled and solved – in the minimum square sense – for each pixel. Specifically, the system is solved only if its eigenvalues are greater than a given set of thresholds. This allows discarding image areas where derivatives are too close to zero or too similar to each other – e.g. due to a lack of motion or because there are no distinguishable features – and, at the same time, increasing

computational efficiency by avoiding unnecessary calculations. The “Lucas-Kanade” and “Horn and Shunck” implementations are available in recent Matlab versions as Simulink blocks.

The “Lucas-Kanade” is fairly similar to the “Gradient” implementation with the main difference being the assignment of numerical weights to give different weights to the neighborhoods as function of their distances from the center.

The “Horn and Schunck” algorithm combines equation (57) with a global smoothness term λ with the goal of constraining the estimated velocity. This algorithm also features an iterative procedure that is halted when the maximum number of iterations is reached.

The “Proesmans” method is in its concept similar to the “Horn and Schunck” method since it requires the minimization of a global energy functional, the main difference is it takes into account the bias in the direction of motion due to correlations in the finite difference approximation [50]. The algorithm was originally developed in C++ at the University of Otago, New Zealand, and it was later revised and adapted. Specifically, all the sub-functions were included within a single C++ file and an interface that allowed the algorithm to be called from Matlab was added.

The classic advantage of this class of algorithms is their computational speed. Their main disadvantages are that they need to cope with the aperture problem; additionally, the calculation of the spatial and temporal derivatives is usually very prone to errors due to the presence of a number of noises from different sources.

3.2.2 Phase Methods

Phase techniques are based on the idea that 2D image velocity can be modeled as the phase behavior of a band-pass filter output [53]. The idea of using phase information for OF calculation purposes was originally developed by Fleet and Jepson [51]. The resulting Matlab-coded algorithm used within this effort is available from the Matlab Central file exchange site [52]. As outlined in [53], the algorithm calculates the OF estimation using the following three sequential steps. First, a spatial filtering is obtained using the Gabor Filter and the temporal phase gradient is calculated using the estimation of the velocity components. Next, a component velocity is rejected if the corresponding filter pair's phase information is not linear over a given time span. Finally, an interpolation is used to combine the partial velocities obtained using Gabor Filters in order to achieve the optical flow in the *u* and *v* directions.

3.2.3 Matching techniques

For methods belonging to the "Matching" class, the optical flow vector $[\dot{u}, \dot{v}]$ is calculated for a given pixel by finding the displacement of a template around the pixel between two consecutive frames. The template matching between two consecutive frames is usually performed by minimizing a predefined function of the differences between the two templates. Within this effort, the "Difference" and the "Correlation" methods have been considered and implemented. The "Difference" algorithm uses the Sum of the Absolute Differences (SAD) among templates belonging to consecutive frames to find the best matching templates [47]. The "Correlation" algorithm instead calculates the correlation among templates to perform the matching. Both algorithms

were developed at WVU and coded in Matlab. It should be emphasized that algorithms belonging to this category have shown to be computationally more demanding than the algorithms belonging to the "Gradient" category.

3.2.4 Feature-based methods

These methods calculate $\dot{u}$ and $\dot{v}$ by measuring of the displacements of certain image features – as detected by a feature detection algorithm and later associated by a feature matching algorithm - between two consecutive frames. Therefore, they implicitly rely on the assumption that the same image features can consistently be detected and associated over different image frames. It should be emphasized that such methods provide OF results for image points that not only do not belong to an evenly spaced grid – as for the OF provided by previously mentioned classes – but are typically located at different image points for different image frames.

In this effort two different feature based methods – named directly after their internal feature detection algorithms - have been considered, that is the Harris [27] and the SIFT [54] methods. Specifically, the Harris algorithm – coded in Matlab using the description in [55] - is a corner detection algorithm that allows extracting the position of specific corners with some robustness to real world conditions such as, for example, changes in the illumination. Next, a point-matching algorithm has to be used to match the corners detected within to consecutive frames of an image sequence. In this effort, an algorithm previously developed at WVU and coded in C [55] has been used to perform the point-matching task following the corner detection performed by the Harris algorithm.

The other feature detection algorithm is known as SIFT (Scale Invariant Feature Transform). SIFT has been developed with the goal of detecting and associating the same features between different images. Specifically, features are detected using a filtering approach that identifies stable points in the scale space, and are then associated using a descriptor-based approach [54]. Empirical experience has shown that the accuracy of the OF methods belonging to this class strongly depends on the performance of the associated matching algorithm. The version used within this effort was an executable file called from Matlab.

3.3 Derivation of the Ideal Optical Flow

The comparison between the different Optical Flow algorithms is based on the calculation of the "Ideal" Optical Flow (or Ideal Flow) generated by the motion of an object in space. Given any point on the image plane, the ideal flow can be calculated from the position and the velocity - with respect to the camera - of the point in the field of view that generates - by projection - the optical flow.

Specifically, a 'pin-hole' mathematical model of the camera [57] is assumed:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{f}{x_c} \begin{bmatrix} y_c \\ z_c \end{bmatrix} \tag{58}$$

where f is the camera focal length, u and v are – as previously described - the horizontal and vertical coordinates of a point in the image plane resulting from the projection of the point ${}^{C}P=[x_c, y_c, z_c]^T$ on such plane.

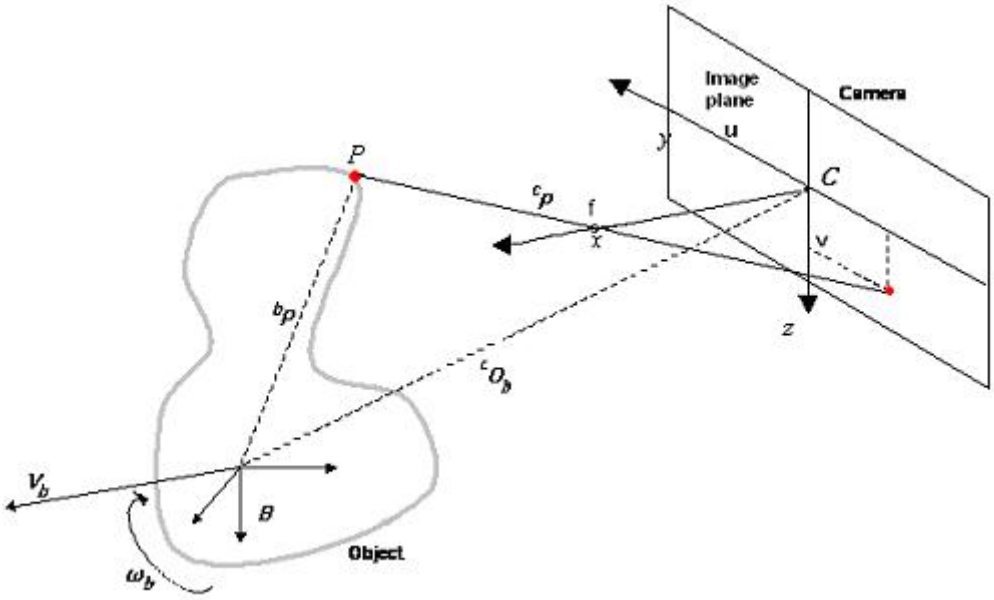


Figure 44: Pinhole model

Note that the left superscript "*C*" in ${}^{C}P$ indicates that the point is expressed with respect to a camera-fixed reference frame, which is centered in the camera plane and has its *x*-axis pointing in the direction of view, and its *y* and *z*-axis pointing respectively in the directions of *u* and *v* of the image plane. Assuming that ${}^{C}P$ is part of a rigid body centered in ${}^{C}O_{B}$ and moving with respect to the camera reference frame with a linear velocity ${}^{C}V_{B/C}$ and angular velocity ${}^{C}\omega_{B/C}$, differentiating (58) with respect to time, and using standard kinematics relationships to express the derivative of ${}^{C}P$ yields:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = f \begin{bmatrix} -\frac{y_c}{x_c^2} & \frac{1}{x_c} & 0 \\ -\frac{z_c}{x_c^2} & 0 & \frac{1}{x_c} \end{bmatrix} \left[{}^{C}V_{B/C} + {}^{C}\omega_{B/C} \otimes \left({}^{C}P - {}^{C}O_{B} \right) \right] \tag{59}$$

where $\dot{u}$ and $\dot{v}$ represent the *ideal* optical flow (at the image coordinates *u* and *v*) generated by the motion of ${}^{C}P$ and $\otimes$ indicates the three-dimensional cross product.

3.3.1 Calculation of the points relative to the ground

Applying (59) to calculate the "ideal" OF generated by the relative motion between the camera and a generic terrain point P, requires the calculation of the coordinates of the point P with respect to the camera, that is ${}^{C}P$.

Assuming that the terrain is flat with a constant altitude z_e^*, the homogeneous coordinates [82] of the point P with respect to the earth reference frame are ${}^{E}P = [x_e, y_e, z_e^*, 1]^{\mathrm{T}}$. The homogeneous coordinates of P in camera reference frame are given by ${}^{C}P = {}^{C}T_E(\psi,\theta,\varphi,{}^{E}O_C){}^{E}P$, where ${}^{C}T_E(\psi,\theta,\varphi,{}^{E}O_C)$, is the 4 by 4 matrix transforming earth-frame coordinates in camera-frame coordinates, the Euler angles ψ, θ, and φ, express the orientation of the camera reference frame with respect to the earth reference frame, and the vector ${}^{E}O_C = [x_o, y_o, z_o, 1]^{\mathrm{T}}$, express the position of the origin of the camera reference frame with respect to the earth reference frame.

The coordinates x_e, and y_e can then be found by setting the projection on the image plane of the point ${}^{C}P = {}^{C}T_E(\psi,\theta,\varphi,{}^{E}O_C){}^{E}P$ to the current image point $[u, v]$ from which the OF vector originates. Specifically, the MATLAB® Symbolic Toolbox [58] was used to obtain a formula yielding the earth frame coordinates x_e and y_e of a generic terrain point ${}^{E}P$, as a function of the image plane coordinates u, v, as well as ψ, θ, and φ, and ${}^{E}O_C$. In this process, the image points that did not correspond to physical points belonging to the terrain (such as for example points above the horizon) were automatically discarded.

Once the coordinates x_e and y_e were found as a function of the variables $f,u,v,\psi,\theta,\varphi,{}^{E}O_C$, the point ${}^{C}P$ was calculated using the formula ${}^{C}P = {}^{C}T_E(\psi,\theta,\varphi,{}^{E}O_C){}^{E}P(f,u,v,\psi,\theta,\varphi,{}^{E}O_C)$, resulting in:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \frac{\left(u\,\mathrm{c}_\theta\, z_o\, \mathrm{s}_\varphi - u z_e^*\, \mathrm{s}_\varphi\, \mathrm{c}_\theta - z_e^* v\, \mathrm{c}_\theta\, \mathrm{c}_\varphi + z_o v\, \mathrm{c}_\theta\, \mathrm{c}_\varphi - f\, \mathrm{s}_\theta\, z_e^* + f\, \mathrm{s}_\theta\, z_o \right)}{\left(-u^2 c_\theta{}^2 + u^2 c_\theta{}^2 c_\varphi{}^2 - 2 s_\varphi v c_\theta{}^2 c_\varphi u - f^2 c_\theta{}^2 - v^2 c_\theta{}^2 c_\varphi{}^2 + f^2 \right)} \begin{bmatrix} f \\ u \\ v \end{bmatrix} \tag{60}$$

where the letters "c" and "s" denote respectively the cosine and sine functions of the Euler angle indicated as a subscript.

The first three coordinates of ${}^{C}P$ were then used within (59) - along with the known translational and rotational velocity of the camera with respect to the terrain - to calculate the "ideal" OF at the point [*u, v*].

3.4 Optical Flow Comparison in Simple Motion

Three specific sets of experiments were developed with the objective of comparing the difference of the OF algorithms listed above. Specifically, the 1st set of experiments investigated the OF produced by a rotating disk, while the 2nd set of experiments investigated the OF produced by attaching the disk to a cart sliding in the horizontal image direction (*u* direction). The 3rd set of experiments was designed for the analysis of the OF produced by a forward translation of the disk toward the camera. Within each experiment, 3 different videos were recorded and analyzed. The videos were recorded using a "Qware EasyCam WB-001" along with its data acquisition software – with a focal length of 864.2 pixels, horizontal and vertical sizes respectively of 320 and 240 pixels and a frame rate of 10 frames per second. Each video of the same set differs from the other from the picture glued to the wooden disk. In fact, in the first video of each set of experiment, the glued picture is a chessboard having 0.64 cm wide black-and-white squares. In the second video of each experiment, the glued picture features a

picture having a white background and 1.27 cm wide squares each with different gray intensity. In the third video of each experiment, the glued picture features an image of the Jackson's Mill (WV) airstrip taken in flight from a camera mounted on the WVU-YF22 UAV aircraft model [56]. The nine recorded videos (three for each experiment) were selected to represent the different types of motion that an object can typically undergo in a 3D space. The results are summarized in tables; in each table, the algorithms providing the best performance are marked in bold.

3.4.1 Rotating Disk experiments

For this set of experiments, a given picture was glued to a wooden disk, which was in turn attached to a DC motor. Different rotational velocities and different pictures were selected for each video. A section of 120 frames was then selected from each video for performing the analysis. Two different criteria were selected for comparing the results from the OF algorithms, that is the overall angular velocity error, angular and magnitude errors with respect to the 'ideal' flow. The criteria are briefly discussed below.

3.4.1.1 Overall Angular Velocity Error:

An estimate of the disk angular velocity was calculated using the OF vectors supplied by each of the algorithms. Specifically, for each OF vector the point ${}^{C}P$ was calculated using the coordinates of the center of the disk – which was also set to be the center of rotation - and the initial position of the optical flow vector. Then, for each OF vector, a corresponding estimated disk angular velocity was calculated by setting to zero the last two components of the vector $\omega_{B/C}$ and the three components of the vector $V_{B/C}$ in

(59) and pseudo-inverting the formula. Specifically, zeroing out ${}^{C}V_{B/C}$ and the last two components of $\omega_{B/C}$ in (59) yields:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = f \begin{bmatrix} -\frac{y_c}{x_c^2} & \frac{1}{x_c} & 0 \\ -\frac{z_c}{x_c^2} & 0 & \frac{1}{x_c} \end{bmatrix} \begin{bmatrix} 0 \\ -\left({}^{C}\omega_{B/C}\right)_x \left({}^{C}P - {}^{C}O_B\right)_z \\ \left({}^{C}\omega_{B/C}\right)_x \left({}^{C}P - {}^{C}O_B\right)_y \end{bmatrix} \tag{61}$$

where $\left({}^{C}\omega_{B/C}\right)_x$ is the unknown, and $\left({}^{C}P - {}^{C}O_B\right)_y$ and $\left({}^{C}P - {}^{C}O_B\right)_z$ are the second and the third component of the vector $\left({}^{C}P - {}^{C}O_B\right)$. Multiplying the terms in (61) yields:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \frac{f}{x_c} \begin{bmatrix} -\left({}^{C}\omega_{B/C}\right)_x \left({}^{C}P - {}^{C}O_B\right)_z \\ \left({}^{C}\omega_{B/C}\right)_x \left({}^{C}P - {}^{C}O_B\right)_y \end{bmatrix} \tag{62}$$

At this point, it is possible to solve for $\left({}^{C}\omega_{B/C}\right)_x$ in two different ways, that is, by using the first equation in (62) (in the $\dot{u}$ component only) or by using the second equation in (62) (in the $\dot{v}$ component only). In this study, both approaches were pursued and the final value for the angular velocity for a given optical flow vector was obtained by averaging the two outcomes:

$$\left({}^{C}\omega_{B/C}\right)_x = \frac{1}{2}\left(\frac{\dot{v}x_c}{f\left({}^{C}P - {}^{C}O_B\right)_y} - \frac{\dot{u}x_c}{f\left({}^{C}P - {}^{C}O_B\right)_z} \right) \tag{63}$$

The total angular velocity was then calculated by averaging the estimated velocities over the whole image frame and over all the video frames, and compared with the known (recorded) disk angular velocity. This yielded the first evaluation criteria for the OF algorithms.

3.4.1.2 Angular and Magnitude Errors w.r.t the Ideal Flow:

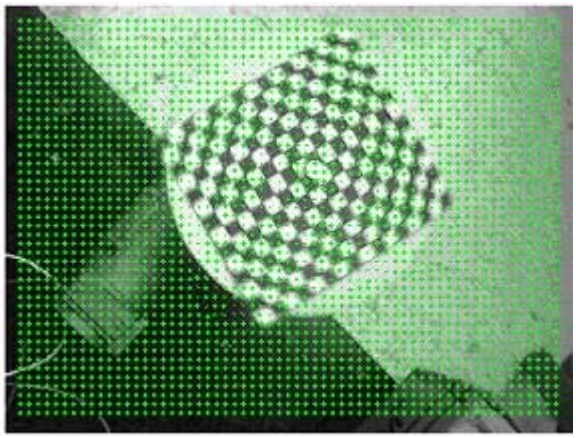
For each frame of the video, the "ideal" OF was calculated for each image point by substituting the known disk angular velocity and the point position in (59). Both the "ideal" flow vector and the flow vector detected from the OF algorithms were then expressed in polar coordinates, resulting in a magnitude and an angle value – with respect to the u axis - for each vector. Finally, the errors in the magnitude and the angle were calculated as the differences between the "ideal" and the "detected" values.

3.4.1.3 Results of the test

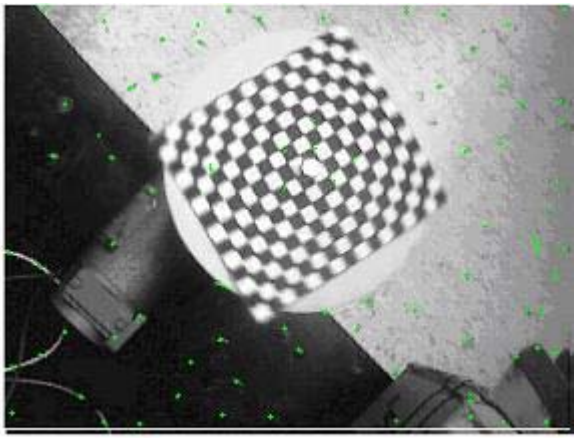
Examples of the OF field obtained for the rotating disk experiment - using the nine algorithms and the video #1 - are shown in Figure 45.

Correlation OF

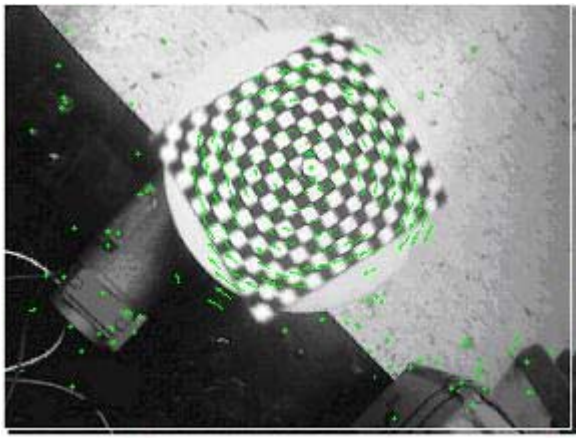
Difference OF



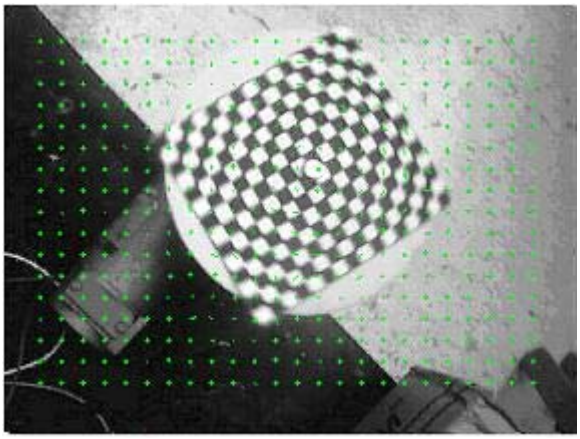
Gradient OF



Harris OF



SIFT OF



Phase OF

Proesmans OF

Lucas-Kanade

Horn and Shunk OF

Figure 45: A frame of the video #1 with the OF derived from the nine algorithms

The results of the analysis are reported in the following Table 15 - Table 17. Specifically, each table reports the results obtained by all the nine algorithms when applied to a given video.

	ω_{real}	ω_{calc}	*Std*	*Err %*	Ang_{err}	Mag_{err}
Correlation	-0.112	-0.057	0.015	49.6	25.8	1.65
Difference	-0.112	-0.044	0.009	60.8	56.4	4.10
Gradient	-0.112	-0.012	0.001	89.3	41.7	5.26
Harris	-0.112	-0.084	0.062	25.0	50.5	2.14
SIFT	**-0.112**	**-0.109**	**0.004**	**2.9**	**5.9**	**0.51**
Phase	-0.112	-0.035	0.004	69.0	50.4	3.72
Proessmans	-0.112	-0.061	0.009	45.3	28.2	2.78
Lucas- Kanade	-0.112	-0.017	0.002	85.2	68.1	4.13
Horn -Schunck	-0.112	-0.037	0.006	67.0	73.7	5.45

Table 15: Video #1 (rotating disk experiment)

	ω_{real}	ω_{calc}	*Std*	*Err%*	Ang_{err}	Mag_{err}
Correlation	**0.0357**	**0.0325**	**0.003**	**9.0**	**11.0**	**0.45**
Difference	0.0357	0.0343	0.005	3.8	29.4	1.75
Gradient	0.0357	0.0129	0.001	64.0	12.0	1.18
Harris	**0.0357**	**0.0353**	**0.010**	**1.1**	**30.8**	**0.83**
SIFT	**0.0357**	**0.0322**	**0.022**	**9.8**	**11.7**	**1.27**
Phase	0.0357	0.0224	0.003	37.2	20.5	0.67
Proessmans	0.0357	0.0136	0.002	62.0	14.7	1.20
Lucas- Kanade	0.0357	0.0104	0.002	70.8	54.7	1.23
Horn -Schunck	0.0357	0.0077	0.001	78.6	60.2	1.53

Table 16: Video #2 (rotating disk experiment)

	ω_{real}	ω_{calc}	*Std*	*Err %*	Ang_{err}	Mag_{err}
Correlation	**0.0503**	**0.0381**	**0.003**	**24.1**	**16.4**	**0.84**
Difference	0.0503	0.0417	0.005	17.1	28.5	1.62
Gradient	0.0503	0.0121	0.002	76.0	24.4	2.08
Harris	**0.0503**	**0.045**	**0.015**	**10.5**	**25.6**	**1.05**
SIFT	**0.0503**	**0.0495**	**0.023**	**1.5**	**10.1**	**0.44**
Phase	0.0503	0.0282	0.003	43.9	26.9	1.13
Proessmans	0.0503	0.0197	0.003	60.8	21.5	1.64
Lucas- Kanade	0.0503	0.0132	0.002	73.7	52.9	1.65
Horn -Schunck	0.0503	0.0163	0.002	67.5	51.5	1.80

Table 17: Video #3 (rotating disk experiment)

In the above tables, ω_{real} is the 'true' rotational velocity of the disk calculated off-line in radiant per frame. ω_{calc} is instead the average value over 120 frames of the rotational velocity which calculated from the OF field provided by each algorithm. The standard deviation (*Std*) and percentage error between the 'true' and the calculated velocity are also shown in the table. The columns Ang_{err} and Mag_{err} contain the average angular and magnitude errors - measured respectively in degrees and pixels - for each algorithm.

3.4.2 Sliding Cart Experiments

This set of experiments consisted in attaching the same pictures of the previous experiment to a toy train which was used as a moving cart. Since the camera was positioned on the train side, a pure translational motion of the picture on the *v* axis was recorded. Each video featured a different picture while the train velocity was kept constant among the different videos. Only a subset of the video frames, specifically the 15 frames in which the train was within the camera field of view, was used for this analysis. As for the previous experiment, the overall velocity error and the angular and magnitude errors with respect to the 'ideal' flow were used as performance metrics:

3.4.2.1 Overall Velocity Error:

An estimate of the train velocity was calculated using the OF vectors supplied by each algorithm. Specifically, for each optical flow vector, a corresponding estimated train velocity is calculated by setting to zero both $\omega_{B/C}$ and the first component of $V_{B/C}$ in (59) and inverting the formula. Specifically, zeroing-out ${}^{C}\omega_{B/C}$ and the first component of ${}^{C}V_{B/C}$ yields:

$$\begin{bmatrix} \left({}^{C}V_{B/C}\right)_y \\ \left({}^{C}V_{B/C}\right)_z \end{bmatrix} = \frac{1}{f}\begin{bmatrix} x_c & 0 \\ 0 & x_c \end{bmatrix}\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} \tag{64}$$

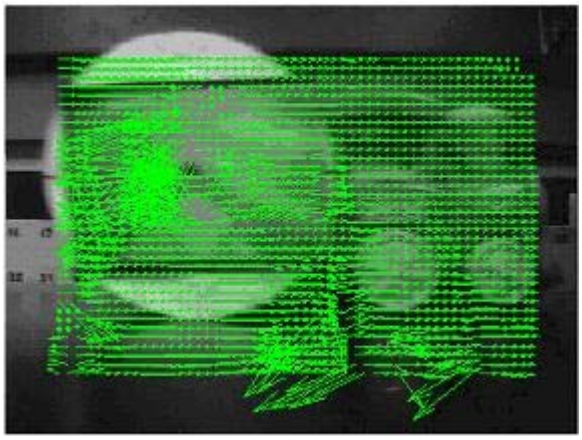
For each axis, the total velocity is then calculated by averaging the estimated velocities over the whole image frame and over all the video frames. Comparing the total velocity with the known recorded train velocity, yields the first evaluation criteria for the OF algorithms.

3.4.2.2 Angular and Magnitude Errors w.r.t the Ideal Flow:

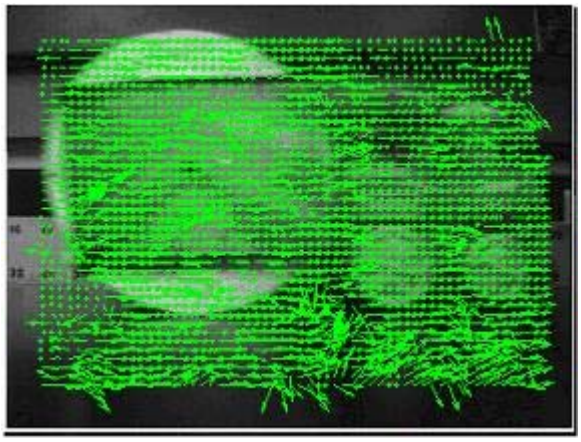
For each video frame, the "ideal" OF was calculated in each image point using (59) along with the known train velocity. Both the "ideal" flow vector and the flow vector detected from the algorithms were then expressed in polar coordinates, resulting in a magnitude and an angle value, measured respectively in pixels and degrees. Finally, the errors in magnitude and angle were calculated as the difference between the "ideal" and the "detected" values, and averaged over all the OF vectors belonging to all the used video frames, yielding the second evaluation criteria for the considered OF algorithm

*3.4.2.3 **Results of the test***

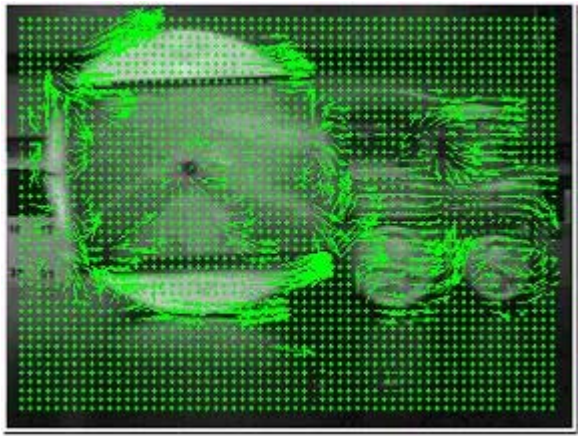
Examples of the OF field obtained for the sliding cart experiment in the video #3- using the nine algorithms - are shown in Figure 46, while Table 18 - Table 20report the results obtained from all the nine algorithms for each of the 3 videos.



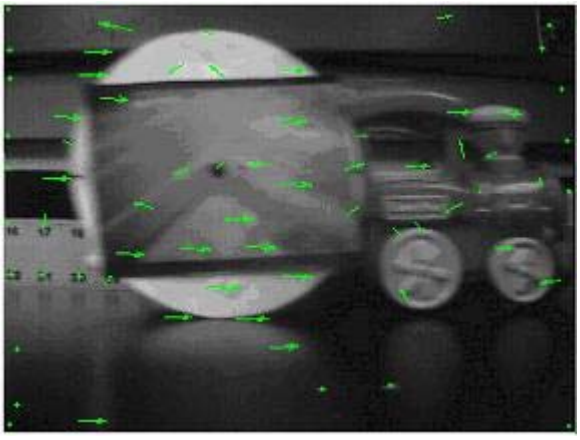
Correlation OF



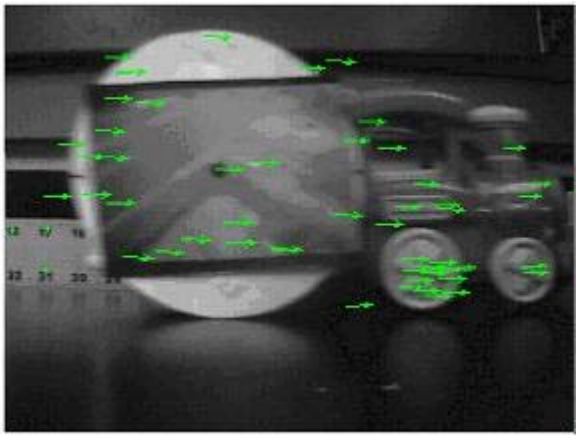
Difference OF



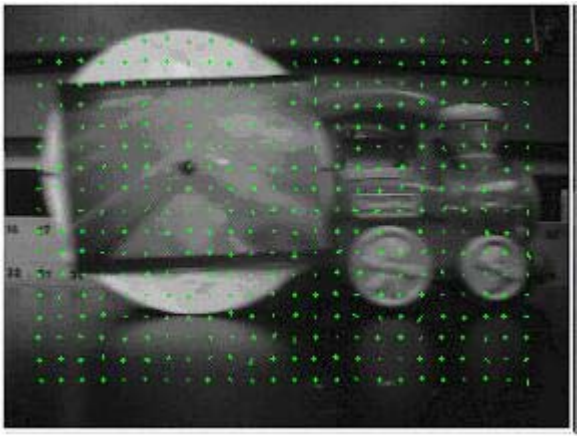
Gradient OF



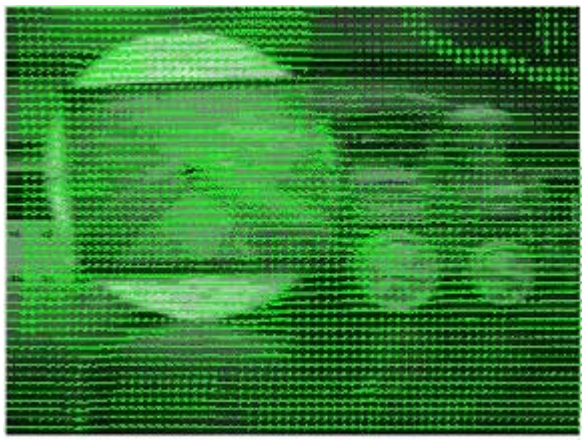
Harris OF



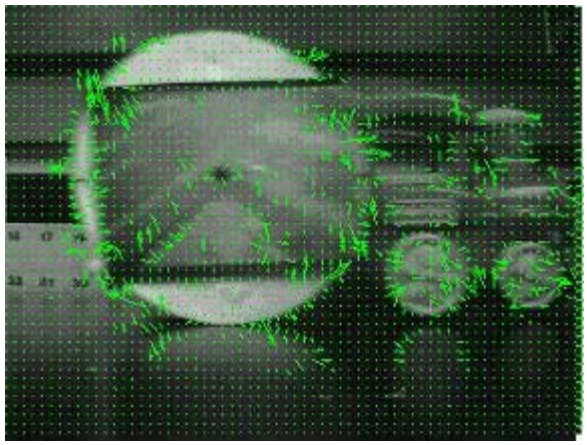
SIFT OF



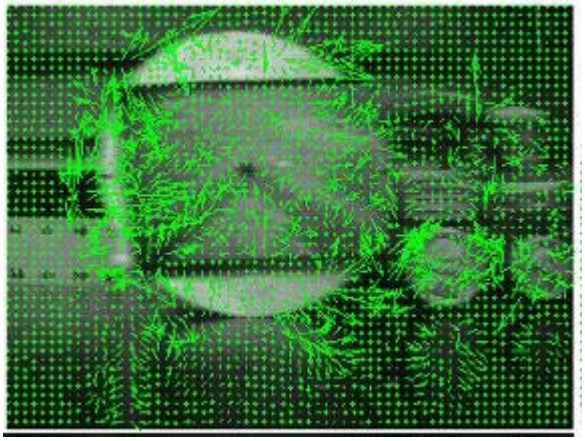
Phase OF



Proesmans OF



Lucas-Kanade



Horn and Shunk OF

Figure 46: A frame of the video #3 with the OF derived from the nine algorithms

	V_z	*Real* V_y	V_y	Std_y	Ang_{err}	Mag_{err}
Correlation	0.73	14.55	5.15	1.05	70.48	11.13
Difference	0.34	14.55	6.18	2.10	55.41	10.03
Gradient	0.41	14.55	-0.53	0.11	135.57	15.08
Harris	0.76	14.55	4.59	2.10	64.98	11.52
SIFT	**0.14**	**14.55**	**15.44**	**0.46**	**0.93**	**1.14**
Phase	1.21	14.55	-0.95	0.23	123.44	15.57
Proessmans	2.11	14.55	0.87	1.89	100.72	14.26
Lucas- Kanade	0.55	14.55	-0.95	0.21	130.19	15.56
Horn -Schunck	0.89	14.55	-0.94	0.18	122.07	15.89

Table 18: Video #1 (sliding cart experiment)

	V_z	*Real* V_y	V_y	Std_y	Ang_{err}	Mag_{err}
Correlation	0.73	14.55	3.72	2.07	78.20	12.06
Difference	**0.02**	**14.55**	**10.35**	**1.93**	**31.54**	**5.27**
Gradient	0.24	14.55	-0.15	0.23	117.93	14.72
Harris	1.38	14.55	-0.17	2.93	113.18	15.33
SIFT	**0.13**	**14.55**	**15.33**	**0.44**	**1.36**	**1.32**
Phase	0.58	14.55	-1.28	0.42	125.14	15.87
Proessmans	**-0.02**	**14.55**	**10.88**	**2.99**	**10.14**	**4.34**
Lucas- Kanade	0.21	14.55	0.019	0.44	90.79	14.83
Horn -Schunck	0.65	14.55	1.36	0.74	86.31	16.38

Table 19: Video #2 (sliding cart experiment)

	V_z	*Real* V_y	V_y	Std_y	Ang_{err}	Mag_{err}
Correlation	**0.29**	**14.55**	**10.51**	**0.71**	**10.26**	**6.57**
Difference	**-0.89**	**14.55**	**11.84**	**0.96**	**17.34**	**4.24**
Gradient	0.020	14.55	0.35	0.069	37.77	14.21
Harris	0.38	14.55	7.47	2.87	47.38	8.39
SIFT	**0.12**	**14.55**	**14.61**	**2.89**	**1.87**	**1.84**
Phase	0.12	14.55	-0.045	0.12	92.33	14.64
Proessmans	**-1.10**	**14.55**	**8.70**	**1.68**	**9.83**	**6.15**
Lucas- Kanade	0.033	14.55	0.26	0.13	79.17	14.44
Horn -Schunck	-0.63	14.55	1.52	0.39	71.68	14.13

Table 20: Video #3 (sliding cart experiment)

In the above tables, $RealV_y$ is the 'true' recorded velocity in the y direction. Note that the real velocity along the vertical camera axis z, that is $RealV_z$, has been considered to be 0. The columns V_z and V_y contain the average –throughout the frames where the sliding picture is visible - of the velocities extracted from the OF field in the y and z directions, measured in pixels/frame. Finally, Std_y is the standard deviation of the velocity in the u direction.

3.4.3 Forward Translation Experiments

Within this set of experiments, the pictures were attached in front of the train with the camera positioned along the longitudinal direction of the train. Therefore, the images of the picture became closer as the train moved forward. A section of 60 frames was

selected from each video to perform the analysis. As for the previous experiment, each video featured a different picture, while the train velocity was kept constant. The same performance metrics were used to compare the different OF algorithms as in the previous two sets of experiments. To calculate the 'ideal' OF from the known train position and velocity, a pin-hole mathematical model of the camera was used as described in previous section. Knowing the center of the disk and the radius it is possible to use (59) in order to find the Ideal Optical Flow for the forward translation movement.

3.4.3.1 Overall Velocity Error:

An estimate of the train velocity was calculated using the OF vectors supplied by each algorithm. Specifically, for each optical flow vector, a corresponding estimated velocity was calculated by setting to zero both $\omega_{B/C}$ and the third component of $V_{B/C}$ in (59) and pseudo-inverting the formula. Specifically, this yields:

$$\left({}^{C}V_{B/C}\right)_x = -\frac{x_c^2}{f}\begin{bmatrix} y_c \\ z_c \end{bmatrix}^{\dagger}\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} \tag{65}$$

where the sign † indicates the pseudo-inverse operation.

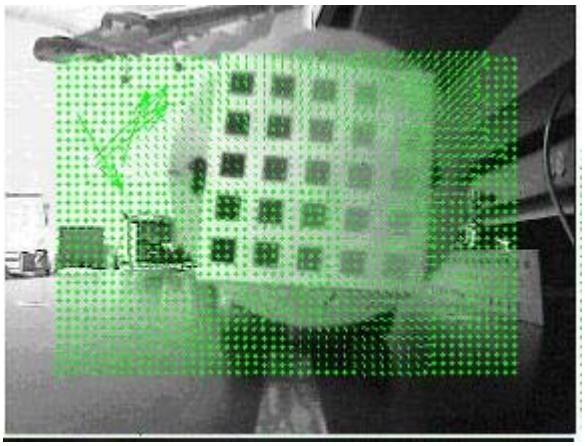
The total velocity was then calculated by averaging the estimated velocities over the entire frame and over all the used video frames and compared with the known train velocity, yielding the first performance criteria for the comparison of the OF algorithms.

3.4.3.2 Angular and Magnitude Errors w.r.t the Ideal Flow:

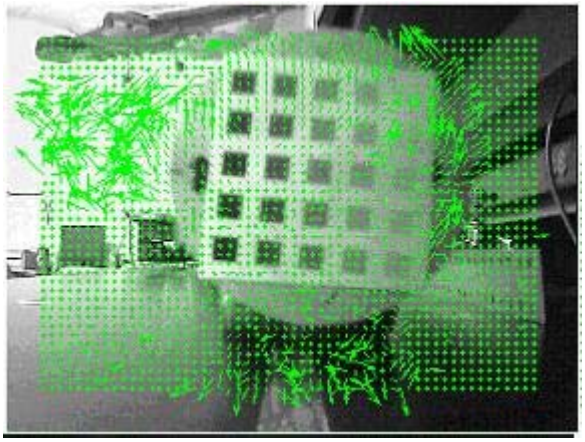
For each used video frame, the "ideal" OF was calculated at each image point - by substituting the true train velocity in (59) - and then expressed in polar coordinates. As for the previous experiments, the errors in magnitude and in angle were then calculated as the difference between the "ideal" and the "detected" OF magnitudes and angles.

*3.4.3.3 **Results of the test***

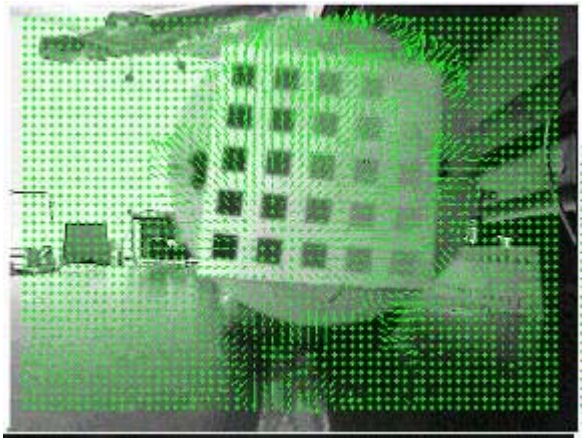
Examples of the OF field obtained for the forward translation experiment - using the nine algorithms and video #2- are shown in Figure 47.



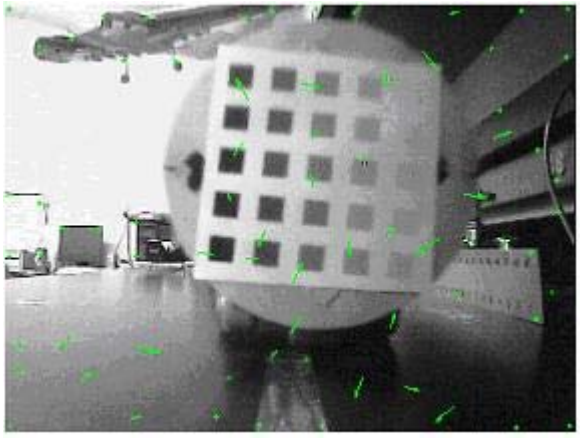
Correlation OF



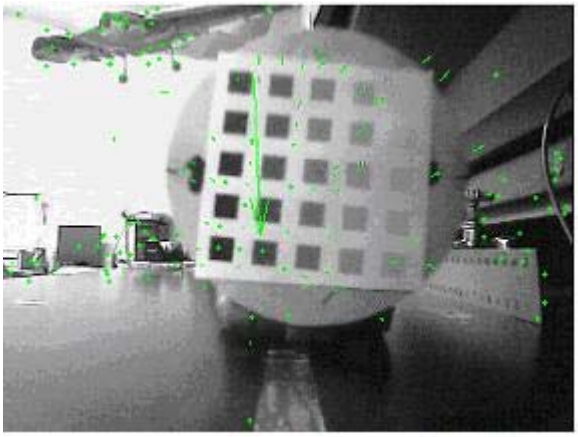
Difference OF



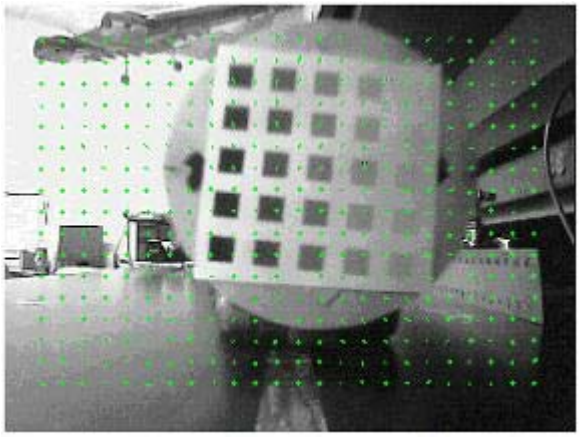
Gradient OF



Harris OF



SIFT OF



Phase OF

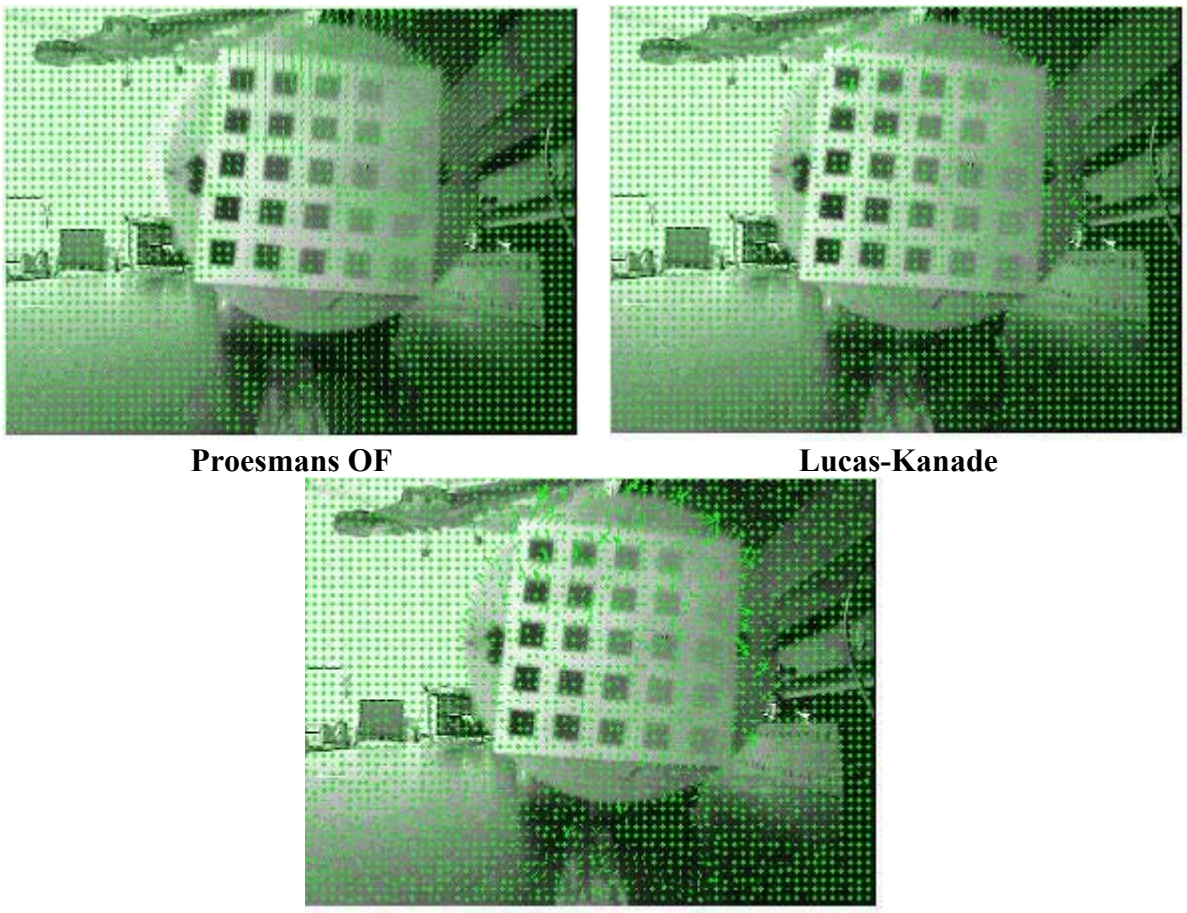


Figure 47: The nine algorithms in the forward translation experiment and video #2

Table 21 - Table 23 report the results obtained by the nine OF algorithms for each of the three videos.

	Real V_x	V_x	Std_x	*ERR* %	Ang_{err}	Mag_{er}
Correlation	-0.168	-0.34	1.42	-103	39.8	2.73
Difference	-0.168	-0.0085	0.48	95.0	75.1	8.50
Gradient	-0.168	-0.077	0.02	54.0	35.8	0.77
Harris	**-0.168**	**-0.164**	**0.77**	**2.7**	**63.8**	**3.73**
SIFT	**-0.168**	**-0.159**	**0.06**	**5.4**	**27.8**	**0.63**
Phase	**-0.168**	**-0.147**	**0.07**	**12.6**	**41.5**	**0.78**
Proessmans	-0.168	-0.135	0.01	19.6	37.7	0.60
Lucas- Kanade	-0.168	-0.103	0.02	38.8	43.7	0.82
Horn -Schunck	-0.168	-0.126	0.04	25.1	57.7	1.43

Table 21: Video #1 (forward translating experiment)

	Real V_x	V_x	Std_x	*ERR* %	Ang_{err}	Mag_{er}
Correlation	-0.168	0.004	0.75	102	26.0	0.78
Difference	-0.168	-0.006	0.62	96.3	49.3	3.70
Gradient	-0.168	-0.077	0.01	54.5	33.4	0.65
Harris	**-0.168**	**-0.160**	**0.32**	**5.23**	**54.2**	**1.29**
SIFT	**-0.168**	**-0.105**	**0.12**	**37.8**	**42.0**	**0.64**
Phase	**-0.168**	**-0.143**	**0.03**	**15.3**	**34.3**	**0.54**
Proessmans	-0.168	-0.094	0.01	44.3	41.3	0.66
Lucas- Kanade	-0.168	-0.084	0.02	50.4	53.3	0.84
Horn -Schunck	-0.168	-0.082	0.02	51.1	60.8	1.03

Table 22: Video #2 (forward translating experiment)

	Real V_x	V_x	Std_x	*ERR* %	Ang_{err}	Mag_{er}
Correlation	**-0.168**	**-0.231**	**0.07**	**-36.9**	**30.9**	**0.74**
Difference	-0.168	-0.101	0.23	39.9	52.1	2.91
Gradient	-0.168	-0.072	0.02	57.3	40.6	0.74
Harris	-0.168	-0.113	0.46	32.7	56.0	1.23
SIFT	**-0.168**	**-0.116**	**0.05**	**31.1**	**41.8**	**0.57**
Phase	**-0.168**	**-0.138**	**0.03**	**18.0**	**40.4**	**0.59**
Proessmans	-0.168	-0.056	0.02	66.6	61.3	0.82
Lucas- Kanade	-0.168	-0.059	0.02	65.0	63.0	0.92
Horn -Schunck	-0.168	-0.076	0.05	54.7	62.9	1.09

Table 23: Video #3 (forward translating experiment)

In the above tables, *Real* V_x is the 'true' recorded velocity in the x direction. Note that the real velocity along the axis y and z, that is *Real* V_y and *Real* V_z, has been considered to be 0 since the cart motion is parallel to the longitudinal depth axis of the camera. The column V_x contains the average – over the frames where the forward translation is analyzed - of the velocities extracted from the OF field in the x direction, measured in meter/second. Finally, Std_x is the standard deviation of the velocity in the x direction. The columns Ang_{err} and Mag_{err} contain the average angular and magnitude errors - measured respectively in degrees and pixels - for each of the algorithms.

3.5 Optical Flow Comparison in Complex Motion

3.5.1 Virtual Images Analysis

The algorithms are executed within a simulation environment that is linked to a Virtual Reality Toolbox® (VRT) [67] interface. Such interface allows the position and orientation of a flying aircraft in the simulation to drive the position and orientation of an associated visual model of the aircraft in a "virtual world", which was described in previous section.

In particular, within this effort, the scenario consisted only of a planar terrain situated 700m above the sea level, and featuring a repeated picture of a natural landscape, taken from Google Earth ®, the aircraft flies at initial altitude of 100 m above the terrain.

A window on this virtual scenario - featuring the view from a virtual camera placed on the aircraft - was made available to the user. Using functions provided by the Virtual Reality Toolbox, images from such camera – with horizontal and vertical sizes respectively of 320 and 240 pixels and focal length equals to 289.7 pixels - were continuously acquired – at a frame rate of 10 frames per second - and fed to the different optical flow algorithms during the simulation. Specifically, as represented in Figure 41, the Optical Flow was continuously calculated from each couple of consecutive images, using each of the nine available algorithms. The Ideal Flow was also computed using the procedure described in previous section.

3.5.1.1 Overall Velocity Error

It should be noticed that, contrary to the previous experiments, the (rotational and translational) velocity vectors could not be expressed directly – that is by directly inverting (59) - as a function of the instantaneous OF field *in a single point* in the image

plane. This happens because in this case there is no additional a-priori information about the structure of the motion (e.g. motion constrained along a certain known axis) that could be used to compensate for the information lost during the projection.

However, the overall information of *all* OF vectors in the image plane can still be used to estimate both the translational and rotational velocities in the image. Specifically, rearranging (59) by collecting the velocity terms in a 6 by 1 vector yields:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = M(f, {}^{C}P, {}^{C}O_B) \begin{bmatrix} {}^{C}V_{B/C} \\ {}^{C}\omega_{B/C} \end{bmatrix} \tag{66}$$

where $M(f, {}^{C}P, {}^{C}O_B)$ is the following 2 by 6 matrix:

$$M(f, {}^{C}P, {}^{C}O_B) = \frac{f}{x_c} \begin{bmatrix} -\frac{y_c}{x_c} & 1 & 0 & z_o - z_c & \frac{y_c(z_o - z_c)}{x_c} & \frac{y_c(y_c - y_o)}{x_c} + x_c - x_o \\ -\frac{z_c}{x_c} & 0 & 1 & y_c - y_o & \frac{z_c(z_o - z_c)}{x_c} + x_o - x_c & \frac{z_c(y_c - y_o)}{x_c} \end{bmatrix} \tag{67}$$

and the rotation center of the object is expressed as ${}^{C}O_B = [x_o, y_o, z_o, 1]^{\mathrm{T}}$.

Collecting - in a columnwise fashion - the optical flow vectors $[\dot{u}_i \;\; \dot{v}_i]^{\mathrm{T}}$ generated by the considered OF algorithm, along with their corresponding matrices $M(f, {}^{C}P_i, {}^{C}O_B)$ yields:

$$\begin{bmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \vdots \\ \dot{u}_N \\ \dot{u}_N \end{bmatrix} = \begin{bmatrix} M(f, {}^{C}P_1, {}^{C}O_B) \\ \vdots \\ M(f, {}^{C}P_N, {}^{C}O_B) \end{bmatrix} \begin{bmatrix} {}^{C}V_{B/C} \\ {}^{C}\omega_{B/C} \end{bmatrix} \tag{68}$$

Pseudo-inverting (68) yields

$$\begin{bmatrix} {}^{C}V_{B/C} \\ {}^{C}\omega_{B/C} \end{bmatrix} = \begin{bmatrix} M(f, {}^{C}P_1, {}^{C}O_B) \\ \vdots \\ M(f, {}^{C}P_N, {}^{C}O_B) \end{bmatrix}^{\dagger} \begin{bmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \vdots \\ \dot{u}_N \\ \dot{u}_N \end{bmatrix} \tag{69}$$

which, assuming $N \geq 6$, yields an estimation of the translational and rotational velocities of the object (i.e. of the terrain) at a certain time instant. For each time step, the velocities were then calculated according to (69) and compared with the velocities produced by the simulation, yielding the first performance criteria for the comparison of the OF algorithms.

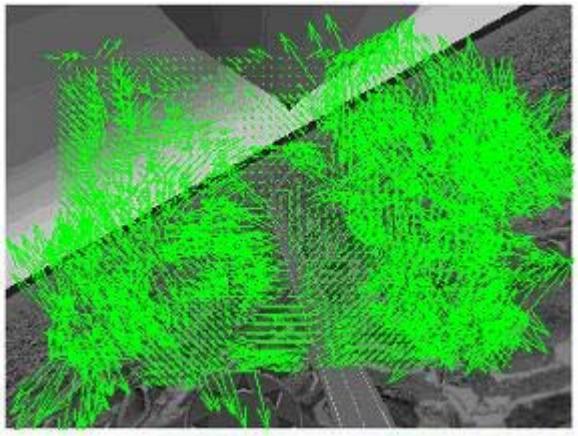
It should be noticed that, in the considered experiment, form the camera point of view, the object (that is the terrain) revolves around the camera, and therefore the first three coordinates of the rotation center ${}^{C}O_B$ are zero. Finally, to conclude this section, it should also be mentioned that the related problem of estimating a sequence of relative *positions and orientations* from the motion of several points in the image is typically classified as a "structure from motion" problem [83].

3.5.1.2 Angular and Magnitude Errors w.r.t the Ideal Flow

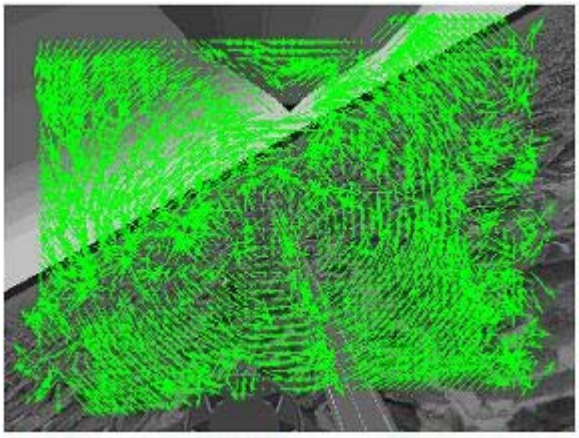
The first three coordinates of ${}^{C}P$ were used within (59) - along with the known translational and rotational velocity of the camera with respect to the terrain - to calculate the "ideal" OF at each point [*u, v*]. Expressing in polar coordinates both the ideal and the detected OF vectors - as for the previous experiments - allowed for the calculation of the errors in magnitude and in angle for each detected OF vector. Averaging the errors over all the OF vectors calculated during the simulation yielded the second performance metric for the evaluation of the different OF algorithms.

3.5.1.3 Results of the test

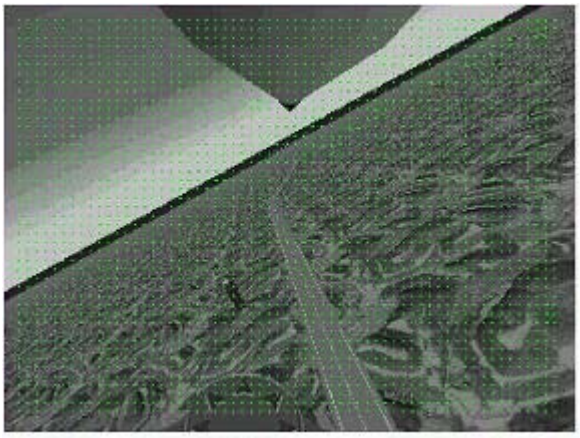
Samples of the results of the analysis conducted within this effort are shown in Figure 48. Figure 48 shows, also, the Ideal Optical Flow calculated as described in the previous sections, note that the Ideal OF can be derived only for the ground.



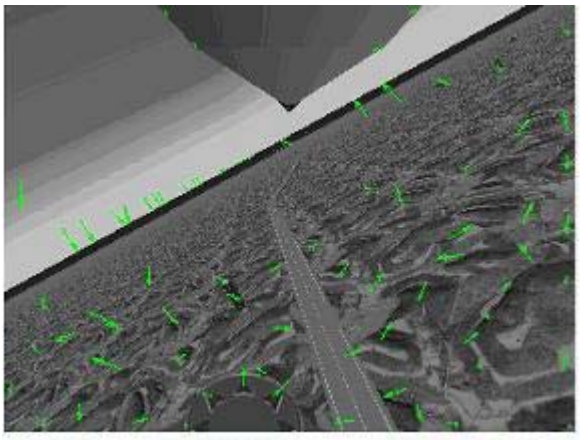
Correlation OF



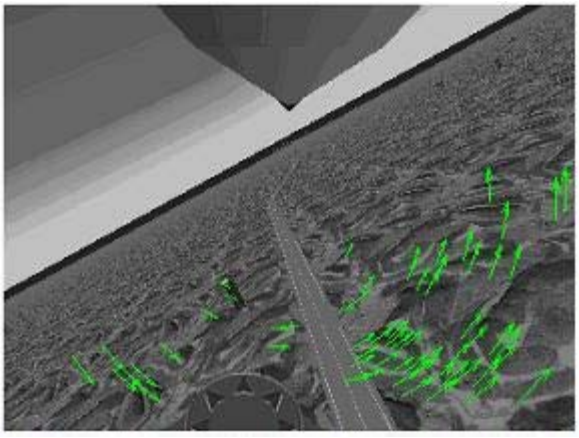
Difference OF



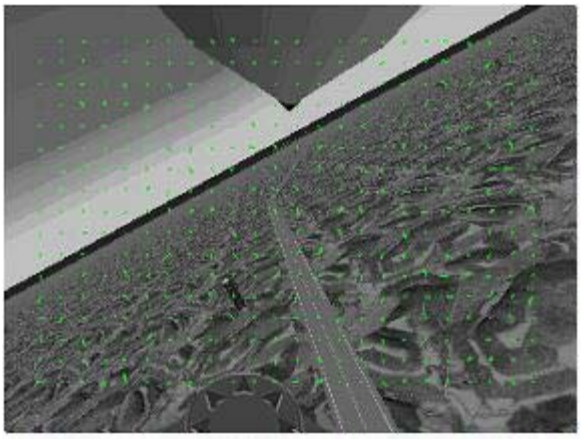
Gradient OF



Harris OF



SIFT OF



Phase OF

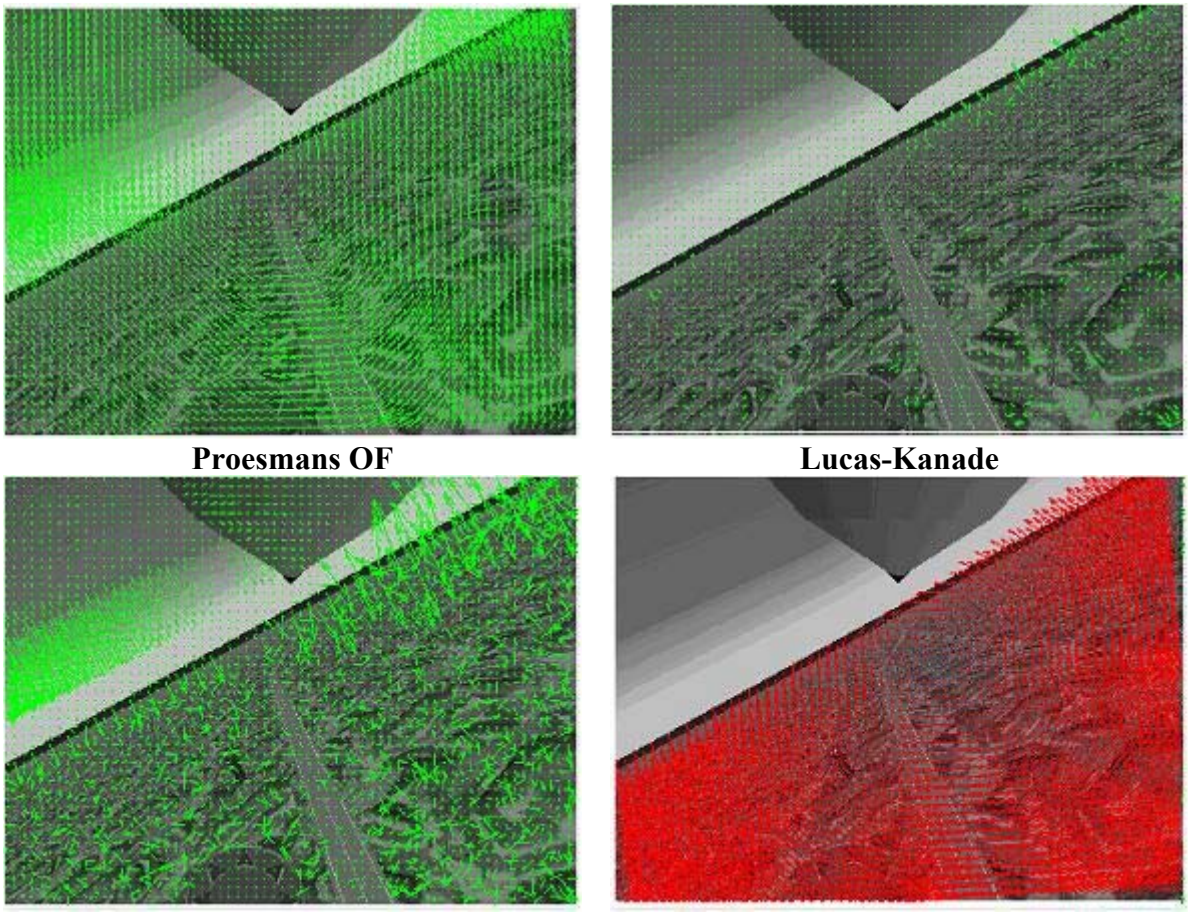


Figure 48: A frame of the complex motion experiment with virtual images with the OF derived from the nine algorithms

	Mean Angular Error (deg)	STD Angular Error	Mean Magnitude Error (pix)	STD Magnitude Error	*Norm of Mean error Velocity (m/s)*	*Norm of Mean error Angular Velocity (rad/s)*
Correlation	**24.48**	**17.13**	**1.65**	**3.02**	**9.57**	**0.62**
Difference	46.04	12.57	3.88	1.27	33.92	7.91
Gradient	53.599	14.13	3.93	3.14	3.87	3.15
Harris	61.81	13.82	2.69	1.54	7.31	2.85
SIFT	**21.46**	**16.13**	**3.31**	**5.79**	**21.09**	**3.47**
Phase	51.62	15.45	2.58	2.77	10.01	3.21
Proesmans	**30.76**	**11.18**	**3.08**	**2.26**	**10.22**	**3.54**
L- K	83.91	11.80	3.54	2.66	3.05	2.94
H S	54.85	40.46	2.22	2.53	3.21	3.08

Table 24: Experiment in the VRE

In Table 24, the algorithms are compared in terms of the mean and standard deviations of the errors, which were obtained by comparing the OF produced by the 9 algorithms with the Ideal OF calculated as described in the previous sections. A visual analysis revealed that the Difference, SIFT, and Proesman algorithms performed better. However, the statistical analysis in Table 24 showed that the Correlation and Proesman algorithms provided the best results.

It should be noticed that, contrary to the experiments regarding the simple motion, the velocity vectors could not be expressed directly – that is by directly inverting (59) - as a function of the instantaneous OF field *in a single point* in the image plane. This happens because in this case there was no additional a-priori information about the structure of the motion (e.g. motion constrained along a certain known axis) that could be used to compensate for the information lost during the projection. Estimating the relative position and motion from the motion of several points in the image plane can be considered as a "structure from motion" problem, which has been extensively investigated in the last decade [83].

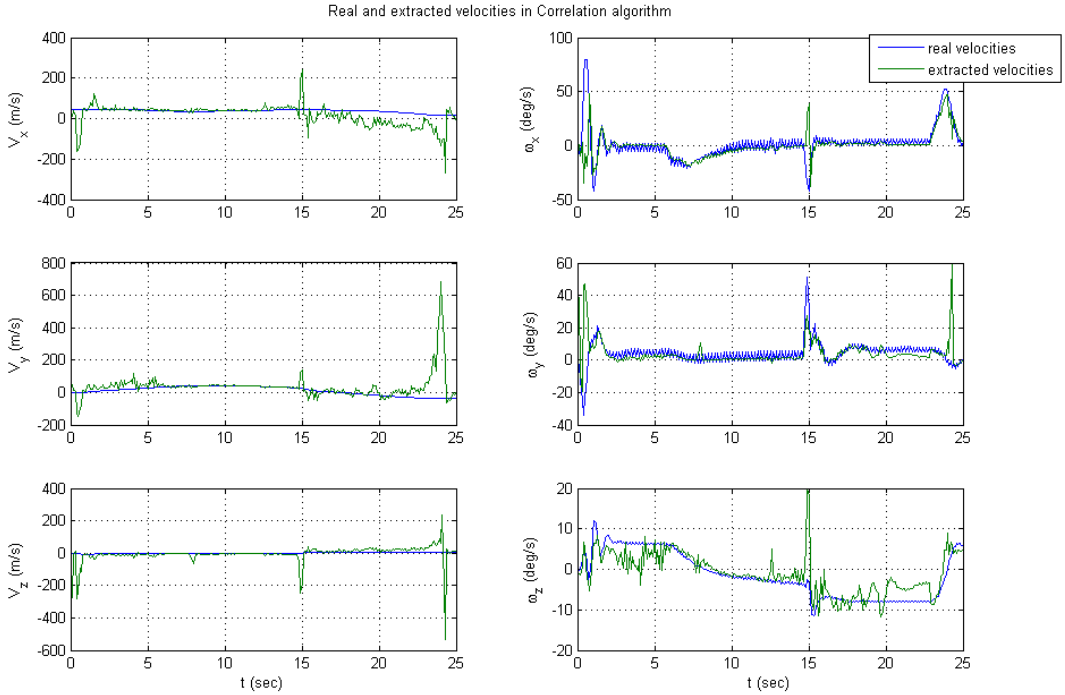


Figure 49: Comparison between real and extracted linear and angular velocities in the Correlation algorithm

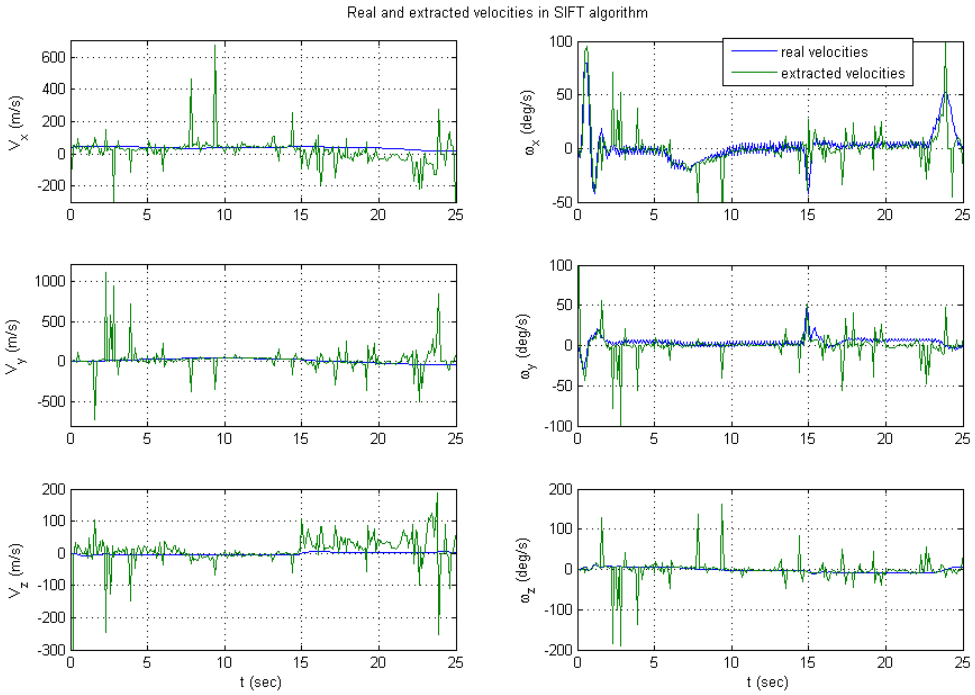


Figure 50: Comparison between real and extracted linear and angular velocities in the SIFT algorithm

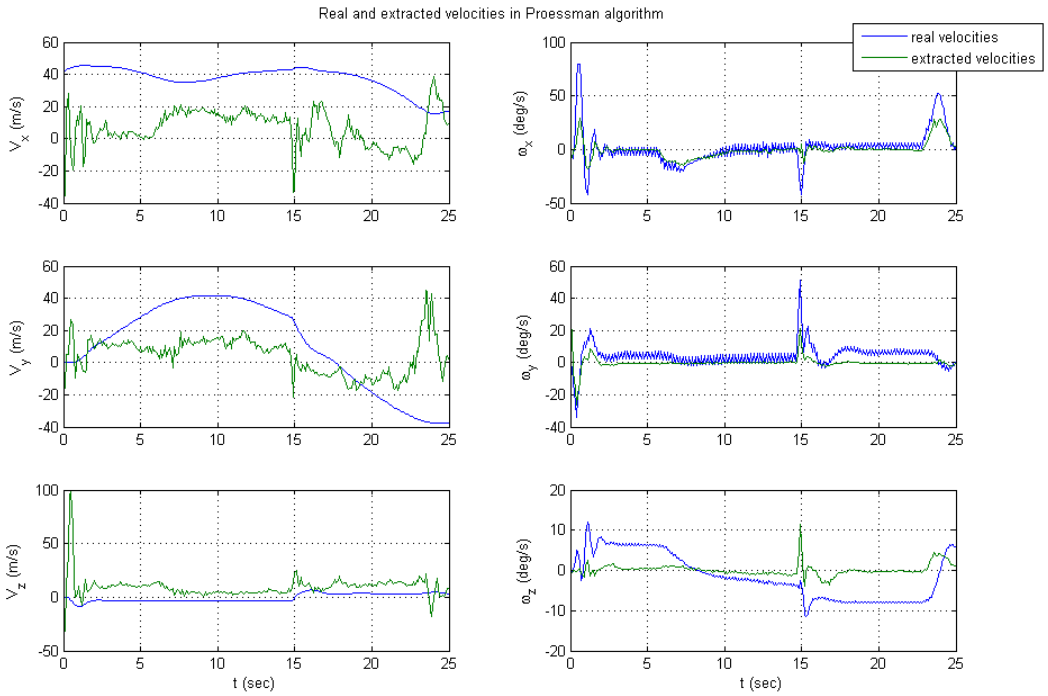


Figure 51: Comparison between real and extracted linear and angular velocities in the Proesman algorithm

Figure 49 - Figure 51 show comparisons between real and extracted linear and angular velocities for the best algorithms of the test. The velocities were extracted using the method described in previous section. Particularly the angular velocities in the Correlation and in the Sift algorithms seem comparable to the data provided by the simulated sensors.

3.5.2 Real Images Analysis with Data from the Sensors

In this experiments the algorithms were executed on image frames from a video recorded during the one of the flight tests performed for the WVU YF-22 Formation Flight Program [56]. A picture of the aircraft in flight is shown in the left part of Figure 52. The right part of Figure 52 shows the position of the camera.

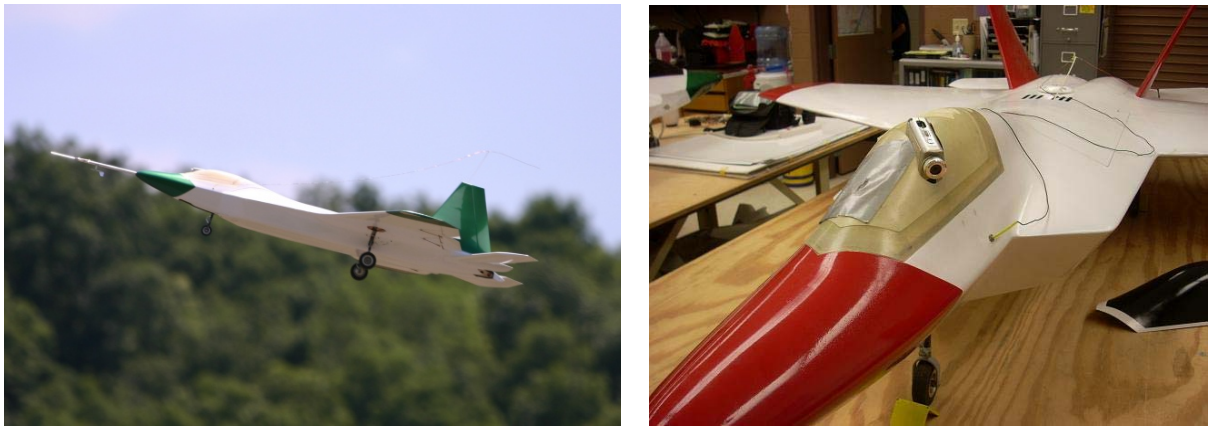


Figure 52: WVU YF-22 in flight conditions (left) and position of the camera in the WVU YF-22 (right).

The aircraft was equipped with an Inertial Measurement Unit (IMU), which allowed for acquiring the acceleration in x, y and z direction and the angular rates p, q, and r. The Vertical gyro provided measurements for the aircraft Euler's angles, and the GPS provided the translational position and velocity measurements x, y, z, V_x, V_y, V_z with respect to the earth reference frame. Furthermore, the nose probe provided measurements for the α, β angles, and absolute and differential pressure sensors were used to provide measurements for H and V [56]. The camera – which had a focal length equals to 847.5 pixels - was placed one meter in front of the aircraft center of gravity, with orientation with respect to the aircraft body frame consisting of yaw and pitch angles of -45° and -14.5° respectively.

To perform the OF experiment, a 25-seconds 320x240 video - acquired at the rate of 15 frames per second - was extracted from an original 607-seconds video, and the 9 OF algorithms were continuously executed for each couple of consecutive images. The Ideal Flow was calculated using the method described in previous sections and the (translational and rotational) position and velocity data acquired during the flight session.

3.5.2.1 Overall Velocity Error

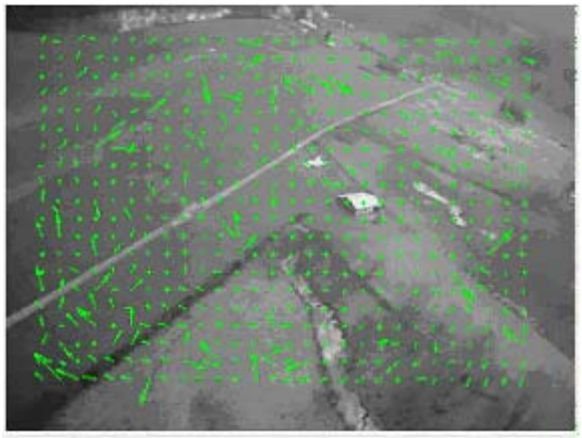
The procedure described in the virtual image analysis can be used in order to extract the linear and angular velocities of the aircraft in the real image analysis. It should be noticed that (69) provides the velocities in CRF and they cannot be directly compared to the data provided from the real sensors. In fact, the sensors provide the linear velocity in ERF and the angular velocity in URF. With the purpose of comparing the two quantities, the extracted velocities have to be pre-multiplied by the consistent transformation matrix.

3.5.2.2 Angular and Magnitude Errors w.r.t the Ideal Flow

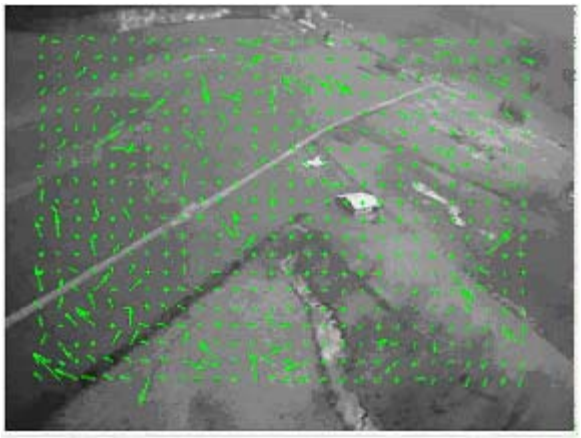
Expressing in polar coordinates both the ideal and the detected OF vectors - as for the previous experiments - allowed for the calculation of the errors in magnitude and in angle for each detected OF vector. Averaging the errors over all the OF vectors calculated during the simulation yielded the performance metric for the evaluation of the different OF algorithms.

3.5.2.3 Results of the test

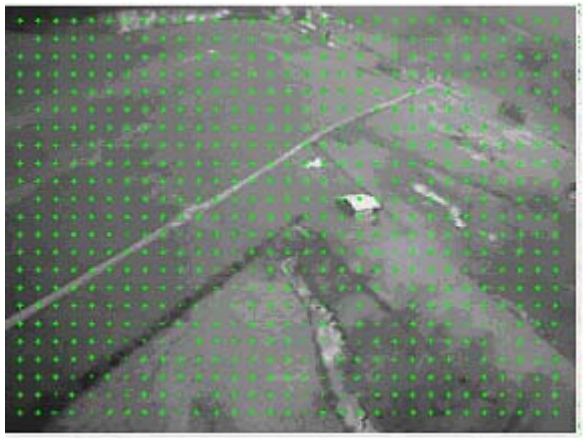
Samples of the results of the analysis conducted within this effort are shown in Figure 53. Figure 53 shows, also, the Ideal Optical Flow calculated as described in the previous sections, note that the Ideal OF can be derived only for the ground.



Correlation OF



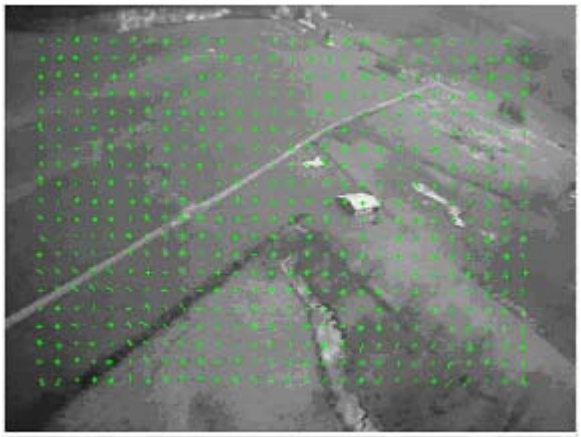
Difference OF



Gradient OF

Harris OF

SIFT OF



Phase OF

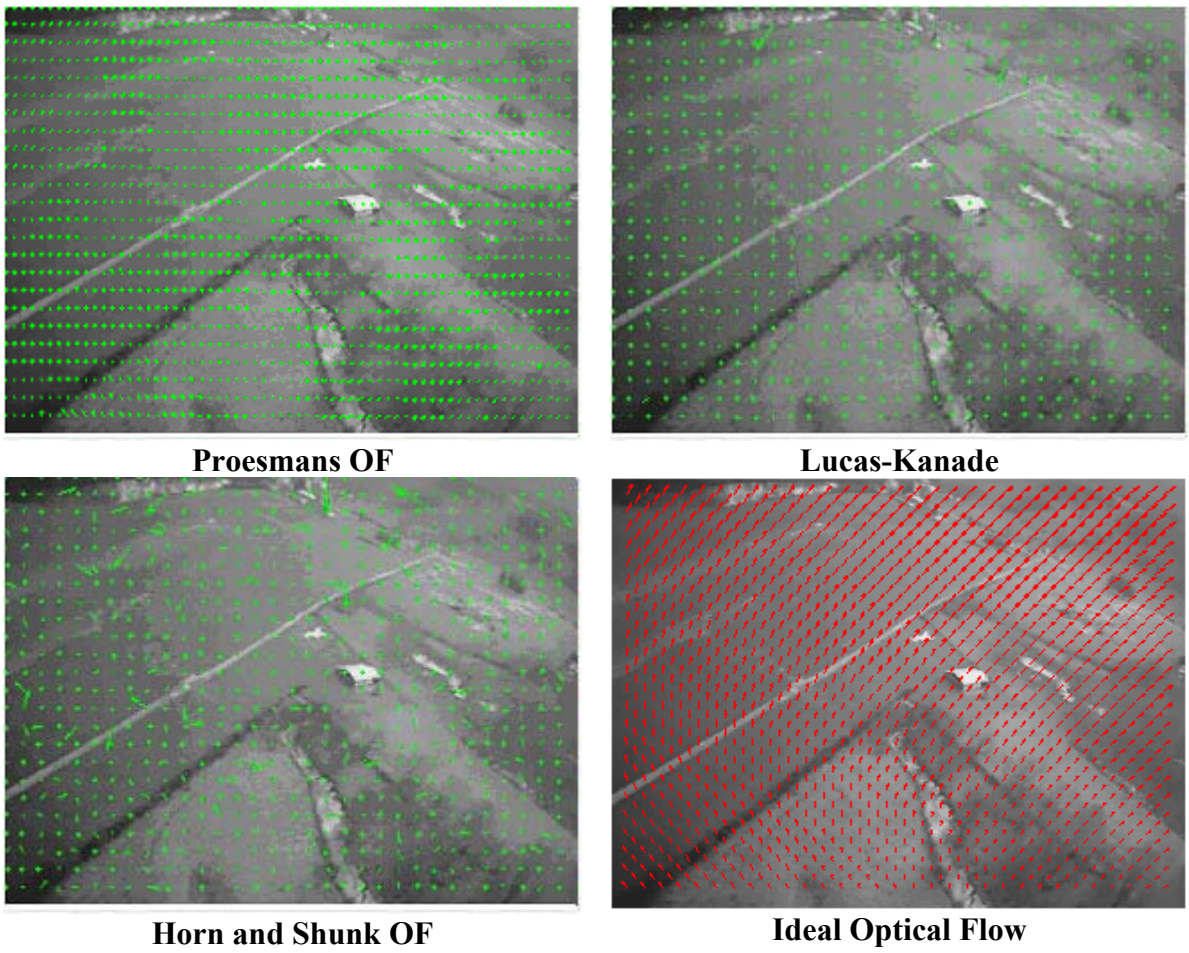


Figure 53: The nine algorithms and Ideal Flow in the complex motion experiment and real images

	Mean Ang Error (deg)	*STD Ang Error*	*Mean Mag Error (pix)*	*STD Mag Error*	*Norm of Mean error Velocity (m/s)*	*Norm of Mean error Angular Velocity (rad/s)*
Correlation	**35.67**	**32.66**	**6.42**	**4.97**	**10.72**	**2.48**
Difference	47.38	25.93	6.40	4.58	16.19	4.19
Gradient	61.23	20.28	11.86	6.98	15.19	9.28
Harris	55.48	22.56	6.93	5.10	17.40	6.29
SIFT	**32.58**	**31.70**	**6.10**	**4.73**	**8.68**	**2.81**
Phase	78.43	26.81	10.48	7.05	17.01	8.95
Proesmans	**40.07**	**31.46**	**9.12**	**6.30**	**12.53**	**7.44**
L- K	82.23	7.37	10.86	6.68	15.01	9.41
H S	82.62	7.99	6.57	7.46	15.16	9.43

Table 25: Experiment in the Real Images and Real data from the sensors

Table 25 shows the results obtained from a statistical analysis where the algorithms were compared - in terms of the mean and standard deviations of the errors in angle and magnitude - with the Ideal Flow during a 25 seconds simulation with a sampling time of 0.1 seconds. While a visual analysis of the pictures shows that the SIFT, and Proesmans algorithms provided the best performance, the statistical analysis in Table 25 highlights the fact that the Correlation algorithms also performs well in this experiment.

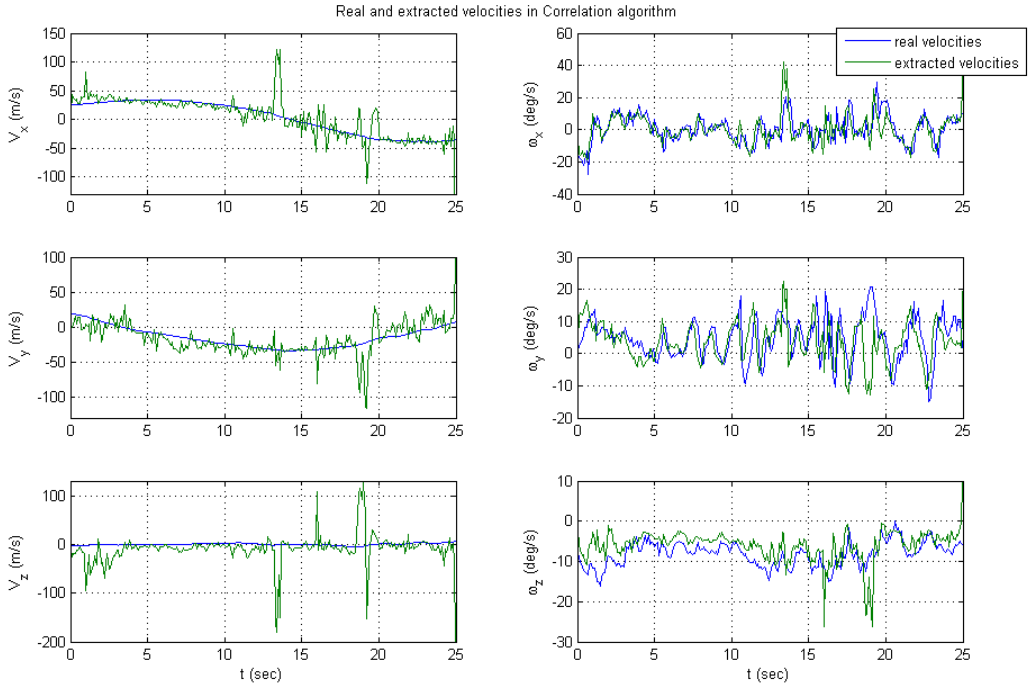


Figure 54 Comparison between real and extracted linear and angular velocities in the Correlation algorithm

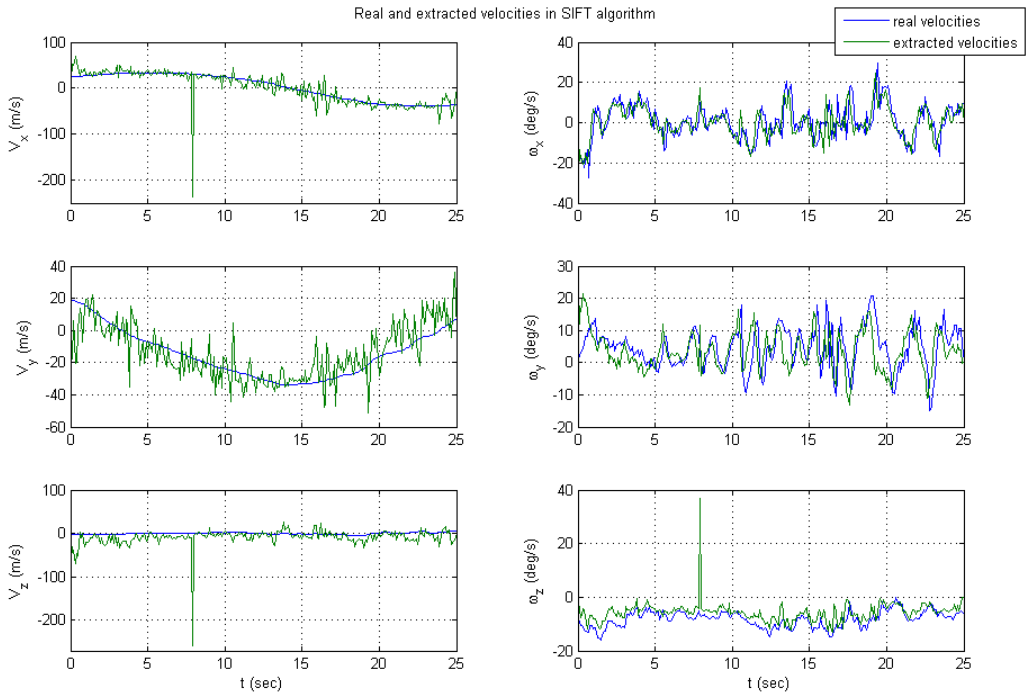


Figure 55 Comparison between real and extracted linear and angular velocities in the Sift algorithm

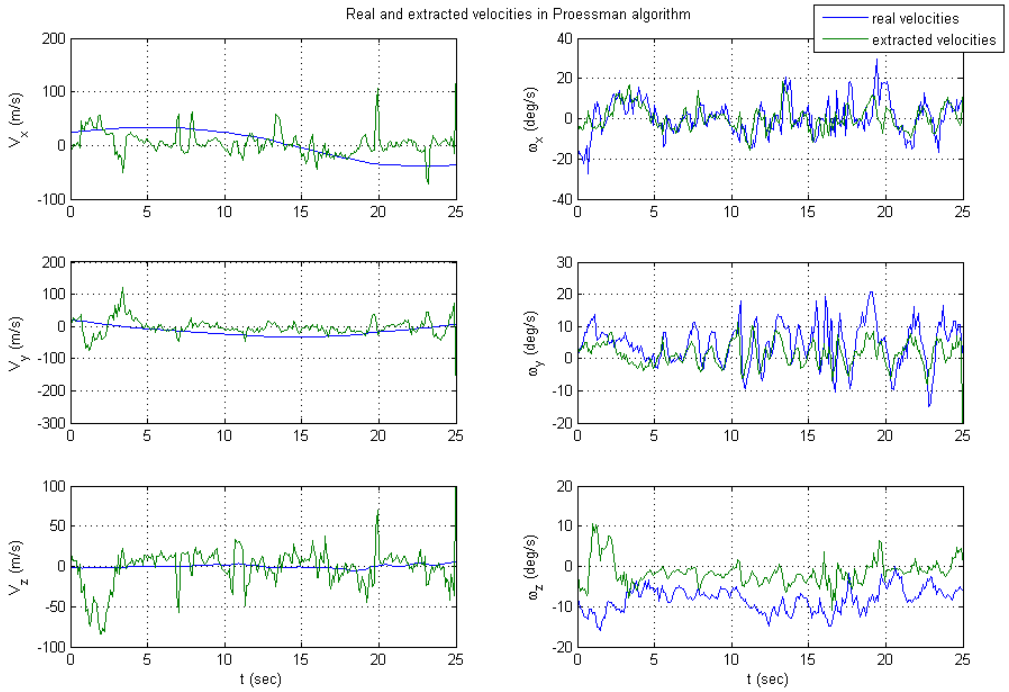


Figure 56 Comparison between real and extracted linear and angular velocities in the Proesman algorithm

Figure 54 - Figure 56 show comparisons between real and extracted linear and angular velocities for the best algorithms of the test. The velocities were extracted using the method described in previous section. Particularly the angular velocities in the Correlation and in the Sift algorithms seem comparable to the data provided by the real sensors.

3.6 Computational Requirements Analysis

Table 26 summarizes the results of the computational requirements analysis. Specifically, for each algorithms, the average computational time in seconds and the average number of produced OF vectors for each frame are reported in the first and second row respectively.

Algorithm	*Corr*	*Diff*	***Grad***	*Harris*	*Sift*	*Phase*	*Proes*	***L-K***	***H-S***
Time (sec)	9.77	9.36	**0.237**	0.615	1.45	2.95	1.30	**0.122**	**0.142**
# Points	1872	2240	2745	67.3	31.6	408	3072	3072	3072

Table 26: Time analysis

This analysis was performed on a 120-frames video, where each frame has dimension 320x240, using a Pentium 4 dual processor 3.4 GHz with 2GB of RAM memory. It should be emphasized that in order to minimize the computational overhead due to interpreter calls for the algorithms coded in Matlab (that is Correlation, Difference and Gradient), a considerable effort was undertaken to avoid explicit for-loops whenever possible, to pre-allocate arrays, to store data in column format, and to avoid the instantiation of unnecessary variables. Furthermore, the Matlab Profiler® was consistently used to locate and optimize bottlenecks in the code. As a result, the majority of the execution time for the final Matlab codes was spent within Matlab built-in functions, which consist in highly optimized routines stored in pre-compiled dynamic linked

libraries (DLLs). As a consequence, the algorithms written in Matlab could be reasonably compared with the algorithms written in C or C++.

As it can be seen from Table 26, algorithms belonging to the same class have similar computational requirements. As expected, the "matching" algorithms are the slowest ones. A profiling analysis of such algorithms confirmed that almost the totality of time was used to perform operations like SAD or Correlation, which are implemented as built-in files and, therefore, can be considered to be optimized.

The Gradient, Lucas-Kanade and Horn-Schunck are faster, which is not surprising since they rely only on basic operations on relatively small matrices. The Proesmans algorithm is more sophisticated, and, therefore, more computationally intensive. Finally, the performance of the algorithms based on feature detection is strictly correlated to the operations required to find the features. For example, corners are computationally easier to find than scale-invariant features. Therefore, SIFT required generally more computational effort than Harris.

With the exceptions of the Gradient, Lucas Kanade, and Horn and Shunck algorithms, the results of this analysis are not encouraging for the purpose of deploying the OF algorithms on currently available embedded computers, which typically have very limited computational resources.

3.7 Adaptation of the SIFT algorithm for Real-Time purpose

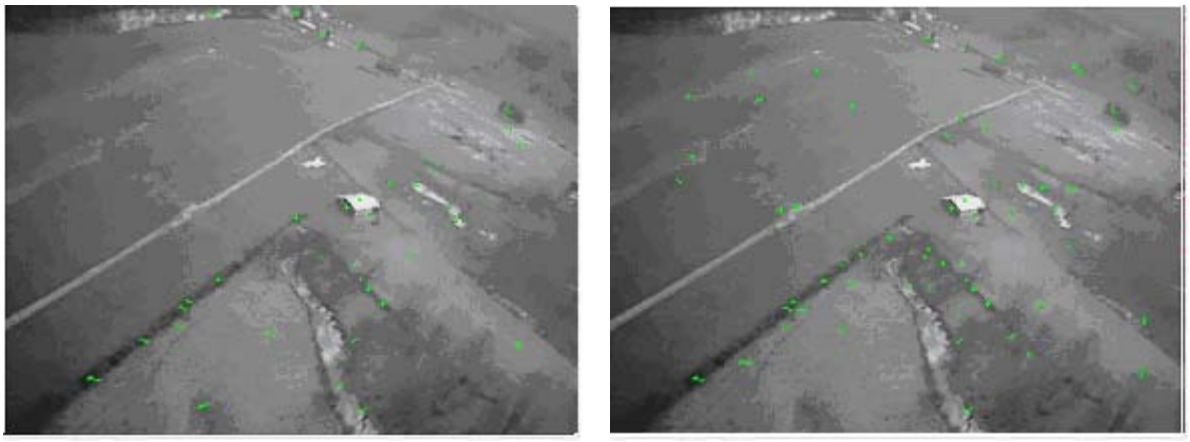
The original implementation of the SIFT algorithm is provided by Lowe [54]. The algorithm is in executable form that read a file .pgm and computes the Scale Invariant Feature Transform descriptor. A Matlab interface is used to write the image into a .pgm

file and run the executable file. The algorithm was developed in order to provide a unique descriptor for every detected feature. Matching the features found in different images is possible recognize the same point in two different images and calculating the difference in the position the Optical Flow in the matched features can be computed.

The executable implementation cannot be placed in on-board computers since it cannot be compiled as Matlab S-Function. The implementation provided by Vedaldi [68] is, instead, open source and written in Matlab and C languages; therefore can be used within Matlab S-Functions and consequently easily converted for Real Time purpose. On the other hand, the code written by Vedaldi is slower until it is compiled for Real Time application. In Appendix A the description of the SIFT algorithm implementation is provided [68].

3.7.1 Comparison between the two SIFT implementations

Despite of the fact it is possible tune the Vedaldi's implementation in order to obtain the same results of the of the code provided by Lowe, it was observed that the Vedaldi's implementation provide much more versatility and the parameters can be tuned in order to achieve better results. As shown in Figure 57, the code provided by Vedaldi can be tuned in order to find more features compared by the implementation provided by Lowe for the same image.



Lowe's SIFT OF **Vedaldi's SIFT OF**

Figure 57: Comparison of the OF provided by the two SIFT implementations

The computational time required to calculate the OF for both implementations and a portion of the video recorded during formation flight program was analyzed. Actually, the algorithms runs in a Simulink scheme where at each step the previous and the actual image are analyzed in order to extract the features. In a Real –Time implementation the algorithms can be optimized analyzing only the actual image for each step and recording the features of the previous instant in order to be used in the next iteration. This method was already implemented and was actually used in the Computational Requirements Analysis, but it cannot run in a Simulink scheme. Consequently, the computational time requirements published in Table 27 for the Lowe's SIFT implementation is different from the one provided in Table 26.

	Lowe's SIFT	*Vedaldi's SIFT*
Time (sec)	1.88	23.10
# Points	56.4	88.2

Table 27: Computational Time in the two SIFT implementation

In Table 28 are reported the data relative to the analysis of a portion of the Formation Flight video for the two version of the SIFT implementations. It should be

noticed that the performance of the two implementations are very similar in terms of mean and standard deviation of the angular and the magnitude error.

	Mean Ang Error (deg)	*STD Ang Error*	*Mean Mag Error (pix)*	*STD Mag Error*
Lowe's SIFT	32.58	31.70	6.10	4.73
Vedaldi's SIFT	33.63	30.84	6.22	4.68

Table 28: Comparison between in term of performance between the two SIFT

4 CONCLUSION

A number of Machine Vision (MV) techniques for specific applications in flight controls were evaluated in this effort. Particularly, different MV techniques were implemented and tested within simulation environments for the specific problems of the MV-based Aerial Refueling (AR) and Collision Identification for Unmanned Aerial Vehicles (UAVs). The first section of the document describes the AR problem; within these analyses, two different algorithms to solve the Point Matching and the Pose Estimation problem were proposed, implemented and evaluated.

The performances of the two Point Matching algorithms – the Mutual Nearest Point (MNP) and the Maximum Clique Detection (MCD) – were compared using different tests featuring virtual and real images. The results from this detailed comparison showed that the accuracy of the two algorithms is very similar. However, the MCD algorithm was able to generally recognize more corners and to provide better matching if the projected points were closer to the points detected in the image, and this is especially true for real images. On the other hand, the MNP algorithm had provided a more consistent overall matching and it generally allowed a smaller pose estimation error, while at the same time requiring a lower computational effort. These considerations led to the choice of the MNP algorithm within the Machine Vision-based Aerial Refueling.

In the Pose Estimation analysis, the attention was focused on the analysis of the performance of two widely used pose estimation algorithms - the GLSDC and the LHM algorithm - in terms of accuracy and robustness. The results from this comparison showed that the accuracy of the two algorithms is substantially similar; however, the LHM algorithm had provided a substantially higher level of robustness at the expense of

a larger required computational effort. Therefore, the LHM was preferred whenever its additional computational requirements were not a key issue in this particular problem.

A statistical analysis of the component of the Machine Vision sensor showed that the error could be considered white and gaussian suggesting the use of EKF for sensor fusion purpose. Therefore, the sensor fusion system based on the use of Extended Kalman Filtering was designed. It provided reliable position information through integration of the measurements supplied by the GPS system and the MV system, as well as from other aircraft sensors. The sensor fusion system has been described with details. A closed-loop simulation study using the simulation environment for the analysis of MV-based AR problem was performed. Results show that the proposed sensor fusion system allowed an improvement of more than one order of magnitude in the precision of the position estimates when compared to a previously used interpolation-based sensor fusion system. Furthermore, the results from simulations studies performed by changing some key tuning parameter suggest that the filter presents desirable robustness characteristics; in addiction, more robustness should be reached whether the Euler angles provided by the MV system are used into the sensor fusion system.

In the second section, the Collision Identification problem was analyzed in detail. The innovative solution was the use of Optical Flow algorithms for the identification of the possible risks collision. Particularly, a novel method to calculate the Ideal Flow generated by the motion of a rigid body was developed. This method allows the definition of standard approaches for the direct comparison of the velocity vector fields in both simple and complex scenarios. Nine of the most used algorithms were analyzed with different type of motion. In particular, real experiments on simple motion – rotational,

translational and forward translational motion – as well as in complex 6DOF motion – using both a virtual simulation environment and real video recorded during the flight testing – were developed and studied. An interesting method derived from the inversion of the Ideal Flow formula for extract the linear and angular velocities in complex motion from the Optical Flow was developed. The extracted velocities present a good level of accuracy using the Optical Flow algorithms that provide the better performance (Correlation, SIFT and Proessman). In all the tests, the level of accuracy was not enough for the development of a Collision Identification system based on Optical Flow. Hence, the Machine Vision can help in the identification of possible collision but needs to be integrated with more accurate sensor such as radars. In any case, the SIFT algorithm was the one that generally provided the best performance. A second version of the SIFT algorithm – provided by Vedaldi – was adapted at the desired purpose and compared with the original version – provided by Lowe. In the analysis was found that the version provided by Vedaldi was able to detect more features reaching the same level of accuracy. On the other hand, the version provided by Lowe was much faster than the other version. The big difference in the execution time was due to different implementation, in fact, the Lowe's version is a compiled executable file, and the Vedaldi' s version is instead written in Matlab code and need to be compiled in order to provide better performance.

5 APPENDIX A

This appendix was provided by Vedaldi and published in [68].

5.1.1 Scale Space

A scale space is a function $F(x,\sigma) \in \mathbb{R}$ of a spatial coordinate $x \in \mathbb{R}^2$ and a scale coordinate $\sigma \in \mathbb{R}^+$. Since a scale space $F(\cdot,\sigma)$ typically represents the same information at various scales $\sigma \in \mathbb{R}$, its domain is sampled in a particular way in order to reduce the redundancy.

The scale coordinate σ is discretized in logarithmic steps according to:

$$\sigma(s,o) = \sigma_o 2^{o+s/S}, \quad o \in \mathbb{Z},\ s = 0,\ldots,S-1 \tag{70}$$

where o is the octave index, s is the scale index, $S \in \mathbb{N}$ is the scale resolution $\sigma_o \in \mathbb{R}^+$ and is the base scale offset. Note that it is possible to have octaves of negative index.

The spatial coordinate *x* is sampled on a web with a resolution that is a function of the octave. x_o which is the spatial index for octave *o*. This index is mapped to the coordinate *x* by

$$x = 2^o x_o, \quad o \in \mathbb{Z},\ x_o \in [0,\ldots N_o - 1] \times [0,\ldots M_o - 1] \tag{71}$$

where (N_o; M_o) is the spatial resolution of octave *o*. If (M_0; N_0) is the resolution of the base octave $o = 0$, the resolution of the other octaves is obtained as

$$N_o = \left\lfloor \frac{N_0}{2^o} \right\rfloor, \quad M_o = \left\lfloor \frac{M_0}{2^o} \right\rfloor \tag{72}$$

It is useful to store some scale levels twice, across different octaves. This is done allowing the parameter *s* to be negative or greater than *S*. Formally, the range of *s* is

$[s_{\min}, s_{\max}]$. It can also be denoted the range of the octave index o as $[o_{\min}, o_{\min}+O-1]$, where $O \in \mathbb{N}$ is the total number of octaves. The SIFT detector makes use of the two scale spaces described next.

5.1.1.1 Gaussian Scale Space

The Gaussian scale space of an image *I(x)* is the function

$$G(x,\sigma) \triangleq (g_\sigma * I)(x) \tag{73}$$

where the scale $\sigma = \sigma_o 2^{o+s/S}$ is sampled as explained in the previous section and the symbol $*$ represents the convolution operation. In practice, it is assumed that the image is already pre-smoothed at a nominal level σ_o, so that $G(x,\sigma) \triangleq \left(g_{\sqrt{\sigma^2-\sigma_n^2}} * I\right)(x)$. As suggested in [54], the pyramid is computed incrementally from the bottom by successive convolutions with small kernels.

5.1.1.2 Difference of Gaussians Scale Space

The Difference of Gaussians (DOG) scale space is the scale "derivative" of the Gaussian scale space $G(x,\sigma)$ along the scale coordinate σ. It is given by:

$$D(x,\sigma(s,o)) \triangleq G(x,\sigma(s+1,o)) - G(x,\sigma(s,o)) \tag{74}$$

Remark 1 (Lowe's parameters): Lowe's implementation uses the following parameters:

$$\sigma_n = 0.5, \quad \sigma_0 = 1.6 \cdot 2^{1/S}, \quad o_{\min} = -1, \quad S = 3 \tag{75}$$

In order to compute the octave *o*=-1, the image is doubled by bilinear interpolation (for the enlarged image $\sigma_n = 1$). In order to detect extrema at all scales, the Difference of Gaussian scale space has $s \in [s_{\min}, s_{\max}] = [-1, S+1]$. Since the Difference of Gaussian scale space is obtained by differentiating the Gaussian scale space, the latter

has $s \in [s_{\min}, s_{\max}] = [-1, S+2]$. The parameter O is set to cover all octaves (i.e. as big as possible.)

5.1.2 The Detector

The SIFT frames (or "keypoints") are a selection of (sub-pixel interpolated) points (x,σ) of local extremum of the DOG scale-space $D(x,\sigma)$, together with an orientation θ derived from the spatial derivative of the Gaussian scale-space $G(x,\sigma)$. For what concerns the detector (and being in general different for the descriptor), the "support" of a keypoint (x,σ) is a Gaussian window *H(x)* of deviation $\sigma_w = 1.5\sigma$. In practice, the window is truncated at. $|x| \leq 4\sigma_w$.

The Gaussian and DOG scale spaces are derived as in previous section. In this Section, the parameters *S*; *O*; $s_{\min}, s_{\max}$, $o_{\min}, \sigma_0$ refer to the DOG scale space. The Gaussian scale space has exactly the same parameters of the DOG scale space except for $s_{\max}^{DOG}$ which is equal to $s_{\max} - 1$. The extraction of the keypoints is carried one octave per time and articulated in the following steps:

- Detection: Keypoints are detected as points of local extremum of $D(x,\sigma)$. In the implementation, the function extracts such extrema by looking at 9x9x9 neighborhoods of samples. As the octave is represented by a 3D array, the function returns indexes k (in Matlab convetion) that are to be mapped to scale space indexes (x_1; x_2; s) by

$$k - 1 = x_2 + x_1 M_o + (s - s_{\min}) M_o N_o \tag{76}$$

Alternatively, another function can be used to map the index k to a subscript $(i, j; l)$ and then use

$$x_1 = j - 1, \quad x_2 = i - 1, \quad s = l - 1 + s_{\min} \tag{77}$$

Because of the way such maxima are detected, one has always $1 \le x_2 \le M_o - 1,\ 1 \le x_1 \le \mathrm{N_o}\text{-}2\ s_{\min} + 1 \le s \le s_{\max} - 1$.

Since both local maxima and minima are searched, the process is repeated for $-G(x,\sigma)$. (If only positive maxima and negative minima are of interest, another option is to take the local maxima of $|G(x,\sigma)|$ directly, which is quicker.)

- Sub-pixel refinement. After being extracted, the index $(x_1; x_2; s)$ is fitted to the local extremum by quadratic interpolation. At the same time, a threshold on the "intensity" $D(x,\sigma)$ and a test on the "peakedness" of the extremum is applied in order to reject weak points or points on edges. The edge rejection step is explained in detail in the paper [54]. The sub-pixel refinement is an instance of Newton's algorithm.
- Orientation. The orientation θ of a keypoint (x,σ) is obtained as the predominant orientation of the gradient in a window around the keypoint. The predominant orientation is obtained as the (quadratic interpolation) maximum of the histogram of the gradient orientations $\angle G(x_1, x_2, \sigma)$ within a window around the keypoint. The histogram is weighted both by the magnitude of the gradient $|\nabla G(x_1, x_2, \sigma)|$ and a Gaussian window centered on the keypoint and of deviation 1.5σ (the Gaussian window defines the region of interest as well). After collecting the data in the bins and before computing the maximum, the histogram

is smoothed by a moving average filter. In addition to the global maximum, each local maximum with a value above 0.8% of the maxium is retained as well. Thus for each location and scale multiple SIFT frames might be generated.

5.1.3 The Descriptor

The SIFT descriptor of a keypoint (x,σ) is a local statistic of the orientations of the gradient of the Gaussian scale space $G(x,\sigma)$.

- Histogram layout. The SIFT descriptor (Figure 58) is a histogram of the image gradients orientations and locations (these are tuples $(x,\theta)\in\mathbb{R}^2\times\mathbb{R}/\mathbb{Z}$). The histogram bins form a three dimensional lattice with N_P = 4 bins for each spatial direction and N_o = 8 bins for the orientation for a total of $N_P{}^2 N_o = 128$ components (these numbers can be changed by setting the appropriate parameters). Each spatial bin is square with unitary edge. The window *H(x)* is Gaussian with deviation equal to half the extension of the spatial bin range, that is N_P=2.

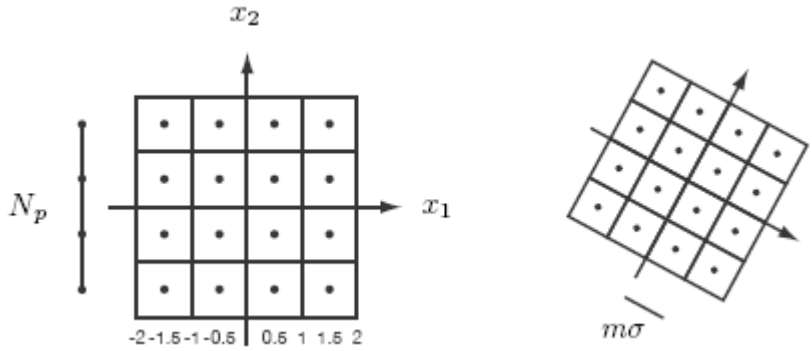


Figure 58: SIFT descriptor layout. The actual size of a spatial bin is mσ where σ is the scale of the keypoint and m=3.0 is a nominal factor

- Keypoint normalization. In order to achieve invariance, the histogram layout is projected on the image domain according to the frame of reference of the keypoint. The spatial dimensions are multiplied by $m\sigma$ where σ is the scale of the keypoint and m is a nominal factor (equal to 3.0 by default). The layout is also rotated so that the axis x_1 is aligned to the direction θ of the keypoint.
- Weighting. The histogram is weighted by the gradient modulus and a Gaussian windowed and tri-linearly interpolated. More in detail, each sample $\left(x_1, x_2, \angle G\left(x,\sigma\right)\right)$ is
    - weighted by $\left|\nabla G\left(x,\sigma\right)\right|$;
    - weighted by the Gaussian window *H(x)*;
    - projected on the centers of the eight surrounding bins;
    - summed to each of this bins proportionally to its distance from the respective center.

Remark 2. (Lowe's implementation) In order to achieve full compatibility with Lowe's original implementation, the users has to pay attention to many little details as the memory layout of the descriptor and the convention for the gradient orientations.

6 REFERENCES

[1] U.S Department Of Defense: http://www.defenselink.mil/transcripts/transcript.aspx?transcriptid=2064 , March 2003.

[2] U.S Department Of Defense: http://www.defenselink.mil/releases/release.aspx?releaseid=8665 , July 2005.

[3] U.S. Air force: http://www.airforce.com/ , March, 2008

[4] U.S. Navy: http://www.navy.mil/navydata/fact_display.asp?cid=1100&tid=2100&ct=1, August, 2007.

[5] U.S. Army: http://www.army.mil/2003TransformationRoadmap/Chpt8.pdf , 2003.

[6] Hovercam: http://www.hovercam.com/Hovercam/Home.html , March, 2008.

[7] Cyber Aerospace: http://www.proxygen.com/37/Editorial.asp?aff_id=37&this_cat=Company&type_id=733&cat_id=735&action=sub&list_type=name. March, 2008.

[8] U.S Department of Homeland Security: http://www.dhs.gov/xnews/speeches/press_release_0446.shtm, June, 2004.

[9] Heyhurst K.J., Maddalon J.M, Miner P.M, Szatkowsky G.N, Ulrey M.L, DeWalt M.P, Spitzer C.R. "Preliminary Considerations of classifying hazards of Unmanned Aircraft System", February, 2007

[10] Korbly, R. and Sensong, L. "Relative attitudes for automatic docking," *AIAA Journal of Guidance Control and Dynamics*, Vol. 6, No. 3, 1983, pp. 213-215.

[11] Kimmett J., Valasek J., Junkins J.L., "Autonomous Aerial Refueling Utilizing a Vision Based Navigation System", Proceedings of the 2002 AIAA GNC Conference, Paper 2002-5569, Monterey (CA), August 2002.

[12]Fravolini, M.L., Ficola, A., Campa, G., Napolitano M.R., Seanor, B., "Modeling and Control Issues for Autonomous Aerial Refueling for UAVs Using a Probe-Drogue Refueling System," *Journal of Aerospace Science Technology*, Vol. 8, No. 7, 2004, pp. 611-618.

[13]Philip N.K., Ananthasayanam M.R., "Relative Position and Attitude Estimation and Control Schemes for the Final Phase of an Autonomous Docking Mission of Spacecraft", *Acta Astronautica*, vol. 52, 2003, pp. 511-522.

[14] Sinopoli B., Micheli M., Donato G., Koo T.J., "Vision Based Navigation for an Unmanned Aerial Vehicle", Proceedings of the 2001 IEEE International Conference on Robotics and Automation, Vol. 2, 1757-1764, Seoul, South Korea, May 2001.

[15]Stevens, B.L., Lewis, F.L., *"Aircraft Control and Simulation,"* John Wiley & Sons, New York, 1987.

[16] Rauw, M.O.: "FDC 1.2 - A Simulink Toolbox for Flight Dynamics and Control Analysis". Zeist, The Netherlands, 1997. ISBN: 90-807177-1-1, http://www.dutchroll.com/

[17]Addington, G.A., Myatt, J.H., "Control-Surface Deflection Effects on the Innovative Control Effectors (ICE 101) Design, "Air Force Report", AFRL-VA-WP-TR-2000-3027, June 2000

[18] Stengel, R.F., "*Optimal control and estimation*", Dover Publication Inc. New York, 1994.

[19] Asada H.J., Slotine J.E., "Robot Analysis and Control", Wiley, New York, 1986, pp. 15-50.

[20] Spong M.W., Vidyasagar M. "Robot Dynamics and control", Wiley, New York, 1989, pp. 62-91.

[21] Roskam J. "*Airplane Flight Dynamics and Automatic Flight Controls – Part II*", DARC Corporation, Lawrence, KS, 1994.

[22] Blake W, Gingras D.R., "Comparison of Predicted and Measured Formation Flight Interference Effect", Proceedings of the 2001 AIAA Atmospheric Flight Mechanics Conference, AIAA Paper 2001-4136, Montreal, August 2001.

[23] Gingras D.R., Player J.L., Blake W., "Static and Dynamic Wind Tunnel testing of Air Vehicles in Close Proximity", Proceedings of the 2001 AIAA Atmospheric Flight Mechanics Conference, Paper 2001-4137, Montreal, Canada, August 2001.

[24] Virtual Reality Toolbox Users Guide, 2001-2006, HUMUSOFT and The MathWorks Inc.

[25] The VRML Web Repository, Dec. 2002: (http://www.web3d.org/x3d/vrml/)

[26] Vendra, S., Campa, G., Napolitano, M.R., Mammarella, M., Fravolini, M.L., "Addressing Corner Detection Issues for Machine Vision based UAV Aerial Refueling" Submitted to *Machine Vision and Application*, October 2005.

[27] Harris, C and Stephens, M, "A Combined Corner and Edge Detector", *Proc. 4th Alvey Vision Conference*, Manchester, pp. 147-151, 1988.

[28] Noble, A. "Finding Corners", *Image and Vision Computing Journal,* 6(2): 121-128, 1988.

[29] Hutchinson S., Hager G., Corke P., "A tutorial on visual servo control", *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 5, 1996, pp. 651-670.

[30] Pla, F., Marchant, J.A., "Matching Feature Points in Image Sequences through a Region-Based Method," *Computer vision and image understanding*, Vol. 66, No. 3, 1997, pp. 271-285.

[31] Fravolini, M.L., Campa, G., Napolitano, M.R., Ficola, A., "Evaluation of Machine Vision Algorithms for Autonomous Aerial Refueling for Unmanned Aerial Vehicles", Submitted to: *AIAA Journal of Aerospace Computing, Information and Communication,* April 2005.

[32] Mammarella, M., Campa, G., Napolitano, M.R., Fravolini, M.L., Pollini L., Perhinschi, M., "Addressing Pose Estimation Issues for Machine Vision based UAV Autonomous Aerial Refueling", The Aeronautical Journal, pp 389-396, June 2007.

[33] Kimmett J., Valasek J., Junkins J.L., "Autonomous Aerial Refueling Utilizing a Vision Based Navigation System", Proceedings of the 2002 AIAA GNC Conference, Paper 2002-5569, Monterey (CA), August 2002.

[34] Branca, A., Stella, E., Distante, A., "Feature Matching by Searching Maximum Clique on High Order Association Graph," *iciap*, p. 642, 10th International Conference on Image Analysis and Processing, 1999

[35] "Image Processing Toolbox version 5.2", The Mathworks, http://www.mathworks.com/ access/helpdesk/help/toolbox/images/, March 2006.

[36] Broida, T., Chellappa, R., "Estimating the kinematics and structure of a rigid object from a sequence of monocular images", *IEEE transaction on Pattern Analysis and Machine Intelligence*, vol. 13 no. 6, pp.497-513, 1991.

[37] Soatto, S., Perona, P., "Reducing "structure from motion": a general framework for dynamic vision, Part 1: modeling", *IEEE transaction on Pattern Analysis and Machine Intelligence*, vol. 20 no. 9, pp.993-942, 1998

[38] Haralick, R.M et al., "Pose Estimation from Corresponding Point Data", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 19, No. 6, 1989, pp. 1426-1446.

[39] Lu, C.P., Hager, G.D., Mjolsness, E., "Fast and Globally Convergent Pose Estimation from Video Images," *IEEE Transactions On Pattern Analysis and Machine Intelligence*, Vol. 22, No. 6, 2000, pp. 610-622.

[40] Dell' Aquila, R., Campa, G., Napolitano, M. R., Mammarella, M., "Real-Time Machine-Vision-Based Position Sensing System for UAV Aerial Refueling", *Journal of Real-Time Image Processing*, In Press as of Feb 2007.

[41] Barrows, G., and Neely, C.: 'Mixed-mode VLSI optic flow sensors for in-flight control of a micro air vehicle', *Proceedings. SPIE*, 2000, 4109, pp. 52–63

[42] Khalil F. F, and Payeur P. "Optical Flow Techniques in Biomimetic UAV Vision", *IEEE International Workshop on Robotic and Sensors Environments*, pp. 14-19,2005.

[43] "Global Aerial Surveillance to Implement Collision Avoidance on Advanced UAV Prototypes Intended for Military Use". *Market Wire*. August 2005. FindArticles.com. 16 Jan. 2008. http://findarticles.com/p/articles/mi_pwwi/is_200508/ai_n14941925.

[44] "Flight Demonstrations Evaluate UAV Collision -Avoidance Technology" Relese 03-20, April, 2003, http://www.nasa.gov/centers/dryden/news/NewsReleases/2003/03-20.html

[45] Fasano G., Accardo D., Moccia A., Carbone C., Ciniglio U., Cozzaro F., Luongo S., "Multi-Sensors Based fully Autonomous Non-Cooperative Collision Avoidance System for UAVs", *AIAA Infotech@Aerospace 2007 Conference and Exhibit*, May 2007.

[46] Galvin, B., McCane, B., Novins, K., Mason, D. and Mills, S. "Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms". In *Proceedings of the 9th British Machine Vision Conference,* pp. 195-204, 98, *1998*.

[47] Barron J.L., Fleet, D.J and Beauchemin, S.S. "Performance of Optical Flow Techniques", *Intl Journal of Computer Vision*, vol. 12, pp. 43-77,1994.

[48] Lucas, B., Kanade, T., "An iterative Image Restoration Technique with an Application to Stereo Vision", *Proc. of DARPA Image Understanding Workshop*, p. 121-130, 1981.

[49] Horn B.K.P., Schunck B.G. "Determining optical flow" *AI,*. 17. pp.185-204, 1981.

[50] Proesmans, M., Van Gool, L., Pauwels E. and Oosterlinck A., "Determination of optical flow and its Discontinuities using non-linear diffusion". In *Proceedings of the 3rd European Conference on Computer Vision,* Vol. 2, pages 295-304, 1994.

[51] Fleet D.J., and Jepson A.D. "Computation of component image velocity from local phase information" Int. J. Comp. Vision 5, pp. 77-104, 1990.

[52] Gautama T., Phase-based Optical Flow, Matlab Central http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectType=file&objectId=2422 (2002).

[53]Gautama T., Van Hulle M.M., "A Phase-based Approach to the Estimation of the Optical". *IEEE Transaction on neural network. Vol.13 No.5*. 2002.

[54] Lowe D.G., "Object Recognition from Local Scale - Invariant Features", *Proc. of the International Conference on Computer Vision,* 1999.

[55] Campa, G., Mammarella, M., Napolitano, M. R., Fravolini, M. L., Pollini, L., Stolarik, B., "A comparison of Pose Estimation algorithms for Machine Vision based Aerial Refueling for UAV", *Mediterranean Control Conference 2006*, 2006.

[56] Campa, G., Gu, Y., Seanor, B., Napolitano, M. R., Pollini, L., Fravolini, M. L., "Design And Flight Testing Of Nonlinear Formation Control Laws", *Control Engineering Practice*, pp 1077-1092, Vol. 15, Issue 9, 2007.

[57] Hutchinson S., Hager G., Corke P., "A tutorial on visual servo control", *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 5, 1996, pp. 651-670.*The Virtual Reality Toolbox User's Guide*, 2001-2007, HUMUSOFT and The MathWorks Inc.

[58] Symbolic Math Toolbox User's Guide, 1993-2007, The MathWorks Inc.

[59] Kenney, C.S., Manjunath, B.S., Zuliani, M., Hewer, G.A., Van Nevel, A., "A Condition Number for Point Matching with Application to Registration and Postregistration Error Estimation", *IEEE Transactions On Pattern Analysis and Machine Intelligence*, Vol. 25, No. 11, 2003, pp. 1437-1454.

[60] Campa, G., Napolitano, M. R., Fravolini, M. L., "A Simulation Environment for Machine Vision Based Aerial Refueling for UAV", Submitted to: IEEE Transaction on Aerospace and Electronic Systems, April 2006.

[61] Ross, G.J.S., "*Nonlinear Estimation*", Springer-Verlag, New York, 1990.

[62] Denison, D.D., "*Nonlinear Estimation and Classification*", Springer, New York, 2003.

[63] Krakiwsky, E. J., Harris, C.B., Wong, R.V.C., "A Kalman Filter for integrating Dead Reckoning, Map Matching and GPS Positioning", IEEE PLANS '88 Position Location and Navigation Symposium Record `Navigation into the 21st Century, Dec. 1988.

[64] Cooper S, Durrant-Whyte H, "A Kalman filter model for GPS navigation of land vehicles", In IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, pages 157-163, September 1994.

[65] Abbott E., Powell D., "Land-vehicle navigation using GPS," Proceedings of the IEEE, Vol. 87, No. 1, pp. 145-162, January 1999.

[66] S. Panzieri, F. Pascucci, G. Ulivi, "An Outdoor Navigation System Using GPS and Inertial Platform," IEEE/ASME Trans. on Mechatronics, vol. 7, n. 2, pp. 134-142, 2002.

[67] *The Virtual Reality Toolbox User's Guide*, 2001-2007, HUMUSOFT and The MathWorks Inc.

[68] Vedaldi, A., "An open implementation of the SIFT detector and descriptor", UCLA CSD Tech. Report 070012, 2006.

[69] J. L. Junkins, H. Schaub, and D. Hughes, "Noncontact position and orientation measurement system and method," U.S. Patent 6,266,142, 2001.

[70] J. Valasek, K. Gunnam, J. Kimmett, M. Tandale, J. L. Junkins, and D. Hughes, "Vision-based sensor and navigation system for autonomous air refueling," Journal of Guidance, Control, and Dynamics, 2005,vol. 28,n 5,pp.979-989.

[71] Bowers, R.E., "Estimation Algorithms For Autonomous Aerial Refueling Using a Vision Based Relative Navigation System" Master Thesis at Texas A&M University, 2005.

[72] H. Seraji, R. Steele, and R. Ivlev, "Sensor-based collision avoidance: Theory and experiments", J. Rob. Syst. 13: 9, pp .571–586, *1996.*

[73] S. Bouaziz, M. Fan, Roger Reynaud, T. Maurin: Multi-Sensors and Environment Simulator for Collision Avoidance Applications. Computer Architectures and Machine Perception, 2000: 127-130.

[74]Connolly, C. "Collision avoidance technology: from parking sensors to unmanned aircraft" Sensor Review, Vol, 27,Num. 3, pp. 182-188(7), 2007.

[75] Moldovan, E. Tatu, S.-O. Gaman, T. Wu, K. Bosisio, R. G., "A New 94-GHz Six-Port Collision-Avoidance Radar Sensor", IEEE Transactions on Microwave Theory and Techniques, vol. 52; num. 3, pages 751-759, 2004

[76] Kwag, Y.K., Chung, C.H., "UAV based collision avoidance radar sensor", IEEE Geoscience and Remote Sensing Symposium, 23-28 July 2007.

[77] Fasano, G., "Multisensor based Fully Autonomous Non-Cooperative Collision Avoidance System for UAVs", Ph.D. Thesis, University of Naples, May 2008.

[78] Lee, H.-C., “Implementation of collision avoidance system using TCAS II to UAVs”, The 24th Digital Avionics Systems Conference, 30 Oct – 3 Nov, 2005.

[79] Griffiths, S., Saunders, J., Curtis, A., Barber, B., McLain, T., & Beard, R., “Maximizing miniature aerial vehicles—obstacle and terrain avoidance for MAVs”. *IEEE Robotics and Automation Magazine*, *13*, 34–43. 2006.

[80] Mammarella, M, Campa, G., Napolitano, M. R., Fravolini, M. L., Perhinschi, M. G., Gu, Y., " Machine Vision / GPS Integration Using EKF for the UAV Aerial Refueling Problem ", In Press: IEEE Transaction on Systems Man and Cybernetics, Jul. 2008.

[81]Warwick G., “AFRL Advanced Autonomous Aerial Refueling”, Aviation week, June 2008.

[82] Sciavicco L., Siciliano B.: *Modeling and control of robot manipulators*, New York (N.Y.): McGraw-Hill, 1996.

[83]Chiuso, A., Favaro, P., Jin, H., and Soatto, S. 2002. Structure from motion causally integrated over time. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 24(4), 523–535.