2010:063

MASTER'S THESIS

Development of Hardware-in-the-Loop Facility for Optical Navigation with the Study of Insect Inspired Algorithm for Spacecraft Landing

Nawarat Termtanasombat

Luleå University of Technology

Master Thesis, Continuation Courses
Space Science and Technology
Department of Space Science, Kiruna

2010:063 - ISSN: 1653-0187 - ISRN: LTU-PB-EX--10/063--SE

LULEÅ UNIVERSITY OF TECHNOLOGY

Development of Hardware-in-the-Loop Facility for Optical Navigation with the Study of Insect Inspired Algorithm for Spacecraft Landing

by

Nawarat Termtanasombat

A thesis submitted in partial fulfillment for the
degree of Master of Science

in the
Department of Space Science
Kiruna

September 2010

Abstract

The thesis objective was to develop a real-time interface to the robot controller and related software components in the hardware-in-the-loop simulation facility for optical navigation. For demonstration of the developed components, an insect-inspired landing algorithm using a camera was chosen to study along with the hardware-in-the-loop simulation development. The testing of the algorithm in the hardware-in-the-loop simulation provided the evaluation of the algorithm itself as well as the functional test of the hardware-in-the-loop software components.

In this thesis, two simulation platforms were presented. First, the simulation platform using planetary image generation software was setup. The landing algorithm acquired image data from the image generation software by providing spacecraft dynamics information to the software. After image processing, the algorithm provided the feedback control to control the spacecraft. Second, the hardware-in-the-loop simulation software components were setup in the facility in German Aerospace Centre (DLR). The robotic arm carrying a camera was controlled to approach the scaled terrain model. Simple spacecraft dynamics was performed and transformed to the robotic arm movement. Reference frames for the transformation were defined. The real-time control software for the robotic arm was developed and the closed-loop control in the hardware-in-the-loop simulation was setup. Finally, control signals from the image processing module were transferred to change spacecraft dynamics in the simulation resulted in changing the movement of the robotic arm.

An insect-inspired algorithm using optical flow from a camera for controlling the spacecraft during the landing was presented. The image processing for the optical flow determination and interpretation were described. The characteristics of the optical flow were studied. The control law for controlling the spacecraft velocity according to the optical flow was designed according to the optical flow characteristics. The delay in the control system was considered in the design of the control rule.

The implementation of the algorithm with the tests in software-in-the-loop simulation and in the hardware-in-the-loop facility was presented. The landing algorithm can slow down the spacecraft and land softly both with image generation software and in hardware-in-the-loop simulation. The system with the delay was studied. The developed control rule can control the delayed system effectively. The thesis has shown that the insect-inspired landing algorithm using only the optical flow calculation is feasible for the landing of the spacecraft.

Acknowledgements

This thesis work was a part of Erasmus Mundus Master programme, Joint European Master in Space Science and Technology (SpaceMaster). I would like to thank the European Commission for the opportunities and scholarship that enabled me to pursue the programme and perform this thesis work.

This thesis work was performed at the Department of Navigation and Control Systems, Institute of Space System, German Aerospace Center (DLR) in Bremen. I wish to thank DLR and Dr. Stephan Theil, head of the Navigation and Control Department for the support on instruments, laboratory facilities and the opportunities for me to complete my thesis work there.

I would like to express my gratitude to Hans Krüger, my project advisor at DLR, Bremen for the opportunities and fully support in term of advice, discussions and encouragement as well as in administration issues. He is very nice and helpful in all aspects. His efforts and advice were very valuable for me to carry out the thesis work.

I wish to thank Ansgar Heidecker and Stephen Steffes (DLR) for advice and efforts helping me during the development on dSPACE system. Thanks to friends at DLR for the nice time I spent during the thesis work.

Very special thank to Anita Enmark, my examiner at the Department of Space Science (IRV) at Luleå University of Technology in Kiruna for her guidance and advice. I wish to thank Victoria Barabash and Sven Molin for the coordinating effort throughout the SpaceMaster course. I would like to give personal thanks to Maria Winnebäck and Anette Snällfot-Brändström for the best administrative persons ever helping me in all issue during the course and a very great time in Kiruna.

I would like to thank Dan Veal for proofreading of the language in the thesis.

Thanks to Katrin and Bilal for very nice smiles and warm welcome to Bremen. I would like to thank them for housing me when I was looking for the permanent accommodation in Bremen. Thank to P'Jeab, Im, Ta, Guni, Miri, and other friends in Bremen for our great time, encouragement and every support during my stay in Bremen. Thanks to SpaceMaster friends for the friendship and encouragement until the end of the course.

Last but not least, I would like to thank my family for their love and supporting me even from the other side of the world.

Contents

List of Figures

Abbreviations

AOCS	**A**ttitude and **O**rbit **C**ontrol **S**ystem
DLR	**D**eutsches Zentrum für **L**uft- und **R**aumfahrt (German Aerospace Center)
DOF	**D**egree **O**f **F**reedom
EKF	**E**xtended **K**alman **F**ilter
FOE	**F**ocus **O**f **E**xpansion
FOV	**F**ield **O**f **V**iew
GBAS	**G**round-**B**ased **A**ugmentation **S**ystem
GPS	**G**lobal **P**ositioning **S**ystem
IMU	**I**nertial **M**easurement **U**nit
IP	**I**nternet **P**rotocol
KRL	**K**UKA **R**obot **L**anguage
LIDAR	**Li**ght **D**etection **A**nd **R**anging
PANGU	**P**lanet and **A**steroid **N**atural Scene **G**eneration **U**tility
ROI	**R**egion **O**f **I**nterest
RSI	**R**obot**S**ensor**I**nterface
TCP	**T**ool **C**entre **P**oint
TRON	**T**estbed for **R**obotic **O**ptical **N**avigation
UAV	**U**nmanned **A**erial **V**ehicle
UDP	**U**ser **D**atagram **P**rotocol
XML	**E**xtensible **M**arkup **L**anguage

Chapter 1

Introduction

Space exploration is a curiosity of mankind and it is a way to learn and develop the technology which can be related to the life on earth. Humans start from looking into the sky and expanding the optical exploration to physical exploration by sending humans or robotic systems out from the earth to space. In the future, more missions are planned to land a rover or probe on another planetary surface, moon or asteroid. The exploration missions in the future require an accurate and safe landing system without the support of absolute global navigation system like GPS that is available on earth. Because of high delay on the communication link between the earth and the landing spacecraft, the landing operation needs to be done autonomously by the intelligent system installed on the spacecraft. The instruments carried on board may be very limited in mass and type. With the limited sensors onboard the spacecraft, one way to solve the problem is the use of optical navigation techniques that can navigate the spacecraft using the camera images of the planetary surface.

For the development and testing of the optical navigation algorithms, the testing facility for evaluation of the algorithm is needed. In DLR (German Aerospace Center) in Bremen where this thesis is performed, the Testbed for Robotic Optical Navigation (TRON), a hardware-in-the-loop test facility for optical navigation is building up. In this thesis, part of the development of TRON will be presented.

1.1 Optical navigation

Optical navigation is the method using the optical devices such as a camera or LIDAR to look at the environment to perform navigation tasks. In the context of the thesis, optical navigation means for the navigation of a flying body such as aircraft and spacecraft

looking at the terrain surface under it. The aim of the optical navigation is to control the spacecraft trajectory to follow the desired manoeuvre.

The manoeuvre such as moving to the flat spot on the surface and land there safely can be performed by the optical navigation methods. The LIDAR image of the terrain can determine the high variation of the area and evaluate whether the area is flat or not. The LIDAR or camera image sequence can tell about the movement of the spacecraft relative to the identified flat area. Then the spacecraft can be controlled to move to the flat area and touch the surface at low velocity. This operation may need the use of another sensors such as gyrometer and accelerometer.

Another example of optical navigation is the localization according to the terrain map. The aircraft or spacecraft can look at the terrain under it and determine the position and orientation of itself by comparing the terrain and the map. It can determine and control its manoeuvre according to the image taken by the camera onboard.

The problem considered in this thesis is the landing of the spacecraft using optical flow from the image of the terrain under the spacecraft regardless the identification of the landing site. The method is inspired by the ability of the insect to fly and land safely with very small brain and no stereo vision ability.

1.1.1 Literature on landing algorithm

On earth, the autonomous landing problem is an interesting topic in the field of aircraft and helicopter landing. The method used nowadays in the commercial aircraft is the use of GPS system with local ground base enhancement e.g. Ground-Based Augmentation System (GBAS). This method depends on the GPS system and also an expensive ground base system. On another planetary, moon or asteroid, the GPS system and the ground base facility are not available. An alternative method for the landing problem is the vision based method.

The vision for navigation is starting from the attempt to determine 3D motion of the camera from the image sequence by mathematical methods (Negahdaripour, 1987). However, humans as well as insects are able to land the aircraft or themselves without knowing their 3D motion. The study about the ability to navigate of the insect was introduced (Srinivasan et al., 1991). Entzinger and Suzuki (2009) have also studied several visual perceptions that human uses during the landing of the aircraft. From the literature, optical flow is one of the interesting clues that humans or insects use for their navigation during the landing.

There have been many studies on the use of machine vision to navigate and land the aircraft, notably the Unmanned Aerial Vehicle (UAV). Rives and Azinheira (2002) studied by comparison between the method of using the image sequence to reconstruct the aircraft position and the method of direct control the aircraft to track the target image. Chahl et al. (2004) has studied the landing of insects which use only the optical flow and implemented it to be tested with the robotic gantry. Beyeler et al. (2009) has implemented the insect navigation algorithm using only optical flow sensors from the optical mouse, rate gyros and airspeed sensor. They was able to take off and land a small UAV successfully.

For the planetary mission to date, most of the robotic landers have used the integration of the acceleration and angular velocity measurements from inertial measurement units (IMU) together with the velocity and altitude sensors such as Doppler radar or laser ranging device. The integration of the IMU measurement gives very large error due to model error, measurement noise and the error in the initial condition. A new method for precise landing under development is the use of camera images. A camera is a light weight sensor compared to the altitude measurement instrument and can give more information about the position relative to the terrain.

Many researches focus on the use of cameras to apply to planetary lander technology. The research on autonomous landing technology with the help of optical navigation is used in MUSES-C mission (Misu et al., 1999). The study aimed to design a system for autonomous landing on the asteroid near the earth. They extracted feature points from the camera image together with depth measurement of the features by a laser range finder instrument. The image features are tracked to determine the spacecraft inertial status.

Later, Mourikis et al. (2009) has studied the use of camera image by detecting the feature in the image and comparing it with the prior map. An extended Kalman filter (EKF) is used to integrate visual features and IMU measurements together. The algorithm provides estimation of the position, attitude and velocity of spacecraft relative to terrain with the test on sounding rocket.

From the literature research both from the technology on earth and the planetary missions, it is interesting to study the insect-inspired navigation method which uses the optical flow to apply with the spacecraft landing.

1.1.2 Insect navigation

Insects are animals with surprising navigation abilities. They cannot determine the depth of the visual image like human with two eyes because they are too small to have sufficient distance between their eyes. They have no sonar mechanism to detect the obstacle like in the bat. However, insects have never crashed to their environment when they are flying. They can land smoothly and softly without knowing the distance to contact. With very small size of their brain, they can navigate themselves to the food and fly back to their home precisely.

There are experiments trying to study navigation behaviour of the honeybee (Thakoor et al., 2005). The big conclusion about how the insects navigate themselves is the use of optical flow of the environment that changed by the movement of the insects. The insect tries to control its movement according to the amount of optical flow which automatically controls its movement to the correct manoeuvre as can be seen from Fig. 1.1.

From the studies, the insect tries to control the amount of optical flow to be constant during both flight and while landing. This results in automatically decreasing the approach velocity. When the insect is far away from the surface, a big amount of movement (high velocity) produces low amount of optical flow. When the insect is near the surface, just a small movement can produce high amount of optical flow. This is the reason behind the method of keeping the optical flow to be constant during the landing.

In this thesis, a landing algorithm inspired by the insect behaviour will be studied, implemented and tested in the hardware-in-the-loop simulation.

1.2 Hardware-in-the-loop simulation

During the development of optical navigation algorithm, the evaluation process of the method is needed to prove the concept as well as to confirm correct design and implementation. In reality, the navigation algorithm is dealing with sensor hardware and actuators hardware. There are many factors in the system with real hardware that are different from the software simulation such as realistic delay time, connection and communication logic, response time of the actuator, detail characteristic of material and etc. These factors cannot be completely modelled into the software simulation as they may be too complicated or have non-deterministic behaviour.

Software simulation is also important in terms of quick and easy to access at the beginning phase of development. However, a further proof of the navigation algorithm is needed after the software simulation. Hardware-in-the-loop simulation is a further

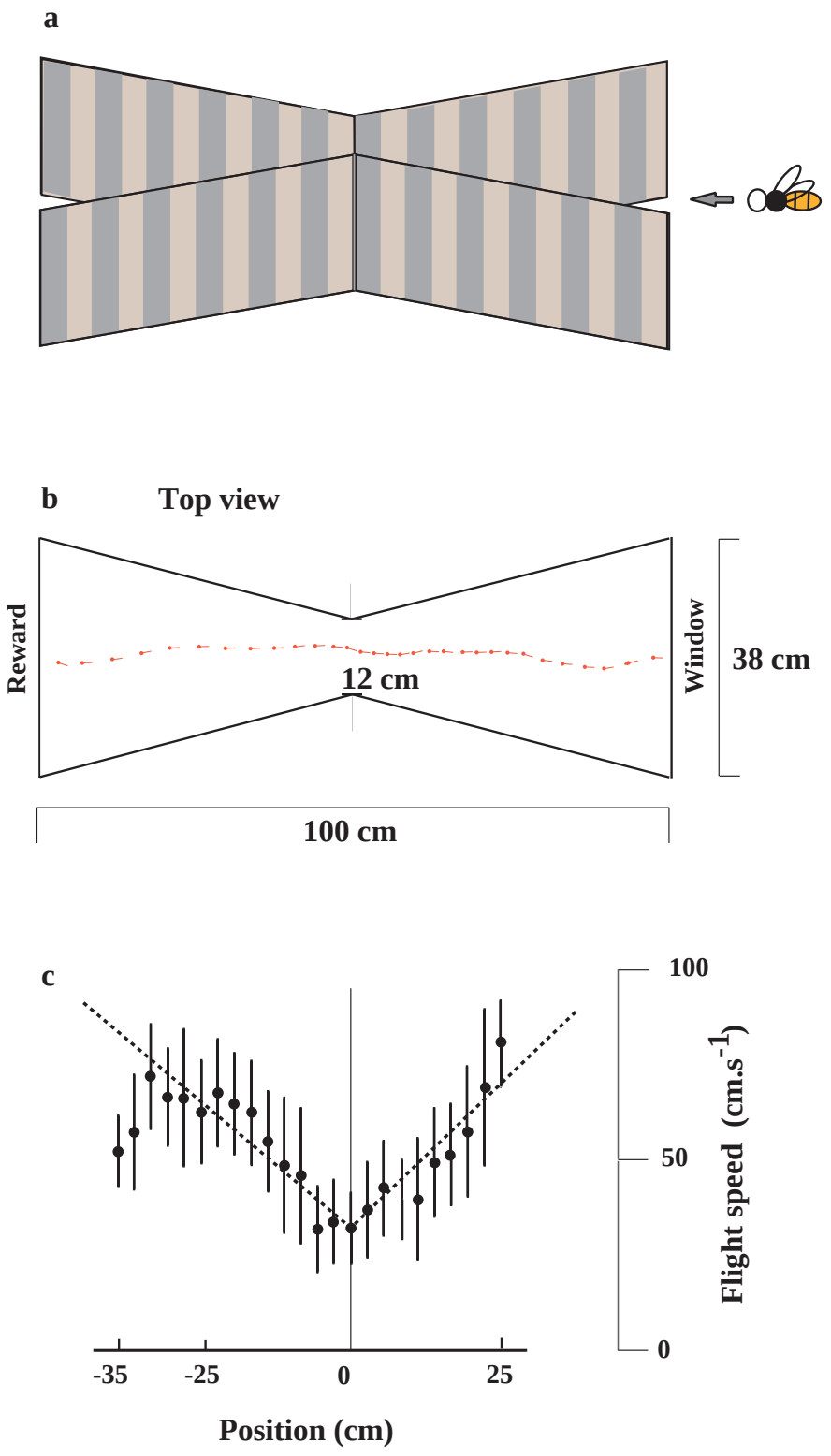


FIGURE 1.1: Experiment investigating visual control of flight speed. Bees flying through a tapered tunnel decrease their flight speed when the tunnel narrows, and increase the speed when it widens. The data indicate that bees control flight speed by holding constant the angular velocity of the image of the environment. Adapted from Srinivasan et al. (1996).

step before testing with the whole system environment which in the case of optical navigation is sending the spacecraft to space. Hardware-in-the-loop simulation is the simulation that tries to include as much as hardware to be used in the real situation into the control loop. The environment of the Hardware-in-the-loop simulation will be controlled to enabled cost and time effective development.

There is a hardware-in-the-loop test facility for the planetary surface and landing (de Lafontaine et al., 2008) but they are focusing on LIDAR application of the terrain. Another method for hardware-in-the-loop simulation is using a helicopter to simulate the spacecraft dynamic for testing landing algorithm (Saripalli et al., 2002).

1.2.1 Hardware-in-the-loop test facility in DLR

In DLR, a hardware-in-the-loop test facility (TRON) is under the development. A robotic arm for simulation of the dynamics of the spacecraft in 7-DOF has been installed. At the end of the robotic arm, an optical camera has been installed for looking at the scaled model of the planetary surface. The facility is designed to be able to perform the simulation of the spacecraft around planetary, moon and asteroid. The laboratory is also capable for testing of the optical navigation instruments such as LIDAR and camera. The current status of TRON can be seen in Fig. 1.2.

The robotic arm can be controlled by external system via Ethernet connection. The real-time communication between the spacecraft dynamic simulation and the robot will be implemented in this thesis. The algorithm for converting the dynamic of the spacecraft to the robot command will be studied. The mathematical definition of the terrain model also will be studied. The software implemented in this thesis is designed for general testing of any optical navigation algorithms e.g. crater detection algorithm, hazard avoidance and landing algorithm.

In this thesis, the insect-inspired landing algorithm will be tested in the hardware-in-the-loop facility. The dynamic of the spacecraft will be simulated by the movement of the robotic arm. The scaled terrain models manufactured to be similar to the planetary surface enable the camera to see the similar image as it is in another planetary.

1.3 Scope of the thesis

This chapter introduces the challenges in navigation of the spacecraft for landing in another planet with the limited instruments onboard. The method of insect-inspired

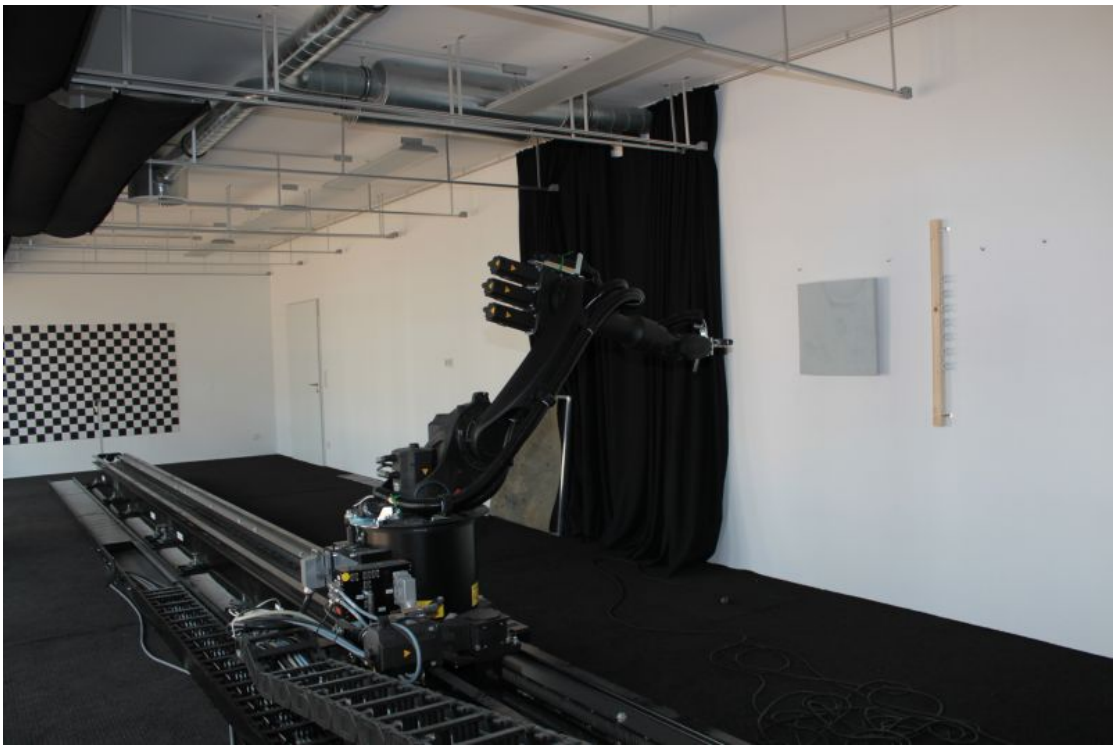


FIGURE 1.2: The laboratory facility of the hardware-in-the-loop simulation for optical navigation (TRON).

navigation by optical flow is chosen to study in this thesis with testing in hardware-in-the-loop simulation. The objective of this thesis is to setup and operate the hardware-in-the-loop simulation facility by applying the process of studied in insect-inspired landing algorithm. So, the thesis will divided into two mains part. The first is to setup the platform for the simulation which will be described in Chapter 2, Chapter 3 and Chapter 4. The second is the design and implementation of landing algorithm using optical flow described in Chapter 5 and Chapter 6. The testing of the designed algorithm will be discussed in Chapter 7. The conclusions and future work will be presented in Chapter 8.

The project in this thesis is feasibility studies of the landing algorithm and the proof of operation in the hardware-in-the-loop test facility. The project in this thesis focuses in high level system integration and operation of the test facility. The project does not consider very accurate detail of each subsystem component. For example the spacecraft dynamic simulation module is not the high accuracy dynamic simulation and the low level control of the spacecraft attitude and velocity is assumed to be perfect control. Simple control rule for the landing algorithm is implemented and test in the facility. The design and implementation of the image processing unit is not the most efficient algorithm that can be achieved. However, every system components are connected to

each other and perform a simple landing scenario in the hardware-in-the-loop implementation.

Chapter 2

Spacecraft Dynamic Simulation

To simulate the spacecraft movement in the laboratory, the dynamic of it has to be known. In the landing context, the spacecraft dynamic is mostly under the gravitational field and the thruster actions which can be represented as differential equations. From the differential equation, to get the position of the spacecraft, the equation needs to be integrated. In this thesis, the Runge-Kutta method has been chosen. The implementation of the dynamic simulation will be discussed in this chapter. Mission profile of the landing trajectory will be also discussed. In order to be able to control the spacecraft from the optical flow calculation, the actuator need to be simulated and integrated into the dynamic simulation.

2.1 Dynamic of point mass in a simplified gravitational field

A spacecraft is orbiting around its host because of its velocity and the acceleration due to the gravitational force. In this thesis, the spacecraft position is simulated from the differential equation related to the gravitational force using Newton's law of universal gravitation as

$$F = \frac{GMm}{r^2} \tag{2.1}$$

where F is force between two objects, G is the gravitational constant of $6.673 \times 10^{-11} \frac{m^3}{kg \cdot s^2}$, M is mass of the bigger object (host planetary) in kg, m is mass of the smaller object (spacecraft) in kg and r is the distance between two objects in m. The force of interaction between the spacecraft and its host object produces the motion dynamic according to Newton's first law of motion by

$$F = ma \tag{2.2}$$

where a is acceleration of the mass moving suppose to the force F. By putting force in Eq. 2.1 to be equal to Eq. 2.2 produces the differential equation of the spacecraft dynamic as

$$a = \ddot{x} = \frac{GM}{r^2} \tag{2.3}$$

where x is the position of the spacecraft in the gravitational field. To get the spacecraft position, the Eq. 2.3 needs to be integrated. The method chosen to solve the differential equation in this thesis work is Runge-Kutta 4th order.

2.1.1 Runge-Kutta 4th Order method

The Runge-Kutta 4th order method has been chosen instead of Euler method because the efficiency and quality of calculation is better. It does not require a very small time step in each integration loop. An implementation of the orbit simulation using this method can be found in Voesenek (2008).

The Runge-Kutta 4th order method specified the differential equation in the form

$$\begin{aligned} y' &= f(t, y), \\ y(t_0) &= y_0 \end{aligned} \tag{2.4}$$

where y is the state of the system, in this case is the dynamic of the spacecraft, and t is time. The integration of this problem is given by

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\ t_{n+1} &= t_n + h \end{aligned} \tag{2.5}$$

where h is the time step size and the coefficients k_1, k_2, k_3, k_4 defined as follows

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + \frac{h}{2}, y_n + k_1\frac{h}{2}), \\ k_3 &= f(t_n + \frac{h}{2}, y_n + k_2\frac{h}{2}), \\ k_4 &= f(t_n + h, y_n + k_3 h). \end{aligned} \tag{2.6}$$

The method estimates the coefficient k_1, uses it for the computation of coefficient k_2 then uses k_2 for the computation of k_3 and so on. The coefficients are used for calculation the value of the next time step in Eq. 2.5.

2.1.2 Implementation of the dynamic in gravitational field

The differential equation of the spacecraft dynamic in Eq. 2.3 can be written in the form of Eq. 2.4 by let the state y consists of position and velocity in Cartesian coordinate system which has the origin $(0, 0, 0)$ at the centre of the host object. The state vector y and y' defined as

$$y = \begin{pmatrix} x & \dot{x} \\ y & \dot{y} \\ z & \dot{z} \end{pmatrix}, y' = \begin{pmatrix} \dot{x} & \ddot{x} \\ \dot{y} & \ddot{y} \\ \dot{z} & \ddot{z} \end{pmatrix}. \tag{2.7}$$

By using the definition in Eq. 2.7, the spacecraft dynamic is described by function $f(t, y)$ as

$$y' = f(t, y) = \begin{pmatrix} \dot{x} & -GM\frac{-x}{|x|^3} \\ \dot{y} & -GM\frac{-y}{|y|^3} \\ \dot{z} & -GM\frac{-z}{|z|^3} \end{pmatrix}. \tag{2.8}$$

From the initial condition, y_0, the position and the velocity of the next time step can be computed. By using the current state to compute the state of next time step according to Eq. 2.5, the spacecraft position and velocity is computed at each time step.

2.2 Actuator Simulation

In this thesis, the scope of work does not cover the low level control of the spacecraft. The simulation assumed that the AOCS can respond accurately to the command from the navigation algorithm. The thesis will study only one actuator command which is changes of velocity (Δv). Δv are directly added to the simulation mathematic to change the state of the spacecraft. In this thesis, only the position of the spacecraft is simulated, the orientation is not considered.

2.3 Mission Profile

The landing trajectory will consist of three parts: parking orbit, descent orbit and touch down trajectory. In this thesis, the mission profile used in the simulation is for a lunar landing mission which has a circular parking orbit at an altitude of 100 km. The spacecraft will perform a Hohmann transfer to be closer to the surface at the altitude of 10 km. Then the spacecraft changes its mode into preparation for touchdown mode which will use the gravity turn manoeuvre.

Another scenario that will be studied in this thesis is the asteroid landing. Asteroid can be considered to have very low gravitational force, so the gravity turn maneuver cannot be used. The spacecraft that will be landed on the asteroid cannot carry heavy instruments or sensor. So the algorithm needs to be able to perform in limited set of sensors. In this case, the spacecraft will control its landing on the asteroid using just the optical flow.

Chapter 3

Simulation Platform

The aim of the simulation platform is to be able to acquire the image according to the camera view from different spacecraft position in the orbit and during landing manoeuvres. The spacecraft position is changing according to orbital dynamics and control signals which in this case is derived from the calculation of the image that has been acquired from the former spacecraft positions. However, during the development of the navigation algorithm, the spacecraft cannot be in the orbit and produce the image for testing. So, the simulation platform is setup for generation of the images that spacecraft is expected to see from the simulated spacecraft position in Section 2.1. Then the navigation algorithm uses the image generated to compute the control signal and uses that signal to calculate new spacecraft position. The new image will be generated using current spacecraft position and the new control signal will be computed and so on.

The simulation mentioned above can be done using two methods. First is to calculate everything using computer graphic to generate the image from the terrain model and camera position (software simulation). Second is to have a real camera looking at a small scaled terrain model in which the camera position is scaled down from the spacecraft position in the dynamic simulation (hardware-in-the-loop simulation). During the development of the hardware-in-the-loop simulation, a system to control the robot is needed to be developed. A robot simulation system is used before apply the system to the real robot hardware.

3.1 Software simulation

To complete the simulation task, Planet and Asteroid Natural Scene Generation Utility (PANGU) (Parkes et al., 2004) has been chosen. It supports the network interface,

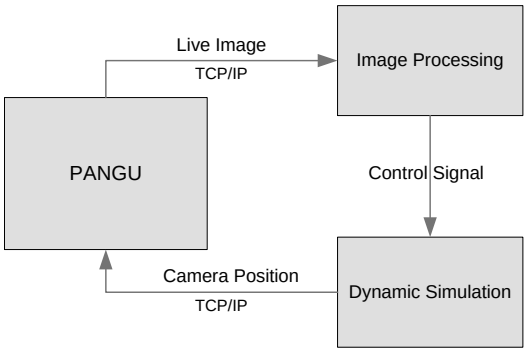


FIGURE 3.1: Block diagram of the simulation platform using PANGU.

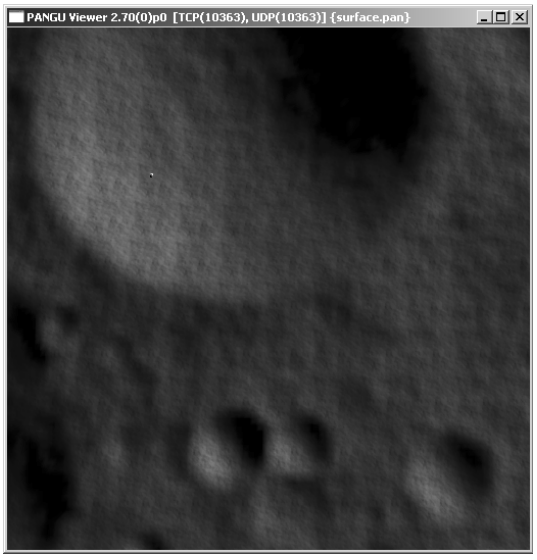


FIGURE 3.2: An image, generated by PANGU.

enabling user applications to input the spacecraft position and get the image sequence according to the input position. The software supports a wide range of object from small surface terrain plate to the planet or asteroid model. From the three-dimension model, the image can be generated using camera projection mathematics to simulate the image view from different camera position.

In software simulation, the spacecraft dynamic is calculated and passed it to PANGU. Then, PANGU generates an image resulted from the spacecraft movement. The image processing unit calculates the control signal from the sequence of image that changes the dynamic of the spacecraft. The new position of the spacecraft is passed to PANGU again and the new image is generated. The new control signal is calculated from the image and so on. The block diagram of the closed loop software simulation can be seen in Fig. 3.1. The resulting image from PANGU (Fig. 3.2) seems to be realistic but it has not included the important factors such as system delay, image grabbing time and the detail is lost which can be seen from the recurring square pattern over the image. In PANGU,

the level of detail in the terrain is limited due to the limited number of polygon that the software can handle. The lighting properties of the material also has not supported in PANGU. One big disadvantage of the software simulation is that it cannot completely provide natural properties of the system. The real world situation is too complicated to be modelled as a computer program including every factor in the same time. The small system properties such as dirt in the optic system and its consequence cannot be modelled properly in the simulation.

The software simulation is very useful in the process of developing the algorithm that needs a lot of proving of the principle concept and a certain robustness level. After that, hardware-in-the-loop simulation will be used in order to perform the algorithm in the environment that is closer to the reality.

3.2 Hardware-in-the-loop simulation

The Hardware-in-the-loop simulation is an important step for testing the navigation algorithm. The algorithm will experience almost realistic system delay and give rise to unexpected problems. This thesis will setup the software interface and components for performing the Hardware-in-the-loop simulation for any optical navigation related mission in the TRON laboratory.

3.2.1 Laboratory setup

The laboratory is setup to be able to simulate the optical, geometric and dynamic condition of the spacecraft during landing. The robot arm KUKA KR16 on the KL250/2 rail is used for simulate the dynamics of the spacecraft. The robot carries optical camera and navigation sensor package as optional at the end of the arm. The 3D terrain model plate is placed on the wall will simulate the geometric view that is equivalent to the view from the camera in orbit. The lighting and shadow also can be controlled using the light source to simulate the sun. The desired layout of the laboratory can be seen in Fig. 3.3. For the detail design of the laboratory can be found in (Krüger and Theil, 2010). The setup for hardware-in-the-loop simulation in this thesis can be found in Fig. 3.4.

3.2.2 Robot Control System Architecture

The robot controller used is KRC2 which consists of Windows XP embedded OS and the VxWorks real-time OS. The robot controller is running under the VxWorks context

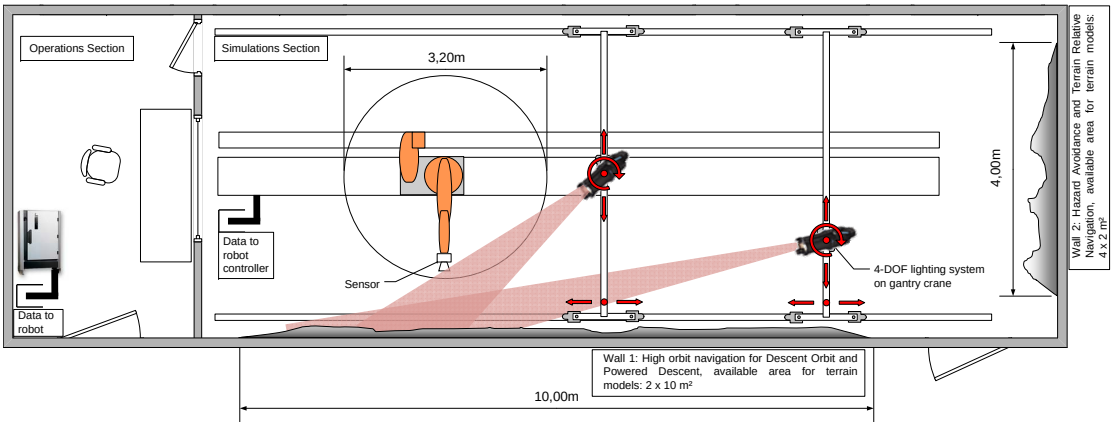


FIGURE 3.3: Schematic top view of TRON, adapted from (Krüger and Theil, 2010).

FIGURE 3.4: The laboratory setup for the landing simulation in this thesis.

which can be command via a user interface for creating KRL programmes and manual movements running on Windows XP embedded. The robot is controlled by dSPACE real-time system via the RSI-XML interface of the KUKA robot controller over the local area network. The reason for using dSPACE is that the robot controller required the real-time commanding system which the normal PC without real-time operating system cannot perform the task. Furthermore, dSPACE system provides an interface to the development on MATLAB/Simulink[1] which can be directly converted to run in the dSPACE system by Real-time workshop. Because a lot of model and simulation has been already done in MATLAB/Simulink, dSPACE system is perfectly choice for the laboratory. The system architecture of the robot control from dSPACE system can be found in Fig. 3.5. The configuration of the network connection in the laboratory can be found in Fig. 3.6.

[1]MATLAB/Simulink is a trademark of The MathWorks, Inc.

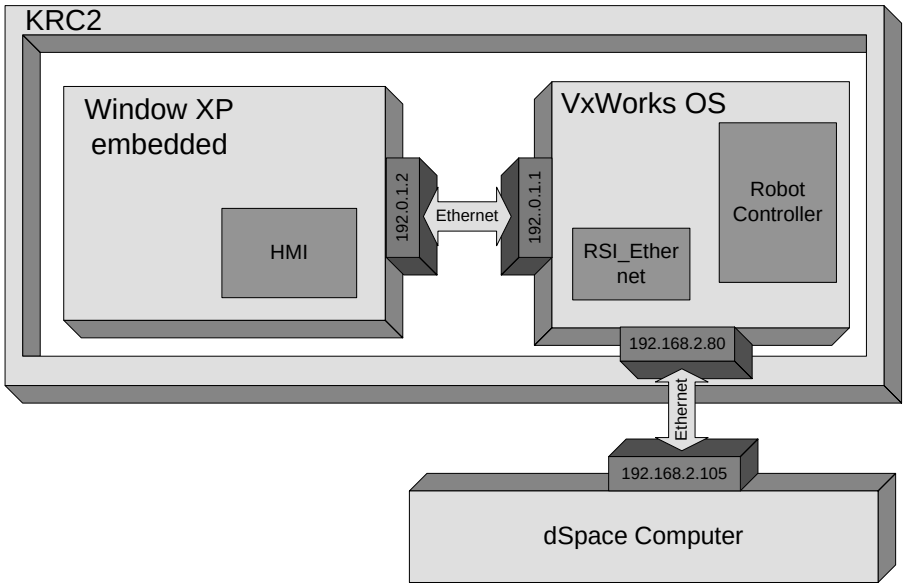


FIGURE 3.5: System components in TRON laboratory for the hardware-in-the-loop simulation.

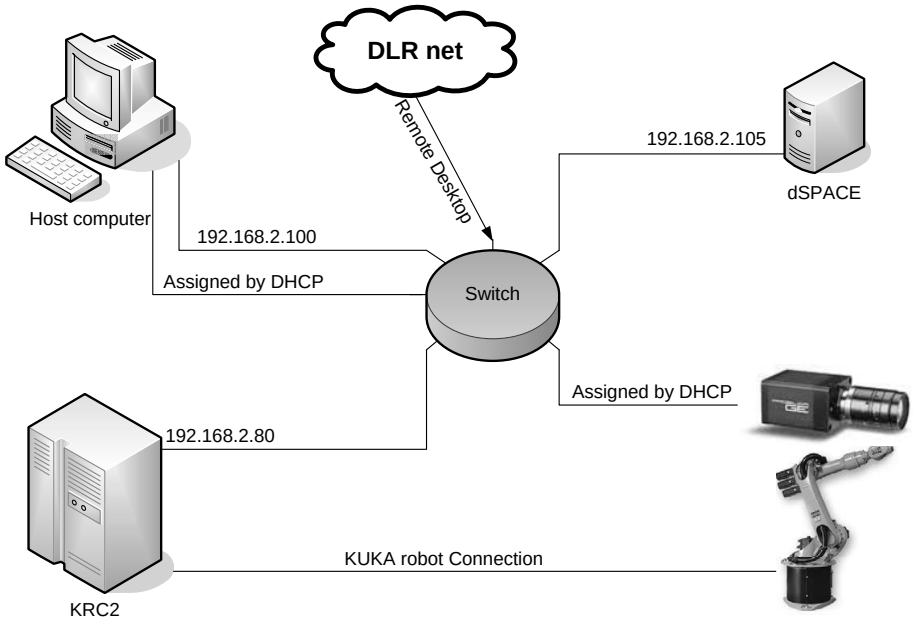


FIGURE 3.6: Network connection and the IP address of the components in the laboratory.

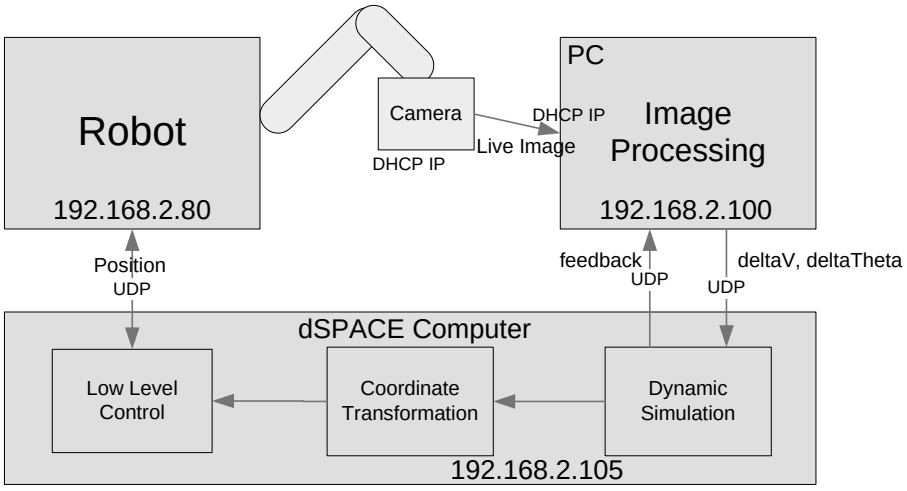


FIGURE 3.7: Control scheme of the hardware-in-the-loop simulation.

3.2.3 Closed-Loop Control Scheme

The dSPACE system controls the robot that carries the camera by the command from image processing unit as can be seen in Fig. 3.7. The camera moves and gives the new image to process in image processing unit. This is the desired hardware-in-the-loop simulation for optical navigation. The navigation algorithm can be developed and tested using this platform.

3.3 Robot simulation during the development

During the development of the hardware-in-the-loop simulation which involves robot control task, the robot controller simulation software (KUKAsim and OfficeLite) which has the same configuration as the real robot controller is used to ensure the safety during preliminary testing and speed up the development. The control scheme during the testing allows the control signal calculated from software simulation with PANGU to control the robot position in the simulation. The robot in the simulation will move the same as the spacecraft moves in PANGU. In the future, this control scheme also can be used to set up the comparison between the image from PANGU and the image from the laboratory with the same spacecraft trajectory. The diagram of this control scheme can be seen in Fig. 3.8.

The software KUKAsim is a graphical user interface that enables the user to analyze the movement of the robot. The software OfficeLite performs robot dynamic simulation, calculation of the control rule and calculation of the inverse kinematic. OfficeLite

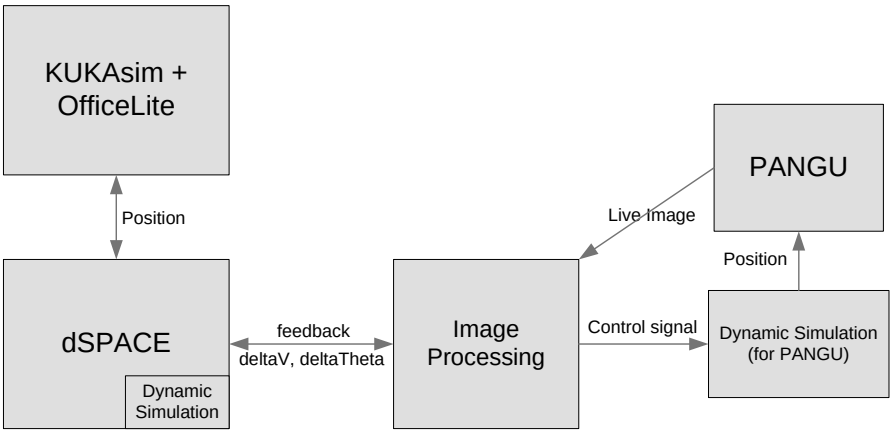


FIGURE 3.8: Control scheme during the testing with simulated robot controller.

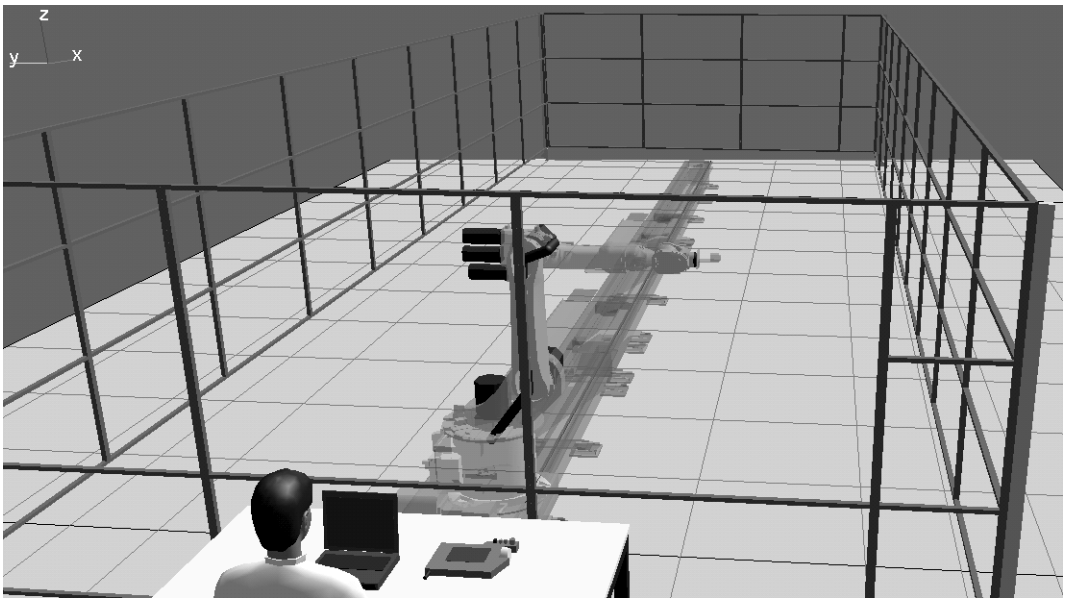


FIGURE 3.9: Laboratory setup in KUKAsim.

also provides simulated user interface to control the robot called KUKA HMI (Human Machine Interface). The laboratory setup in KUKAsim is shown in Fig. 3.9.

The robot controller itself is running on a real-time operation system, VxWorks, and KUKA HMI is a user interface to command the controller in the VxWorks. VxWorks in the controller simulation is running on the virtual machine inside the host operating system which is Windows XP. KUKAsim communicates with robot controller in VxWorks via a virtual network adapter. The diagram showing the connection between the components in the simulation platform is shown in Fig. 3.10

In order to control the robot from the dSPACE system, the control data has to be propagated from the real network interface on Windows XP to the VxWorks. A small

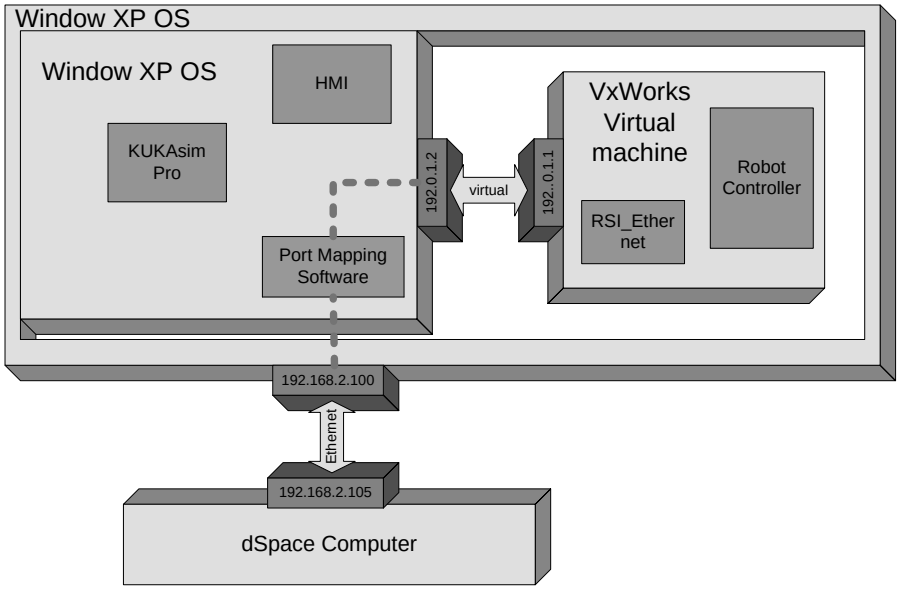


FIGURE 3.10: System components for the robot simulation using KUKASim.

program that can map the package from one port on one interface to another port on another interface is needed. That program is the port mapping software block in Fig. 3.10 developed during this thesis work.

Chapter 4

Robot Control for Hardware-in-the-Loop Simulation

In this chapter, the design and implementation of the laboratory system for hardware-in-the-loop simulation will be discussed. The aim of the simulation is to get the real-time image from the spacecraft dynamic from the simulation in Chapter 2 to be used in the navigation algorithm. To do this in the hardware-in-the-loop simulation, the robot that carries the camera has to be controlled to move correctly in the laboratory. The transformation from the spacecraft dynamic simulation coordinate system to the laboratory coordinate system will be described. The terrain model definition will be defined. The communication to the robot controller will be considered here. The control signal from the image processing module processing the optical navigation algorithm will be integrated to the robot control loop. The telemetry containing the robot state from the robot propagated to the image processing module for evaluation of the algorithm will be described. The synchronization of the optical navigation control loop and the robot control loop will be discussed.

4.1 Reference coordinate system

In hardware-in-the-loop simulation, there are two reference coordinate systems to be considered which are world reference and laboratory reference. The spacecraft simulation is done in the world reference. World reference in this thesis, the zero point $(0, 0, 0)$ is fixed to be at the centre of the moon or the asteroid. The simulated spacecraft position is transformed into the laboratory reference which is used for robot control.

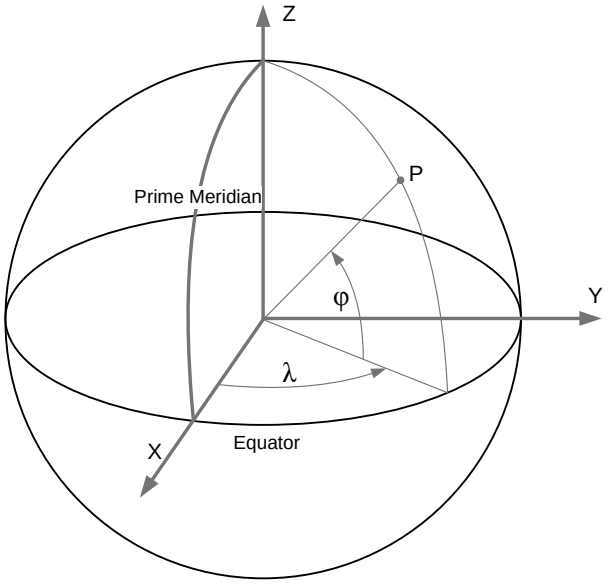


FIGURE 4.1: World reference system and its relation to latitude and longitude.

4.1.1 World reference system

The world reference system has the origin $(0, 0, 0)$ at the centre of the object of interest. The +Z axis is the north pole of the object of interest. The +X axis is pointing along the equator and the prime meridian is start from the point that the equator intersect with the +X axis. The +Y axis is at the direction that give the cross product of $\vec{X} \times \vec{Y} = \vec{Z}$ according to the right hand rule. Figure 4.1 shows the orientation of the axis and its relation to the latitude, λ, and longitude, φ, coordination.

4.1.2 Laboratory reference system

The spacecraft position in orbit has to be transformed in to the laboratory reference system which define as in Fig. 4.2. The origin point is placed at the zero point of the KL250/2 rail. The +X axis is pointing toward the wall that installed the terrain model. The +Y axis is point along the robot rail outward from the control room. The +Z axis is point up to the ceiling. This reference system is exactly the same as the reference system defined in the KUKA Roboter system used for controlling the robot.

4.2 Terrain model definition

The terrain model is a scaled area of the surface of the object of interest that will be installed in the laboratory wall. The model definition needs to define the surface area

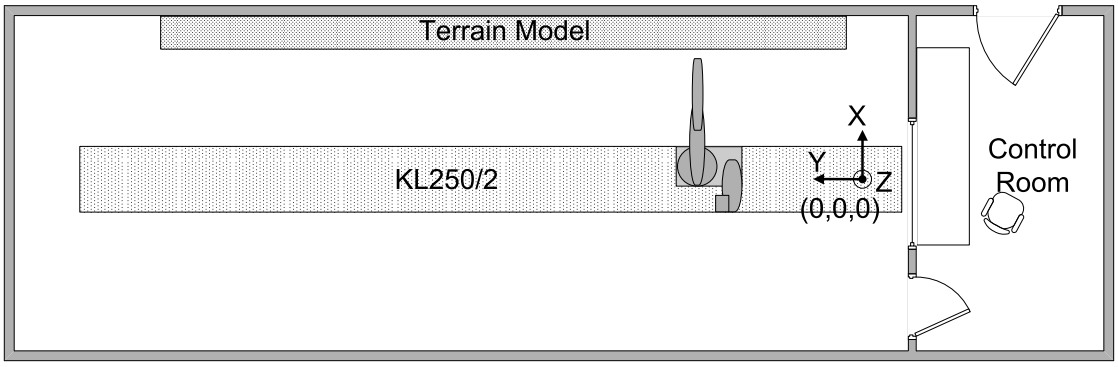


FIGURE 4.2: Laboratory reference system.

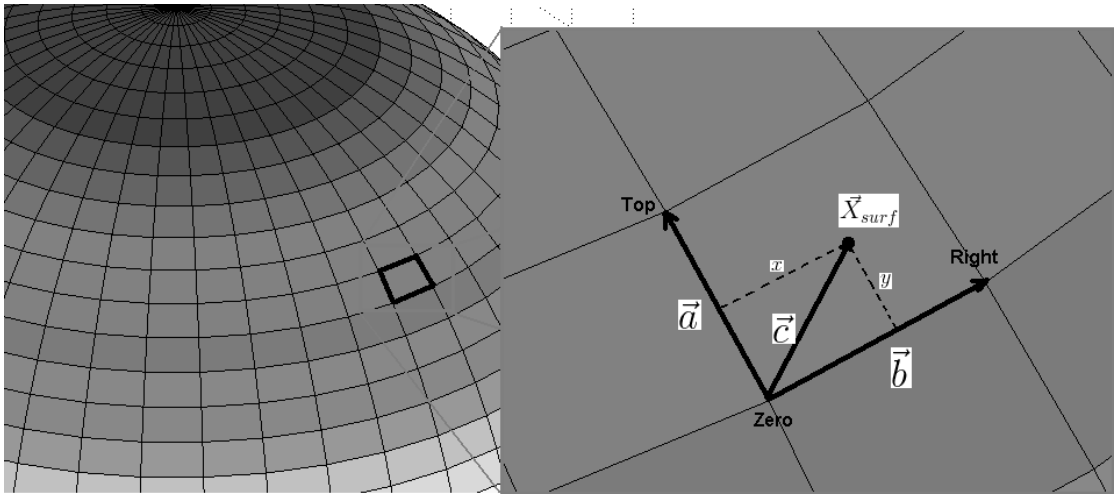


FIGURE 4.3: Surface definition and vectors of the terrain model in the world reference system.

in the world reference system which matches to the position of the terrain model in the laboratory reference system.

4.2.1 Definition in world reference system

The terrain model definition in the world reference is defined by three points of latitude and longitude on the surface area which are points "Zero", "Top" and "Right" of the rectangle. Note that three points are needed to define the rectangle in latitude and longitude system if the latitude of the terrain model is high, the surface cannot be rectangle with only two corner points of definition. Illustration of the surface area definition and related vector for further calculation can be seen in Fig. 4.3.

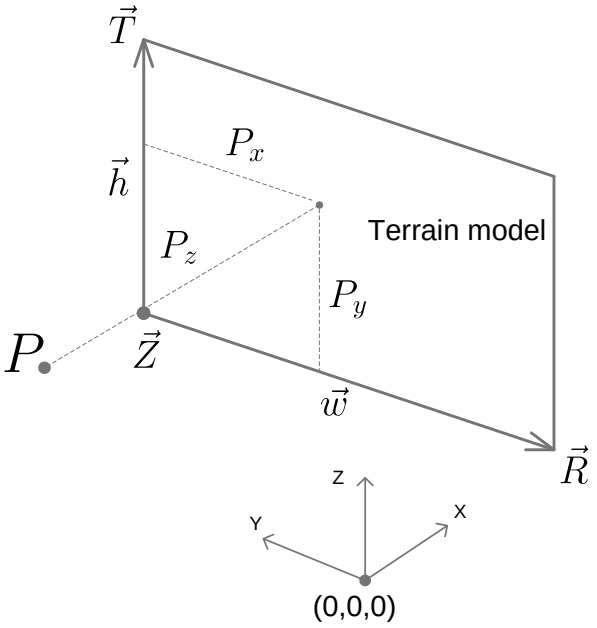


FIGURE 4.4: Vectors and points of the terrain model in laboratory reference system.

4.2.2 Definition in laboratory reference system

The terrain model in laboratory is a plate of scaled terrain installed on the laboratory wall. A flat plate rectangle also can be defined by three points of "Zero" ($\vec{Z}$), "Top" ($\vec{T}$) and "Right" ($\vec{R}$) in laboratory reference system. The plate definition and related vector for further calculation is shown in Fig. 4.4. Vectors $\vec{h}$ and $\vec{w}$ are the vector of height and width of the terrain model in the laboratory reference system.

4.3 Transformation from world reference system into laboratory reference system

The transformation uses the relation of three points matching from the world reference system to laboratory reference system. The point "Zero" in world reference system is corresponding to the point "Zero" in the laboratory reference system. The same also applies to the point "Top" and "Right". With this relationship, the transformation for two reference systems can be derived.

The process of transformation begins from the transformation of latitude and longitude data into world reference system. The spacecraft position will be projected to the surface. Then the projected point on the surface area is transformed to the surface plate reference system. Then it is transformed by scaling into the terrain model reference

system. The position in terrain model reference system is then transformed into the laboratory reference system.

4.3.1 Latitude and longitude transformation into world reference system

First, the latitude, φ, and longitude, λ, of the surface area definition need to be transformed into world reference system (x, y, z) by

$$\begin{aligned} x &= R \cdot cos(\varphi) cos(\lambda) \\ y &= R \cdot cos(\varphi) sin(\lambda) \\ z &= R \cdot sin(\varphi) \end{aligned} \tag{4.1}$$

where R is the radius of the object of interest.

4.3.2 Transformation of spacecraft position into terrain reference system

The spacecraft position, $\vec{X}_{world}$, is projected to the surface as in Fig. 4.5. Assuming that the projected point, $\vec{X}_{surf}$, is on the surface around the interested surface area, then X and Y component of $\vec{X}_{surf}$ in the plate reference system can be calculated by

$$\begin{aligned} \vec{X}_{surf}(x) &= \frac{\vec{a} \cdot \vec{c}}{|\vec{a}|} \\ \vec{X}_{surf}(y) &= \frac{\vec{b} \cdot \vec{c}}{\left|\vec{b}\right|} \end{aligned} \tag{4.2}$$

by the help of vector $\vec{a}$, $\vec{b}$ and $\vec{c}$ shown in Fig. 4.3. Then it is transformed into the terrain model reference system in the laboratory by

$$\begin{aligned} P_x &= \frac{\vec{X}_{surf}(x) \left|\vec{h}\right|}{|\vec{a}|} \\ P_y &= \frac{\vec{X}_{surf}(y) \left|\vec{w}\right|}{\left|\vec{b}\right|} \end{aligned} \tag{4.3}$$

where (P_x, P_y) is X and Y component of the scaled position in terrain model reference system. The illustration of the notation can be found in Fig. 4.4.

The altitude of the spacecraft position, $X_{alt} = \left|\vec{X}_{world} - \vec{X}_{surf}\right|$, is scaled into the Z component, P_z, of the terrain model reference system by the scale factor, $\mu = \frac{|\vec{a}|}{|\vec{h}|} = \frac{|\vec{b}|}{|\vec{w}|}$,

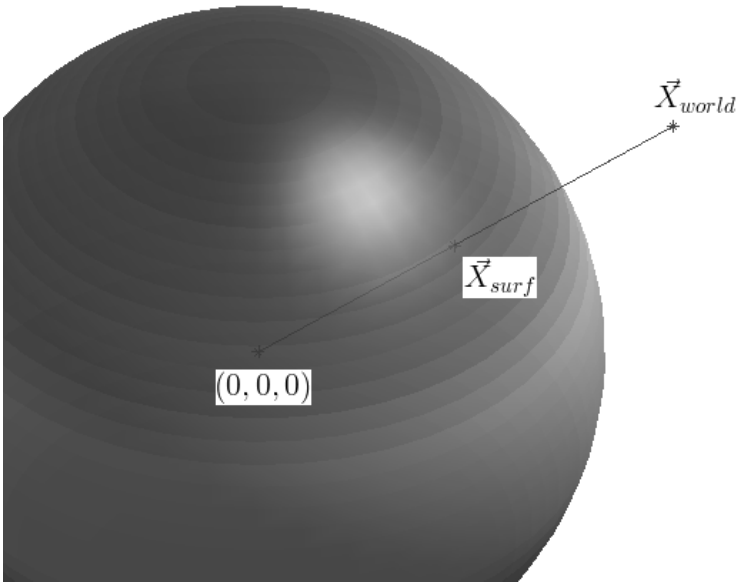


FIGURE 4.5: The projection point on the surface of the spacecraft position.

of the terrain model as

$$P_z = \mu \cdot X_{alt}. \tag{4.4}$$

4.3.3 Transformation of position in terrain model reference system into laboratory reference system

Let $\vec{x}$ represent the spacecraft position in terrain model reference system as

$$\vec{x} = \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} \tag{4.5}$$

where P_x, P_y, P_z have been calculated in the former section. The desired laboratory coordinate of the spacecraft position is

$$\vec{x}_{lab} = A\vec{x} \tag{4.6}$$

where A is a transform matrix from the terrain model reference system to the laboratory reference system.

Matrix A can be solved by using the relation of the point "Zero", "Top", "Right" which are in the laboratory reference system will be represented by $\vec{Z}, \vec{T}, \vec{R}$ consequently.

At the "Zero" point which is $\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}^T$ in the terrain model reference system, the relationship can be written by

$$\begin{bmatrix}Z_x\\Z_y\\Z_z\\1\end{bmatrix} = \begin{bmatrix}? & ? & ? & ?\\? & ? & ? & ?\\? & ? & ? & ?\\0 & 0 & 0 & 1\end{bmatrix} \times \begin{bmatrix}0\\0\\0\\1\end{bmatrix}. \tag{4.7}$$

It can be seen that the last column of the transformation matrix are solved.

The same for the "Top" and "Right" points which are $\begin{bmatrix}\left|\vec{h}\right| & 0 & 0 & 1\end{bmatrix}^T$ and $\begin{bmatrix}0 & |\vec{w}| & 0 & 1\end{bmatrix}^T$ in the terrain model reference system, the relationships can be written by

$$\begin{bmatrix}T_x\\T_y\\T_z\\1\end{bmatrix} = \begin{bmatrix}? & ? & ? & Z_x\\? & ? & ? & Z_y\\? & ? & ? & Z_z\\0 & 0 & 0 & 1\end{bmatrix} \times \begin{bmatrix}\left|\vec{h}\right|\\0\\0\\1\end{bmatrix}, \tag{4.8}$$

$$\begin{bmatrix}R_x\\R_y\\R_z\\1\end{bmatrix} = \begin{bmatrix}? & ? & ? & Z_x\\? & ? & ? & Z_y\\? & ? & ? & Z_z\\0 & 0 & 0 & 1\end{bmatrix} \times \begin{bmatrix}0\\|\vec{w}|\\0\\1\end{bmatrix}. \tag{4.9}$$

From the relation of the "Top" and "Right" points, the first and second column of the transformation matrix can be solved as

$$A = \begin{bmatrix}\frac{\vec{h}_x}{|\vec{h}|} & \frac{\vec{w}_x}{|\vec{w}|} & ? & Z_x\\ \frac{\vec{h}_y}{|\vec{h}|} & \frac{\vec{w}_y}{|\vec{w}|} & ? & Z_y\\ \frac{\vec{h}_z}{|\vec{h}|} & \frac{\vec{w}_z}{|\vec{w}|} & ? & Z_z\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{4.10}$$

with the knowledge of vector $\vec{h}$ and $\vec{w}$ calculated by

$$\begin{aligned}\vec{h} &= \vec{T} - \vec{Z},\\ \vec{w} &= \vec{R} - \vec{Z}.\end{aligned} \tag{4.11}$$

With the knowledge of the scaled altitude in terrain model reference system and in laboratory reference system is the same, the relationship for solving the last column of

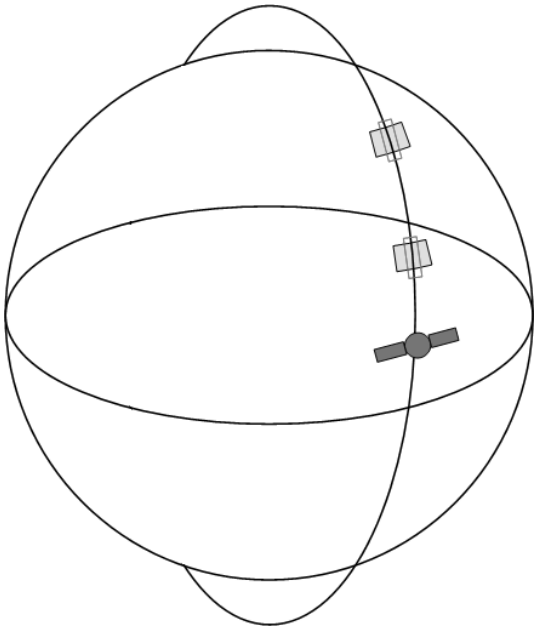


FIGURE 4.6: Parts of the orbit chosen to be simulated in the laboratory.

the transform matrix can be written by

$$\begin{bmatrix} Z_x + (\vec{h} \times \vec{w})_x \\ Z_y + (\vec{h} \times \vec{w})_y \\ Z_z + (\vec{h} \times \vec{w})_z \\ 1 \end{bmatrix} = \begin{bmatrix} a & d & ? & Z_x \\ b & e & ? & Z_y \\ c & f & ? & Z_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ \left|\vec{h} \times \vec{w}\right| \\ 1 \end{bmatrix}. \tag{4.12}$$

Finally, the transformation matrix, A, has been solved as

$$A = \begin{bmatrix} \frac{\vec{h}_x}{|\vec{h}|} & \frac{\vec{w}_x}{|\vec{w}|} & \frac{(\vec{h}\times\vec{w})_x}{|\vec{h}\times\vec{w}|} & Z_x \\ \frac{\vec{h}_y}{|\vec{h}|} & \frac{\vec{w}_y}{|\vec{w}|} & \frac{(\vec{h}\times\vec{w})_y}{|\vec{h}\times\vec{w}|} & Z_y \\ \frac{\vec{h}_z}{|\vec{h}|} & \frac{\vec{w}_z}{|\vec{w}|} & \frac{(\vec{h}\times\vec{w})_z}{|\vec{h}\times\vec{w}|} & Z_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.13}$$

From the complete transformation matrix in Eq. 4.13, the spacecraft position in laboratory reference system which can be used for control the robot is calculated by Eq. 4.6.

4.4 Transition between different orbit parts

Because the terrain model in the laboratory cannot cover the whole flight, some selected parts of the orbit are chosen to be simulated in the laboratory. Figure 4.6 shows an

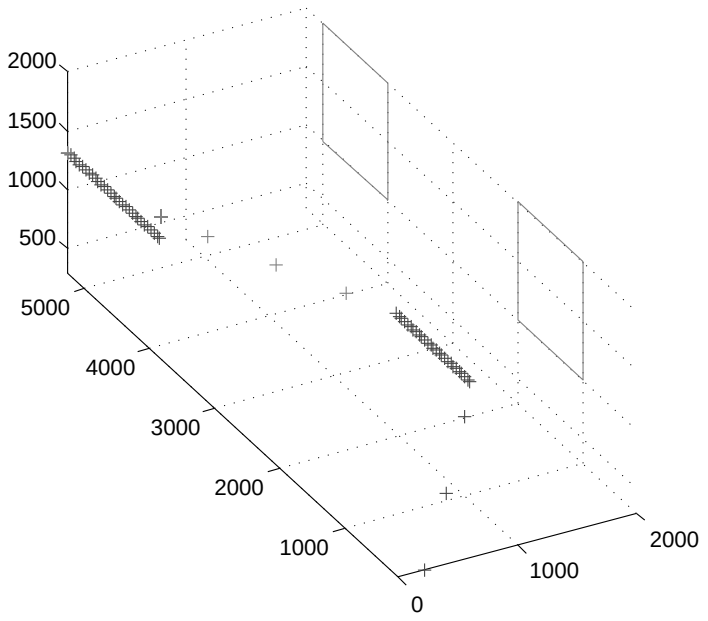


FIGURE 4.7: Robot trajectory in the laboratory moving along 2 terrain model.

example of the selected orbit parts and the terrain surface associated with them. The robot trajectory for an orbit with two orbit parts which are simulated in the laboratory with different scale is shown in Fig. 4.7. The rectangles on the wall represent the terrain model and the plus signs shows the position of the robot.

In the laboratory, the robot movement between each part of the orbit is planned according to the knowledge of the scale of terrain model, the spacecraft velocity and the robot limiting factors. Because the robot does not know the simulation result in the future, it knows just only the current position of the spacecraft, it is impossible to go to the correct position in the future.

The strategy used here is to move the robot to standby at the point near the next terrain model. The altitude of the spacecraft can be predicted from the orbital parameter and the terrain model scale. The distance of the starting point from the edge of the terrain model is proportional to the predicted spacecraft velocity divided by the ability of the robot to accelerate. If the orbital velocity is high and the limited acceleration of the robot is low, then the distance that the robot needs to accelerate to the desired velocity is long. This also affects to the time that the robot should start to move to acquire the desired velocity.

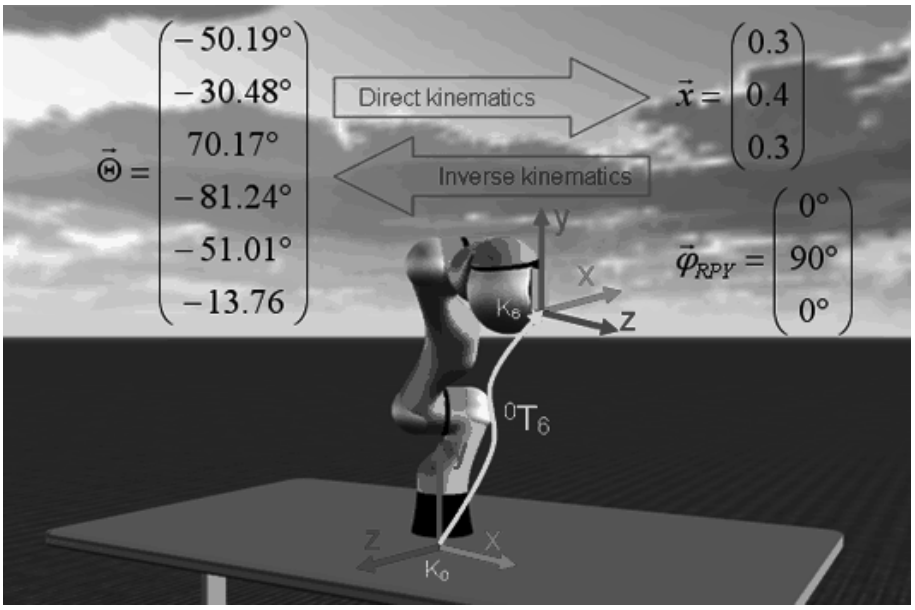


FIGURE 4.8: Direct kinematics and inverse kinematics calculation (KUKA Robotics Corp).

4.5 Command Interface

In this thesis, RobotSensorInterface (RSI) for KUKA robot is chosen for controlling the robot because of the support in real-time control over the Ethernet. The KUKA robot controller sends out the XML string containing robot status and IPOC[1] number as UDP packets over the IP network to the specified IP address at 85 Hz. Then it waits for a reply from a server application which in this case is the dSPACE computer. The server needs to send the command to control the robot also in XML format with the same IPOC number as it got from the request within 12 ms. The coordinate system chosen to command the robot is the absolute coordinate system. The KUKA robot controller provides the possibility to control the tool centre point (TCP)[2] in Cartesian coordinates relative to the starting point of the RSI context[3]. Once the position of the spacecraft is transformed into laboratory reference system which is the TCP in Cartesian coordinate system (x, y, z, A, B, C), the robot controller will calculate inverse kinematics from the TCP position to each joints that can be used to control the robot. Figure 4.8 shows the description of direct and inverse kinematics calculation. The detailed setup configuration and protocol of the interface can be found in KUKA Roboter GmbH (2009).

[1]IPOC number is a packet number from the KUKA robot controller, it is increasing over time.

[2]TCP is the position of the instrument at the end of the robot arm. This position in this thesis is the position of the camera that the robot is carrying.

[3]RSI context in this thesis is the context that the robot receives the command position from the Ethernet

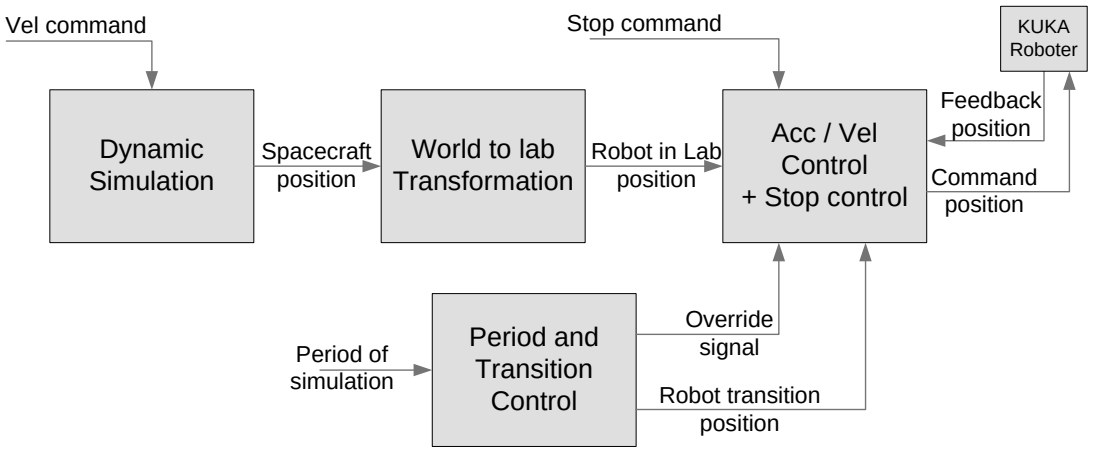


FIGURE 4.9: Block diagram of the transformation from the dynamic simulation to the robot control command.

4.6 Transformation from laboratory coordinates to robot command

The robot is controlled by sending an absolute position relative to the starting position of RSI context. This coordinate system is defined as robot coordinate system. The dSPACE computer remembered the start position from the first package that it received from the KUKA robot controller. Then dSPACE computer transforms robot position from the laboratory reference system to the robot coordinate system by simple subtraction. The current state of the robot is also considered for the next command to be limited in acceleration and velocity. Boundary limitation checking is also implemented in the last process before the command is sent to the robot controller. The whole picture of the transformation from the dynamic simulation until the position is transformed to the robot command can be seen in Fig. 4.9.

4.6.1 Acceleration and velocity control

Because the robot has limited ability in acceleration and velocity, the trajectory position that is sent to robot need to be under these limitations, otherwise, the robot controller will reject the command and stop the robot. In this thesis, the transformed position from the simulation into laboratory reference system is filtered by the acceleration and velocity limit module without concerning the actual simulation. So, the robot position may not exactly match to the actual simulation at the beginning of the simulation when the robot needs time to gain the desired velocity. The robot needed velocity is calculated from the current robot state received from the KUKA robot. Then the velocity is compared to the maximum limitation and the acceleration limit with the last

frame velocity. If the needed velocity exceeds the limit, the command position will be modified to allow just maximum velocity and acceleration. The position in the dynamic simulation is not affected by this operation. The robot position in the laboratory and in the simulation will reach the same position at some point that the robot gains enough velocity to catch the simulation position. This shall happen only before the robot can see the terrain model during the acceleration phase. The margin distance is planned as mention in Section 4.4.

4.6.2 Stop control

The robot also cannot be immediately stopped, it has to be slow down enough to stop. The stop strategy used here is to always command the robot at the same position as its current state and let the acceleration and velocity control to adjust the command to the maximum acceleration and velocity allowance. This method has a small problem when the robot is very slow and gives slightly bias measurement of its state. The robot will move slowly away and not completely stop because the command moves it away by the wrong measurement. The small threshold is set to command the robot to be at the same position if the robot state is changed less than the threshold. The method can effectively stop the robot from moving.

4.7 Control feedback from image processing module

During the simulation, the robot can be controlled from the external system which can process the real-time signal from the sensors. From the project in this thesis, a Gigabit Ethernet camera (Prosilica GC1350) installed at the end of the robot arm is the only sensor used for navigation and control of the spacecraft. The live image from the camera is processed on a desktop computer producing the control data to be sent over the Ethernet to dSPACE computer. This command is a UDP packet which provides the information about change of velocity (Δv).

4.8 Telemetry from the robot controller

To be able to compare and study the efficiency of the algorithm, the real current robot state is send back to the image processing module. The current robot position and the control signal can be recorded for analyzing of the response time. The relation between the control signal and the robot state also can be analyzed. The recorded data also can be used by many other data analysis methods. The packet from dSPACE computer is

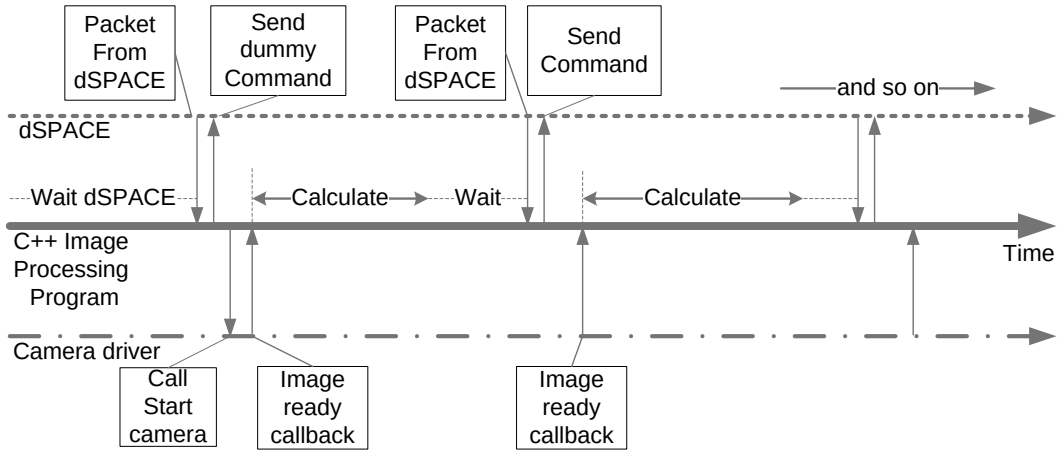


FIGURE 4.10: Time sequence event on image processing module (C++ program) interacts with dSPACE and the camera driver.

sent to the image processing module equal to the processing rate which in this thesis set at 8.5 Hz. Future development on the dSPACE computer can be done to enable the data recording on the dSPACE system at 85 Hz.

4.9 Synchronization of simulation and control loop

Because the image processing task takes a lot of time in calculation, it cannot provide control data rate as high as the simulation loop running at 85 Hz. The image processing program will provide the command at the rate of 8.5Hz instead which is enough for concept proving. The simulation in dSPACE computer sends one telemetry packet out over ten simulation executions to the image processing module which mean, it send the telemetry packet out at the rate of 8.5 Hz. The image processing module waits for the packet from dSPACE and then tells the camera to start the continuous acquisition at the rate of 8.5Hz. When the camera finishes acquiring image data, it will send a signal to call a callback function that runs the image processing algorithm. The result from the algorithm is the control data which will be send back to dSPACE computer when the next UDP packet is arrived. The time line sequence of events that happen on the image processing module can be seen in Fig. 4.10. A dummy packet is sent out to maintain the time between the point that the packet from dSPACE is arriving and the image ready callback signal.

Chapter 5

Optical Flow Determination

To study the navigation of the insect, the optical flow computation from the image sequence is needed. In this chapter, the optical flow problem definition will be discussed first then the survey of algorithm to solve this problem will be discussed. After the selection of the algorithm, the implementation and interpretation will be discussed.

5.1 Problem definition

Optical flow problem is to track a set of points in an image in the former time which is present in the image at current time. The movement of the pixel in an image can result from the movement of the object in the camera field of view or from the movement of camera itself. Figure 5.1 shows the tracking of some points between two images. The works in this thesis is mainly looking at the surface of the planet. It can be assumed that the surface is flat compared to the camera field of view and height variation. The optical flow calculated will be used for controlling the motion of the spacecraft.

5.2 Algorithm selection

Many methods are present in the scientific community for solving the optical flow problem. There are different classes of techniques commonly used which are differential methods, region-based matching methods, energy-based methods and phase-based methods (Barron et al., 1994).

The differential method defines the optical flow problem as a differential equation from the assumption of constant image intensity. To solve the equation, another constraint is needed besides the constant intensity ($dI/dt = 0$) assumption. Horn and Schunck (1981)

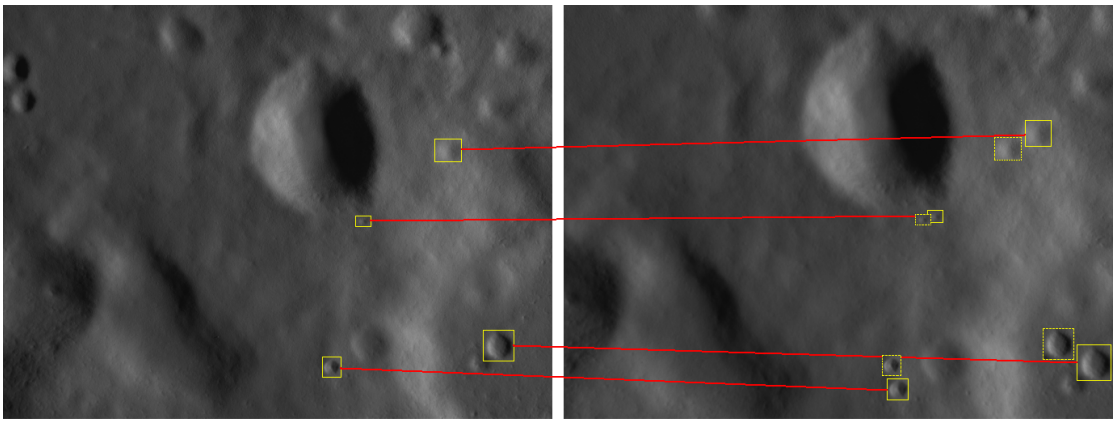


FIGURE 5.1: Tracking of some points shows the definition of optical flow

used a global smoothing factor as the constraint while Lucas and Kanade (1981) used the local constant intensity as the constraint. There are many other authors trying to solve the equation by different method but the most sufficient differential method in time and performance is the Lucas and Kanade (1981) algorithm (Barron et al., 1994).

According to Barron et al. (1994), the region-based matching and energy-based method give not good result and are not very robust. The phase-based method gives good results but need a lot of computation effort (Zhao and Spetsakis, 2001). So these sets of algorithm have been ruled out.

In this thesis, the goal of calculating the optical flow is to be able to navigate and control the spacecraft. There are some methods that can determine the egomotion of the camera without calculating the optical flow, which is the image interpolation technique (Nagle et al., 1997, Srinivasan, 1994). However, this method needs the six convolutions of the image to generate the reference images which is expensive in the calculation or needs additional hardware which is not suitable for the hardware-in-loop simulation laboratory.

From the literature research, the Lucas-Kanade algorithm has been chosen for implementation and testing the idea of an insect-inspired landing algorithm. However the disadvantage of the differential method is that it cannot handle quick movement because of the assumption used. One method for overcome this problem is to apply the pyramid down operation to the image and apply the Lucas-Kanade algorithm (Bouguet, 2000). The result from this method can handle the large displacement of motion.

5.3 Lucas-Kanade algorithm

The Lukas-Kanade algorithm is a differential method for the determination of the optical flow. The differential method uses an assumption that the image intensity at a pixel is a spatial movement of the intensity of a pixel in a former image. Assume $I(x, y, t)$ is the image intensity at the pixel (x, y) at time t and it moves to pixel $(x + \delta x, y + \delta y)$ in the image at time $t + \delta t$, the image intensity is still constant which can be written as

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t). \tag{5.1}$$

Equation 5.1 can be expanded using Taylor series and assumed that the non-linear parts are zeros.

Lukas-Kanade algorithm uses another assumption that the local intensity around the interesting image position, (x, y), is also constant. This assumption gives a number of differential equations which can be written as

$$I(x + i, y + j, t) = I(x + i + \delta x, y + j + \delta y, t + \delta t), \tag{5.2}$$

$$i, j \in (-w..w, -w..w) \tag{5.3}$$

where w is the considered window size of the local intensity constraint. With over constraint equations, the optical flow $(\delta x, \delta y)$ can be solved by the least-square technique. In solving these equations, the weight is applied to give the importance to the pixel near the centre of constraint window more than the pixel that is far from the centre of constraint window.

More detail on solving the weighted least-square condition on these differential equations can be found in Lucas and Kanade (1981).

5.3.1 Pyramidal implementation

Because the Lucas-Kanade algorithm has limited ability in detection of large displacement of the optical flow, the pyramidal implementation by Bouguet (2000) has been introduced to solve this problem. The original image is filtered by the pyramid down operation which is inexpensive in the calculation effort. The pyramid image will be small down in size but still contains the big picture of the whole image. The illustration of the pyramid down operation is shown in Fig. 5.2.

The pyramidal of Lukas-Kanade algorithm is performing on the high level of pyramid image first as it will determine the coarse position of the feature. Then it uses the position

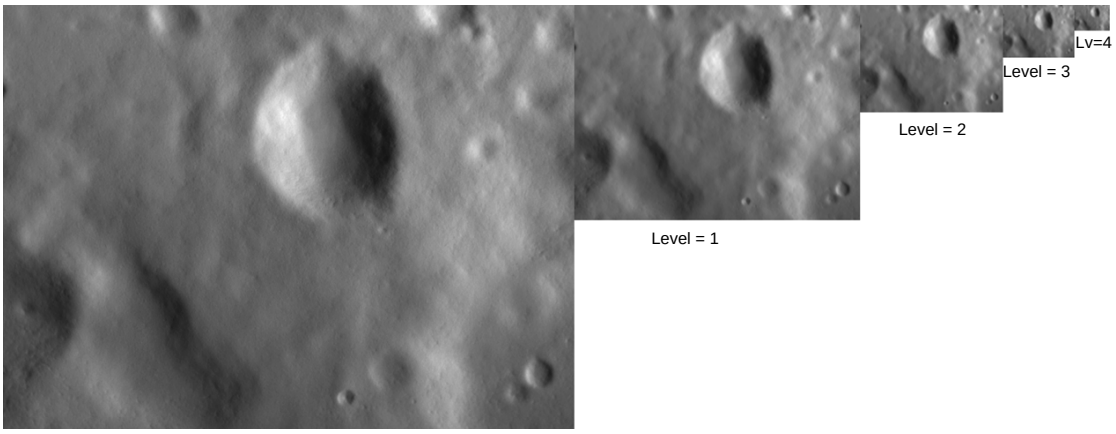


FIGURE 5.2: Pyramid down operation of a image.

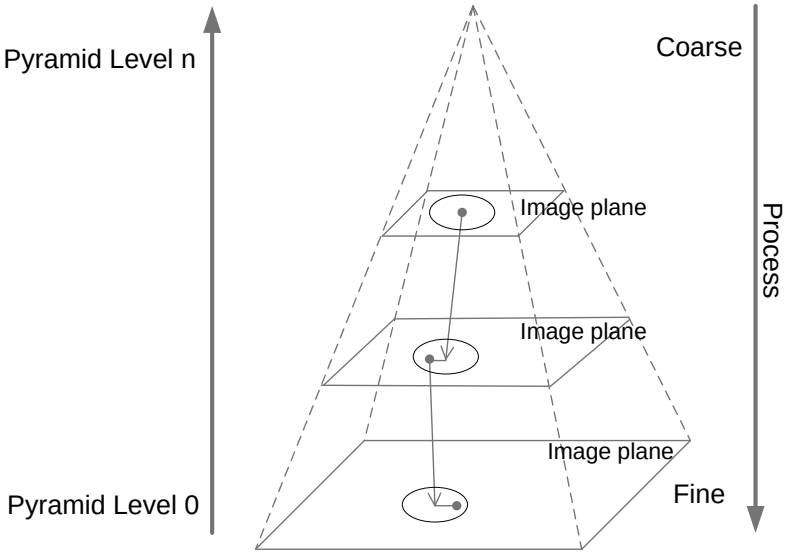


FIGURE 5.3: The summarized operation of the pyramidal of Lukas-Kanade algorithm.

in the higher pyramidal image for the initial condition of the least-square calculation for the next lower pyramid level image. This operation gives the result in ability to detect the large movement from the high level pyramid image but still maintains the accuracy in the low level pyramid image. The pyramidal of Lukas-Kanade algorithm can be summarized in Fig. 5.3.

5.4 Implementation

The important steps in the optical flow implementation consist of acquisition of the image which has a suitable characteristic with the algorithm, the algorithm implementation itself and the process related to the use of algorithm which in this case is the feature selection process.

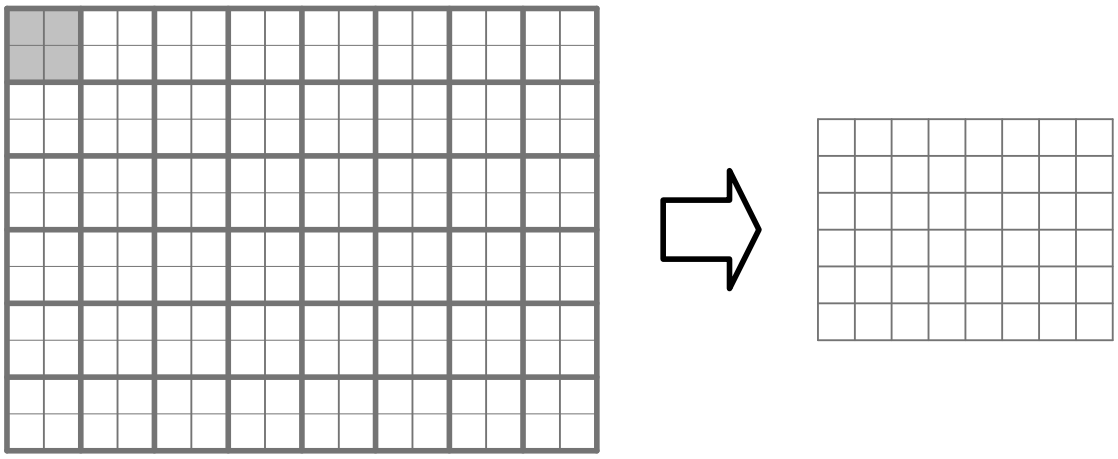


FIGURE 5.4: Image binning in the acquisition process.

5.4.1 Camera acquisition

According to the characteristic of Lucas-Kanade algorithm that handled the flow velocity in the order of pixel and limited the maximum detectable velocity as the level of pyramid image, therefore the acquired image should not be too large in size. The size of image that still presents the local feature is enough. Also, a smaller image size results in better calculation time. In the project presented in this thesis, the camera chosen is Procillica GC1350 which supports pixel binning. In this thesis, 2x2 pixel binning is selected resulted in reducing the acquired image in size from the full resolution of 1360x1024 pixels to 680x512 pixels as seen in Fig. 5.4. The FOV of the camera is preferred to be maximum (see the discussion in Section 6.2), so the ROI of the camera is set to 100% of the FOV. The trigger mechanism to acquire image is set to fixed rate trigger mode, so the time between the acquired images is constant and can be used for state prediction in the future.

5.4.2 Lucas-Kanade optical flow implementation

The selected algorithm has been already implemented by OpenCV[1] with intensive optimization. The implementation constructs the pyramid image and applies the Lucas-Kanade optical flow algorithm to each level by using the initial guess in the iterative process from the higher pyramid image. The detail implementation can be found in Bouguet (2000).

[1] http://opencv.willowgarage.com

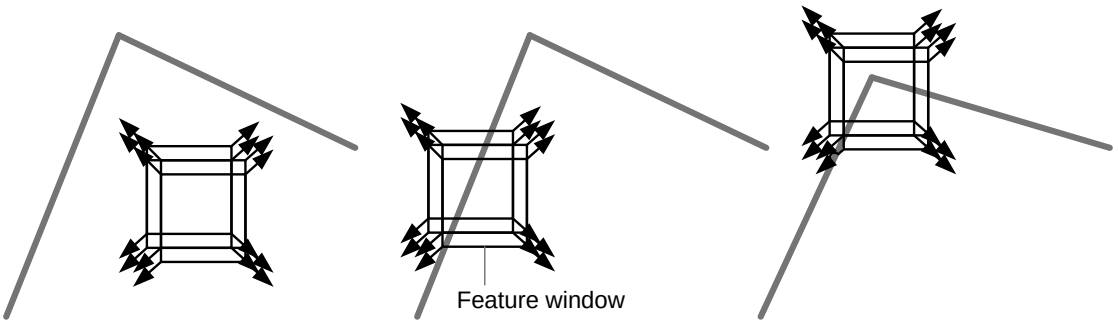


FIGURE 5.5: The illustration of the "corner" feature.

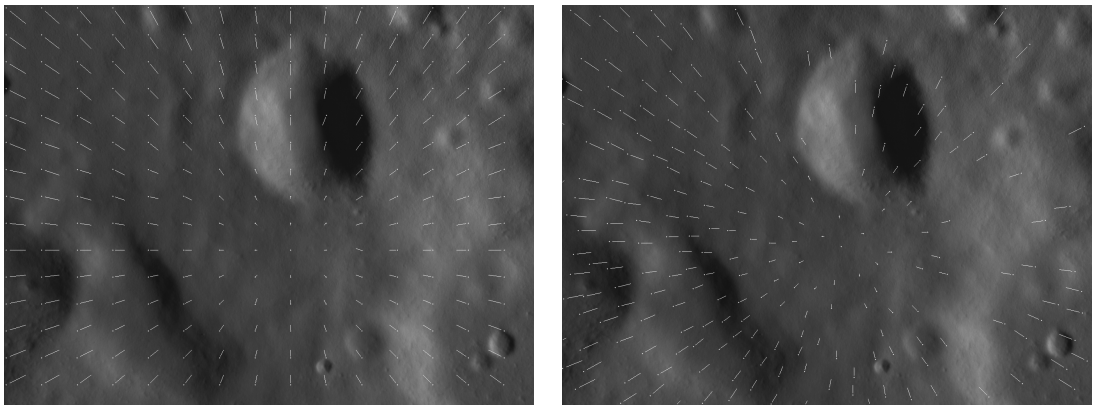
5.4.3 Feature selection

The optical flow is a feature tracking problem which needs some pattern to be able to recognise the flow. For example, if the camera is looking at the completely black or white region, it will never know the movement of the image because the image always looks the same in every direction it moves. Similarly to the line only image, if the window moves along the line, the movement along the line cannot be detected. The good feature that can be detected is the region that has a "corner". The "corner" in this case means the characteristic of the feature that can be easily detected when the window is moving as seen in Fig. 5.5. It can be a sharp edge corner, a spot or even a circle.

One possibility to select the appropriate feature is the Harris corner detection or the use of Eigen value of the pixel and neighbourhood (Harris and Stephens, 1988, Shi and Tomasi, 1994). However, from the image dataset of the moon or asteroid surface, the features are spread all over the image. In this thesis, the total optical flow over the whole image is the interested part not the specific feature. Moreover, from the experiment with image dataset, the result indicated that it does not need the feature detection process. The features can be randomly picked up and most of the features can be tracked except the one in shadow. Without the need to pick up the feature, the overall calculation time is decreased dramatically (more than 20 ms per image on Intel Core 2 Quad 2.83 GHz computer). The comparison between the optical flows of the same image determined by feature selection algorithm and spread pattern feature selection can be found in Fig. 5.6.

5.5 Interpretation

The calculated optical flow is in the form of local image velocity at the image pixels. These values cannot be used directly, they need to be interpreted to another meaningful data. In this thesis, the overall flow indication number which is used for spacecraft



(a) The selection of feature using pattern spots.

(b) The selection of feature using corner detection algorithm.

FIGURE 5.6: Comparison of the result flow from different feature selection methods.

velocity control is extracted. The Focus Of Expansion (FOE) is also another interested feature to be discussed. The characteristic of the flow is also considered for the attitude control.

5.5.1 Flow number calculation

A very simple way used here to calculate how much optical flow in the image is to average over the magnitude of the velocity vectors in each direction. From now on, define this value as average optical flow number, ω_{avg}. Figure 5.7 shows the average flow in each direction. Because the change of the trajectory cannot be very high, the averaged optical flow number can be used to effectively compare the amount of the flow in two consecutive frames. The average optical flow number can be calculated by

$$\omega_{avg} = \frac{\omega_{+x} + \omega_{-x} + \omega_{+y} + \omega_{-y}}{n} \tag{5.4}$$

where the average of the flow in each direction can be calculated by

$$\begin{aligned} \omega_{+x} &= \frac{\sum |v_{+x}|}{N_{+x}} \\ \omega_{-x} &= \frac{\sum |v_{-x}|}{N_{-x}} \\ \omega_{+y} &= \frac{\sum |v_{+y}|}{N_{+y}} \\ \omega_{-y} &= \frac{\sum |v_{-y}|}{N_{-y}} \end{aligned} \tag{5.5}$$

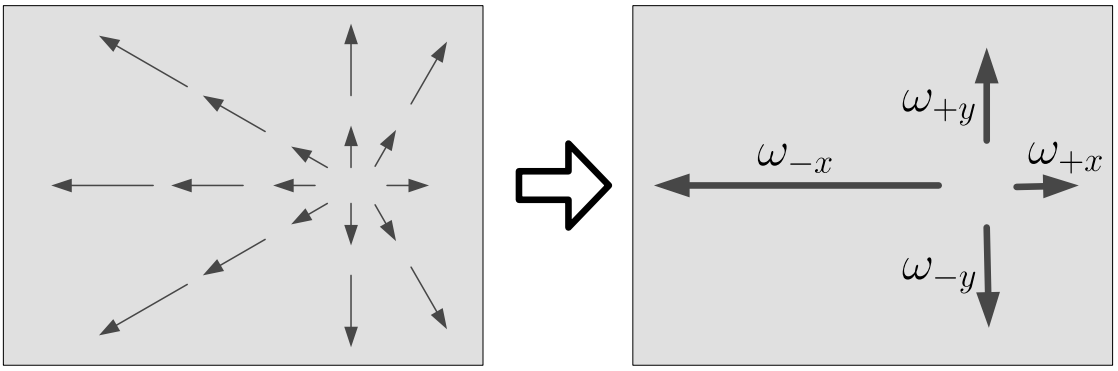


FIGURE 5.7: The average flow number calculation in each direction

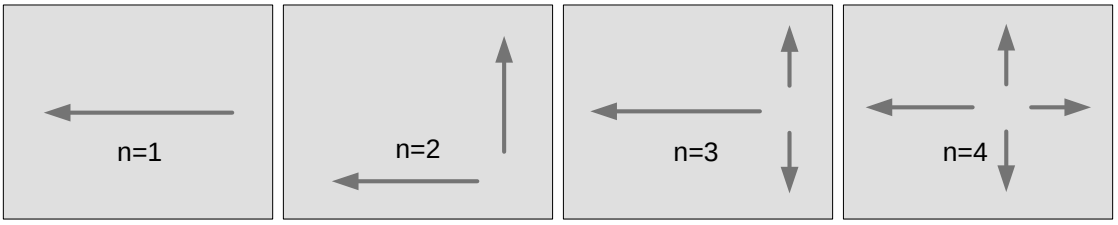


FIGURE 5.8: The illustration of value n in the Eq. 5.4

where v_x and v_y are the local image velocities of each feature, $(N_{+x}, N_{-x}, N_{+y}, N_{-y})$ are the number of vector in each direction and n is the number of flow direction that has non-zero value. Figure 5.8 shows the meaning of value n.

5.5.2 Focus Of Expansion calculation

The Focus Of Expansion (FOE) is an interesting feature of the optical flow, it can infer the movement of the camera. There are many researches in the determination of FOE. Some methods are the direct method which can determine the FOE directly from the image without calculating the optical flow (Negahdaripour and Horn, 1989). Because the former calculation has already determined the optical flow, the simple method can be used by calculate the average intersection point of all velocity vectors in the image. Figure 5.9 shows the calculation of the FOE. This method is not the best way to determine the FOE of the video sequence because it will give jumping position of FOE over time because of nominal noise in the optical flow vectors. However, the determination of the FOE is not the focus in this thesis work. The better method can be investigated and used in the future.

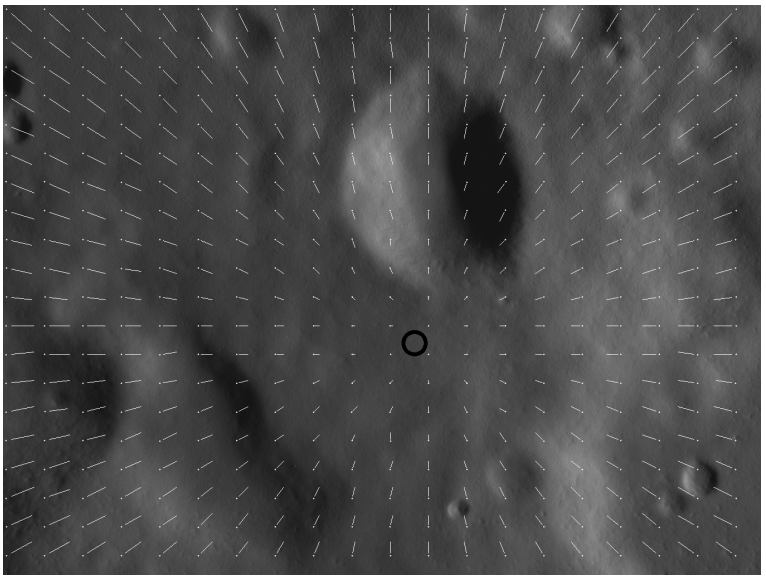
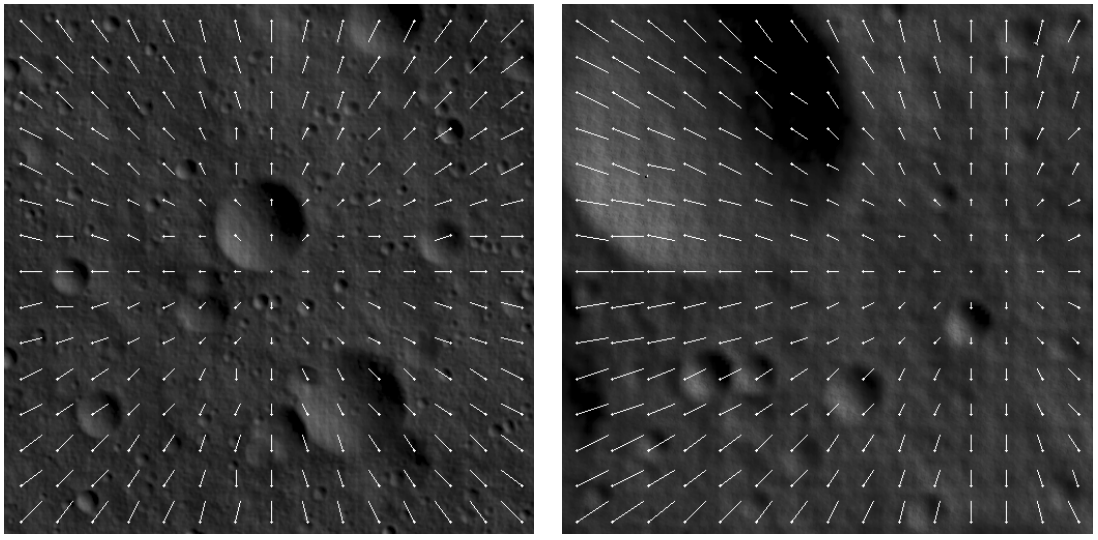


FIGURE 5.9: Calculation of Focus Of Expansion (FOE), the FOE is marked with circle.



(a) Balance flow (b) Shifted flow

FIGURE 5.10: The optical flow pattern of the balance and shifted flow.

5.5.3 Balance and shifted flow

When calculating the average flow number in the way mentioned in Section 5.5.1, the average flow number in each direction $(\omega_{+x}, \omega_{-x}, \omega_{+y}, \omega_{-y})$ can also determine the characteristic of the flow. The flow that is shifted will have the value of flow in some direction bigger that the other. An example for the different optical flow pattern can be seen in Fig. 5.10.

It also can be seen from Fig. 5.10 that the optical flow which is shifted to some side

will give the number of optical flow vector in opposite direction significantly different from each other. So, the number of vectors in each direction, $(N_{+x}, N_{-x}, N_{+y}, N_{-y})$, can be used as the rough indicator about the shifted flow. The balance flow image has the number of vector in each direction roughly equal to each other. The shifted flow image has a different in number of vectors in each direction. The number of vector counting can be used for roughly control the spacecraft attitude to give the desired number of vectors in each direction instead of controlling the FOE.

Chapter 6

Spacecraft Control Using Optical Flow

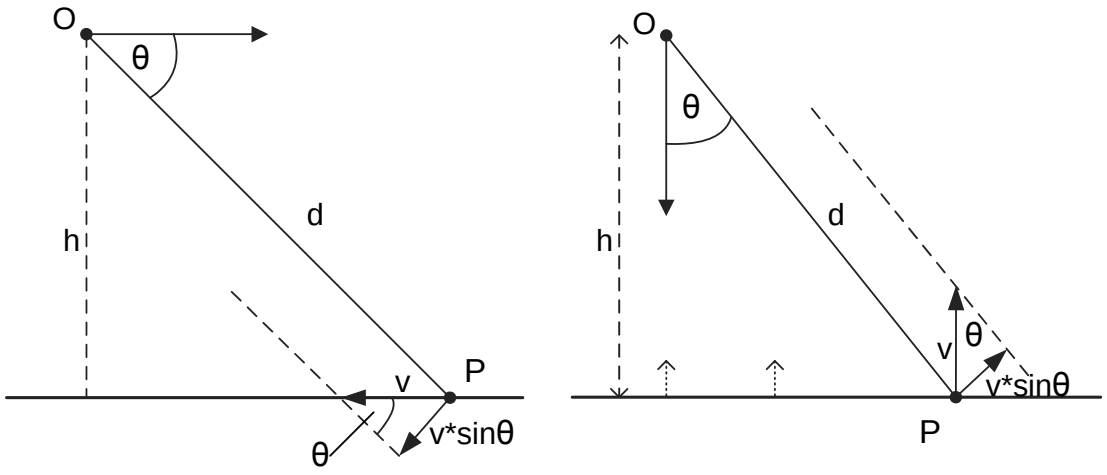
To control the spacecraft by using the optical flow, the characteristic of the flow according to the motion has to be studied. The attempt to determine the camera motion from the optical flow will be discussed. Some relations between optical flow and camera motion which are used in the control strategy for the control of spacecraft for landing will be presented. Mathematical background of the control strategy which is derived from the optical flow characteristic relation to the motion will be discussed. Finally, the control law designed for the landing of the spacecraft using the optical flow from the image sequence will be described.

6.1 Optic flow characteristic of the different motion

When a camera is moving in different manoeuvres, the optical flow produced from each motion is different. To be able to control the spacecraft according to the optical flow, the characteristics of the flow from the motion are studied. The 1-dimension motion parallax is studied resulting in the mathematical relation between the camera motion and the optical flow. The position of FOE related to the motion is also studied, resulting in attitude and moving direction determination from the optical flow.

6.1.1 1-dimension motion parallax

When the camera is moving, the different points in the Field Of View (FOV) will move differently to each other depending on their angle and distance to the camera. The



(a) The camera moves parallel to the ground. (b) The camera moves forward to the ground.

FIGURE 6.1: The parallax motion of a point in FOV when the camera moving differently.

far object will seem to move slower than the object near the camera. This is called, motion parallax. Whiteside and Samuel (1970) talked about the parallax motion of the moving aircraft and derived an equation which can be adapted to describe the optical flow characteristic. Continuing from Whiteside and Samuel (1970) work, the optical flow equations can be derived.

The optical flow, ω, of a point on the ground, P, seen by a camera, O, moving parallel to the ground with the velocity of v as in Fig. 6.1(a) can be described by

$$\omega = \frac{v \cdot sin\theta}{d} \tag{6.1}$$

where θ is the angle that the position of point P in the image makes to the centre of the FOV and d is the distance from the camera to point P. Assuming that the velocity vector is a part of a big circle, the unit of optical flow, ω, is in radian per second. The relation of height, h, and the distance, d, is

$$h = d \cdot sin\theta,\ d = \frac{h}{sin\theta}. \tag{6.2}$$

Then substitute Eq. 6.2 into Eq. 6.1 produces the relation between the optical flow and the height which is

$$\omega = \frac{v}{h} sin^2\theta. \tag{6.3}$$

For the direct movement down towards the ground (Fig. 6.1(b)), it is the same situation as the ground is moving up towards the camera. The parallax motion is still the same as described in Eq. 6.1 but the relation between the height and the distance to the point

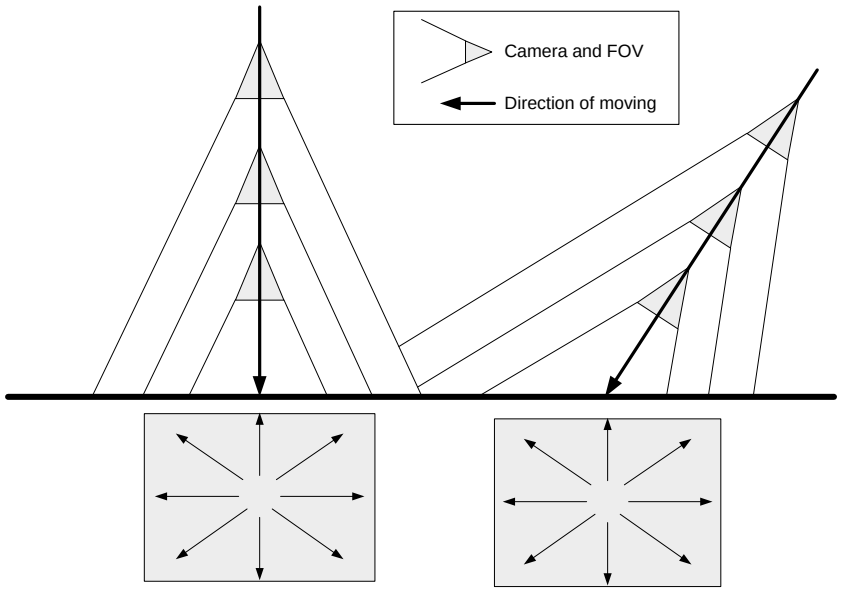


FIGURE 6.2: The camera is moving and looking in the same direction, the FOE is in the centre of the FOV.

is changed to

$$h = d \cdot cos\theta,\ d = \frac{h}{cos\theta} \tag{6.4}$$

and gave the relation of optical flow and height as

$$\omega = \frac{v}{h} sin\theta cos\theta. \tag{6.5}$$

It can be seen that the optical flow have a certain relation to the ratio of moving velocity and the height regardless the moving direction. The equation can explain the situation when the camera is moving near to the ground that the optical flow is very high compare to when it is moving at the same velocity but far above the ground. If the camera is moving in a straight line, the factor apart from $\frac{v}{h}$ will be just a constant and can be ignored when using optical flow to control camera velocity.

6.1.2 Looking in the moving direction

When the camera is looking in the same direction as the moving direction, FOE will be always in the middle of FOV. This characteristic does not depend on the angle that the direction of movement made to the surface. The illustration of this situation can be seen in Fig. 6.2. It can be seen that the points in the image is moving out from the centre of the FOV.

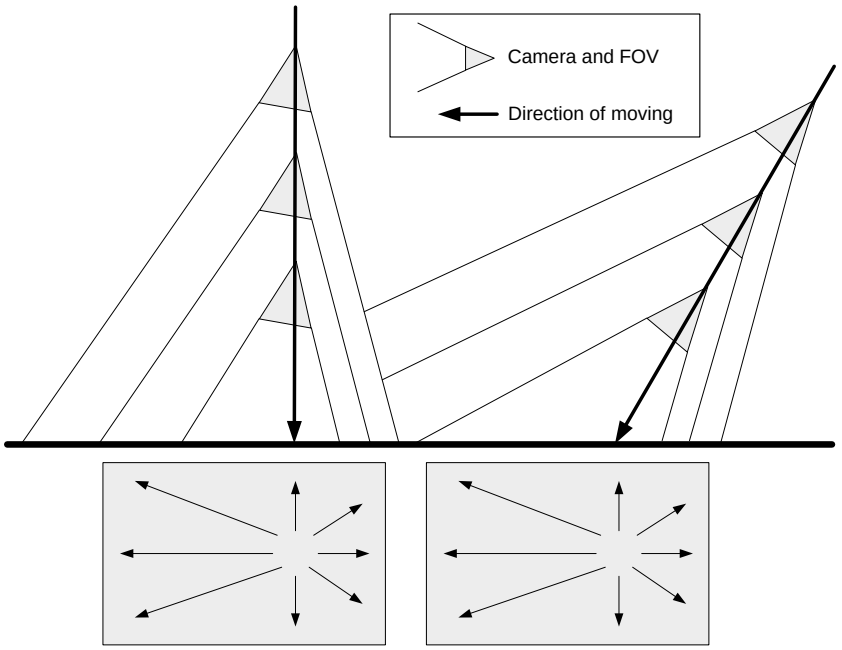


FIGURE 6.3: The camera is moving and looking in different direction, the FOE is stay at the same place inside the FOV.

6.1.3 Looking in the different direction from the moving direction

When the camera is moving always in straight line but looking in the different direction constantly from the moving direction, the FOE will deviate from the camera centre point equal to the angle between the direction of moving and the direction of looking. The illustration of this situation can be seen in Fig. 6.3. The FOE also can be outside the FOV as in Fig. 6.4 and the statement mentioned above is still true. Furthermore, the FOE will stay at the same place in the image if the camera is moving in straight line but the FOE position will be moving if the camera is moving in curve. This knowledge can be used for correction of the moving direction when knowing the target point to land. This can be done by looking at the target point and try to move the FOE to be at the middle of the field of view by adjusting the moving direction. Also it can be used to control the trajectory of the movement to follow a straight line by trying to control the FOE to be at the same position in the image.

6.2 Problem of egomotion determination

One way to be able to control the spacecraft is to determine self motion from the camera image sequence. There are many researches on using optical flow to determine the egomotion of the camera. Negahdaripour (1987) has solved the egomotion equation in closed form of the camera looking at a planar surface. Futhermore, Adiv (1989) has

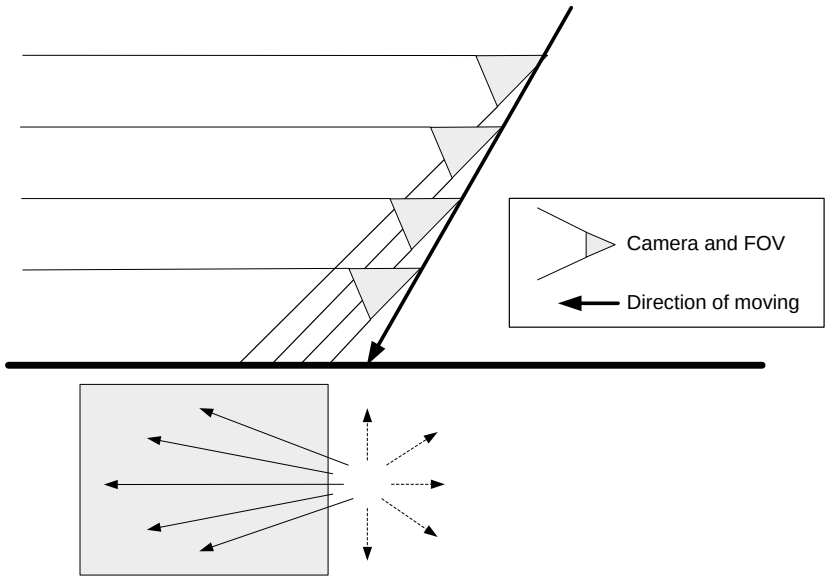


FIGURE 6.4: The camera is moving and looking in different direction, the FOE is stay at the same place outside the FOV.

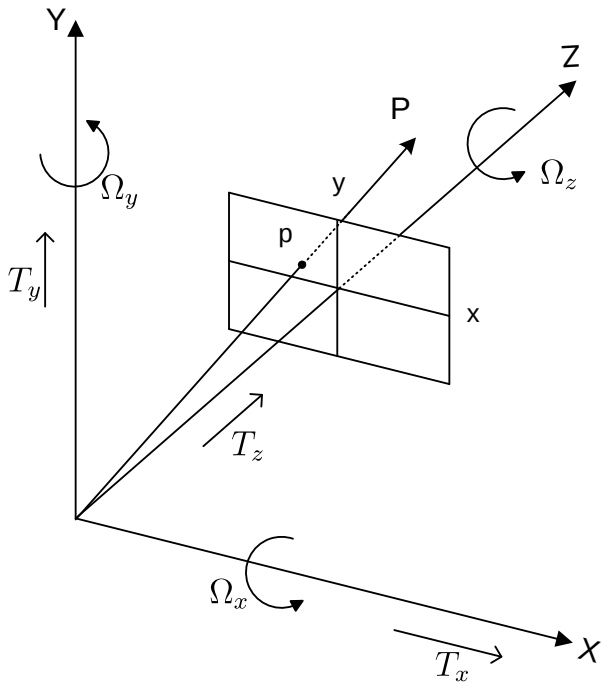


FIGURE 6.5: A coordinate system shows the image position of point P and the notation used in Eq. 6.6(Adiv, 1989)

described the optical flow from the motion in the mathematical form and study the ambiguities in retrieving the egomotion from noisy optical flow. The equation related to the egomotion determination described by Adiv (1989) when (X, Y, Z) are the position of point P in the camera coordinate (Fig. 6.5) and the projection of point P on the image plane is at (x, y) with the image velocity of (α, β) are

$$\alpha_R = -\Omega_x xy + \Omega_y(1 + x^2) - \Omega_z y, \tag{6.6a}$$

$$\beta_R = -\Omega_x(1 + y^2) + \Omega_y xy + \Omega_z x, \tag{6.6b}$$

$$\alpha_T = (T_x - T_z x)/Z, \tag{6.6c}$$

$$\beta_T = (T_y - T_z y)/Z \tag{6.6d}$$

where (α_R, β_R) is the rotational component and (α_T, β_T) is the translational component of the image velocity. It can be seen that knowledge of the image velocity field can be used for solving the movement of the camera. However, the mathematical study by Adiv (1989) shows that in some situations, completely different movements of the camera can produce very similar optical flows. This means, given a noisy optical flow field, a wrong camera movement may be recovered from the flow instead of the correct one. The situations that cause the ambiguity included: the FOV of the camera is small, the absolute translation is small, the object is far away, etc.

In current robotic research, there is a set of algorithm called SLAM (Simultaneous Localization And Mapping). Some of them uses only one video camera and can determine egomotion in the same time as the structure of the environment (Davison et al., 2007, Eade and Drummond, 2006). But the method is also limited by the situations that provide the ambiguity described above. Furthermore, it is believed that the insect also does not consider it height from ground or time to contact to the ground when it is landing (Chahl et al., 2004, Thakoor et al., 2005). Due to the outcome of the research, it was decided to use a simple approach of the optical flow control without the determination of the egomotion. However, this is interesting subject to be studied in the continuing work.

6.3 Control strategy

The control strategy in this thesis considers first the trajectory of the landing. The trajectory is similar to the insect landing trajectory. The control of velocity is also the main focus in this topic, the velocity at landing needs to be slow enough in every direction, so the spacecraft is not crashing to the ground. During the approach to the landing site, sometimes it is impossible to move to it directly in straight line. The

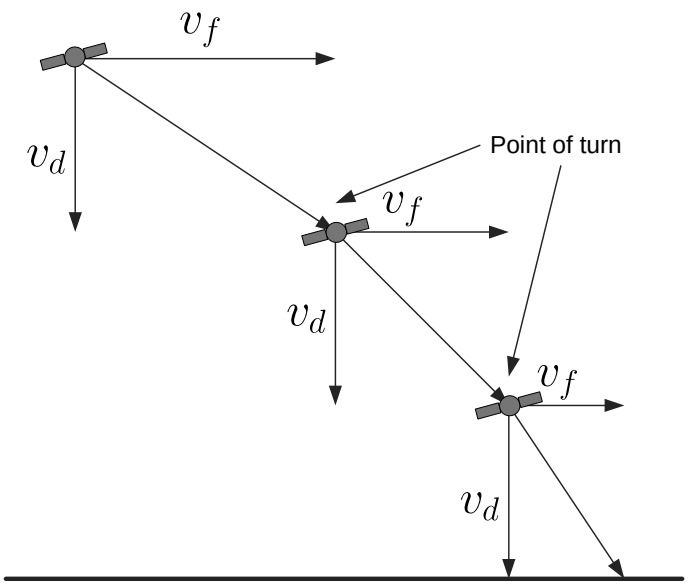


FIGURE 6.6: The velocity ratio for descent angle control and changes of the descent angle.

attitude and the direction of moving also need to be controlled. This is possible using the optical flow information from the image sequence. The last important point in the control strategy is the delay in the control system. The simple strategy to overcome the delay will be discussed.

6.3.1 Trajectory selection

Trajectory chosen here is a linear motion direct to the ground to maintain similar flow characteristic during the entire landing. A linear movement will produce the optical flow with FOE at the same place during the approach to the ground. This results in the average flow number calculated in Eq. 5.4 is represent the comparable value to each other during the entire landing. The forward velocity and the falling down velocity is decrease with the same ratio. The descent angle can be selected by adjusting the ratio between forward velocity and falling down velocity as can be seen in Fig. 6.6. If a necessary change of direction of movement is needed, the trajectory will be treated as linear motion with rotation at the point of turn as in Fig. 6.6.

6.3.2 Velocity control

The desired velocity at touch down should be nearly zero in all directions. Similar to the insect, the control law here is to control the optical flow to be constant. This results in the spacecraft velocity to be fast when the spacecraft is far away from the surface then

linearly slower when it moved nearer to the surface and getting close to zero at touch down. The optical flow will be controlled to be at a desired value, ω_{aim} via the control of spacecraft velocity.

6.3.2.1 Constant optical flow

Assuming that the landing trajectory approaches the surface in a straight line or the change of the descent angle is small enough. The average flow number, ω_{avg}, calculated using Eq. 5.4 of two consecutive frames can be simplified from the parallax motion equation as

$$\omega_{avg} = c \cdot \frac{v}{h} \tag{6.7}$$

where c is a constant combine average factor and angle deviation factor (e.g. $sin\theta cos\theta$ in Eq. 6.5) together. The altitude can be estimated from the initial guess velocity, v_{guess}, and calculated average optical flow number as

$$h = c \cdot \frac{v_{guess}}{\omega_{avg}}. \tag{6.8}$$

A new control velocity, $v_{control}$, that will control the optical flow to be equal to the desired flow, ω_{aim}, can be calculated by

$$\begin{aligned} \omega_{aim} &= c \cdot \frac{v_{control}}{h} \\ v_{control} &= \frac{h \cdot \omega_{aim}}{c} \end{aligned} \tag{6.9}$$

and using the same altitude which is calculated by Eq. 6.8 given $v_{control}$ as

$$v_{control} = \frac{c \cdot v_{guess} \cdot \omega_{aim}}{c \cdot \omega_{avg}} \tag{6.10}$$

which finally gives the control rule for the control velocity as

$$v_{control} = v_{guess} \cdot \frac{\omega_{aim}}{\omega_{avg}}. \tag{6.11}$$

If there is no clue about the real velocity from another sensor, the control velocity is used for being the guessed velocity in the next calculation loop. The velocity that is used to control the spacecraft calculated by Eq. 6.11 always corrects the optical flow even the guess velocity is not the realistic value but in the same magnitude. If the guess velocity is very wrong then the correction will give the wrong value of optical flow which can be detected and can be used to adjust the guess velocity in the next control loop.

Equation 6.11 derived here is very similar to the equation using by Chahl et al. (2004) which studied the landing behaviour of the honeybees. In Chahl et al. (2004), the velocity in the equation is the forward velocity but in this thesis, the equation is applied to the true velocity on the assumption stated before.

To control the optical flow to be constant, the desired velocity is very high at high altitude and very low near the ground. This needs both acceleration and deceleration of the spacecraft velocity to be equal to the desired velocity all the time. Because the landing trajectory is normally only to decelerate the spacecraft velocity from the orbit velocity to be able to land safely, there is no actuator for the acceleration of the velocity. The control strategy here is modified to just decelerate the spacecraft velocity. Even though the optical flow is too low, the spacecraft will not be accelerated to increase the optical flow. The spacecraft velocity will be untouched until the spacecraft is close to the ground enough to increase the optical flow to the desired level. Then the deceleration is activated again.

6.3.2.2 Desired flow number selection

The selection of ω_{aim} depends on starting velocity, altitude and fuel consideration. The value of ω_{aim} controls the slowing down phase to be quick or slow. Figure 6.7 shows the perfect control of the velocity according to the altitude of various ω_{aim} values when the spacecraft moved directly towards the ground. More complex trajectories also can be designed by changing ω_{aim} value during the landing.

In the case of change of the direction of moving, the same desired image velocity (ω_{aim}) will give slightly different velocity control as the angle factor is different in the parallax motion study (Eq. 6.1). In this thesis, this difference is neglected. Further mathematical study can be investigated to perform better control of the spacecraft.

6.3.3 Attitude control

The attitude of the spacecraft here is assumed to be in the direction that the camera is looking to which may not be the same as the direction that the spacecraft is moving. The FOE is used for controlling the spacecraft attitude and the direction of moving to move forward to the landing site. The control of the attitude requires rotating the camera. Therefore the compensation of the optical flow from the rotation is needed and will be discussed here. Due to the limitation in time, the theoretical studied here will not be implemented and presented in the result section.

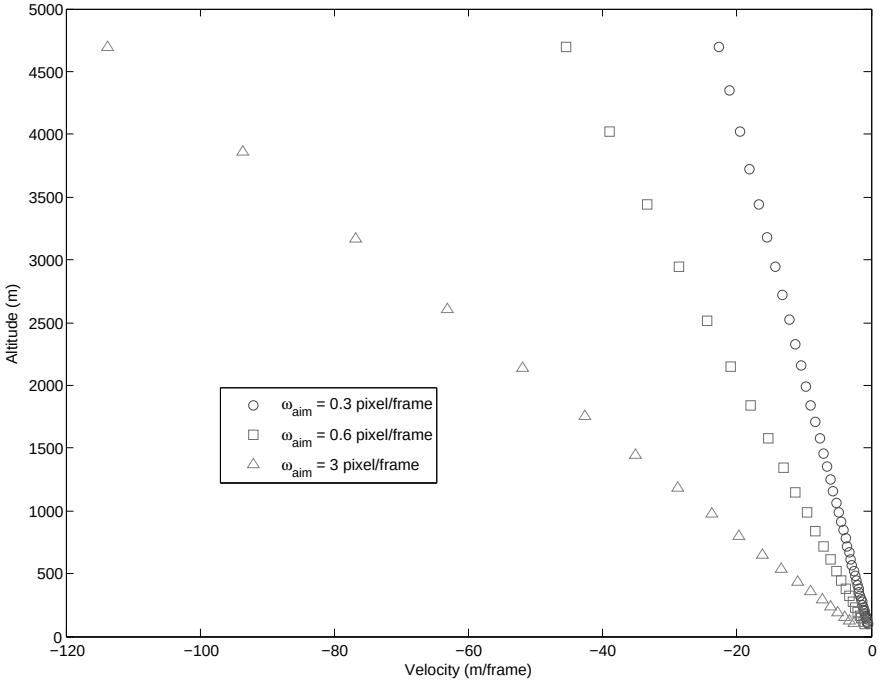


FIGURE 6.7: A plot shows the relation between velocity and altitude when choosing different ω_{aim}

6.3.3.1 Control of the direction of moving

As described in Section 6.1 that the FOE represents the direction of moving. If the certain landing site is known, then to control the spacecraft to move to that landing site is to control the FOE to be at the same spot as the desired site. If the attitude of the spacecraft is holding which means that the camera is always looking in the same direction, then the direction of moving can be controlled to by control the FOE spot to make the certain angle to the centre of the image.

A simple example for the attitude control during landing is when the camera has the desired landing site in its FOV. The spacecraft attitude is always controlled to track the landing site in the middle of FOV. To control the spacecraft to move toward the landing site is to control the FOE to be always in the centre of FOV. This strategy can be applied in similar way to another trajectory design to control the attitude and direction of moving of the spacecraft.

The angle between the FOE and the centre of image is equal to the angle that the spacecraft needs to adjust it direction of moving to make the FOE to be at the centre of the FOV. The direction of moving is changed by adjusting the forward velocity and downward velocity ratio as in Fig. 6.6 where v_d is the downward velocity and v_f is the

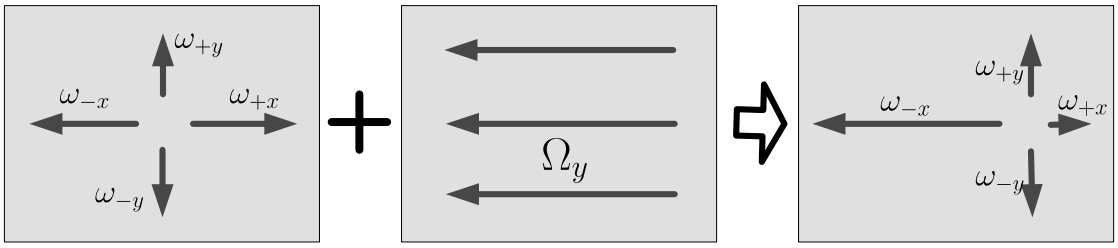


FIGURE 6.8: The rotation affects the average flow number, the compensation is needed.

forward velocity. In the control loop, the direct velocity measurement is not available. The feedback signal for controlling the direction of moving is the position of the FOE.

6.3.3.2 Rotation compensation of the flow

Because the optical flow is changed significantly according to the rotation of the camera, the Eq. 6.11 is not valid anymore. To use the equation, the rotation measurement of the camera is needed. In this thesis, it is assumed that a small IMU is installed on the spacecraft and be able to measure the rotation of the camera. The rotation of the camera gives the optical flow value equal to the angle of rotation regardless of the distance to the object as in Eq. 6.6. The calculation of the average flow number is deviated from a non-rotation movement as shown in Fig. 6.8.

According to Eq. 6.6, the rotation contributes to the optical flow regardless of the distance to target. The rotation known by the means of inertial measurement can be transformed into the rotation component of the optical flow by Eq. 6.6 and then be subtracted it from the measured optical flow. From the average flow number calculated in Eq. 5.4, the average rotation component can be subtracted out from the average flow number. Given $(-a, a)$ and $(-b, b)$ are the boundary of the image plane in the camera coordinate system, the average rotation component can be calculated by

$$\alpha_{Ravg} = \frac{\int_{-b}^{+b}\int_{-a}^{+a}\alpha_R dxdy}{(2a)(2b)} \tag{6.12}$$

where the integration over the image plane is

$$\begin{aligned}\int_{-b}^{+b}\int_{-a}^{+a}\alpha_R dxdy &= \int_{-b}^{+b}\int_{-a}^{+a}(-\Omega_x xy + \Omega_y(1+x^2) - \Omega_z y)dxdy \\ &= (2a + \frac{2a^3}{3})(2b).\end{aligned} \tag{6.13}$$

Finally, the average flow number caused by the rotation is

$$\begin{aligned} \alpha_{Ravg} &= \Omega_y \frac{(2a + \frac{2a^3}{3})(2b)}{(2a)(2b)} \\ &= \Omega_y (1 + \frac{a^2}{3}). \end{aligned} \tag{6.14}$$

Similar to the x direction, the average flow number contributed by the rotation in y direction is

$$\beta_{Ravg} = \frac{\int_{-b}^{+b} \int_{-a}^{+a} \beta_R dx dy}{(2a)(2b)} \tag{6.15}$$

where the integration over the image is

$$\int_{-b}^{+b} \int_{-a}^{+a} \beta_R dx dy = \int_{-b}^{+b} \int_{-a}^{+a} (-\Omega_x (1 + y^2) + \Omega_y xy + \Omega_z x) dx dy. \tag{6.16}$$

The final result for the rotation component in y direction is

$$\begin{aligned} \beta_{Ravg} &= \Omega_x \frac{(2b + \frac{2b^3}{3})(2a)}{(2a)(2b)} \\ &= \Omega_x (1 + \frac{b^2}{3}). \end{aligned} \tag{6.17}$$

To compensate the rotation out from the average flow number, the calculation of the flow in each direction in Eq. 5.5 are modified to

$$\begin{aligned} \omega_{+x} &= \omega_{+x} - \alpha_{Ravg} \\ \omega_{-x} &= \omega_{-x} + \alpha_{Ravg} \\ \omega_{+y} &= \omega_{+y} - \beta_{Ravg} \\ \omega_{-y} &= \omega_{-y} + \beta_{Ravg}. \end{aligned} \tag{6.18}$$

The new value of flow in each direction is then used for calculating the average optical flow value in Eq. 5.4 and can be used for control rule in Eq. 6.11.

6.3.4 System delay issue

The delay in the control system is the time that the system needs from sensing to a successful response to the command. In reality, the delay can be quite high due to the delayed response of the actuators, the data transmission delay and the calculation time. In this thesis, the delay is the time from the sensing of optical flow from the camera image until time that the robot changes its movement according to the optical flow.

If the delay is high, this means that the actuator is slow to respond to the command. A good example of this situation is when the spacecraft performs an approach to the ground and tries to make the optical flow to be constant. When it is nearer to the surface, the optical flow is increase and to make it constant, the spacecraft have to be slowed down. In this phase, the command to spacecraft tries to slow down the spacecraft velocity but the response is delayed and the optical flow is not decreased as the command has not affected the spacecraft velocity yet. The controller thinks that the command is not enough to slow down the velocity, it tries to decrease more and more velocity. Then at the time that the command affects the spacecraft velocity, the magnitude of decreasing in velocity is over the required one, resulting in the velocity becoming too low. At this point the optical flow will be also too low and the spacecraft needs to speed up to maintain the flow to be constant. The same happens to the speed up phase, the speed up will be too much and later on the system will be oscillating between the slowing down phase and the speeding up phase.

There are many parts of the system that can produce the delay in the control loop. Starting from the optical flow characteristic itself, the optical flow calculation needs two images from different times to determine the optical flow number which means that the time between two images is part of the delay in the system. The calculation itself also needs time to proceed. After getting the command ready to be sent, the image processing module waits until the next control loop to send the command. It waits because the system is designed to have a constant delay time which will be a benefit for the further controller design. Then the control command is sent to the dSPACE computer to process the real-time simulation which also produces a delay in the system. The dSPACE computer sends the position command to the robot controller to move the robot. The robot mechanic also needs time to response to the commanded position. When the robot is moved, a new image acquired from the camera will contain the information of the movement from the previous frame. The diagram showing the delay in the system can be seen in Fig. 6.9.

In the reality, the total delay in the system can be much higher than the delay described before. In this thesis, the additional delay will be added to the system to study the effects of the delay to the control rule. The controller designed in this thesis is a simple controller with simple rules to handle the delay. The system stability and control design of the delayed system are not studied here. Modelling of the system with delay is an interesting topic to be explored in the future.

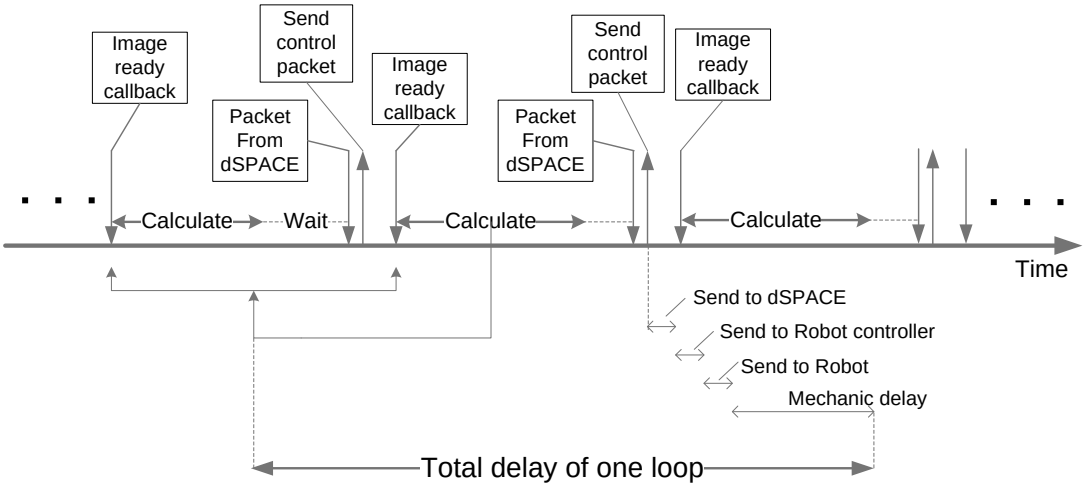


FIGURE 6.9: Delay in one control loop of the hardware-in-the-loop simulation.

6.3.4.1 Control algorithm for the delayed system

Assuming that there is no other clue about current velocity of the spacecraft, old calculated control velocity is the guessed velocity of the next frame. The velocity control equation (Eq. 6.11) can be written in the time sequence by

$$v_{t+\Delta t} = v_t \cdot \frac{\omega_{aim}}{\omega_{avg}} \tag{6.19}$$

where t is the current time and Δt is the time step per frame. Because of the delay in the system, the current measured optical flow, ω_{avg}, is not the interpretation of the current velocity. So, Eq. 6.19 needs to be modified with the correct time delay consideration. Let k be the number of the frame that the current optical flow number is the result of control from the past k frames, Eq. 6.19 can be rewritten into

$$v_{t+\Delta t} = v_{t-k\Delta t} \cdot \frac{\omega_{aim}}{\omega_{avg}}. \tag{6.20}$$

The value k can be vary depend on the system delay characteristic. This method does not predict the state due to the delay. So, with the higher value of k, the controller will give slower response. The state prediction for better control performance should be studied in the future.

Chapter 7

Results and Discussion

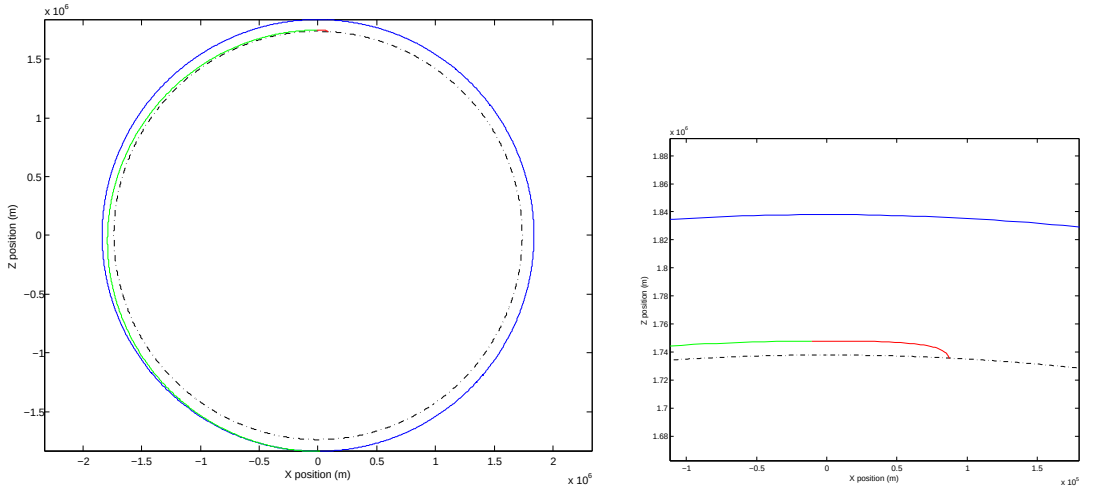
In this chapter, various experiments of spacecraft simulation and control using optical flow will be described and discussed. The same control situation will be performed both in software simulation and in hardware-in-the-loop simulation if applicable. The relation between altitude and velocity control commands will be studied. The trajectories of the spacecraft in different situations will be discussed. The control of delayed system will be also discussed. The results from the image processing algorithm will be presented.

7.1 Orbit simulation and the transition between orbit parts

The dynamic simulation of the parking orbit, transfer orbit and simple gravity turn manoeuvre is simulated using the Runge-Kutta method. The trajectory of the spacecraft in the parking orbit at the altitude of 100 km, during the Hohmann transfer orbit down to the altitude of 10 km and the gravity turn manoeuvre are shown in Fig. 7.1. It can be seen that the calculation gives closed parking orbit indicating that the dynamics simulation and the integration process are correct.

The orbit simulation of parking orbit around the moon which is a circular orbit at the constant altitude of 100 km is chosen to be simulated in the hardware-in-the-loop facility. The spacecraft positions in two orbital parts are selected to be simulated in the laboratory. Two terrain models are defined at the scale of 1:50000 and 1:66666. The positions of the robot compared to the desired transformed position are shown in Fig. 7.2.

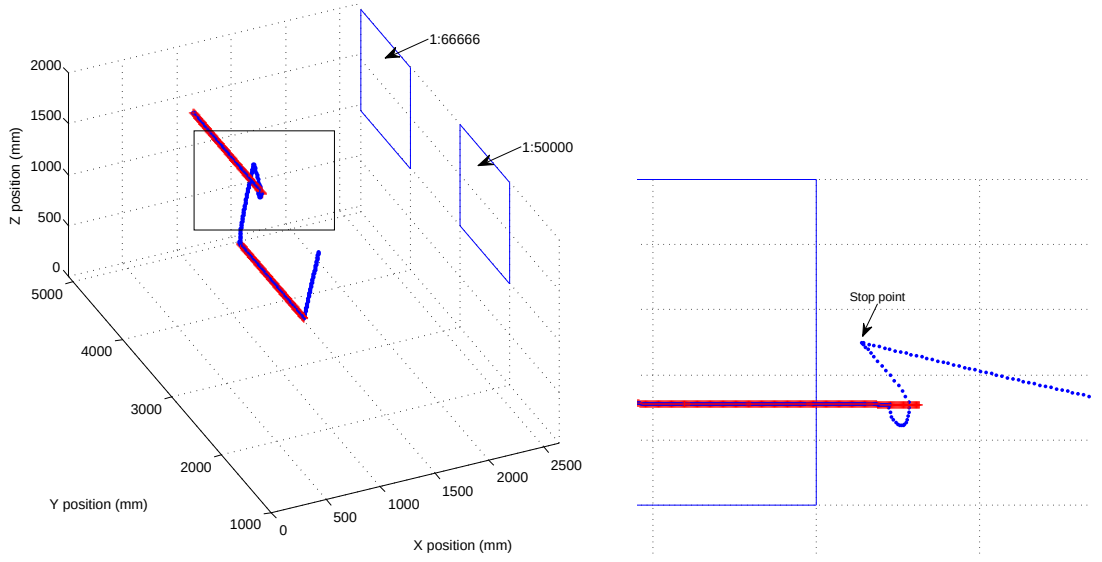
It can be seen from Fig. 7.2 that the robot starts moving from an arbitrary position and follows the designed orbital parts very well. The desired robot velocity to simulate



(a) Overall spacecraft trajectory.

(b) Zoom part of the gravity turn manoeuvre.

FIGURE 7.1: Dynamic simulation of the orbit around the moon. Blue line shows the spacecraft positions in the parking orbit at 100 km altitude. Green line shows the spacecraft positions in Hohmann transfer orbit to 10 km altitude. Red line shows the spacecraft positions in the gravity turn manoeuvre. Black dot line is the surface of the moon.



(a) Robot trajectory of the orbit simulation

(b) Zoom part of robot trajectory before entering the second terrain model.

FIGURE 7.2: Robot trajectory in the hardware-in-the-loop laboratory. Red plus signs show the desired path of the orbit parts. Blue dots show the positions of the robotic arm. The rectangles are the terrain models on the laboratory wall with different scale.

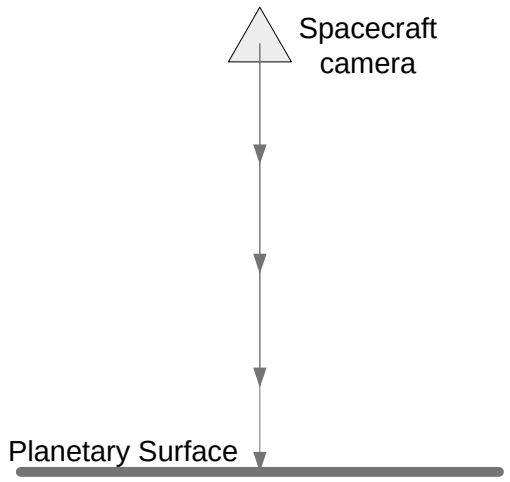


FIGURE 7.3: Diagram shows the landing configuration of the direct linear approach implemented in the thesis.

the parking orbit can be achieved by the robot before the robot is in front of the terrain model (Fig. 7.2(b)). The transition between orbital parts is as designed as follows:

- The robot goes to stop at a point before the terrain model.
- When the simulation time reaches the next orbital part, the robot starts moving to follow the path of the orbit before the robot enter the region of the terrain model.
- The altitudes of two orbital parts are different because the scales of the terrain model are different.

The waiting point of the robot before it moves into the next orbital part can be optimized in the future, so the robot does not need to move back to catch the actual position. This operation proves the ability of the hardware-in-the-loop facility in open loop simulation environment.

7.2 Linear approach to the surface

The linear approach is a very simple movement which can be easily implemented in both software simulation and hardware-in-the-loop simulation. The spacecraft is set to look directly to the ground and falling down toward the ground. Figure 7.3 shows the experiment configuration for the simulation of direct linear approach.

Because of the limitations in the dimensions of the laboratory, the configurations of the spacecraft parameter in software and hardware simulations are slightly different in detail

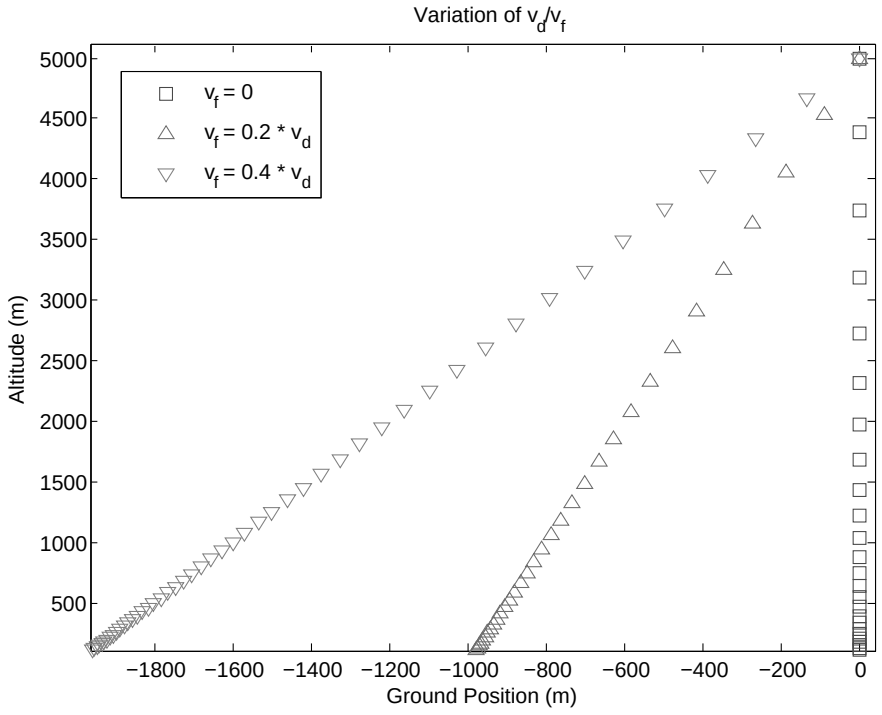


FIGURE 7.4: Trajectories of the spacecraft at different descent angles in software simulation.

of initial condition and the definition of the terrain model. However, the principle of the control rule is still the same and the result is comparable.

7.2.1 Software simulation

In software simulation as described in Sectoin 3.1, the spacecraft is set to start at 5000 metres above the surface. Three experiments with different ratios of v_f/v_d are studied here to compare the effect and efficiency of the landing algorithm.

The desired optical flow in software simulation experiment is 2 pixel/frame. The velocity is controlled by the control rule (Eq. 6.11) to maintain the optical flow to be constant at the desired value. The trajectories of the spacecraft in the simulation at different descent angle are shown in Fig. 7.4. It can be seen that the spacecraft is slowing down when it is nearer to the ground. The average optical flow number can be hold approximately constant as in Fig. 7.5 until the end of the simulation. The velocity is decreasing exponentially with time (Fig. 7.6). Figure 7.7 shows that the velocity and the altitude have a linear relation to each other which means that the velocity will be close to zero when the height is close to zero.

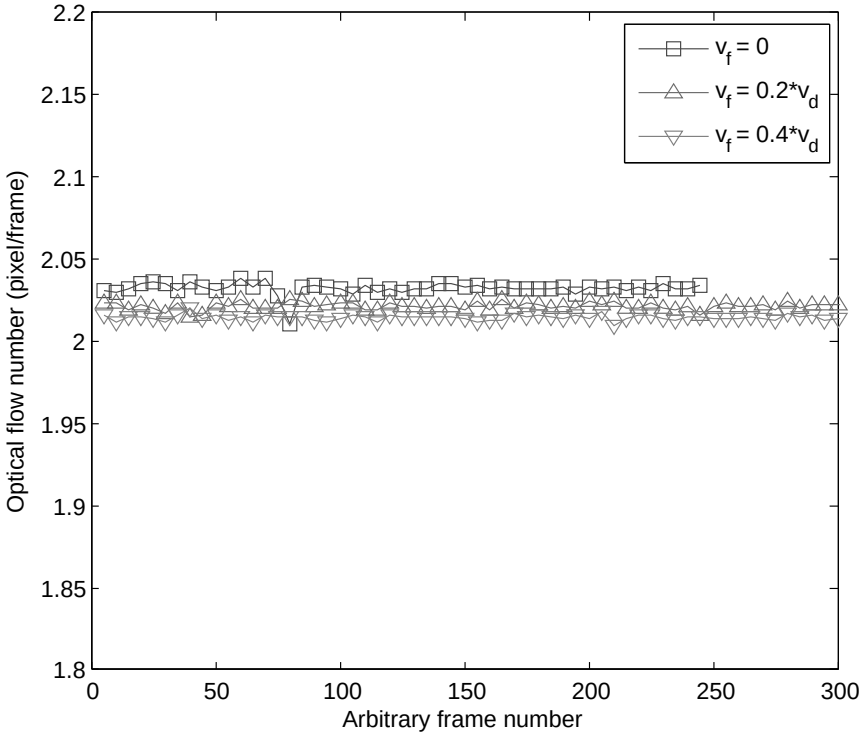


FIGURE 7.5: Optical flow number of the trajectories at different descent angles in software simulation.

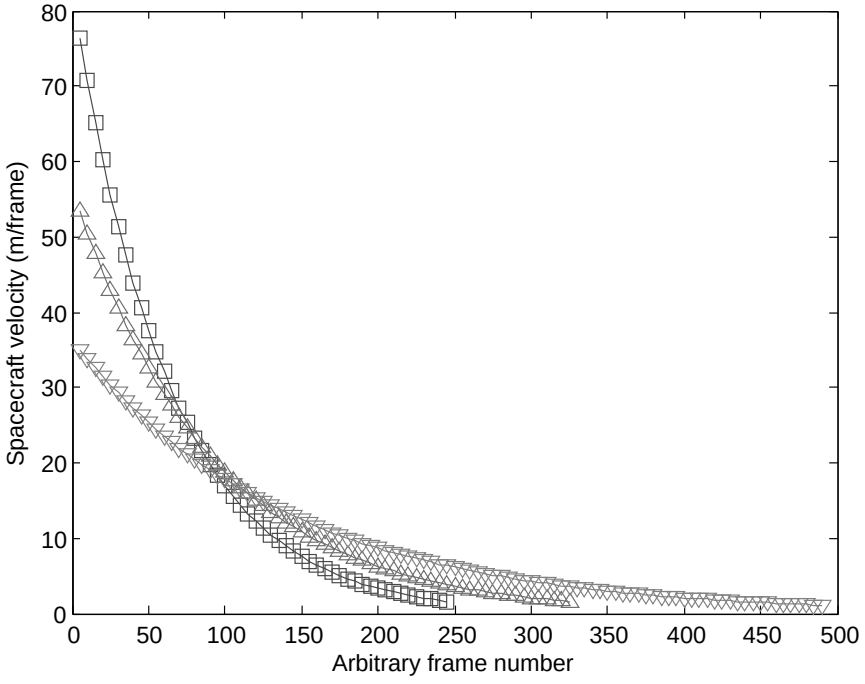


FIGURE 7.6: Spacecraft velocity of the spacecraft of the trajectories at different descent angles in software simulation.

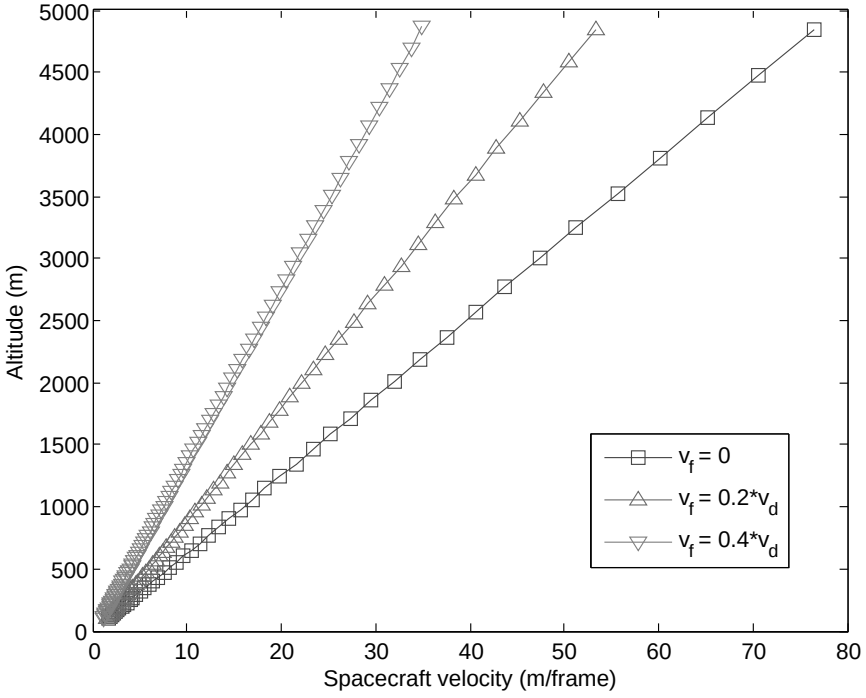


FIGURE 7.7: Relationship between spacecraft velocity and altitude of the trajectories at different descent angles in software simulation.

7.2.2 Hardware-in-the-loop simulation

The same simulation from Section 7.2.1 is applied for the hardware-in-the-loop simulation. The simulated spacecraft is replaced by the robotic arm and the simulated image (from PANGU) is replaced by a real camera at the end of the robotic arm looking a manufactured model with a light source. The simulation setup in the laboratory is shown in Fig. 7.8.

The terrain model in Fig. 7.8 is manufactured from the scaled height elevation data of the moon surface from the KAGUYA mission[1]. The definition of the terrain model is defined as described in Chapter 4. The dynamic simulation of the linear approach is simplified to only depending on the current velocity of the spacecraft regardless the gravitational field. Finally the spacecraft position is transformed into the robot control command and has been controlled by the optical flow number.

It can be seen from Fig. 7.8 that the approach distance is short and the robot cannot move to the position very near the terrain model. The terrain model is also small, so the experiment trajectory here is only part of the direct perpendicular approaching

[1]KAGUYA mission is a Japanese moon mission. The major mission objectives are to obtain scientific data of the lunar origin and evolution and to develop the technology for the future lunar exploration

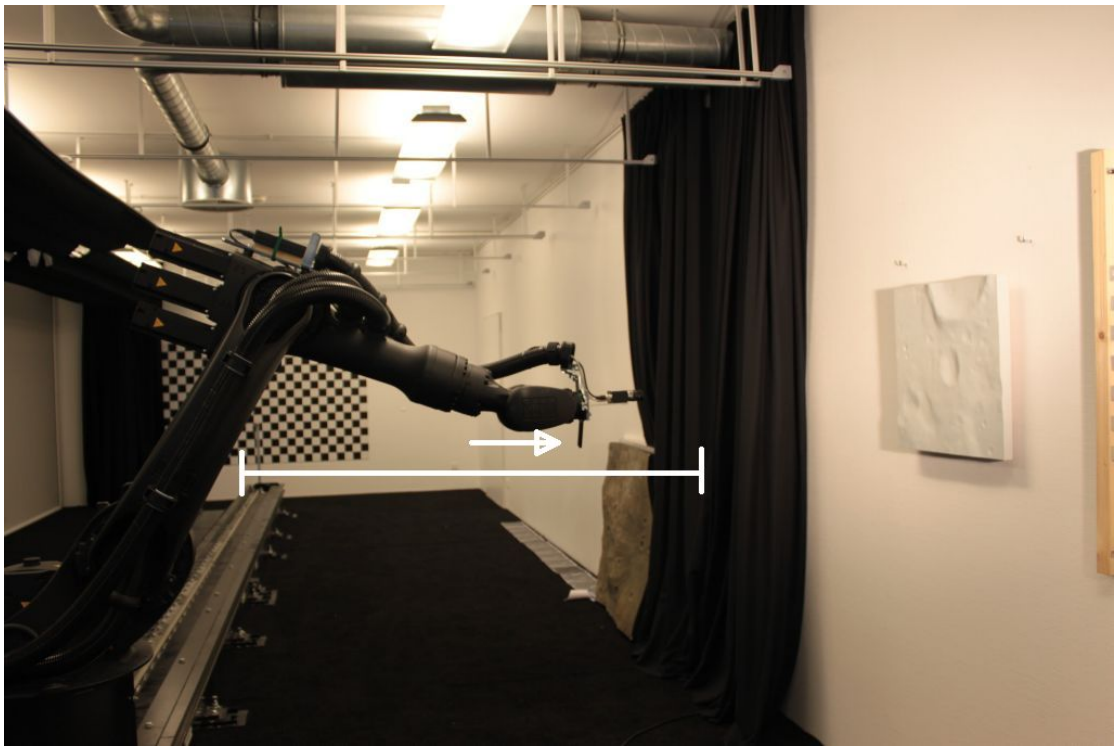


FIGURE 7.8: The hardware-in-the-loop simulation setup of simple landing trajectory with the limits of movement.

manoeuvre. Due to the working space of the robot, the starting altitude of the spacecraft is at 900 metres above the ground with scale of 1:500[2].

In the experiment, the desired optical flow number is 1 pixel/frame. Because the working distance of the robot is limited, to be able to see the slowing down phase of the velocity controlled by the landing algorithm, the desire optical flow number has to be decreased from 2 pixel/frame in the Section 7.2.1 to 1 pixel/frame. The same control law as in the software simulation is applied to control the approaching velocity. Then the changes of velocity (Δv) is transferred to the robot controller causing the robot to change its internal velocity and move to a new position according to the internal velocity. The robot veleocity is measured by acquisition of data from the robot controller. The landing algorithm has no access to the measured velocity.

The trajectory of the robot in the setup above is shown in Fig. 7.9. Even though the trajectory cannot be simulated until the touchdown, it shows that the robot is slowing down during the approach. The average optical flow number which has been calculated from the camera at the end of robotic arm is shown in Fig. 7.10(lower). It can be seen that the optical flow is approximately kept constant. It can be seen that the optical flow number when the robot is stopped is not around zero. Because there are noises in the

[2]the number used for the experiment is not the actual scale of the terrain model.

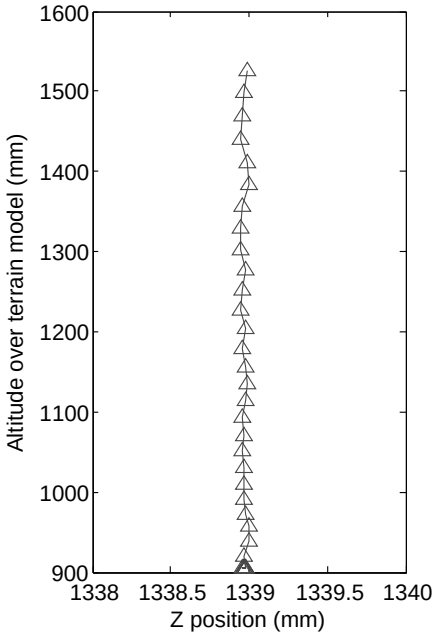


FIGURE 7.9: Trajectory of the robot moving toward the terrain model.

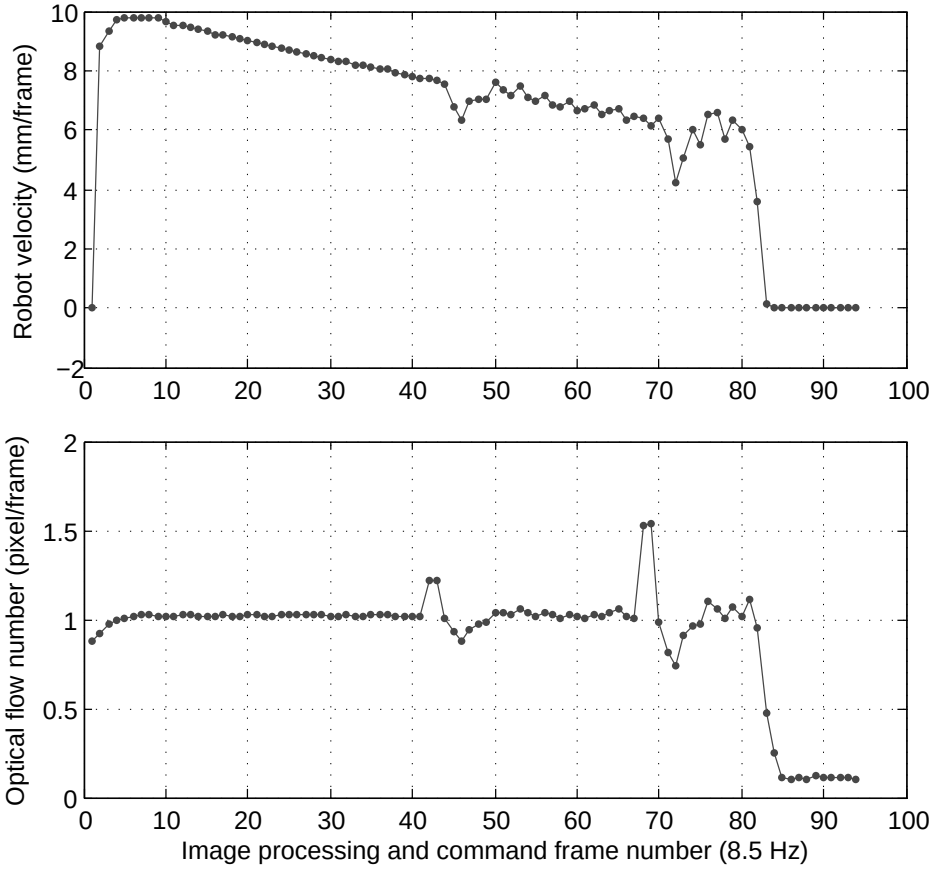


FIGURE 7.10: Robot velocity (upper) and optical flow number (lower) relation in time of the direct approach trajectory in hardware-in-the-loop simulation.

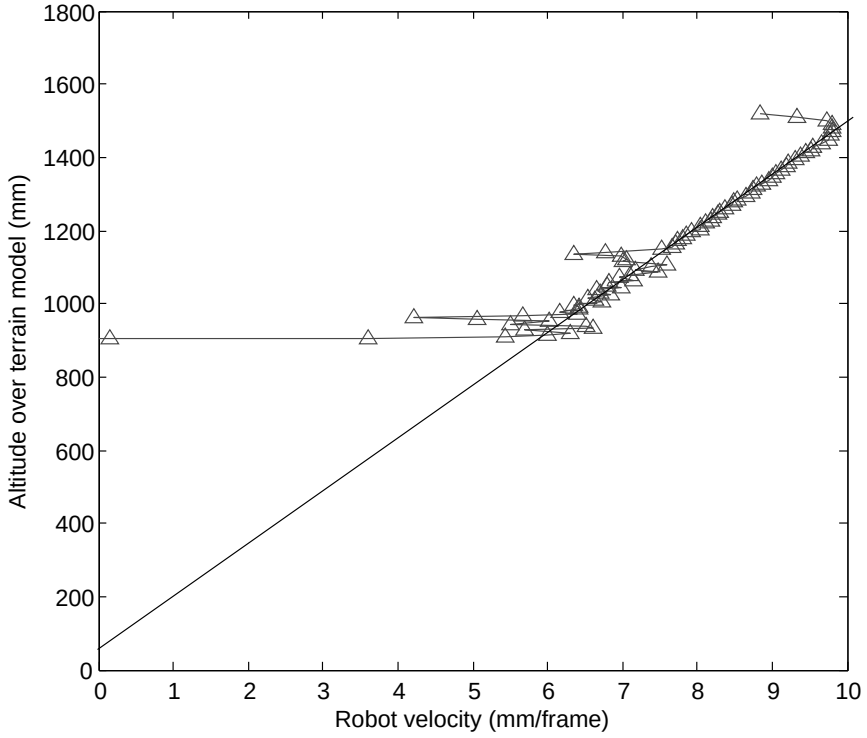


FIGURE 7.11: Relation between robot velocity and the altitude from the terrain model in hardware-in-the-loop simulation.

optical flow and the process of optical flow number calculation performs the summation of the optical flow in each direction, the noises are not cancelling each other resulting the optical flow number is above zero. Further noise filtering algorithm should be studied in the future.

In Fig. 7.10, the optical flow often presents spikes and the spike leads the system to a little bit more oscillation both in optical flow number and robot velocity. The robot velocity is observed to be decreased according to the control of optical flow until the robot cannot move further (Fig. 7.10(upper)). From Fig. 7.10, it can be observed that when there is a spike in the optical flow number, the robot velocity responses to the spike in the opposite direction to keep the optical flow to be constant. Furthermore, the relation between robot velocity and the altitude from the terrain model has linear relation with approximately final speed at the ground of zero as seen in Fig. 7.11.

The spikes as seen in Fig. 7.10 are caused by the shaking because of low level control of the robotic arm. These spikes also present in the delay measurement experiments in the next section. It can be concluded that the spikes do not cause by the optical flow control law. Even with the spike, the controller still can pull the system back to the stable state with some oscillations.

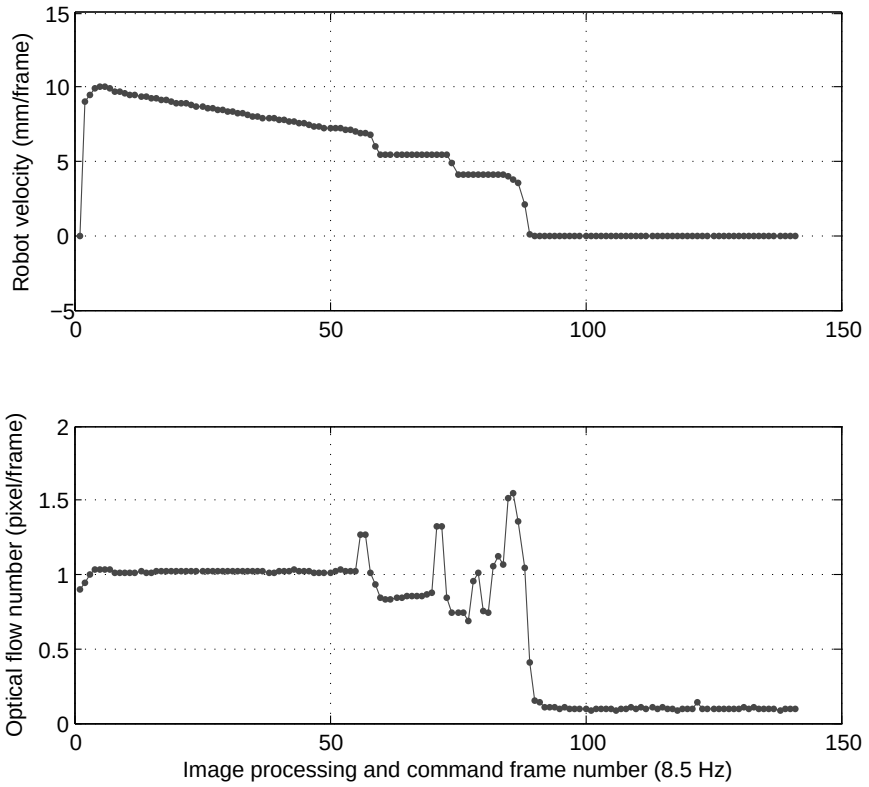


FIGURE 7.12: Robot velocity (upper) and optical flow number (lower) relation in time of the direct approach trajectory controlled only by slowing down the robot velocity (no accelleration) in hardware-in-the-loop simulation.

During the realistic landing, vibrations and sudden jerks can also occur at anytime. One way to easily handle with them is to let the system only slow down even though the optical flow number is lower than the desired value, see Section 6.3.2.1. The result of this strategy is shown in Fig. 7.12. It can be seen that when the spike in the optical flow number showed up, the robot velocity is slowed down and stays constant until the robot moves close enough to the surface that the optical flow number exceeds again the desired number. This behaviour is also a good behaviour that needs a less complicated actuator model. The system also does not suffer from the oscillation after the spike.

7.3 Delayed system

The delay in the control system can significantly change the behaviour of system response. The delay identification will be first introduced. Then the additional delay will be added to the system to study the consequence of the delay to the system response. Finally, the simple control rule for the delayed system is evaluated and discussed in this section.

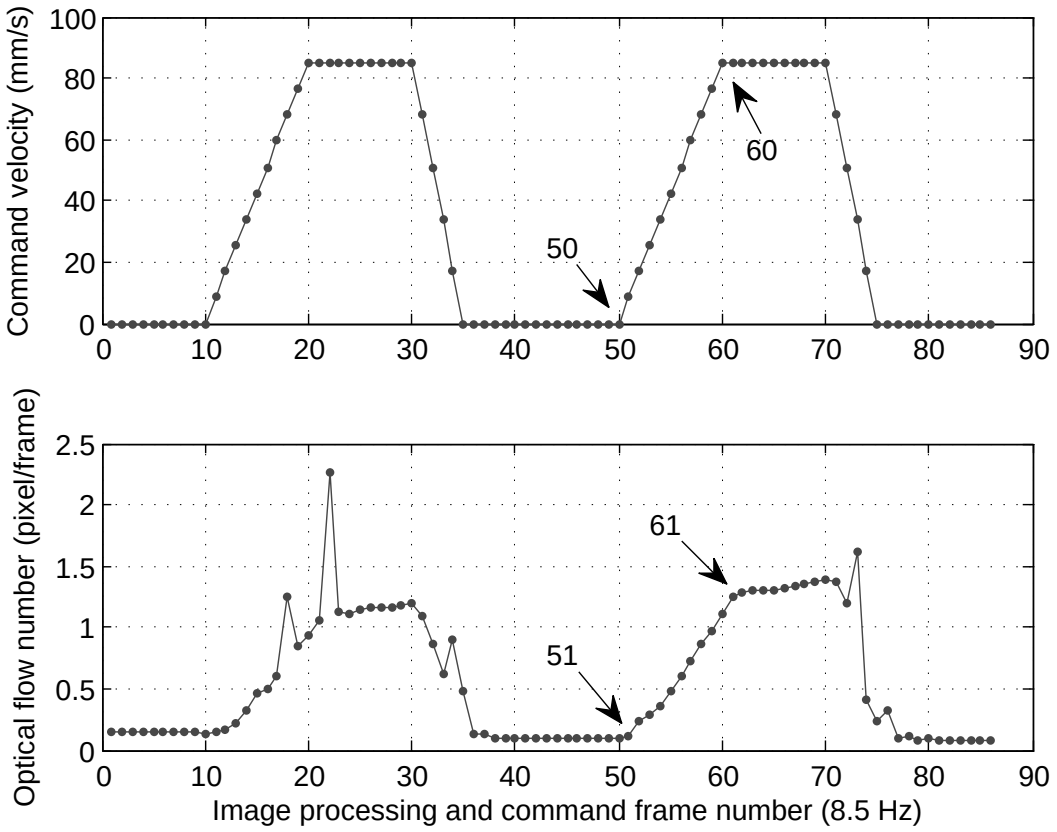


FIGURE 7.13: Delay measurement by comparison of the timing of command velocity and optical flow number. The plot shows nominal system delay.

7.3.1 Delay measurement

In one control loop, the time from the point that the command is sent out, until the measurement of the sensor changes (according to command) is the delay time to be considered in the control rule. To measure this delay, a following experiment has been setup. A certain pattern of velocity commands has been sent to the robot which carries the camera. The pattern in this case is similar to a step-up and step-down signal with limited acceleration that is within the ability of the robot. Then the optical flow resulting from the movement of the robot is recorded and compared with the command-time. The measurement of the nominal system delay without additional delay added is shown in Fig. 7.13. It can be seen that the delay from the point that the control command is sent out until the changes of the optical flow number according to the command is approximately 1 frame at 8.5 Hz.

The total delay in the system is related to the delay of the hardware which in this case is the robotic arm. To study the characteristic of the robotic arm, the delayed response of the robot because of robot controller and mechanical delay in the robot itself is

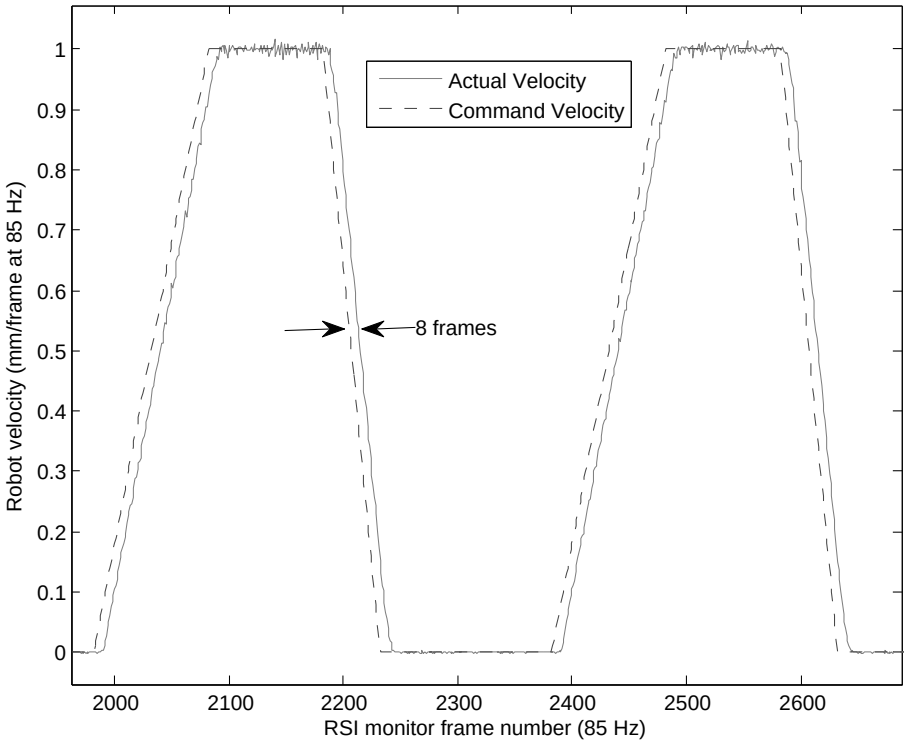


FIGURE 7.14: Delay measurement of robot response time by comparison of command velocity and measured velocity from the RSI monitor.

measured. The robot response time is the minimum delay that can be achieved in the hardware-in-the-loop simulation regardless the delay from another part of the system. With the same technique of measurement, the delay of the robotic arm is measured using the data from RSI monitor[3] and can be seen in Fig. 7.14.

From Fig. 7.14, the delay is 8 frames at 85 Hz which is less than 1 frame of the image processing control loop of 8.5 Hz. This means that the delay in the image processing loop will increase to 2 frames if the image processing and command frequency is higher than 10.625 Hz.

7.3.1.1 Influence of the image frame buffer to the system delay

The same experiment is setup with the additional delay time in the system by using an artificially delayed image. The image buffer is set to be 5 frames and the control loop is waiting for the image buffer to be full before it starts using the image for optical flow calculation. The additional delay is then 5 frames from the command until a measureable effect on optical flow number. The delay measurement is shown in Fig. 7.15. The delayed system will be controlled and its behaviour will be studied in the next

[3]RSI monitor is a program to collect robot data directly from the robot controller.

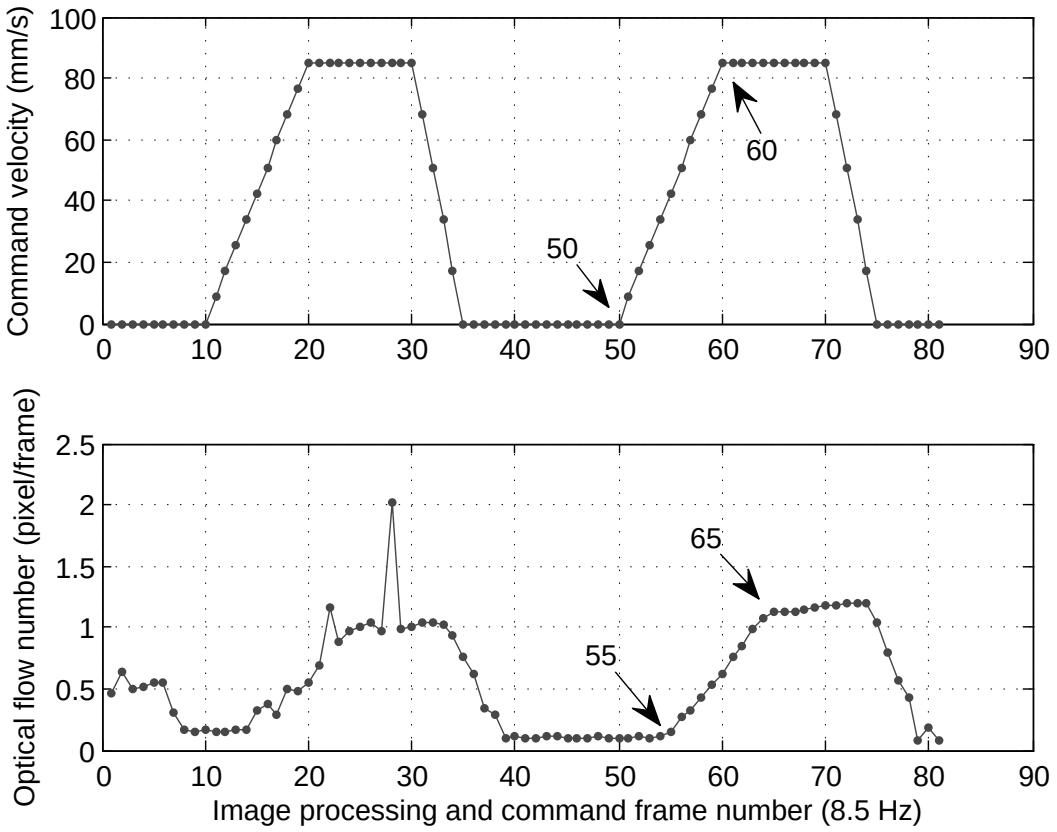


FIGURE 7.15: Delay measurement by comparison of the timing of command velocity and optical flow number. The plot shows the delay of the system with 5 frames image buffer.

section.

7.3.2 Control of the delayed system

The delayed system is controlled by the control rule described in Section 6.3.4.1. Both software simulation using PANGU for image generation and hardware-in-the-loop simulation in TRON will be studied. Different parameters of the control rule will be discussed.

7.3.2.1 Software simulation

The delay is added into the simulation by storing the calculated command in a buffer before applying it 5 frames later to the spacecraft state in the simulation. The velocity profiles of different values of considered delayed frame in the control rule (k in Eq. 6.20) during the direct approach to the ground are studied here. Figure 7.16 and 7.17 show the effect of the value k to the system response which can be seen that if k is too low

or too high, the system will be oscillating. In the case that k is the correct value, the system can be controlled to slow down and to land softly with slow velocity.

One can observe that the frequencies of the oscillation of the system response at different k values are various. With the low number of $k = 2$, the system responses with oscillation in low frequency. When the value of $k = 4$ which is near to the correct delay time, the system response is oscillated very fast. The frequency of behaviour of the system can be various by many factors which can be investigated in the future.

7.3.2.2 Hardware-in-the-loop simulation

The same to the software simulation, different values of k have been applied for the direct approach in the control rule. The velocity profiles of the robot are studied and compared with the software simulation. It can be seen from Fig. 7.18 and Fig. 7.19 that the system is unstable when k is less than the actual delay time. It also can be seen that the behaviour of the system is very similar to the result from software simulation in terms of frequency responses.

For the system response of $k = 7$, it seem that the system is not oscillating in the hardware-in-the-loop simulation but the system is oscillating in the software simulation with PANGU. There are many factors that could improve the situation in the hardware-in-the-loop simulation. The response of the robot may be not as sharp as in the software simulation i.e. the robot has to slowly accelerate. This can make the response to be lower in frequency. The simulation also presents only a short time at the beginning of the operation, the oscillation part may occur in the future if the simulation can be perform with longer distance. The clearer explanation should be studied in the future. At the moment, it can be concluded that with the correct value of k, the system can be successfully controlled to perform a soft landing.

7.4 Curve approach

The landing trajectory also can be designed as a curve approach. It does not need to be always in a straight line. The curve trajectory can be achieved by varying the forward velocity and downward velocity ratio during the approach. Figure 7.20 shows the trajectory of an approach with the decrease of the velocity ratio v_d/v_f from 0.5 to 0 while the velocity is controlled to maintain a constant optical flow. It can be seen that the spacecraft is slowed down during the landing, promising a soft touchdown similar to the linear approach.

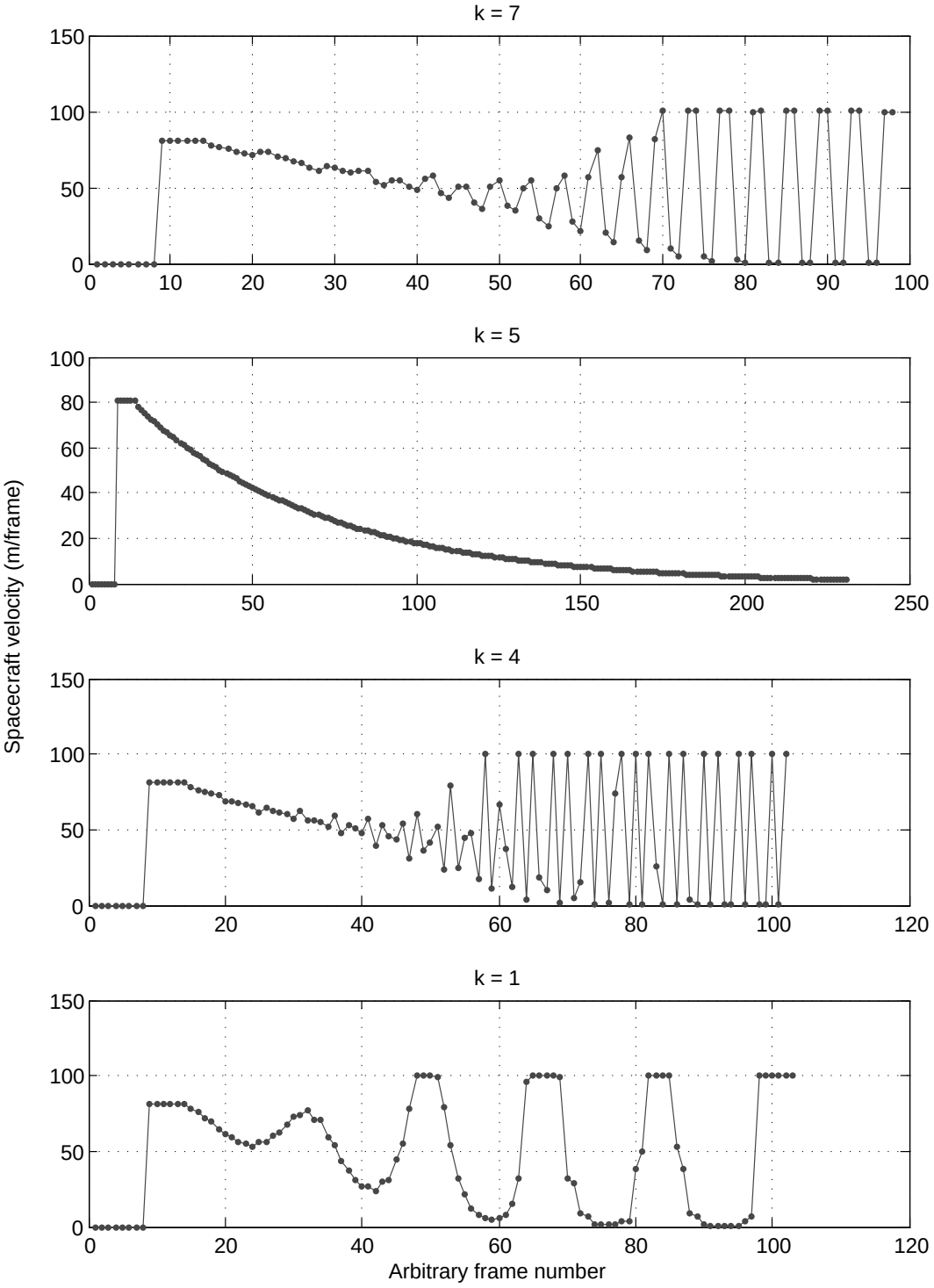


FIGURE 7.16: Velocity profiles of the spacecraft in the software simulation controlled by different k value in the Eq. 6.20.

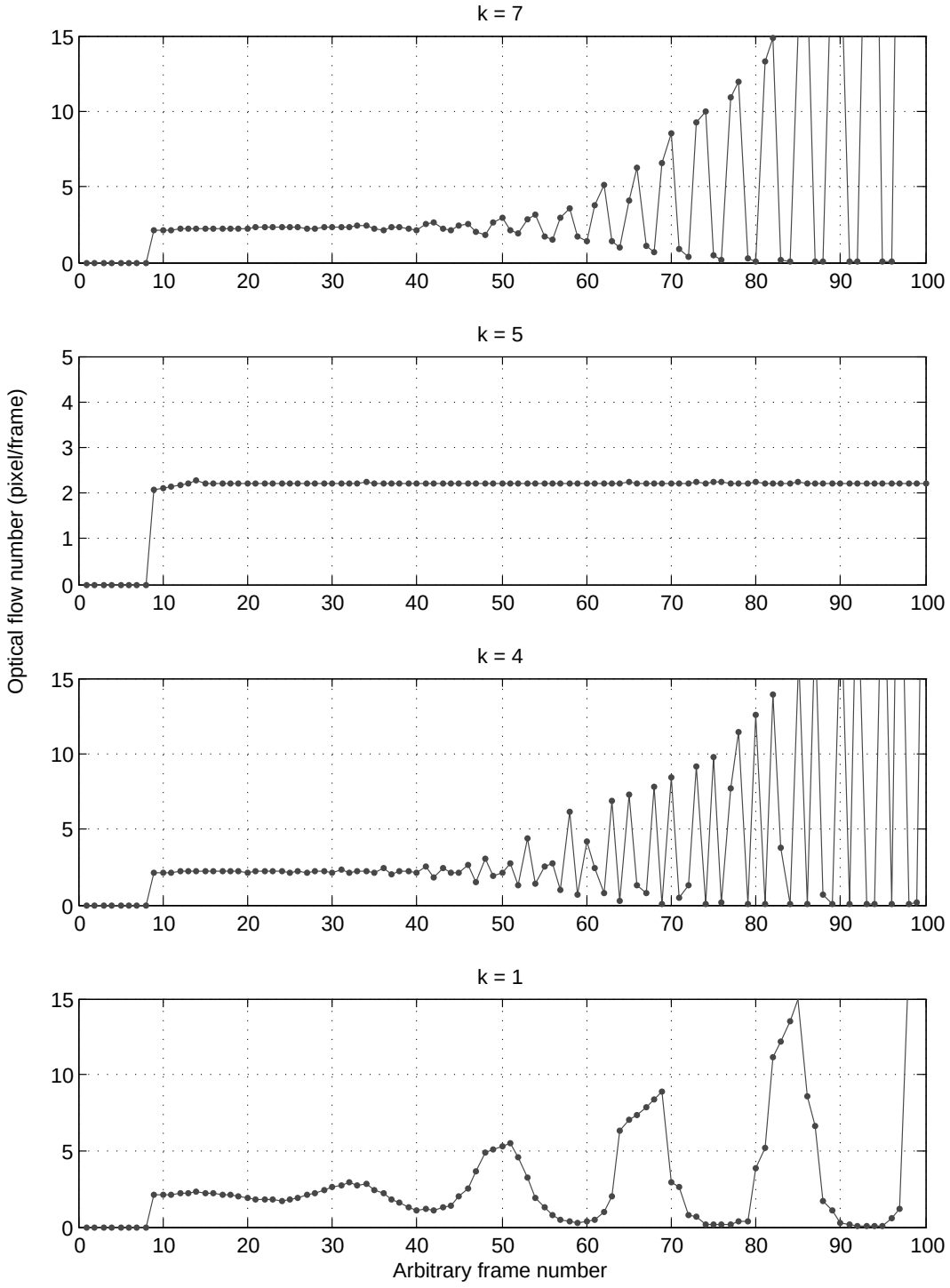


FIGURE 7.17: Optical flow numbers produced from the spacecraft in the software simulation controlled by different k value in the Eq. 6.20.

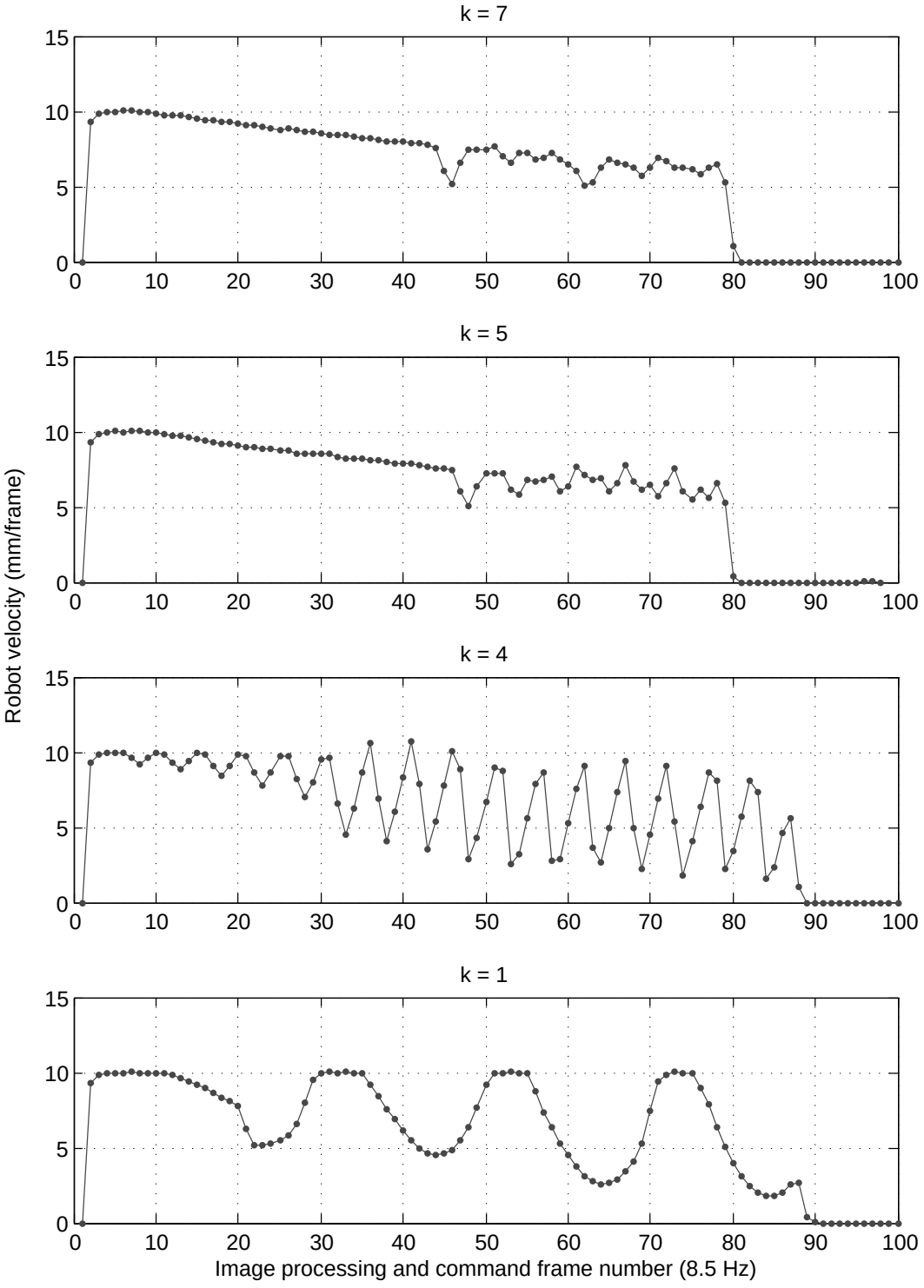


FIGURE 7.18: Velocity profiles of the robot in hardware-in-the-loop simulation controlled by different k value in the Eq. 6.20.

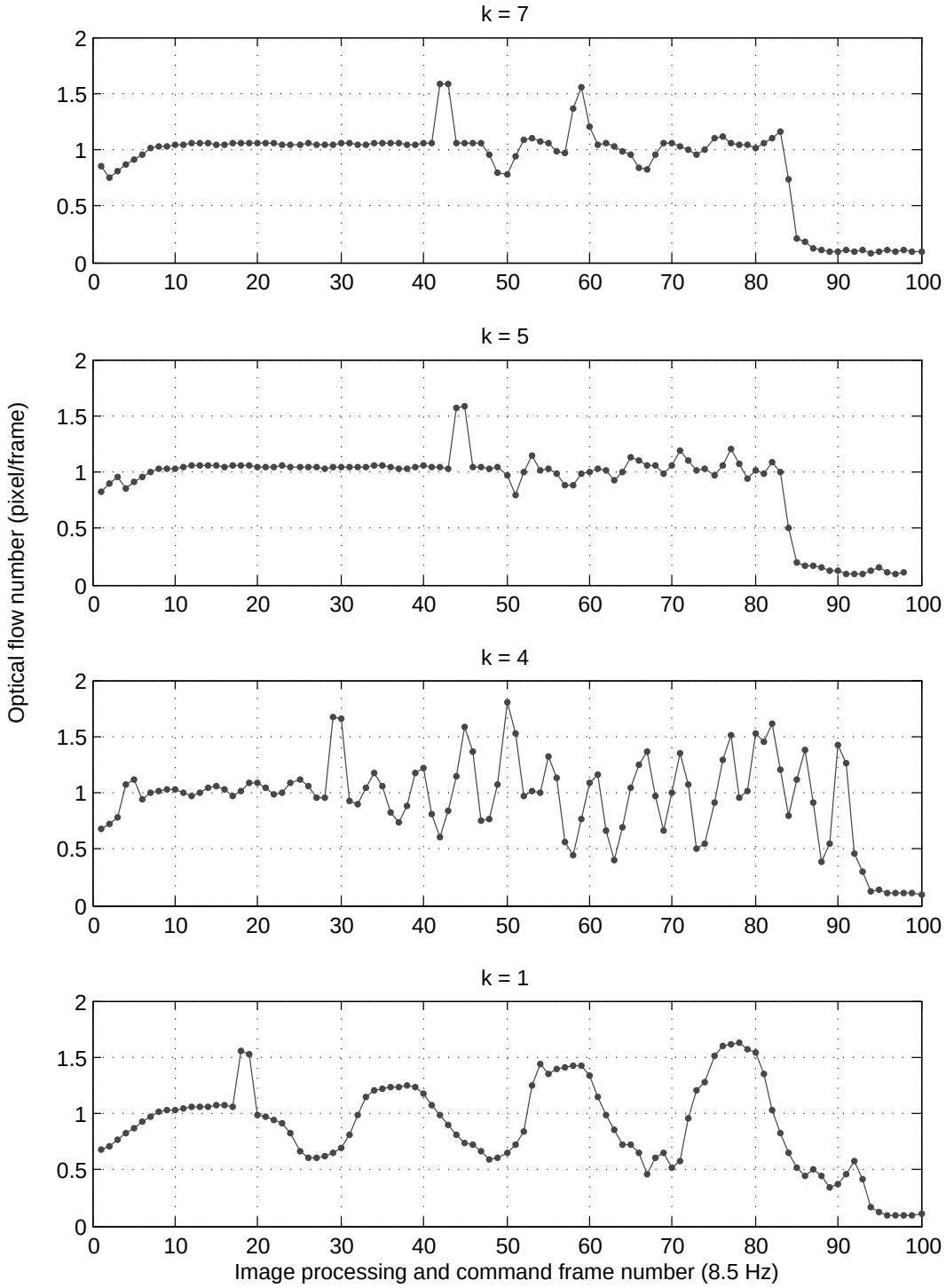


FIGURE 7.19: Optical flow number produced from the robot in hardware-in-the-loop simulation controlled by different k value in the Eq. 6.20.

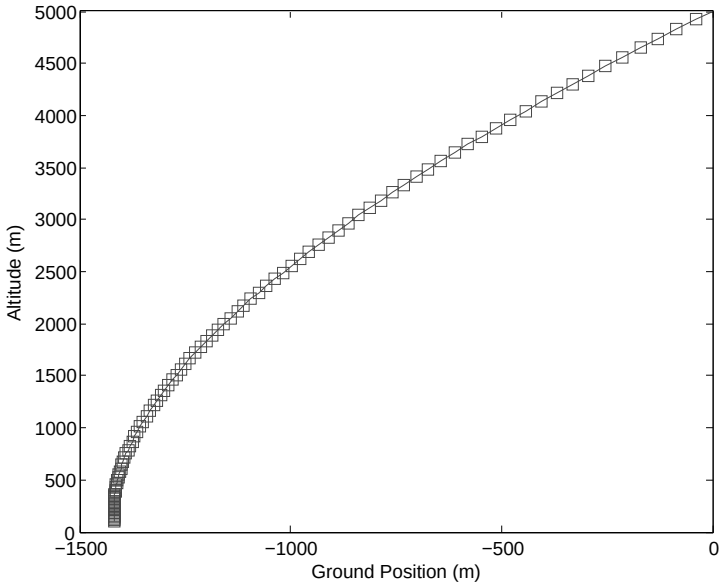


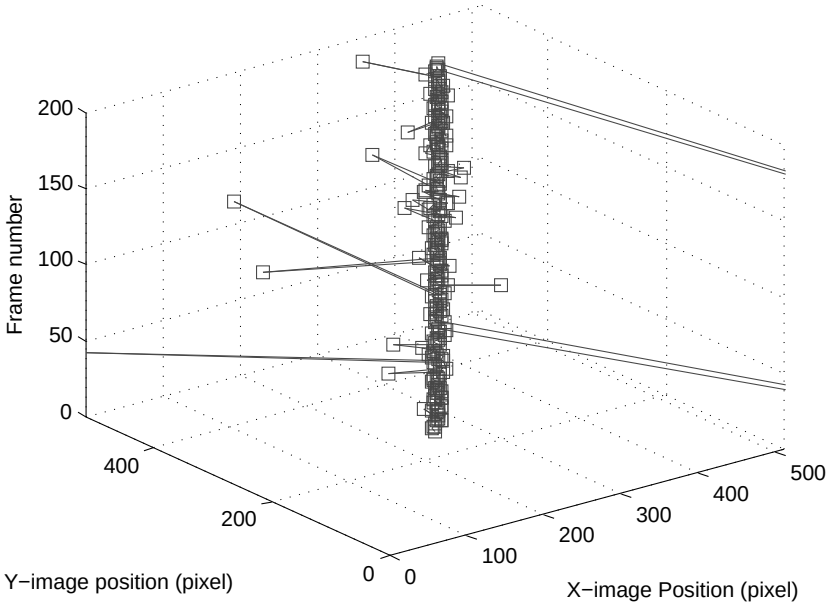
FIGURE 7.20: Curve trajectory produced by adjusting the velocity ratio v_d/v_f from 0.5 to 0 while the optical flow number is controlled to be constant.

7.5 Focus of expansion

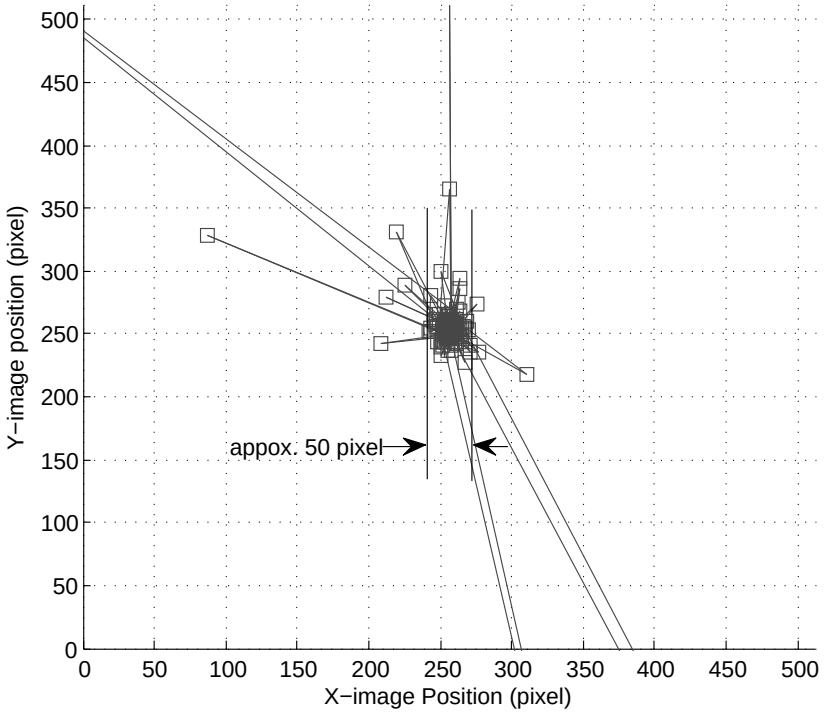
The calculation of the FOE using the image sequence from the direct linear approach in software simulation is shown in Fig. 7.21. It can be seen that the position of the FOE is mostly in the middle of the FOV. There are some frames that the FOE is jumping out of the FOV. That spikes should be able to be filtered out in the future. After the filter is implemented, the accuracy of the FOE position is approximately 50 pixels which is 2° in the FOV of 20°.

For FOE calculation in the hardware-in-the-loop simulation, the FOE during the approach is shown in Fig. 7.22. It can be seen that the FOE is very unstable in vertical direction because of the shaking of the robotic arm. The spikes similar to the result from software simulation still exist. Further improvement of the algorithm is needed to filter out the noise to enable the use of FOE to control the attitude. The accuracy of the FOE calculation from the hardware-in-the-loop simulation in X-direction is also approximately 50 pixels. The accuracy in Y-direction cannot be determined due to the shaking.

The use of the FOE is related to the rotation simulation. The study of the use of FOE should be investigated and tested in the future.

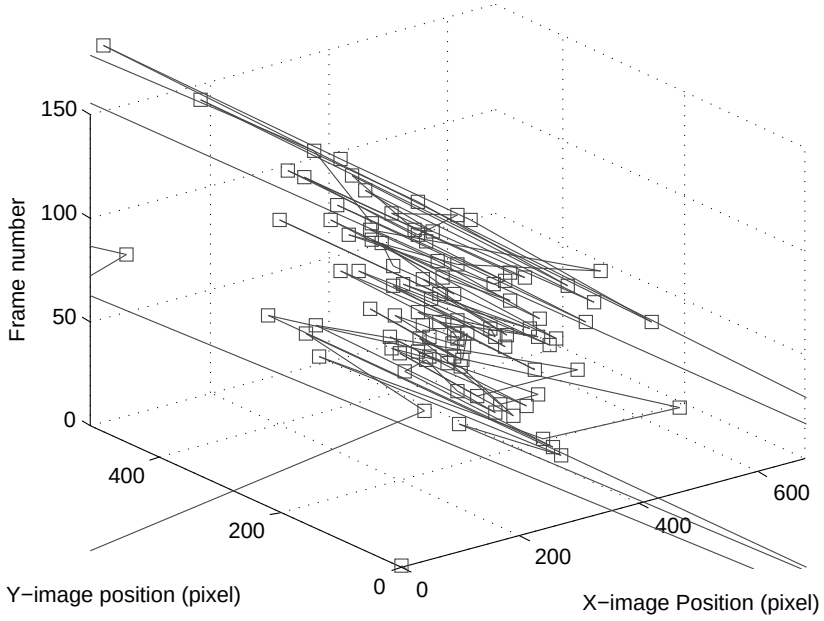


(a) FOE position shown in different frames.

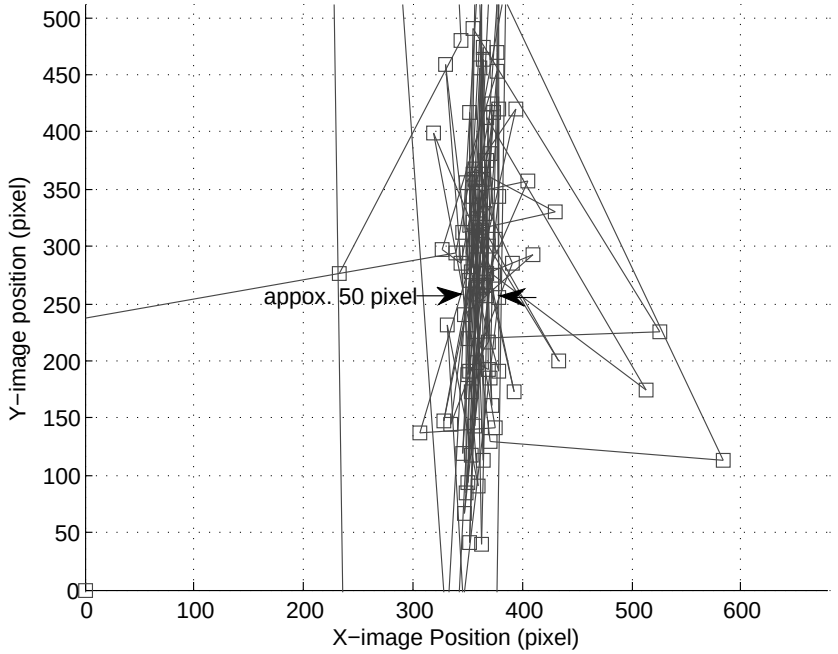


(b) FOE position of different frames shown in one image plane.

FIGURE 7.21: The FOE position of the direct approach landing in software simulation.



(a) FOE position shown in different frames.



(b) FOE position of different frames shown in one image plane.

FIGURE 7.22: The FOE position of the direct approach landing in the hardware-in-the-loop simulation.

Chapter 8

Conclusions and Future Work

In this thesis, the hardware-in-the-loop simulation software components for optical navigation have been developed. The robotic arm carrying a camera looking at a terrain model is controlled. The mathematical definition of terrain model and related reference frames have been described. The real-time robotic arm control interface has been implemented and tested using dSPACE real-time system. The open loop orbit simulation has been performed in the hardware-in-the-loop facility. The closed loop control scheme from the image processing signal to the control of the robot has been setup and tested by applying an insect-inspired landing manoeuvres.

Furthermore, the landing strategy of the insect using optical flow has been studied. The calculation of the optical flow has been implemented using Lukas-Kanade algorithm. The optical flow is controlled to be constant by controlling the velocity of the spacecraft. This method is feasible for the landing of the spacecraft because the optical flow characteristic gives the desired velocity profile for landing which is the high velocity at high altitude and low velocity near the touchdown. The strategy has been tested successfully by linear direct approach to the surface both in software simulation with PANGU and hardware-in-the-loop simulation in TRON. The curve landing trajectory has been tested in the software simulation. From the testing result, it can be seen that the control rule for control the optical flow is sensitive to noise and delay. The delay problem can be tackled when knowing the delay and considering it in the control rule. The algorithm can slow down the spacecraft velocity and be able to land softly on the surface when the correct delay time is considered.

8.1 Future work

The thesis has setup the hardware-in-the-loop software components in terms of real-time robot position control and dynamics transformation to perform the closed loop control operation. There are many details in each component developed in this thesis that can be improved in terms of accuracy and reliability. Firstly, further software development needs to be done to enable rotation related manoeuvres to be simulated by the robot movement. Then the rotation related algorithm can be tested. The second improvement is the accuracy issue. The calibration of the robot position according to the desired trajectory is needed to be studied. At the moment, there is no position feedback sensor except the position from the integration of the robotic arm joint positions. Another factor in the accuracy of the simulation is the transformation of spacecraft position to the robot in laboratory reference system. The current development assumes that the terrain model is flat. However, in reality, the terrain model is part of the planet and has a curved shape. This has to be studied in detail for improving the accuracy of the simulation.

The robot control could also be further developed. At the moment, the robot is controlled by the position without consideration on the velocity which means that the robot will simulate the spacecraft position but will not simulate the spacecraft velocity at that position. A new controller that can control both of them in the same time will be the interesting topic in the future to be explored.

For the optical flow algorithm, the improvement can be done in terms of run-time efficiency. An outlier suppression process can be added to the optical flow field to remove non-agreement optical flow which can be resulted from features in the shadow or in the reflection peak.

The vibration issue of the robotic arm is also a problem to be solved. The method to get rid of the vibration of the robot low level controller should be investigated. A controller design that has ability to handle the vibration noise which can occur in reality during the landing is an interesting study in the future. Also mathematical modelling of the system that can be used in controller design with delay consideration should be investigated in the future.

Bibliography

G. Adiv. Inherent ambiguities in recovering 3-d motion and structure from a noisy flow field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:477–489, 1989.

J. Barron, D. Fleet, and S. Beauchemin. Performance of optical flowtechniques. *International Journal of Computer Vision*, 12(2):43–77, 1994.

A. Beyeler, J.-C. Zufferey, and D. Floreano. optiPilot: control of take-off and landing using optic flow. In *Proceedings of the 2009 European Micro Air Vehicle conference and competition (EMAV '09)*, 2009. URL `http://www.emav2009.org/`.

J.-Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm. Technical report, Microprocessor Research Labs, Intel Corporation, 2000.

J. S. Chahl, M. V. Srinivasan, and S. W. Zhang. Landing strategies in honeybees and applications to uninhabited airborne vehicles. *The International Journal of Robotics Research*, 23(2):101–110, February 2004. doi: 10.1177/0278364904041320.

A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):1052–1067, 2007. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/TPAMI.2007.1049.

J. de Lafontaine, D. Neveu, and J.-F. Hamel. Autonomous planetary landing using a lidar sensor: the landing dynamic test facility. In *7th International ESA Conference on Guidance, Navigation & Control Systems*, 2008.

E. Eade and T. Drummond. Scalable monocular slam. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 469–476, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2597-0. doi: http://dx.doi.org/10.1109/CVPR.2006.263.

J. O. Entzinger and S. Suzuki. Modeling of the human pilot in aircraft landing control. In *Proceedings of the 10th APRU Doctoral Students Conference*. Department of Aeronautics and Astronautics, University of Tokyo, Association of Pacific Rim Universities, 2009.

C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–152, 1988.

B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligent*, 9: 185–203, 1981.

H. Krüger and S. Theil. TRON - hardware-in-the-loop test facility for lunar descent and landing optical navigation. In *18th IFAC Symposium on Automatic Control in Aerospace*, 2010.

KUKA Roboter GmbH. *KUKA.RobotSensorInterface 2.3*, May 2009.

KUKA Robotics Corp. *KUKA Arm Tutorial 2*.

B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *DARPA Image Understanding Workshop*, pages 121–130, 1981.

T. Misu, T. Hashimoto, and K. Nimomiya. Optical guidance for autonomous landing of spacecraft. *IEEE TRANSACTIONS ON AEROSPACE AND ELECTRONIC SYSTEMS*, 35(2):459–473, April 1999.

A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, A. Ansar, and L. Matthies. Vision-aided inertial navigation for spacecraft entry, descent, and landing. *Trans. Rob.*, 25(2):264–280, 2009. ISSN 1552-3098. doi: http://dx.doi.org/10.1109/TRO.2009.2012342.

M. G. Nagle, M. V. Srinivasan, and D. L. Wilson. Image interpolation technique for measurement of egomotion in 6 degrees of freedom. *Optical Society of America*, 14: 3233–3241, 1997.

S. Negahdaripour. *Direct Passive Navigation*. PhD thesis, MIT, 1987.

S. Negahdaripour and B. K. P. Horn. A direct method for locating the focus of expansion. *Computer Vision, Graphic and Image Processing*, 46(3):303–326, 1989. ISSN 0734-189X. doi: http://dx.doi.org/10.1016/0734-189X(89)90035-2.

S. M. Parkes, I. Martin, and M. Dunstan. Planet surface simulation with pangu. In *Eighth International Conference on Space Operations, Montreal, Canada*, pages 1–10, 2004.

P. Rives and J. R. Azinheira. Visual Auto-landing of an Autonomous Aircraft. 0 RR-4606, INRIA, 11 2002. URL `http://hal.inria.fr/inria-00071979/en/`.

S. Saripalli, , S. Saripalli, and G. S. Sukhatme. A testbed for mars precision landing experiments by emulating spacecraft dynamics on a model helicopter. In *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2097–2102, 2002.

J. Shi and C. Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994.

M. V. Srinivasan. An image-interpolation technique for the computation of optic flow and egomotion. *Biological Cybernetics*, 71:401–415, 1994.

M. V. Srinivasan, M. Lehrer, W. H. Kirchner, and S. W. Zhang. Range perception through apparent image speed in freely flying honeybees. *Visual neuroscience*, 6: 519–35, May 1991.

M. V. Srinivasan, S. W. Zhang, M. Lehrer, and T. S. Collett. Honeybee navigation en route to the goal: Visual flight control and odometry. *Journal of Experimental Biology*, 199:237–244, 1996.

S. Thakoor, J. Chahl, G. Stange, M. Srinivasan, B. Hine, and S. Zornetzer. Insect inspired biomorphic altitude hold, hazard avoidance/terrain following and flight navigation/stabilization system. NASA Tech Briefs NPO-30545, NASAs Jet Propulsion Laboratory, Jan 2005.

C. Voesenek. Implementing a fourth order runge-kutta method for orbit simulation, June 2008. URL `http://home.deds.nl/~infokees/useful/OrbitRungeKutta4.pdf`.

T. Whiteside and G. Samuel. Blur zone. *Nature*, 225:94–95, 1970.

P. Zhao and M. E. Spetsakis. Near real-time optical flow. In *Proceedings of 14th International Conference on Vision Interface*, pages 47–55, Ottawa, Canada, 2001.