# Vision-based Automatic Landing of a Rotary UAV

by

© Iryna Borshchova

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

October, 2017

St. John's Newfoundland and Labrador

# Abstract

A hybrid-like (continuous and discrete-event) approach to controlling a small multi-rotor unmanned aerial system (UAS) while landing on a moving platform is described. The landing scheme is based on positioning visual markers on a landing platform in a detectable pattern. After the onboard camera detects the object pattern, the inner control algorithm sends visual-based servo-commands to align the multi-rotor with the targets. This method is less computationally complex as it uses color-based object detection applied to a geometric pattern instead of feature tracking algorithms, and has the advantage of not requiring the distance to the objects to be calculated. The continuous approach accounts for the UAV and the platform rolling/pitching/yawing, which is essential for a real-time landing on a moving target such as a ship.

A discrete-event supervisor working in parallel with the inner controller is designed to assist the automatic landing of a multi-rotor UAV on a moving target. This supervisory control strategy allows the pilot and crew to make time-critical decisions when exceptions, such as losing targets from the field of view, occur. The developed supervisor improves the low-level vision-based auto-landing system and high-level human-machine interface.

The proposed hybrid-like approach was tested in simulation using a quadcopter model in Virtual Robotics Experimentation Platform (V-REP) working in parallel with Robot Operating System (ROS). Finally, this method was validated in a series of real-time experiments with indoor and outdoor quadcopters landing on both static and moving platforms. The developed prototype system has demonstrated the capability of landing within 25 cm of the desired point of touchdown. This auto-landing system is small (100 x 100 mm), light-weight (100 g), and consumes little power (under 2 W).

# Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Siu O'Young, whose insight and expertise greatly assisted my research. I am thankful for his patience, motivation, and immense knowledge. His guidance helped me complete this research project and thesis.

I would also like to thank the Research and Development Corporation (RDC) and Atlantic Canada Opportunities Agency (ACOA) for their financial support of this research project, and express my gratitude to everyone who supported me throughout my PhD studies: Dr. Sam Nakhla, Dr. Tariq Iqbal, Dr. V. Mike Contarino, Dilhan Balage, Bruno Artacho, and the RAVEN II team. I am thankful for the invaluable guidance, constructive criticism, and advice they provided during my research work. I am sincerely grateful to them for sharing their honest and illuminating views on my research project.

Finally, and most importantly, I would like to thank my family and friends: my mother, Tetyana, my sister, Natasha, my fiancé, Aaron, and his parents, Elaine and Leonard, who became my second family. You were always there with words of encouragement or a listening ear. Thank you.

# Table of Contents

# List of Tables

# List of Figures

# List of Acronyms

| | |
|---|---|
| AVO | air vehicle operator |
| AP | Autopilot |
| DES | discrete-event system |
| DPT | desired point of touchdown |
| EP | external pilot |
| FoV | field of view |
| FPV | first-person view |
| GCS | ground control station |
| GPS | global positioning system |
| GUI | graphical user interface |
| HIL | hardware-in-the-loop |
| IBVS | image-based visual servoing |
| IMU | inertial measurement unit |
| LADAR | laser detection and ranging |
| LIDAR | light detection and ranging |
| LSD | line segment detector |
| M&S | modeling and simulation |
| PIC | pilot-in-command |
| PBVS | position-based visual servoing |
| RC | radio control |

| | |
|---|---|
| ROS | robot operating system |
| SFOC | special flight operations certificate |
| SNR | signal to noise ratio |
| SIFT | scale-invariant feature transform |
| SVD | singular value decomposition |
| SWaP | size weight and power |
| TDES | timed discrete-event system |
| UAV | unmanned aerial vehicle |
| UAS | unmanned aerial system |
| V-REP | virtual robotics experimentation platform |

# Chapter 1

# Introduction

## 1.1 Why UAVs?

Unmanned aerial vehicles (UAVs) are gaining popularity and traction in civil, commercial, and military applications (Nex & Remondino, 2014; Saggiani & Teodorani, 2004; Valavanis & Vachtsevanos, 2014). They have been applied in search and rescue, information support during fire containment, the inspection of infrastructure, e.g. power transmission lines (Whitworth, Duller, Jones, & Earp, 2001), and environmental monitoring (Elfes, Bueno, Bergerman, & Ramos, 1998). In military applications, the absence of people on board enables higher g-force combat maneuvering than a human pilot can tolerate. Powerful onboard processors and long-range radio extend their uses far beyond visual line of sight operations when permitted.

An unmanned aerial system (UAS) is composed of a UAV, communications link and a Ground Control Station (GCS). A UAS operation consists of three phases: take-off, mission performance (flight), and landing. Manual UAV operations are done by an external pilot (EP), who remotely controls the aircraft; in an automated flight, an AVO (air vehicle operator) monitors the Ground

Control Station. The EP can obtain information about the state of aircraft with his eyes – radio control (RC), or using an onboard camera - first-person view (FPV). Both manual methods require an experienced pilot due to the expert situational awareness of the airspace required around the UAV (Stevenson, O'Young, & Rolland, 2015).

UAV sizes can be classified as: micro aerial vehicles (MAVs), small or typically under 35 kg, tactical, medium-altitude long-endurance, and high-altitude long-endurance UAVs (Nonami, Kendoul, Suzuki, Wang, & Nakazawa, 2010). The current research is related to small UAV operations and is motivated by their potential civil and commercial applications in different fields, e.g. for wildlife research, environmental monitoring, parcel delivery, police operations and building inspections. However, maximum takeoff weight and power restrictions prevent the use of existing sensor and control technologies to meet all the operational requirements. Novel and "lean" solutions are needed.

UAV types can be classified within four categories: fixed-wing, rotary-wing, airship (balloons), and flapping-wing (Nonami et al., 2010). The most widely used are fixed-wing and rotary-wing UAVs. Fixed-wing UAVs generate lift based on the aircraft's forward airspeed and the shape of the wings. Rotary wing aircraft have wings which form a rotor mounted on a spinning shaft (Kendoul, 2012). This research is related to rotary UAVs, since they have many advantages over fixed-wing unmanned aircraft. These advantages are hovering, low-speed flight, and high maneuverability, which make rotary UAVs an ideal option for inspection and surveillance of industrial objects, icebergs, ice drift, and oil and gas rigs observation. When a UAV flight is performed over the sea or ocean, after the multi-rotor finishes its mission, it is often landed on the water (Bird, 2013) and then retrieved from the water by a human. To remove the risk to the person tasked with retrieving the UAV, the multi-rotor should land automatically on the ship deck. However, even a small error during landing on the ship can lead to damaging of the UAV or the ship's gear or the UAV ditching into the water. The reliable landing of a multi-rotor UAV on a moving platform, such as a ship, is important and is the focus of this thesis.

During landing on a moving vessel the multi-rotor should have a predefined glide-slope and has to align with the moving platform in both the horizontal and vertical directions. Approaching the moving ship at a certain angle is important, since vertical landing of the multi-rotor can result in it hitting the ship's superstructure (e.g. masts and wires). However, autopilots with a non-differential Global Positioning System (GPS) do not provide the desired position accuracy and landing with an accurate glide-slope. Differential GPS, or the addition of acoustic, laser, or radar altimeters is expensive and does not fit within the Size Weight and Power (SWaP) requirements of a small multi-rotor machine. In this research, the SWaP restrictions are determined by the regulations for operating a UAV under a 2 kg maximum take-off weight under exemption rules (Transport Canada, 2014).

In addition to its high cost, differential GPS would have difficulties when used for a UAV landing on a moving platform, since it uses a fixed ground-based reference station to improve a UAV's position estimation. Machine vision could become an alternative to differential GPS in terms of accuracy, because cameras are small, light-weight, and inexpensive compared to other sensors such as LIDAR/LADAR (light/laser detection and ranging). Therefore, a vision-based method is proposed to assist the landing of a multi-rotor on a moving platform.

## 1.2 Literature survey

A “high-level” survey of the literature on visual servoing and camera-guided navigation is given here to give context to the overall thesis. The detailed “low-level” literature surveys will be given in subsequent chapters specific to the technology used in the chapter.

Cameras are widely used for object detection, tracking, and mapping (Chamberlain, Scherer, & Singh, 2011; Kamate & Yilmazer, 2015). One of the most important applications is vision-aided navigation (Silveira, Carvalho, Madrid, Bueno, & Rives, 2001) in order to replace a human pilot (the RC pilot “sees” and makes decisions; camera captures images, and an onboard computer sends commands). They are used for the vision-based control of micro aerial vehicles (MAVs, Ruffier & Franceschini, 2004); under-actuated rigid body systems control applied to helicopter stabilization (Hamel & Mahony, 2002); and UAS navigation and obstacle avoidance (He, Iyer, & Chandler, 2006). Several existing vision-based methods such as visual model feature tracking and line following with complementary sensors are suggested by Mondragon et al. (2007)

and Silveira et al. (2003). Teuliere et al. (2011) described a method to track a moving target through color-based detection. The authors used a multi-part object representation to solve a known problem of color-based detection – the loss of spatial information. Ding et al. (2006) and Lee et al. (2012) described real-time object tracking algorithms from the air based on feature extraction.  The drawbacks of feature-based algorithms are that they are sensitive to noise, changes in the object's appearance and blur. They are also computationally intensive and demand powerful onboard processors; this leads to increasing the cost and weight of the final system.

The combination of image processing methods and control algorithms form a number of approaches that have been developed to solve the problem of automated landing. Current methods use lines as visual features that are detected by an image processing algorithm. Bourquardez and Chaumette (2007) developed a method to detect a runway location from image data (two border lines and the centerline). Meng et al. (2006) proposed a template matching algorithm to detect and track the runway. Dusha et al. (2007) used morphological image processing and Hough Transforms to detect the horizon and estimate the attitude of a UAV. Liu and Li (2012) used a Line Segment Detector (LSD) to detect the main line features. The obvious drawback of these methods is that they cannot be applied to landing in an environment that does not have a runway or line markers (e.g. landing on a moving boat).

Fitzgerald et al. (2005) consider forced landing site selection. This approach uses canny edge detection, followed by a line-expansion algorithm.

However, their method considers general landing sites such as paddocks and does not consider precise landing on a moving platform. Sanches-Lopez et al. (2014) developed a vision-based algorithm to control a UAV during auto-landing on a moving ship. Their approach uses a downward looking camera to estimate the ship's movement, and applies a Kalman filter together with the vision-based method to ensure the accuracy of estimates. Zhou and Zhou (2015) developed a method to predict a ship's movement using a six degrees of freedom ship module. The authors recommend their method to assist in the take-off and landing of a ship-borne helicopter; however, they did not consider automatic landing on a moving ship. Jabbari et al. (2014) presented an image-based visual-servoing scheme to track a moving target based on perspective image moments of a flat object. Their approach uses a linear observer to solve the problems associated with depth uncertainty. Continuation of their work (Jabbari & Bolandi, 2014) improves the method for the design of robust translational and yaw controllers. Olivares Mendez et al. (2014) developed a method for the fuzzy-logic control of a multi-rotor machine during landing on both a static and a moving platform. However, these approaches used a downward looking camera and are not applicable for a multi-rotor descent.

## 1.3 Problems and Solutions

The literature review shows that the majority of vision-based methods for multi-rotor landing use downward-looking cameras, which makes them not applicable for multi-rotor descent (joining the glide-slope while aligning horizontally with a moving target). Approaching the moving ship at a certain angle is important, since vertical landing of the multi-rotor can result in it hitting the ship's superstructure (e.g. masts and wires). Joining the glide-slope has another advantage: the same approach to landing could be further extended to land the fixed-wing UAV after modifying the control scheme to account for inner-loop (pitch/roll) and outer-loop (bearing/altitude) interactions within the autopilot.

The most popular image processing techniques are detecting lines, colors, or recognizing specific markers using feature descriptors. Line markers are not always available to use while landing on the moving ship; color-based methods do not provide spatial information about the object; and feature-based detection is computationally complex, and requires powerful onboard processors.

The drawback of any vision-based method is that wind gusts or controller detuning can lead to undesirable situations such as losing markers from the field of view.  Maintaining large alignment error when the controller is not tuned to wind can lead to hitting the ship's superstructure and damaging the UAV or the ship's gear. Vision-based methods are also sensitive to changes in lighting conditions, which affect object detection. The problems mentioned above can lead to exceptions when the next control input cannot be calculated by the visual servoing algorithm, because the visual marker is missing from the image or cannot be recognized. Since autopilots require a minimum data rate coming from the onboard processor to control the UAV, undefined control input leads to the multirotor crashing into the water.

**Problem statement.** Develop a vision-based automatic landing system that ensures descent with an accurate glide-slope, such that the UAV comes to the desired point of touchdown within +/- 1.5 m or better in both horizontal and vertical directions, and is robust against wind-gust perturbations, changes in lighting conditions, and controller detuning.

The desired accuracy of the auto-landing system, +/- 1.5 m, is stated under the assumption of landing on a representative moving vessel, e.g. an icebreaker without a helideck, with a lack of free space on board the ship, and the possibility of hitting other superstructures e.g. masts or wires.

**Technical challenges:**

1) Complexity of the image processing algorithm.

2) Possible rolling, pitching, and yawing of the moving platform and the UAV.

3) Precision descent within a pre-specified glide-slope and alignment horizontally with the moving vessel.

4) The control system should handle the exceptions, e.g. losing visual markers from the field of view, changes in lighting conditions, the possibility of hitting the ship's superstructure, and battery life limitations.

**Stepped approach to overcoming the technical challenges:**

The vision-based approach to solving the stated problems is depicted in Fig. 1.1; it consists of the paradigm of methods developed in 3 areas: "Sense" ("interpret" information from the camera image), "Control" (send servo-commands to autopilot), and "Discrete-event supervision" (handle exceptions).

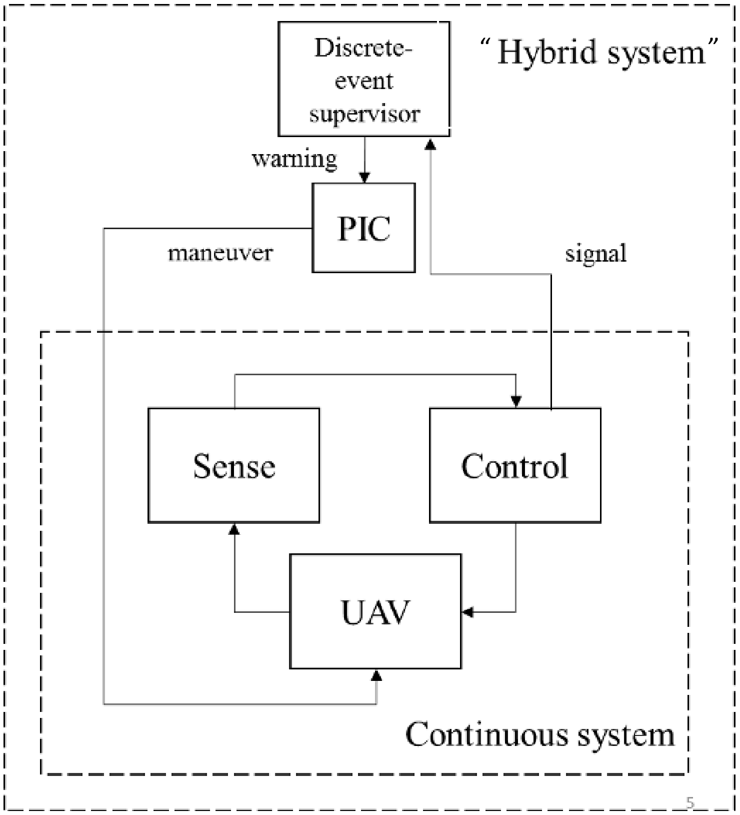


Figure 1.1 – "Hybrid" system for a UAV automated landing

Technical Challenge 1 is solved in the part "Sense" (Fig. 1.1) by applying color-based detection to the geometric pattern. This way, the algorithm can provide all the necessary information while having low computational complexity compared to feature-based methods. Technical Challenge 2 is solved in the part "Sense" by projecting the camera image to the virtual image plane using known roll/pitch angles of the moving vessel and the UAV at a certain moment. Roll/pitch angles of the moving vessel can be obtained from the Inertial Measurement Unit (IMU) located on the moving ship and sent from the Ground

Control Station to the UAV. Related work and solutions to overcome these technical challenges are described in Chapter 2.

Technical Challenge 3 is solved in the part “Control” (Fig. 1.1) by using a front-facing camera instead of a down-facing camera, while applying an image-based visual servoing scheme that is based on the correction angles. This control strategy ensures the UAV joining the glide-slope while being aligned with the moving target in the horizontal direction. Related work and solutions to overcome this technical challenge are described in Chapter 3.

Technical Challenge 4 arises in the following scenarios:

1) “Can’t see”: The pattern is not in the FoV of the camera.

2) “Can’t detect”: The change in lighting conditions affects the detection threshold.

3) “Can’t follow”: If the inner controller is not tuned to the wind/gust conditions, the UAV will maintain alignment error or lose targets from the FoV.

It is assumed that the pattern is within the FoV of the camera while starting the auto-landing mission, since according to the developed method, the UAV uses a GPS for initial alignment.  Exceptions 2 and 3 of Technical Challenge 4 are solved by adding a “Discrete-event supervisor” to the continuous control system (Fig. 1.1), which is described in Chapter 4. The discrete-event supervisor sends timely warnings to the PIC (pilot-in-command) to expedite certain control actions. Such a system is decoupled in a sense that the pilot has absolute authority in performing desirable maneuvers. Since the states of the supervisor are derived from the continuous vision-based control system, and synchronized with its timing, the overall landing system is considered to be hybrid-like.

A human pilot-in-command is overseeing all the landing phases, and is able to abort the landing mission if an exception occurs. The PIC is the human operator who is in charge of the overall safety of the aircraft. Different operators could assume the PIC role depending on the phases of the mission. The AVO usually assumes the PIC role during automatic operations, and the EP could be the PIC during manual operations, e.g. take-off and final touch-down during landing.  In fact, the AVO was the PIC in the “far field” phase, and the EP was the PIC in the “near field” phase during the landing experiments as presented in Chapters 4 and 5.

Because of the PIC oversight at all phases of the landing, the proposed landing process is deemed to be “automatic” and not “autonomous”. The term “automatic” means autonomy is permitted, e.g. way-point following, as long as there is a PIC override. Transport Canada regulations, at the time of this thesis (Transport Canada, 2014), do not permit (fully) autonomous systems because the technology is new and unproven. Limited autonomy without PIC override while satisfying the required safety level may be permitted as the industry matures.

The novelty and the main contribution of this thesis is a creative combination of incremental advances in three fields: image processing, controls, and discrete-event supervisory, as applied to a UAV automatic landing.

The field-specific advances are summarized:

**Image processing:**

- The image processing method was developed to estimate the DPT location on the image. This method uses color-based detection applied to a geometric pattern. It accounts for the rolling, pitching, and yawing of the UAV and the platform and has lower complexity compared to other known methods.

**Control:**

- An IBVS scheme to control a UAV during landing was developed. This approach does not require estimation of the 3D UAV position or calculation of the distance to the objects.

**Discrete-event supervisory control:**

- The landing phase was modeled as a timed discrete-event system (TDES).
- An optimal discrete-event supervisor was synthesized; the applied control mechanism is based on expediting certain control actions to avoid an unrecoverable error.
- The discrete-event layer on top of the continuous controller forms a hybrid-like system. This system is robust to wind/gust, controller detuning, and changes in lighting conditions.

**Validation:**

- The simulator for a vision-based multi-rotor landing on a static and a moving platform was developed. This simulator models wind and gust perturbations, changes in lighting conditions, and visibility constraints.
- The prototype vision-based landing system was developed; it is small (100 x 100 mm), light-weight (100 g), and highly accurate (25 cm from the DPT).

The auto-landing method was tested indoors and outdoors for landings on both static and moving platforms.

**Publications:**

The list of publications with respect to solved technical challenges is given in Table 1.1. The results of this thesis were presented at two conferences (Borshchova, 2014; Borshchova, 2015), and documented in three journal papers (Borshchova and O'Young, 2017 a; Borshchova and O'Young, 2017 b; Borshchova and O'Young, 2017 c). This work was presented at the Student Paper Competition Unmanned Systems Canada, 2014, where it was named one of the top three student papers across Canada.

Table 1.1 - List of publications

| | Borshchova, 2014 | Borshchova, 2015 | Borshchova and O'Young, 2017 a | Borshchova and O'Young, 2017 b | Borshchova and O'Young, 2017 c |
|---|---|---|---|---|---|
| Technical Challenge 1 | ✓ | ✓ | ✓ | ✓ | |
| Technical Challenge 2 | | | | | ✓ |
| Technical Challenge 3 | ✓ | ✓ | ✓ | ✓ | |
| Technical Challenge 4 | | | | | ✓ |

A full design cycle has been completed: starting from theoretical design, to modeling and simulation (M&S), then to implementation over two platforms and ending by validating the M&S predictions through first indoor and then outdoor field trials.

## 1.4 Thesis outline

The approach to a multi-rotor UAV landing on a moving platform is described in Chapter 2. This approach is based on using a single front-facing camera to detect the pattern of red objects on the image. This detailed description of the image processing method that was developed to detect the pattern and determine the desired point of touchdown/aiming point from the image data is presented.

It is shown in Chapter 2 that the unique pattern structure, color-based detection, edge and center extraction, correction for perspective transformation, and affine transformation are combined to form a novel technique that can be applied to process an image taken from an onboard camera to land the UAV. Finally, this image processing method accounts for a UAV and a platform rolling, pitching, and yawing using a projection of the image to the virtual plane.

In Chapter 3, an image-based visual servoing approach to control the UAV during automatic landing is described. According to this approach, the controller will calculate the reference velocities that will be fed to the autopilot based on the determined correction angles. The suggested approach relies on the simplification that the inner loop of the flight controller (velocity loop) is tuned to a certain configurational setup. The suggested control strategy is analyzed from both kinematic and dynamic points of view considering the delay in the control system due to image processing. Notional analysis of the control system is provided, which demonstrates that tracking without a steady-state error and closed-loop stability is viable.

The timed discrete-event layer on top of the continuous controller is modeled in Chapter 4. The discrete-event system (DES) is formally described as a finite automaton represented by a 5-tuple in which the state evolution depends on the occurrence of discrete events over time. The optimal supervisor synthesis is done automatically using timed discrete-event software. The suggested supervisor makes “high-level” decisions and gives instructions to the EP and the AVO to abort the mission when an exception occurs.

The practical aspect of this research is highlighted in Chapter 5. The constraints of the simulation platform and the details of the hardware components are given. The simulation experiments were conducted with both static and moving platforms, using a simulated sensor on board the multi-rotor model. Real-time flight tests were performed both indoors and outdoors for a UAV landing on static and moving platforms. Quantitative and qualitative performance analysis for every scenario are covered in this chapter.

In Chapter 6, a summary of the thesis and discussion on potential directions for future work are presented.

Chapter 2

# Image processing

## 2.1 Related work

The most common methods for object detection are color-based detection and feature-based detection. Using feature-based detection, even a single object can be used to determine a pose. These feature-based methods can be classified as distribution-based descriptors, spatial-frequency techniques, and differential descriptors (Mikolajczyk & Schmid, 2005).

Distribution-based descriptors represent different appearance or shape characteristics. Lowe (2004) developed an algorithm for matching individual features to a database of features using a fast nearest-neighbor algorithm. His method is robust to object rotation and scale. Mondragon et al. (2007) developed an approach to track an object using salient points; their algorithm is robust to noise and helicopter vibration. Frequency methods usually apply Fourier transform, Gabor transform, and wavelets to analyze the texture of the image. Differential descriptors compute derivatives to approximate the neighbourhood of the point. Freeman and Adelson (1991) give an example of designing steering filters based on Gaussian derivatives. The authors applied their filters to determine orientation and detect contours. Se et al. (2001) apply feature tracking to the localization of robots using visual landmarks. The authors use a Kalman filter to track the 3D features and scale-invariant feature transform (SIFT) is used to build a 3D map. The drawback of feature-based detection is that it is computationally intensive and may require large databases; moreover, it is sensitive to changes in object appearance, blur, etc.

Color-based detection is known to be less computationally complex compared to feature-based detection. Azrad et al. (2010) presented an approach to track an object from MAV using color-based detection. The authors use integral-image and color probability distribution to detect the target. Watanabe et al. (2010) developed a tracking method based on the assumption that the target's gray-level is significantly higher than the background. Perez et al. (2002) suggested a method to use the Monte Carlo technique and multi-part color modeling. The main drawback of color-based detection is the loss of spatial information, which leads to difficulties with tracking other movements besides the translational motion.

This chapter pertains to solving the issue of the computational complexity of image processing algorithms (Technical Challenge 1, described in Chapter 1.3) by applying color-based detection to a geometric pattern positioned on the moving platform. This way, the algorithm complexity is lower than that of feature tracking methods; the necessary information needed for alignment is obtained from the geometric position of targets on the image.

The problem of the rolling, pitching, and yawing of the moving platform and the UAV affecting the camera image (Technical Challenge 2, described in

Chapter 1.3) is solved in this chapter. A novel approach to camera-based landing is presented; it highlights the "Sense" part of the overall automatic landing system as a **bolded** box in Fig. 2.1.

This method was documented in (Borshchova and O'Young, 2017 a; Borshchova and O'Young, 2017 b; Borshchova and O'Young, 2017 c) and presented at conferences (Borshchova, 2014; Borshchova, 2015).

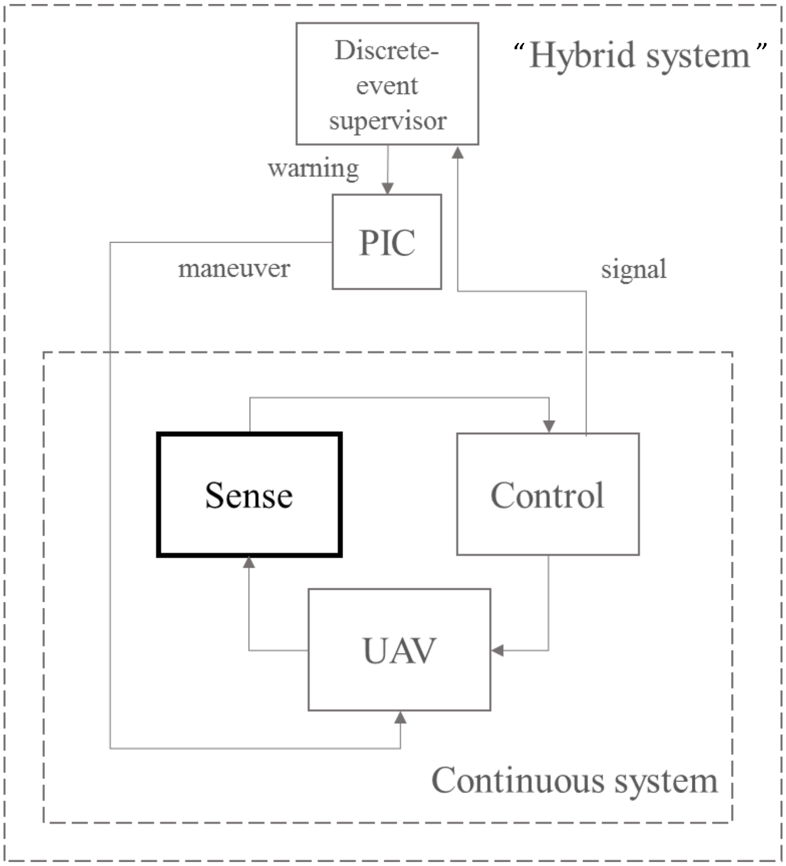


Figure 2.1 – Overall approach to the automatic landing system

## 2.2 Image processing approach

Landing a multirotor on a moving target can be divided in 3 stages:

1) Initial positioning using a GPS.

2) Descent – when the multirotor joins the glide-slope to hit the desired point of touchdown.

3) Touchdown itself.

In this work, the part of landing between descent and touchdown is called the “alignment stage”. In this stage, a multirotor lines up horizontally and vertically with the pattern of red objects located on a moving platform. Touchdown, after the alignment stage, is not considered in this work. The multirotor will hover above and in the vicinity of, e.g. within 1.5 m or better, the DPT. It will then be landed by the external pilot as prompted by the audio alert described in Chapter 4 because of operational safety requirements.

The auto-landing system uses five red visual markers positioned on a moving vessel: four of the markers are located in a square, and a fifth marker is positioned at the end of the landing area in such a way that together with the other two markers in a square it makes up an equilateral triangle. The intersection of the square created by the first four markers is the desired point of touchdown (DPT, Fig. 2.2).

At the early stages of the flight when all five targets are in the field of view of the camera, the next control input is calculated using a perspective transformation technique. In this case, the aiming point is DPT.

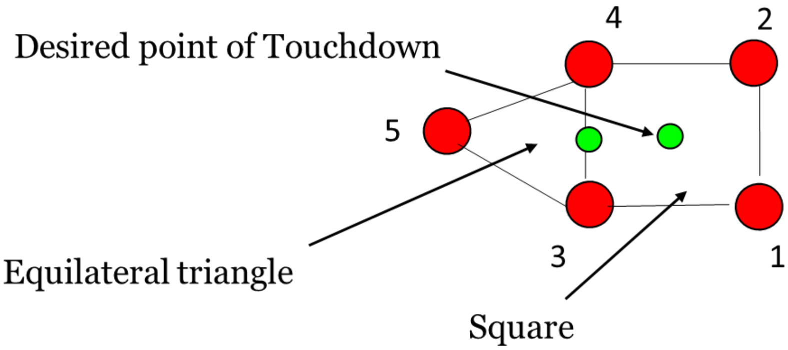


Figure 2.2 - The pattern of red targets on board a moving platform

When getting close to the targets, the first one or two markers will be lost from the FoV; the inner controller will use an equilateral triangle and affine transformation to align with the pattern. The aiming point in the case of three or four targets is the middle of the base of the equilateral triangle (Fig. 2.3).

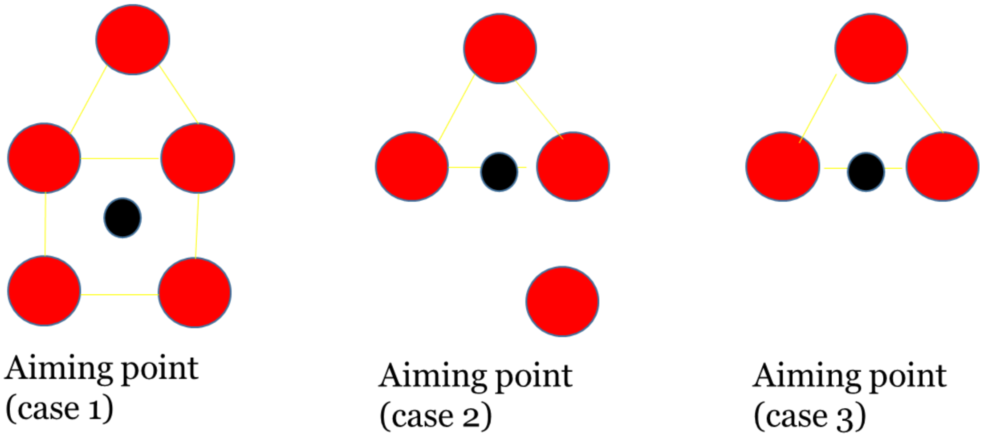


Figure 2.3 – Aiming point depending on the number of detected red targets

The overall image processing algorithm is depicted in Fig. 2.4. The developed software was written in C++ and used OpenCV 2.4.12.1 library (Baggio, 2012; Bradski & Kaehler, 2008; Laganière, 2014).

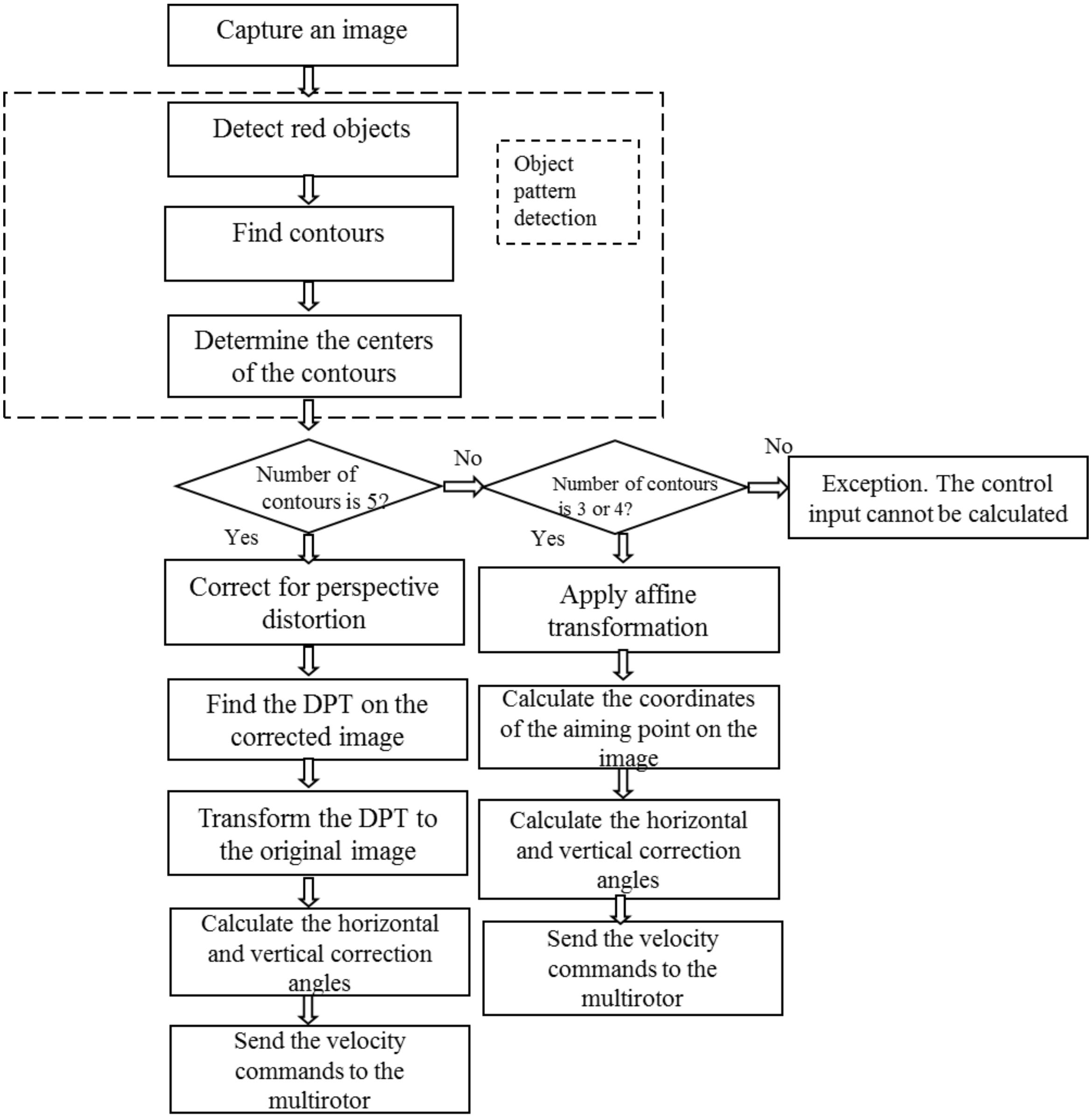

Capture an image
Detect red objects
Object pattern detection
Find contours
Determine the centers of the contours
Number of contours is 5?
No
Yes
Number of contours is 3 or 4?
No
Yes
Exception. The control input cannot be calculated
Correct for perspective distortion
Find the DPT on the corrected image
Transform the DPT to the original image
Calculate the horizontal and vertical correction angles
Send the velocity commands to the multirotor
Apply affine transformation
Calculate the coordinates of the aiming point on the image
Calculate the horizontal and vertical correction angles
Send the velocity commands to the multirotor

Figure 2.4 – Proposed image processing algorithm

The developed image processing algorithm consists of color-based detection, edge and contour detection, and perspective/affine transformation to estimate the aiming point on the image depending on the number of detected red objects.

The 5-point pattern used in the developed auto-landing approach has many advantages over other feasible patterns:

1) The possibility of extension of the developed approach to land a UAV in a GPS-denied environment. When having no GPS to perform initial alignment, it is difficult to determine the correspondence of the points in the pattern due to the distortion of the image. The asymmetric structure of the pattern could become a solution for the robust and computationally effective object pattern recognition in case of a large initial alignment error.

2) The possibility to calculate the complete desired velocity vector (3 rotational and 3 translational velocities) without singularities. When applying to land a fully actuated multi-rotor (the number of degrees of freedom is equal to the number of the controllable variables), or a fixed-wing UAV, the 5-point pattern can be used to calculate the complete desired commanded velocity vector without singularities, while 3 or 4 coplanar points that form a pattern could lead to ambiguities.

3) For the developed image processing approach, the chosen pattern uses the minimum number of points needed to align with the moving platform. Since the developed image processing algorithm first uses perspective transformation, 4 points in a square is the minimum number of the points needed to estimate the

DPT location on the image. After losing 2 targets from the FoV and applying affine transformation, 3 points is the minimum number of points needed to determine the aiming point.

Using a pattern with more points could be beneficial, since in a case when several targets are lost from the FoV such patterns could provide additional positioning reference. This might not be possible because of space limitations, e.g. the maximum beam width at the stern. The design tradeoffs of adding more points to the pattern could be investigated in future works.

### 2.2.1 Color-based detection

To maximize the signal-to-noise ratio (SNR) of detecting visual targets on the background like a ship deck (grey or brown), runway (grey), and grass field (green) ( $SNR = \frac{\mu_{t\arg}}{\sigma_{backgr}}$ , where $\mu_{t\arg}$- average signal value of target, $\sigma_{backgr}$- standard deviation of a background), the targets were chosen to be a distinct red color. The developed algorithm detects a certain shade of "red" to distinguish between the target and the "false alarm". When landing on a vessel with a different color background, the various shades/colors of the targets in the pattern could be used to serve the purpose of effective and robust object pattern detection.

Color-based detection of the red targets was implemented in the following way. In RGB representation, a pixel at image coordinate $(x, y)$ has three integers, $I_g, I_r, I_b$, varying from 0 to 255, respectively. The pixels that belong to the red visual markers vary depending on the lighting conditions, noise, etc. According to the suggested method, a pixel is determined to be red, if:

$$I_g(x,y)+s<I_r(x,y) \text{ and } I_b(x,y)+j<I_r(x,y)\,; \qquad (2.1)$$

where $s$, $j$ - are thresholds that are found experimentally for certain lighting conditions.

As shown by Vinukonda (2011), the total complexity of the SIFT algorithm for an image with dimensions $N \times M$ is $\Theta(N \times M \times s(w^2 + \alpha - \alpha \times \beta + \frac{x^2}{s}(\alpha \times \beta + \gamma) + 3))$, where $\alpha, \beta, \gamma$ - feature fractions, $w$ - Gaussian window, $2x \times 2x$ - neighborhood of a point, and $s \geq 1$. In comparison with the high computational complexity of SIFT, color-based detection has complexity $\Theta(N \times M)$. This allows the suggested approach to be implemented on less powerful processors, thereby reducing the cost of the overall system.

### 2.2.2 Edge and center extraction

After red pixels are detected on the image, the edges of the red objects are determined using a Canny edge detector. The Canny edge detection algorithm (Canny, 1986) can filter noise, but keep the same level of valid information, while ensuring precise positioning on the image.

After the edge detection, the morphology operation of dilation is applied (Gonzalez & Woods, 2007). Assuming $A$ as a binary image, and $B$ as a structural element, the operation of dilation can be expressed by:

$$A \otimes B = \bigcup_{b \in B} A_b \, . \tag{2.2}$$

Dilation reduces the possibility of getting open contours which could create problems in further contour detection.

The image processing algorithm extracts the contours of each of the red objects, using the method of edge approximation that compresses horizontal, vertical, and diagonal segments and leaves only their end points. The software calculates moments $m_i$ of each contour, and obtains the centers of each object using Equation 2.3:

$$x_{c_i} = \frac{\mathrm{m}_{\mathrm{i}10}}{\mathrm{m}_{\mathrm{i}00}}, \; y_{c_i} = \frac{\mathrm{m}_{\mathrm{i}01}}{\mathrm{m}_{\mathrm{i}00}}; \tag{2.3}$$

where $m_{i\,pq} = \int_{\eta_1}^{\eta_2} \int_{\lambda_1}^{\lambda_2} \mathrm{x}^{\mathrm{p}} \mathrm{y}^{\mathrm{q}} \mathrm{F}(\mathrm{x,y}) dxdy$ .

After the centers of each contour are calculated, the points are sorted from the top to the bottom of the image. Since the UAV uses a GPS for the initial alignment, the fifth point is always the one on the top of the image, while the other four points that make up a square are located lower on the image. Having separated the fifth point from the rest of the points, and going clockwise from the top to the bottom of the image, the developed software distinguishes between the points. Such an approach does not require any feature tracking algorithms or methods to solve the correspondence between the points in a pattern, which significantly simplifies the computations.

### 2.2.3 Correction for perspective distortion

In a case when 5 targets are detected, the coordinates of the DPT on the image are estimated using the perspective transformation technique.

Given the image coordinates of four points that make up a square: $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$. Since the shape is known, the coordinates of these points after correction for perspective distortion are also known and are $(x'_1, y'_1), (x'_2, y'_2), (x'_3, y'_3), (x'_4, y'_4)$. The suggested method estimates the $3 \times 3$ perspective transformation matrix $T$ such that:

$$\begin{pmatrix} wx_i \\ wy_i \\ 1 \end{pmatrix} = T^{3x3} \begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & l \\ g & h & \mathrm{i} \end{pmatrix} \begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix}; \qquad (2.4)$$

where $(x'_i, y'_i)$ are corrected coordinates of the point on image, $(x_i, y_i)$ - actual coordinates of the point on the image, index $i = 1,2,3,4$ indicates the numbering of the points that make up a square, and $w$ - non-zero scalar which is needed to express the transformation in homogenous coordinates.

After solving $x_i, y_i$ from Equation 2.4: $x_i = \dfrac{ax'_i + by'_i + c}{gx'_i + hy'_i + \mathrm{i}}$ and $y_i = \dfrac{dx'_i + ey'_i + f}{gx'_i + hy'_i + \mathrm{i}}$.

This gives Equation 2.5:

$$\begin{cases} ax'_i + by'_i + c - gx_i x'_i - hx_i y'_i - \mathrm{i}x_i = 0; \\ dx'_i + ey'_i + f - gy_i x'_i - hy_i y'_i - \mathrm{i}y_i = 0. \end{cases} \tag{2.5}$$

Generalizing for 4-point correspondence, Equation 2.5 can be written in matrix format as $A_i \cdot \mathrm{h} = 0$ for $i = 1..4$, where $\mathrm{h} = [a \quad b \quad c \quad d \quad e \quad f \quad g \quad h \quad \mathrm{i}]^T$ is a $9 \times 1$vector of unknown coefficients, and $A_i$ is a $2 \times 9$matrix of known coordinates for $i^{th}$ point given as:

$$A_i = \begin{bmatrix} x'_i & y'_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x_i y'_i & -x_i \\ 0 & 0 & 0 & x'_i & y'_i & 1 & -y_i x'_i & -y'_i y_i & -y_i \end{bmatrix}. \tag{2.6}$$

After inserting all 4 matrixes $A_i$ into one equation (Geetha & Murali, 2013):

$$\begin{pmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{pmatrix} \cdot \mathrm{h} = 0\,. \tag{2.7}$$

The solution for four points is obtained using Singular Value Decomposition (SVD), where the solution is given as $\mathrm{h}_{1-9} = (a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}, a_{31}, a_{32}, a_{33})^T$ . Since there are 8 linear equations and 9 unknowns, it was necessary to add a constraint $\mathrm{a}_{33} = 1$.

After calculating the transformation matrix, the algorithm extracts the coordinates of DPT (Point 6) on the corrected image, and projects it back to the original image, as is shown in Fig. 2.5:

$$\begin{pmatrix} wx_6 \\ wy_6 \\ 1 \end{pmatrix} = T^{3x3} \begin{pmatrix} x'_6 \\ y'_6 \\ 1 \end{pmatrix}; \tag{2.8}$$

where $(x_6', y_6')$ are corrected coordinates of the DPT on image, $(x_6, y_6)$ - estimated coordinates of the DPT on the image

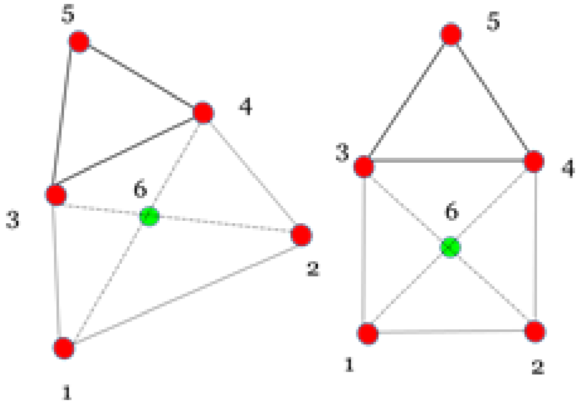


Figure 2.5 - Correction for perspective distortion

### 2.2.4 Affine transformation

The current research considers losing targets from the field of view (a case when three or four targets are detected), which will happen as the UAV comes closer to the moving platform. In this scenario, alignment is completed using the remaining equilateral triangle after applying affine transformation. The aiming point in this case is the middle of the base of the equilateral triangle (Fig. 2.6).

Given the image coordinates of 3 points that make up an equilateral triangle: $(x_1, y_1), (x_2, y_2), (x_3, y_3)$, since the shape is known, the coordinates of these points after correction for affine transformation are also known and are $(x_1', y_1'), (x_2', y_2'), (x_3', y_3')$. The affine transformation matrix $\mathrm{A}^{2x3}$ is calculated based on three-point correspondence so that:

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \mathrm{A}^{2x3} \begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix}; \tag{2.9}$$

where $(x_i, y_i)$ are coordinates of the $i^{th}$ point on the image, $(x_i', y_i')$ - the corrected coordinates of the $i^{th}$ point on the image, index $i = 1,2,3$ indicates the numbering of the points that make up an equilateral triangle.

After solving $x_i, y_i$ from Equation 2.9:

$$\begin{cases} ax_i' + by_i' + c = x_i; \\ dx_i' + ey_i' + f = y_i. \end{cases} \tag{2.10}$$

Generalizing for three-point correspondence, this equation can be written in matrix format as $C_i \cdot \mathrm{h}' = [x_i, y_i]$ for $i = 1,2,3$, where $\mathrm{h}' = [a \quad b \quad c \quad d \quad e \quad f]^T$ is a vector of unknown coefficients, and $C_i$ is a matrix of known coordinates given as

$$C_i = \begin{bmatrix} x_i' & y_i' & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_i' & y_i' & 1 \end{bmatrix}. \tag{2.11}$$

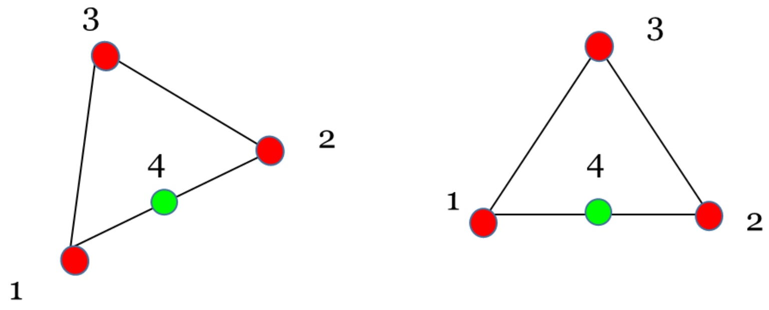


Figure 2.6 – Affine transformation

The solution is $\mathrm{h}' = [a_{11} \quad a_{12} \quad a_{13} \quad a_{21} \quad a_{22} \quad a_{23}]^T$ that allows to obtain the affine transformation matrix. The coordinates of the aiming point on the image are found by:

$$\begin{pmatrix} x_4 \\ y_4 \end{pmatrix} = \mathrm{A}^{2x3} \begin{pmatrix} x'_4 \\ y'_4 \\ 1 \end{pmatrix}; \tag{2.12}$$

where $(x'_4, y'_4)$ are the corrected coordinates of the aiming point on image, and $(x_4, y_4)$ – are the estimated coordinates of the aiming point on the image.

## 2.3 Improvements of the algorithm for a UAV and a platform rolling, pitching, and yawing

The UAV motion can be expressed in the inertial frame $I = \{O_i, X_i, Y_i, Z_i\}$, body frame $B = \{O_b, X_b, Y_b, Z_b\}$, and camera frame $C = \{O_c, X_c, Y_c, Z_c\}$ (Fig. 2.7). $O_c = O_b$ is assumed for the simplicity of modeling. Since the camera image is affected by the rolling, pitching, and yawing of the UAV and the platform, the "virtual camera" approach is used to account for these movements.

Compared to Jabbari et al. (2014) who used virtual projection of a downward looking camera image to eliminate couplings of roll-horizontal, and pitch-forward movement of the under-actuated UAV, the current research extends this approach to account for both a UAV and a target rolling and pitching which affect the front-facing camera image. While the virtual plane described in (H Jabbari Asl, Oriolo, & Bolandi, 2014) was parallel to the target plane, in this work the "virtual image plane" is perpendicular to the pattern/moving vessel plane (further referred to as "virtual orthogonal image plane"), but is rotated with a yaw angle with respect to the centerline of the pattern; the origin of the virtual camera coincides with the origin of the actual camera.

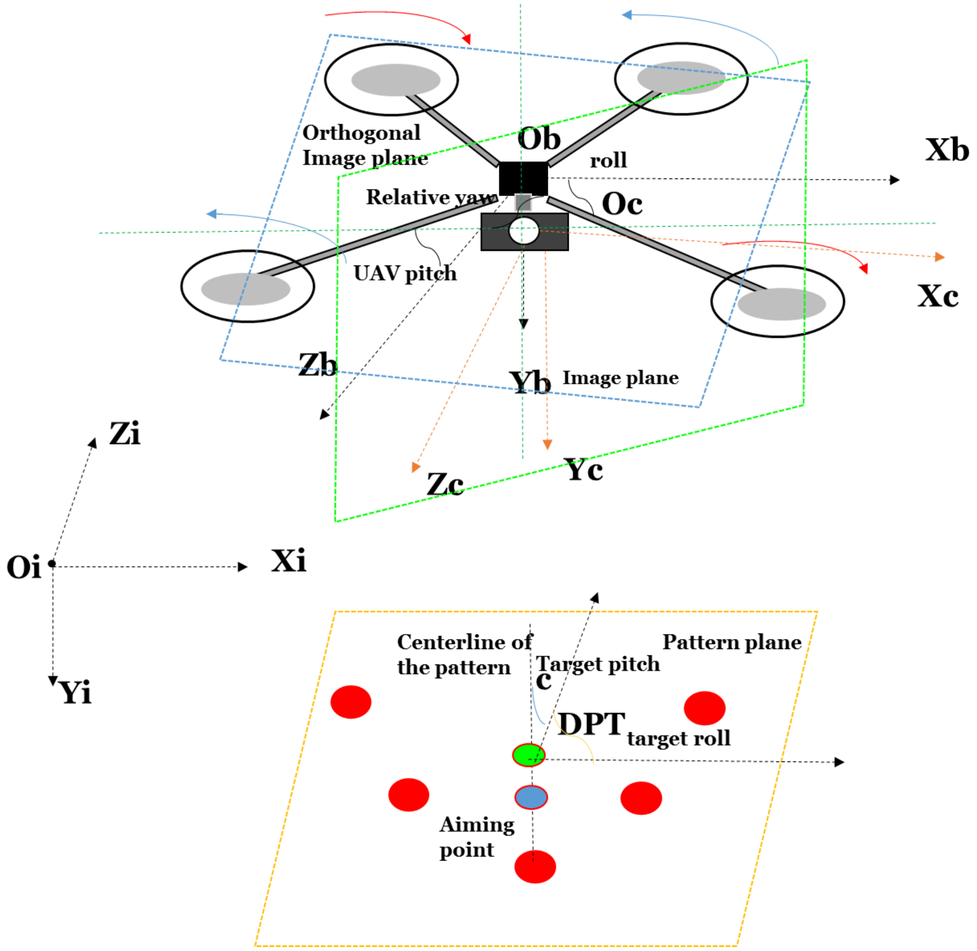


Figure 2.7 - UAV motion frames

Let point $P$ have coordinates $p_i(i_x, i_y, i_z)$ in the inertial frame. The coordinates of this point in the camera frame $p_c(c_x, c_y, c_z)$ can be determined by the Pinhole camera equation (H Jabbari Asl et al., 2014).

The coordinates of $P$ in the orthogonal camera frame are calculated using Equation 2.13:

$$\begin{bmatrix} o_x \\ o_y \\ o_z \end{bmatrix} = R_{\alpha,\beta} \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix}; \tag{2.13}$$

where $\alpha = \alpha_{ship} - \alpha_{UAV}, \beta = \beta_{ship} - \beta_{UAV}$ - roll/pitch of the ship relative to the UAV roll/pitch, respectively, $R_{\alpha,\beta} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix}$ - rotation matrix of relative roll and pitch. The roll/pitch of the UAV can be obtained from the Inertial Measurement Unit (IMU) of the autopilot, and the roll/pitch of the platform can be transmitted to the UAV from the Ground Control Station (GCS) located on the moving platform.

The relationship between the image coordinates $(x_c, y_c)$ of the point $P$ and coordinates of this point on the orthogonal image $(x_o, y_o)$ can be described as follows (Jabbari et al., 2014):

$$\begin{bmatrix} x_o \\ y_o \\ f \end{bmatrix} = kR_{\alpha,\beta} \begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix}; \tag{2.14}$$

where $f$ - focal length of camera, $x_c = fc_x / c_z, y_c = fc_x / c_z$ - image coordinates of the point on camera plane, $k = f / \left( \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} R_{\alpha,\beta} \begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix} \right)$, used to ensure re-projection with a focal length equal to $f$.

Fig. 2.8 demonstrates the example of transition from the camera frame to the virtual orthogonal camera frame for a simulated front-facing camera on board the quadcopter model taken in the V-REP scene (further described in Chapter 5). The images were taken during the simulation of realistic actual roll/pitch angles, which are small in a landing scenario on a moving boat and appear to be identical. However, the difference between the 2 images is evident from the black borders on the right side and the bottom corner of the virtual orthogonal image, as affected by the rotation.

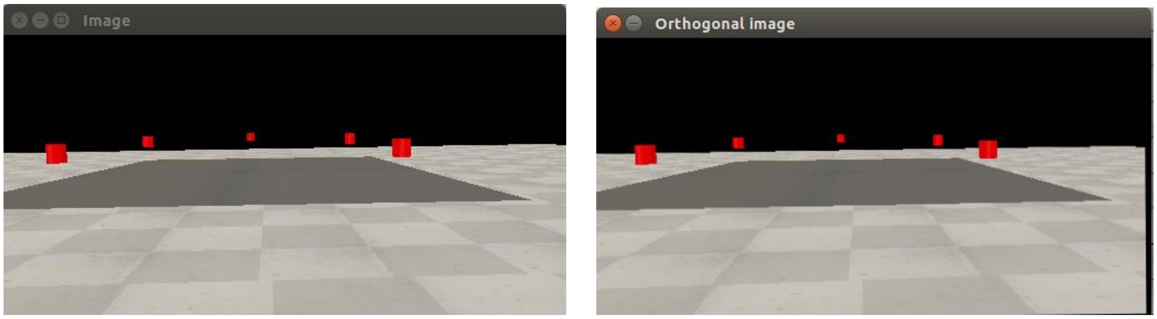


Figure 2.8 – Camera image and virtual orthogonal image for $\alpha = \alpha_{ship} - \alpha_{UAV} \approx -0.01rad, \beta = \beta_{ship} - \beta_{UAV} \approx -0.02rad$

After coming to the orthogonal frame from the actual camera frame, the image processing is done using the techniques described in Chapter 2.2. The orientation of the UAV relative to the pattern can be determined as:

$$\phi = \arccos(\frac{l_1 \cdot l_2}{|l_1| \times |l_2|}) ; \tag{2.15}$$

where $l_1$ - horizontal perpendicular that passes through DPT/aiming point, $l_2$ - horizontal centerline of the pattern that passes through DPT/aiming point (Fig. 2.9).

The flow chart of the software developed to account for the UAV and the target rolling, pitching, and yawing is given in Appendix A.

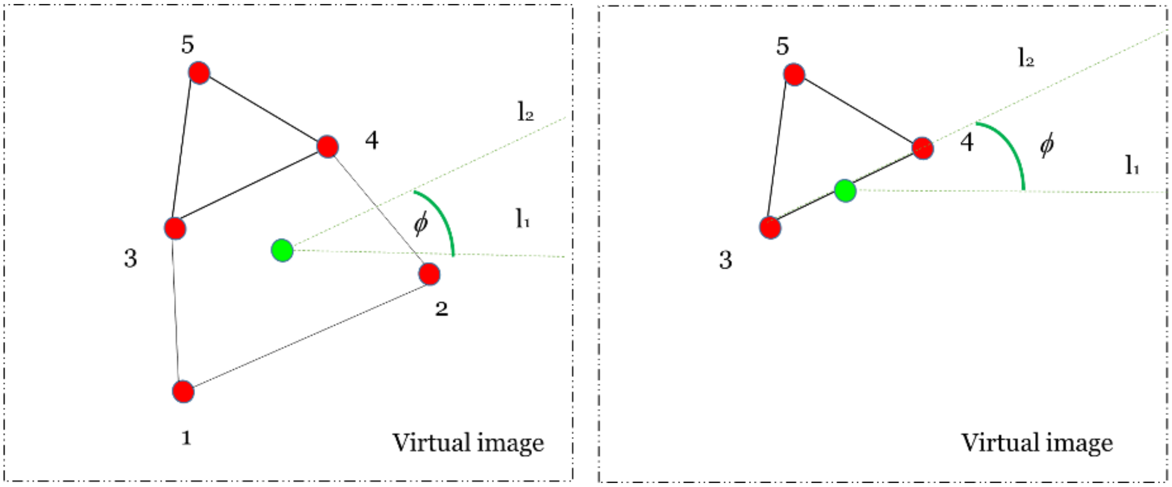


Figure 2.9 – Yaw correction:
a) case when 5 targets are in the FoV; b) case when 3 or 4 targets are in the FoV

## 2.4 Conclusion

The image processing approach for the multi-rotor landing on the moving target is described. The auto-landing system uses a GPS for the initial positioning, then switches to visual servoing. The developed image processing method accounts for the UAV and the platform rolling, pitching and yawing. The contribution of the proposed method is that a color-based detection, applied to detect a geometric pattern, makes the algorithm complexity lower than complexity of feature tracking methods. Compared to the complexity of the SIFT algorithm ($\Theta(N \times M \times s(w^2 + \alpha - \alpha \times \beta + \frac{x^2}{s}(\alpha \times \beta + \gamma) + 3))$), color-based detection applied to the geometric pattern has a complexity of $\Theta(N \times M)$ and is able to provide all the spatial information. This allows the suggested approach to be implemented on less powerful processors, thereby reducing the cost of the overall system.

# Chapter 3

# Visual servoing approach

## 3.1 Related work

Visual information extracted by the onboard camera and used for the control of robot motion, is known as visual servoing (Agin, 1979; Chaumette & Hutchinson, 2006; Peter I Corke & Hutchinson, 2000). There are two basic approaches to visual servoing: position-based visual servoing (PBVS), and image-based visual servoing (IBVS). In PBVS, image data is used to calculate the 3D pose of the object of interest with respect to the robot. Overall error is calculated in the 3D workspace and translated into control commands (Lippiello, Siciliano, & Villani, 2007; Shademan & Janabi-Sharifi, 2005; Thuilot, Martinet, Cordesses, & Gallice, 2002; Wilson, Hulls, & Bell, 1996).

Garcia Carrillo et al. (2011) developed a full-state control method for quadrotor stabilization and trajectory tracking; their approach demonstrated successful performance in quadrotor stabilization indoors. Park et al. (2012) developed a PBVS method using a concept of a 3D visible set. Their method was tested using a six-degrees-of-freedom robot-manipulator; it ensures robust global stability. The advantage of PBVS is that the control laws are designed in the

Cartesian frame, which infers reliable motion control. However, the control law depends on the parameters of the camera and is sensitive to calibration errors.

In IBVS, the control law is derived based on the error that is directly extracted from the location of features on the image (Corke & Hutchinson, 2001; Malis & Rives, 2003; Mariottini, Oriolo, & Prattichizzo, 2007; Park & Chung, 2003). Hashimoto et al. (1991) suggested an IBVS scheme for manipulators with cameras on their hands. The authors used time-variant state feedback to track a moving object. De Luca et al. (2008) developed an IBVS approach that uses a non-linear observer to estimate the depth. Their approach can be used to calculate the focal length of the camera; it is effective in scenarios where the robot is controlled in an unknown environment.

The advantage of IBVS is that it is relatively insensitive to calibration errors. Palmieri et al. (2012) demonstrated that for "dynamic look-and-move" systems, when the vision system provides an external input to the joint closed-loop control of the robot (such as a current auto-landing system), IBVS exhibits better behavior in terms of accuracy. Moreover, in such a case, PBVS gives redundant information, which implies performing unnecessary computations. As compared to PBVS, IBVS does not need as many calculations, since the IBVS method does not reconstruct a 3D pose. This is why to control a multi-rotor during landing on a moving platform, image-based visual servoing was applied.

This chapter pertains to solving the  problem of controlling the multi-rotor in order to correctly join the glide-slope, and maintain a near-zero alignment error on a horizontal plane (Technical Challenge 3 described in Chapter 1.3). The problem is solved by developing an IBVS scheme that is based on correction angles. This IBVS scheme forms the part “Control” of the overall automatic landing system as a **bolded** box in Fig. 3.1. This approach was documented in (Borshchova & O’Young, 2017 a; Borshchova & O’Young, 2017 b) and presented at (Borshchova, 2014; Borshchova, 2015).

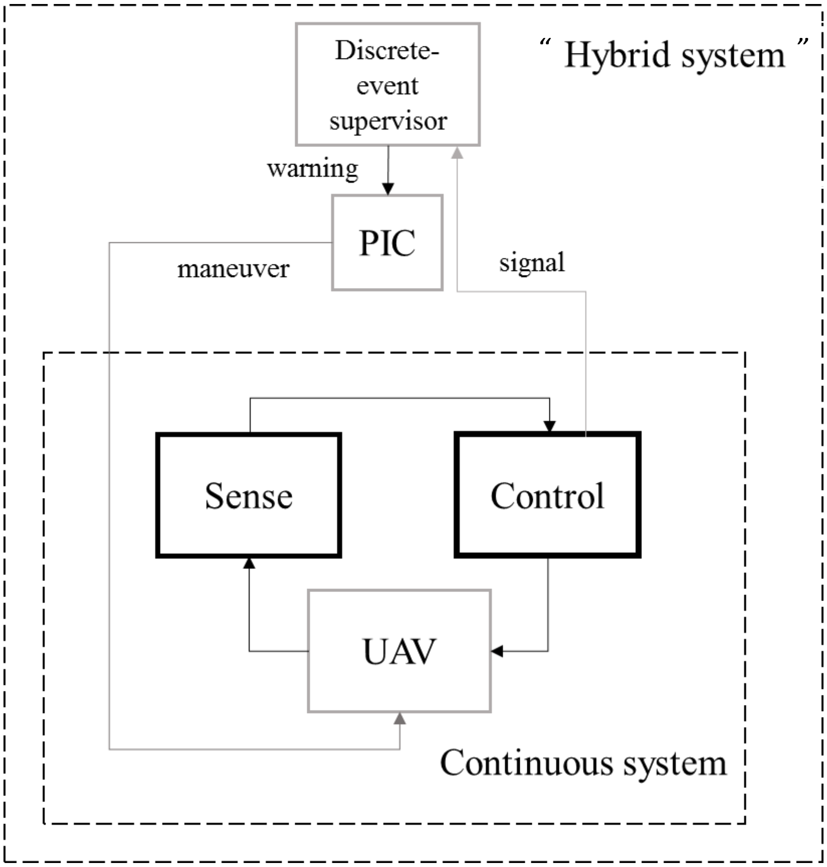


Figure 3.1 – Overall approach to the automatic  landing system

## 3.2 IBVS scheme

In IBVS, the control is designed as a function of image error $e$:

$$e_p = p - p_d; \tag{3.1}$$

where $p_d$ is the desired location of the image feature; $p$ - actual location of the feature on the image.

In the suggested control approach, an image feature is a DPT (aiming point). The desired location of the aiming point on the image is described by angles, since this allows a convenient transition from obtained image data to determine if the UAV is following the glide-slope and is aligned horizontally with the moving platform. In this case, the IBVS scheme can be written as follows:

$$\begin{cases} e_h = \theta - \theta_d, \\ e_v = \varphi - \varphi_d; \end{cases} \tag{3.2}$$

where $\theta_d$, $\varphi_d$ - desired horizontal angle (angle to align with the centerline of the pattern), desired vertical angle (angle to align to a glide-slope), respectively; $\theta, \varphi$ – actual angles.

The developed control approach automates the technique called the "circle of action" that is commonly used by military pilots to land their aircraft. According to this approach, the desired point of touchdown should always remain the center of the aircraft window (stationary point) to maintain the correct slopes. When automating such an approach to land the UAV, this infers that the aircraft should be controlled in such a way that the aiming point always remains the center of the image taken by the front-facing camera. When the aiming point is located at the center of the image, the horizontal and vertical angles, determined by the camera, are zeros. This means that desired alignment angles should always be zero to maintain correct slopes.

Prior to angle calculation, the camera was calibrated as described in (Heikkila & Silvén, 1997; Kannala, Heikkilä, & Brandt, 2008) using MATLAB Camera Calibration Toolbox. Using a Pinhole camera model, the horizontal correction angle is calculated by:

$$e_h = \theta_{cor} = \theta - \theta_d = \arctan\left(\frac{x'}{f}\right) - 0 = \arctan\left(\frac{x'}{f}\right) = \theta\,. \tag{3.3}$$

Vertical correction angle using the Pinhole camera model is given as:

$$e_v = \varphi_{cor} = \varphi - \varphi_d = \arctan\left(\frac{y'}{f}\right) - 0 = \arctan\left(\frac{y'}{f}\right) = \varphi\,; \tag{3.4}$$

where $(x', y')$ - coordinates of the DPT (aiming point) on the image plane;

$f$ – focal length of the camera (Fig. 3.2).

*Note* that for implementation when the target is rolling, pitching, and yawing, these correction angles are calculated from virtual image data as shown in the flow chart in Appendix A.

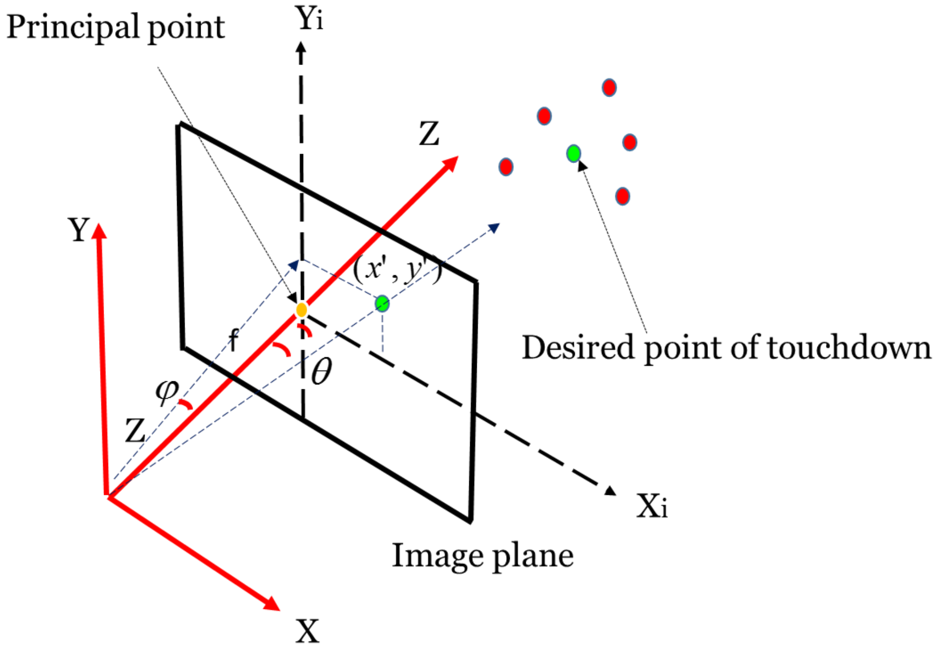


Figure 3.2 – Correlation between image coordinates and correction angles

`

Desired linear velocity vector $v$ for the UAV control is calculated based on the correction angles (Fig. 3.3).

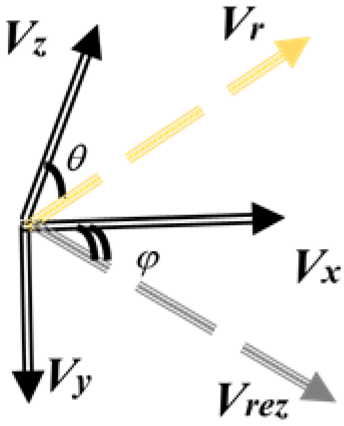


Figure 3.3 – Dependency of the desired velocity vector from correction angles

From Fig. 3.3, the desired velocities can be found using Equation 3.5:

$$\begin{cases} v_z = |v_{rez}| \cos\theta \cos\varphi, \\ v_x = |v_{rez}| \sin\theta \cos\varphi, \\ v_y = |v_{rez}| \sin\varphi; \end{cases} \tag{3.5}$$

where $|v_{rez}|$ is the speed of the multirotor, $v_z$ – the multi-rotor's forward velocity, $v_x$ – the multi-rotor's horizontal velocity, and $v_y$ – the multi-rotor's vertical velocity; all the velocities are represented in a body frame.

It is assumed for this method that the velocity of a moving platform is known and is obtained from the Ground Control Station located on the moving platform. To account for the movement of the platform, the forward velocity of the multirotor $v_z$ has to be larger than the speed of the moving platform in order to catch up with it. In a real-time setup $v_z$ has to account for the limitation of the quadrotor's maximum speed that will vary for different UAVs. Simplifying Equation 3.5 for the already chosen $v_z$:

$$\begin{cases} v_x = v_z \tan\theta, \\ v_y = v_r \tan\varphi; \end{cases} \tag{3.6}$$

where $v_r = v_z / \cos\theta$.

Finally, the system is controlled with a proportional controller:

$$\tilde{\mathrm{F}} = K_p \begin{bmatrix} v_x \\ v_y \end{bmatrix} = K_p \begin{bmatrix} v_z \dfrac{x'}{f}, \\ v_r \dfrac{y'}{f} \end{bmatrix} = K_p \begin{bmatrix} v_z \tan\theta, \\ v_r \tan\varphi \end{bmatrix} = \tilde{K} \begin{bmatrix} \tan\theta, \\ \tan\varphi \end{bmatrix}; \tag{3.7}$$

where $K_p, \tilde{K} - 2\times 2$ diagonal matrices. For small $\theta$ and $\phi$, the matrices can be considered to be diagonal: $K_p = \begin{bmatrix} K_{p_h} & 0 \\ 0 & K_{p_v} \end{bmatrix}, \tilde{K} = \begin{bmatrix} K_{p_h} v_z & 0 \\ 0 & K_{p_v} v_z \end{bmatrix}$.

The overall control system is depicted in Fig. 3.4.

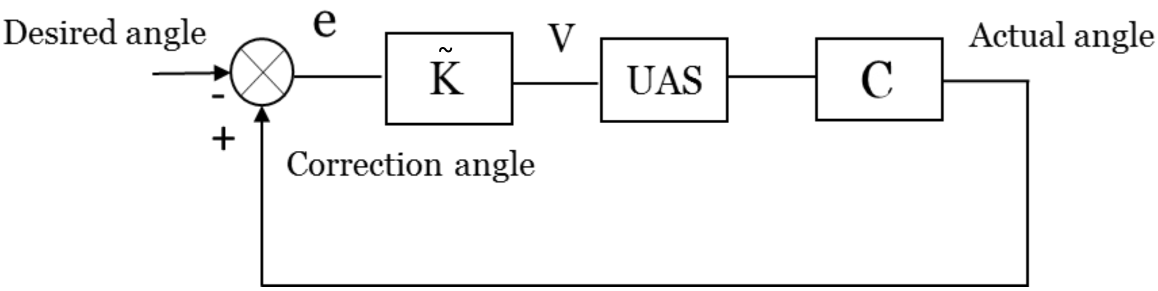


Figure 3.4 - Control diagram of the suggested IBVS approach

Fig. 3.4 shows that actual (horizontal/vertical) angles are measured by the image data from camera C, and are compared to the desired angles (considered zero to ensure following the slopes). The error e (difference between actual and desired angles) is called a correction angle, which is given as an input to controller $\tilde{\text{K}}$. Output of the controller is the desired velocity vector which is sent to the UAS.

## 3.3 Kinematic effects on a closed-loop visual feedback system

Consider the image plane to be a projection of the 3D features of the scene. This allows the use of circles, lines, points, etc. as elementary features that will change their position in the camera frame with the change of position of a multi-rotor. Kinematic effects on the closed-loop vision-based system consider how the UAV should move with respect to the image feature perceived by the camera.

Consider a point on an image as a desired image feature, and define a set of points on the image needed for alignment as $s = s(m, a)$, where $m -$ pixel coordinates of points, $a -$ camera intrinsic parameters. For a 3D point with coordinates $(X, Y, Z)$ after projection on the image plane (Chaumette & Hutchinson, 2006):

$$\begin{cases} x' = fX / Z = u - c_u, \\ y' = fY / Z = h - c_h; \end{cases} \tag{3.8}$$

where $m = (u, h)$ - coordinates of image point in pixels, $f, c_u, c_h -$ focal length and coordinates of the principal point, $a = (f, c_u, c_h)$ - intrinsic camera parameters, $(x', y')$ - image coordinates of the point.

Since a UAV has six degrees of freedom – three rotational and three translational – the UAV velocity matrix relative to a moving target frame $V \in \Re^{1\times6}$ can be expressed as follows:

$$V = [v_x, v_y, v_z, w_x, w_y, w_z]; \qquad (3.9)$$

where $v_x, v_y, v_z$ - horizontal, vertical, and forward velocities, respectively, and $w_x, w_y, w_z$ - angular velocities with respect to the $x, y, z$ axis.

The relation between the change of point parameters $\dot{s}$ ($s \in \Re^{k\times 1}$) on the image and the UAV motion $V$ can be expressed with an interaction matrix $L \in \Re^{k\times 6}$, where $k -$number of parameters that represent image features (Silveira et al., 2001):

$$\dot{s} = L^T V; \qquad (3.10)$$

where a single point is represented by 2 parameters $s = (x', y')$. The relationship between a multi-rotor velocity and time change of error $\dot{e}$ can be expressed as follows:

$$\dot{e} = L_e{}^T V; \qquad (3.11)$$

where $e(t) = s - s_d$, $s_d$ - desired location of the image feature on image, $s$ - actual location of the image feature on the image, and $L^T = L_e{}^T$ from Equations (3.10) and (3.11). Matrix $L$ can be derived from the optical flow equation: $L = \begin{pmatrix} -f/Z_1 & 0 & x'_1/Z_1 & x'_1 y'_1/f & -(f + x'^2_1/f) & y'_1 \\ 0 & -f/Z_1 & y'_1/Z_1 & f + y'^2_1/f & -x'_1 y'_1/f & -x'_1 \\ -f/Z_2 & 0 & x'_2/Z_2 & x'_2 y'_2 & -(f + x'^2_2) & y'_2 \\ 0 & -f/Z_2 & y'_2/Z_2 & f + y'^2_2/f & -x'_2 y'_2/f & -x'_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ -f/Z_n & 0 & x'_n/Z_n & x'_n y'_n/f & -(f + x'^2_n/f) & y'_n \\ 0 & -f/Z_n & y'_n/Z_n & f + y'^2_n/f & -x'_n y'_n/f & -x'_n \end{pmatrix}$, where $Z_i$ – distance to the $i^{th}$ point, $(x_i', y_i')$ - coordinates of the $i^{th}$ point on the image, $i = 1,2,3..n$, $n \in \mathbb{N}^+$ - number of the reference points, and $\mathbb{N}^+$ - a set of positive integers.

Considering a case when the dynamics of the inner loop of the system is much faster than the dynamics of the outer loop, the assumption can be made that multirotor velocity will always go to the desired value. Considering $V$ as a desired commanded velocity of the multirotor, to ensure exponential decoupled decrease of an error $(\dot{e} = -\mu e)$, the desired velocity vector can be calculated as:

$$V = -\mu L_e{}^{+} e; \tag{3.12}$$

where $L_e{}^{+} \in \Re^{6\times k}$, $L_e{}^{+} = (L_e{}^{T} L_e)^{-1} L_e{}^{T}$, and $L_e$ is of full rank (Chaumette & Hutchinson, 2006). When $k = 6$ and $s \in \Re^{6\times 1}$ then it is possible to invert $L_e$ making the desired commanded output velocity $V = -\mu L_e{}^{-1} e$.

It is assumed that this output velocity $V$ can be commanded by a well-tuned autopilot (inner-loop controller) with sufficiently fast dynamics and servo-precision. A control diagram of this approach can be seen in Fig. 3.5.

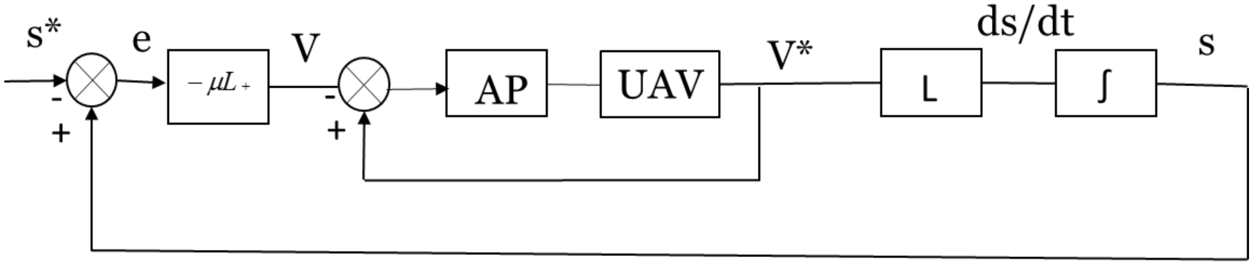


Figure 3.5 – Control diagram of a closed-loop system

If assuming insignificant rolling, pitching, and yawing of the platform (as in Chapter 5.5), and insignificant rolling/pitching of the UAV, considering slow motion during landing in low-wind conditions, using small movement approximation with $-f/Z$ as a dominant term, the interaction matrix could be simplified to $L = \begin{pmatrix} -f/Z & 0 \\ 0 & -f/Z \end{pmatrix}$, where $f$ - focal length of the camera, and $Z$ - distance to the DPT. Thus, the implementation in Fig. 3.6 becomes a simplified special case of the diagram in Fig. 3.5 for the developed IBVS approach that uses correction angles in Fig. 3.4.

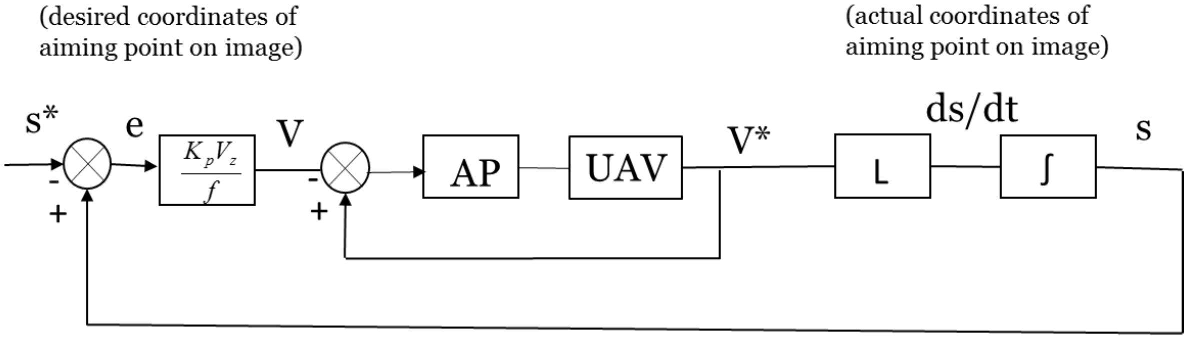

(desired coordinates of aiming point on image)
(actual coordinates of aiming point on image)
s*
e
$\frac{K_p V_z}{f}$
V
-
+
-
+
AP
UAV
V*
ds/dt
L
∫
s

Figure 3.6 – Control diagram of the suggested approach

If the autopilot is well-tuned, then $V = V^*$; the integrator in the diagram is a part of the physics of the camera system. After simplifying the diagram in Fig. 3.6, we will get the diagram in Fig. 3.7.

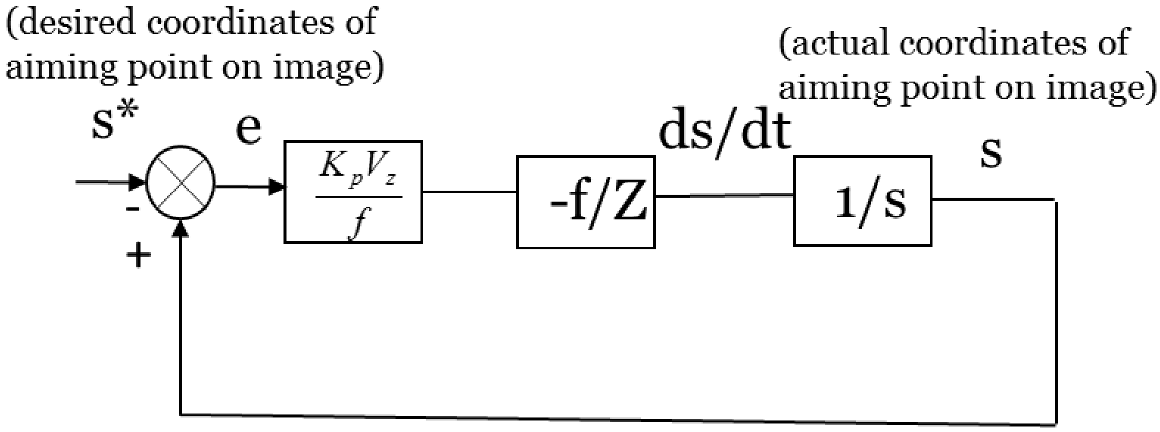

(desired coordinates of aiming point on image)
(actual coordinates of aiming point on image)
s*
e
-
+
$\frac{K_p V_z}{f}$
-f/Z
ds/dt
1/s
s

Figure 3.7 – Control diagram of the suggested approach after simplification

Consider a closed-loop transfer function with $\mathsf{s}$ as a complex number frequency parameter of Laplace transform (Ogata, 1970), for a decoupled horizontal/vertical control structure: $\frac{s}{s^*} = \frac{(1/f)K_p v_z(-f/Z)/\mathsf{s}}{1+(1/f)K_p v_z(f/Z)/\mathsf{s}} = \frac{-K_p v_z/Z}{\mathsf{s}+K_p v_z/Z} = -1$, as $t \to \infty$, which shows that current approach can be used for both regulatory and servo-control systems. Thus, the closed-loop system has zero steady-state error and a time-constant of $\tau = \frac{Z}{K_p v_z}$. Multi-variable control should be considered for future works.

## 3.4 Dynamic effects on a closed-loop visual feedback system

This section analyses dynamic constraints such as inner and outer loop autopilot configurations and camera frame rates as needed for the outer loop performance. The robustness of the controlled system to the delays due to image processing and sampling constraints affecting the performance of the system are considered in the design. The control analysis is represented as applied to a discrete-time continuous system to highlight the aforementioned aspects. In this section, we will use $z$-transform to conduct control system analysis with $z = e^{sT_s}$, where s is a complex number frequency parameter of Laplace transform, and $T_s$ – the sampling interval.

For the convenience of using a control system with negative feedback (as compared to the positive feedback representation used in Chapters 3.2 and 3.3), horizontal and vertical errors are defined as:

$$\begin{cases} e_h = \theta_d - \theta, \\ e_v = \varphi_d - \varphi; \end{cases} \tag{3.13}$$

where $\theta_d$ , $\varphi_d$ - desired horizontal angle (angle to align with the centerline of the pattern), desired vertical angle (angle to align to a glide-slope), respectively; $\theta, \varphi$ – actual angles. Then, the horizontal and vertical correction angles are calculated by:

$$\theta_{cor} = 0 - \arctan\left(\frac{x'}{f}\right) = -\arctan\left(\frac{x'}{f}\right) = -\theta \,. \tag{3.14}$$

$$\varphi_{cor} = 0 - \arctan\left(\frac{y'}{f}\right) = -\arctan\left(\frac{y'}{f}\right) = -\varphi; \tag{3.15}$$

where $(x', y')$ -coordinates of the aiming point on the image plane; $f$ –focal length of camera.

The system is controlled with a proportional controller (Fig. 3.8), which is given as:

$$\mathrm{F} = K_p \begin{bmatrix} -v_z \tan\theta, \\ -v_r \tan\varphi \end{bmatrix} = K_p \begin{bmatrix} v_z \dfrac{-x'}{f} \\ v_r \dfrac{-y'}{f} \end{bmatrix} = K \begin{bmatrix} -x' \\ -y' \end{bmatrix}; \tag{3.16}$$

where $v_r = v_z / \cos\theta$, $v_z$ – multi-rotor's forward velocity, $v_x$ – multi-rotor's horizontal velocity, $v_y$ – multi-rotor's vertical velocity, and $K_p, K$ – $2\times 2$ diagonal matrices, such that $K_p = \begin{bmatrix} K_{p_h} & 0 \\ 0 & K_{p_v} \end{bmatrix}, K = \begin{bmatrix} K_{p_h} v_z / f & 0 \\ 0 & K_{p_v} v_z / f \end{bmatrix}$.

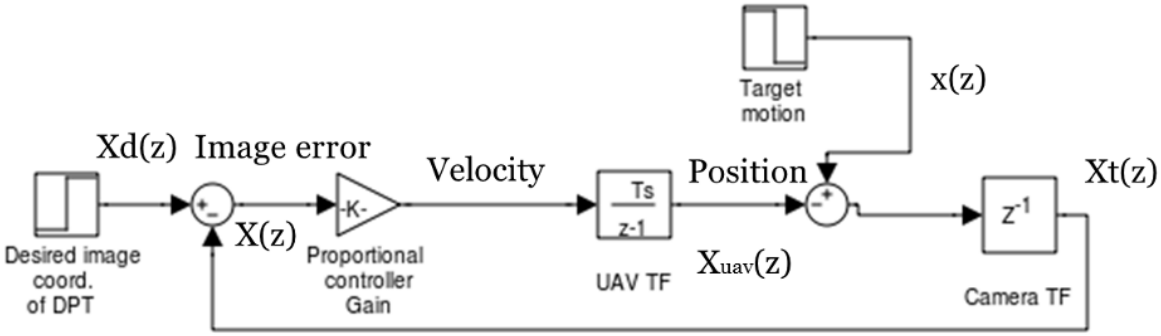


Figure 3.8 - Visual feedback control system for auto-landing

Since the real-time setup described in Chapter 5.5 is capable of processing 10 frames per second, the delay due to image processing is 100 ms. The decision was made to choose the sampling time $T_s$ to be equal to the image processing time, since oversampling does not give any advantage (sending the same value is not beneficial), and sampling at a lower rate than image processing capabilities is undesirable due to the required data rate to stream to the autopilot.

In classical visual feedback control, the camera is represented as a unit delay (Peter Ian Corke, 1994) with transfer function $H(z) = \frac{1}{z}$. The image error $X(z) = X_d(z) - X_t(z)$ is the difference between the desired and current location of the aiming point on the image. Image error $X(z)$ is an input to the proportional controller, which calculates the velocities using Equation 3.16. The UAV transfer function is represented as a discrete integrator that transforms velocity into position $X_{UAV}(z)$, and has a transfer function $G(z) = \frac{1}{z-1}$.

In the real-case scenario, the UAV transfer function consists of the transfer function $Ap(z)$ that represents the dynamics of the multirotor controlled by the autopilot velocity controller. However, if the autopilot is tuned, this block of the diagram is approximately equal to 1 for a closed-loop system (Fig. 3.9).

Since the landing system was implemented as a cascade control with multiple inner loops (sending commands to the outer loop of the autopilot – velocity loop), the control loop with image processing can be slower than the autopilot's velocity loop. The velocity loop of the Pixhawk autopilot (described in

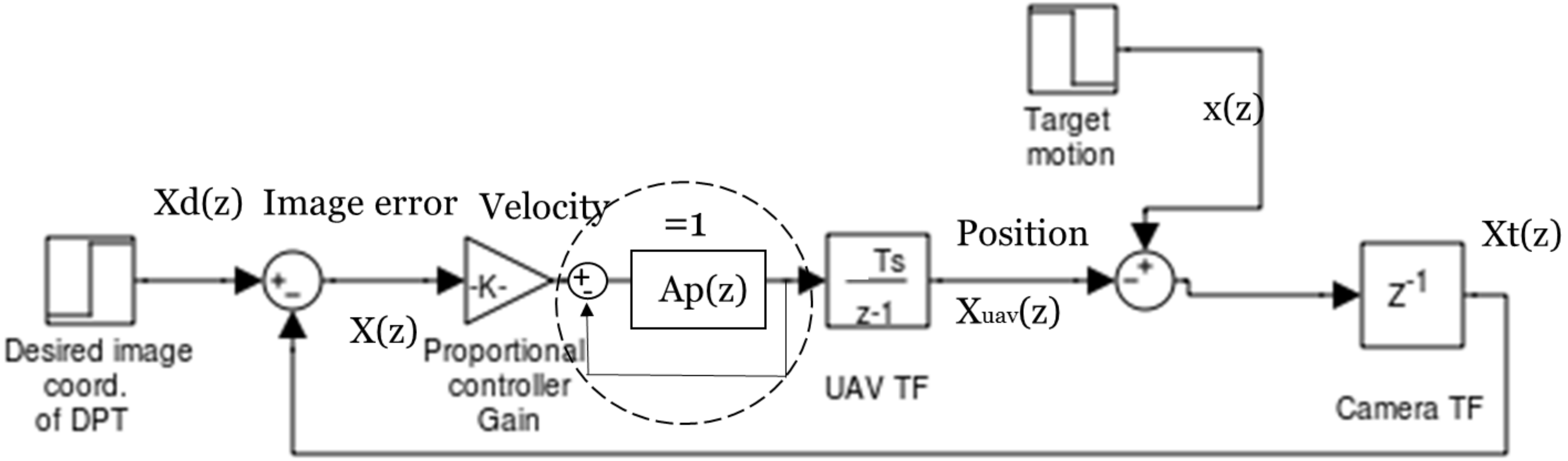


Figure 3.9 - Visual feedback control system for auto-landing (Expanded)

Chapter 5.4.4) runs on 50 Hz. The developers of the Pixhawk recommend that the velocity set-point commands should be sent to the autopilot at least at 4 Hz. Since the developed camera system was capable of 10 Hz, this was enough to achieve a satisfactory performance control system for automatic landing.

The main requirements for such a control system are:

a) Stability; and

b) Tracking without a steady-state error.

Fig. 3.10 shows a typical response of the system in Fig. 3.9 with a proportional controller gain $K = 0.6$, done in Simulink, Matlab, for a decoupled horizontal/vertical control system structure.

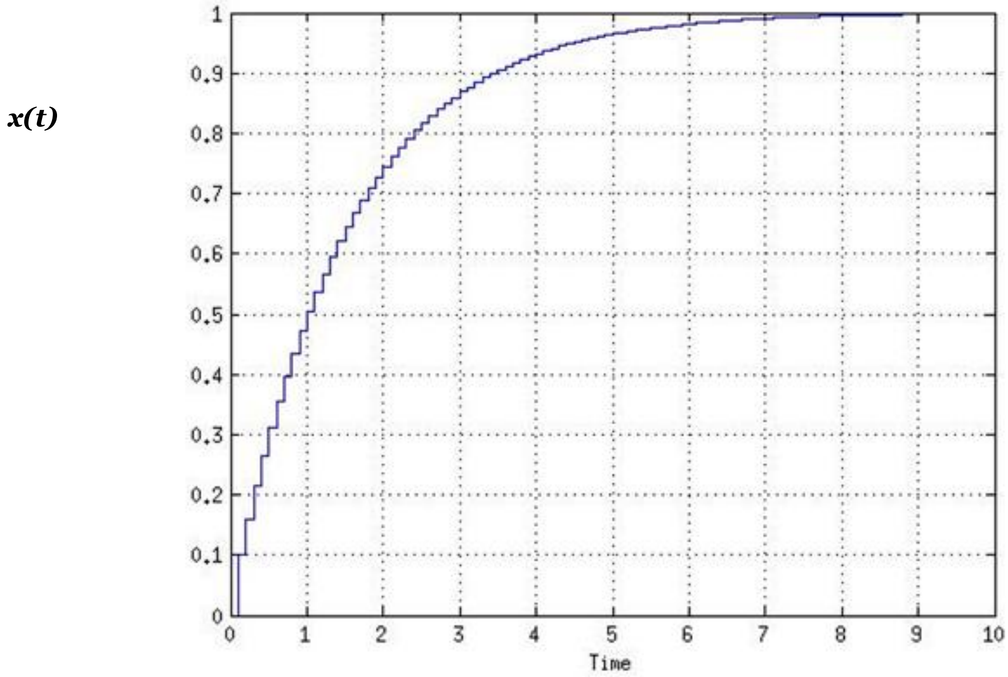


Figure 3.10 - Response of the system with proportional control
($K = 0.6$, $T_s = 0.1\,\mathrm{ms}$)

Consider a closed-loop transfer function that describes the response of the UAV position $X_{UAV}(z)$ to the target position $x(z)$:

$$\frac{X_{UAV}(z)}{x(z)} = \frac{H(z)G(z)K}{1+H(z)G(z)K}. \tag{3.17}$$

After substituting the transfer functions of blocks, this closed-loop transfer function can be written as:

$$\frac{X_{UAV}(z)}{x(z)} = \frac{0.6}{z^2 - z + 0.6} = \frac{0.6}{z(z-1)+0.6}. \tag{3.18}$$

Closed-loop poles are located at $z_1 = 0.5 + 0.6i$, $z_2 = 0.5 - 0.6i$ and indicate a stable closed-loop system (Ogata, 1995). To show that the steady-state error is near-zero, $\frac{X_{UAV}(z)}{x(z)} = \frac{0.6}{z(z-1)+0.6} = 1$ as the discrete time step $n \to \infty$, which ensures that the UAV will come to a target without steady-state error.

The open-loop transfer function of this system is $H(z)G(z)K = \frac{0.6}{z(z-1)}$. The typical bode plot in Fig. 3.11 shows the phase margin of $38^\circ$ with a gain margin of $20\log(1.7) = 4.6$ dB can be achieved.

Consider another closed-loop transfer function that describes the response of the image plane error $X(z)$ to the target motion $x(z)$:

$$\frac{X(z)}{x(z)} = \frac{H(z)}{1+H(z)G(Z)K}. \tag{3.19}$$

After substituting the transfer functions of blocks, this closed-loop transfer function can be written as:

$$\frac{X(z)}{x(z)} = \frac{z-1}{z^2 - z + 0.6} = \frac{z-1}{z(z-1)+0.6}. \tag{3.20}$$

$\frac{X(z)}{x(z)} \rightarrow 0 \text{ as } n \rightarrow \infty$, meaning that as the target moves, the image error is going to zero (Type 1 system), which ensures tracking without a steady-state error.

As can be seen from Fig. 3.10, the closed-loop system has a time constant of the order of 2 seconds. Since the flying time during the outdoor landing experiment (Chapter 5.5.2) was approximately 10 seconds ($k = 100, T_s = 0.1\,s$), the step-response shows that a near zero steady-state error can be achieved with this configuration.

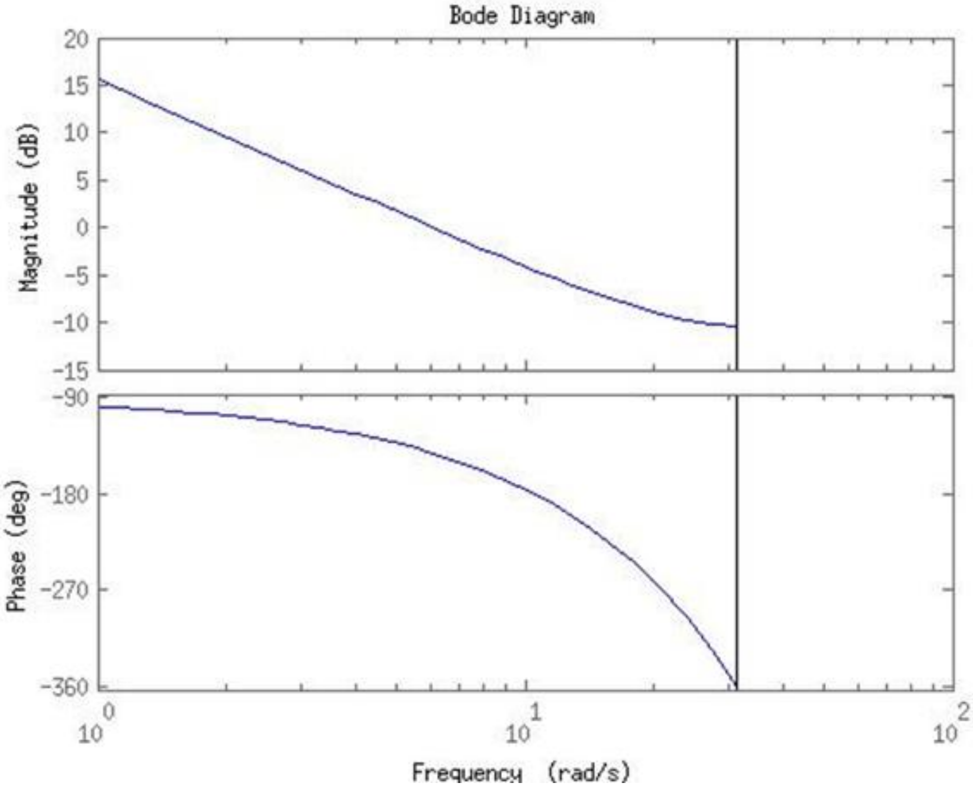


Figure 3.11 - Bode plot of an open-loop transfer function

Diagram 3.9 does not consider wind disturbance. However, the well-tuned autopilot can handle small wind perturbations during flight. High-wind disturbances can cause exceptions such as losing markers from the camera FoV, and are considered in Chapter 4. To prevent losing visual markers from the FoV in high-wind operations, future work should consider using a gimbaled camera to compensate for the UAV and a target rolling and pitching.

The multi-rotor dynamics are simplified for the analysis by assuming a closed-loop autopilot controller will maintain zero steady-state error. Not relying explicitly on the modeled dynamics of a particular UAV helps to generalize the applicability of the developed control scheme to the entire category of the UAVs. However, higher order dynamics could lead to instabilities. Future work should examine the influence of a wide range of disturbances, and parametric variations on the closed-loop stability, considering nonlinearities in the multi-rotor's dynamics, multi-variable control and couplings within IBVS laws.

## 3.5 Conclusion

An image-based visual servoing scheme was described to control a multi-rotor during automatic alignment with a moving platform. An IBVS scheme was chosen over PBVS as it does not require calculating the 3D UAV position, and is less sensitive to camera calibration errors. Since IBVS derives the control laws in the image space, the targets are more likely to stay within the FoV, unless affected by unexpected wind-gust.

The closed-loop visual feedback system is analyzed from two points of view: kinematic and dynamic, considering the delay due to image processing, sampling constraints, and target motion disturbance. The controller design follows a linear controller design, which has limitations to be applied only for small deviations from the desired path. However, according to the developed landing method, the UAV initially aligns with the help of a GPS, thus large deviations from the desired track are not likely. Nonlinear controller design, with potentially applying an adaptive control strategy during a UAV landing on a moving target in a GPS-denied environment, is considered as future work.

To compensate for the delay due to image processing, several methods of dynamical optimization (e.g. Kalman filter) could be used. However, Kalman filtration applied to the sequence of camera images to predict the location of the aiming point at a certain time would lead to increasing the number of operations on the image. For the case of the chosen Odroid processor (details are given in Chapter 5), the image rate was reduced from 10 frames per second to 4-5 frames per second. Additional analysis was needed to determine the trade-offs between increasing the computational complexity and impact of the delay due to image processing on the control system performance. Since the developed auto-landing system operates at the rate of 10 frames per second, a delay of 100 ms is considered to have little impact on the performance of the visual closed-loop system because the dominated dynamics of the aircraft are at least 5 times slower than the camera frame rate.

Even though control system analysis is notional, it demonstrates that tracking without a steady-state error and closed-loop stability through a visual feedback landing system is viable. As shown experimentally in Chapter 5, the prototype system provided reasonable accuracy: 15 cm from the DPT, for landing on a static target, and 25 cm from the DPT, for landing on a moving platform.

Chapter 4

# Discrete-event supervisory control

## 4. 1 Related work

Discrete-event systems represent dynamical systems where signals take values in a discrete set. Examples of such systems are traffic lights (“red”, “amber”, “green”), elevator (“waiting”, “going up”, “going down”), mechanisms for opening gates (“door opened”, “door closed”), etc.

Discrete-event systems can be timed and untimed. Untimed DES considers logical behaviour of the state evolution; it does not account for timing constraints, such as how long the system can stay in a certain state, or when it should enter another state. Timed DES combines both logical and timing details, and thus is more complex.  Theory of discrete-event systems is generally based on the concepts of states, events, and languages.

Ramadge and Wonham (1989) developed a supervisory control strategy for a DES. Their control mechanism refuses actuation requests in order to achieve desired system behaviour; the supervisor’s design is “language-based”. Brave and

Heymann (1988) suggested a control strategy using a counter, which is described with an update map. Brandin and Wonham (1994) developed a timed transition model; the authors generalized the existence of maximally permissive supervisory control. The limitation of their modeling framework is that controllable events should always have an infinite upper time bound, which does not always coincide with processes in the real world. The modeling framework described in this thesis does not have this limitation: when the upper time bound is taken to be infinity, then delay *can* be made indefinite; this special case corresponds to Wonham and Brendin's disablement mechanism. The delay technology described in this thesis allows disablement to be *either* time-limited, or indefinite.

In this chapter, the landing phase is modeled as a timed discrete-event system. The  problems with possible exceptions that can occur during a landing mission (Technical Challenge 4 described in Chapter 1.3) are solved by adding a discrete-event supervisor to the continuous control system. The current modeling framework follows the semantics of O'Young (1991), where the supervisor takes an active role in sending timely warnings to the system for expediting and/or delaying certain control actions. These semantics also model time persistency, which is essential to account for the battery discharge over the sequence of states during landing.

This discrete-event layer forms the part "Discrete-event supervisory control" of the overall automatic landing system as a **bolded** box in Fig. 4.1. This approach was documented in (Borshchova & O'Young, 2017 c).

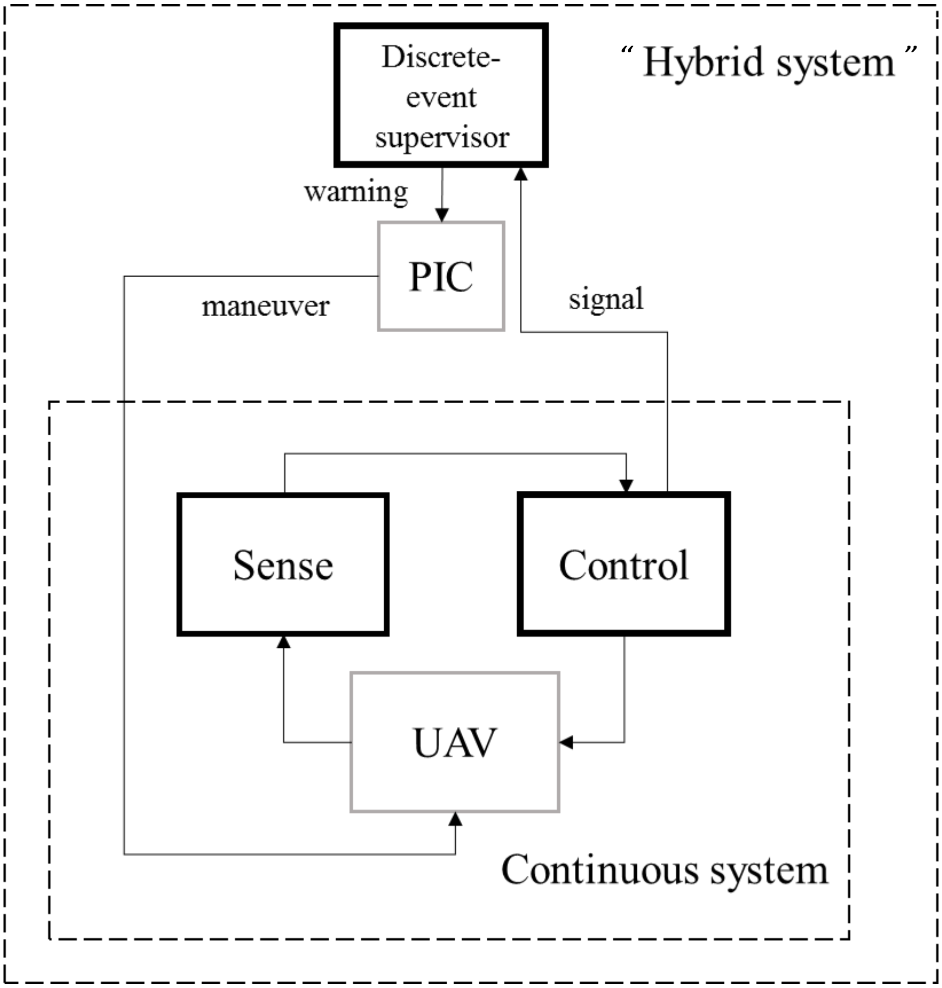


Figure 4.1 – Overall approach to the automatic landing system

## 4. 2 Modeling of a timed discrete-event vision-based landing

A discrete-event system is a discrete-state, event-driven system in which the state evolution depends on the occurrence of discrete events over time. An untimed discrete-event system can be formally described as a finite automaton represented by a 5-tuple

$$G = (A, \sum, \delta, a_0, A_m); \tag{4.1}$$

where $A$ - finite set of activities (states), $\sum$ - finite set of symbols (events), also called “alphabet”, $\delta$ - transition function, so that $\delta : A \times \sum \rightarrow A$, $a_0 \in A$ - start activity, which is the activity before any input was processed, $A_m \subseteq A$ - set of accept activities (marker activities, Wonham & Ramadge, 1987).

Automaton reads a finite string of symbols $\sigma_1\sigma_2\sigma_3...\sigma_n$ where $\sigma_i \in \Sigma$, which is called an input word. Denote $\sum^*$ - set of all finite words, then $\delta : A \times \sum^* \rightarrow A$ (Lin, Vaz, & Wonham, 1988). Events are thought to be instantaneous and occurring at quasi-random moments of time $\Re^+ = \{t \mid 0 \leq t < \infty\}$.

Let $G_1, G_2$ be two DES. The synchronous product of $G_1$ and $G_2$ denoted by $G_1 \| G_2$ is another DES with the initial activity of $G_1 \| G_2$ being a pair $(a_0^1, a_0^2)$, where $a_0^1, a_0^2$ are initial activities of $G_1$ and $G_2$, respectively. An activity

$(a_1, a_2)$ is a marker activity when $a_1$ is marked w.r.t. $G_1$ and $a_2$ is marked w.r.t. $G_2$.

Let ℕ denote the set of nonnegative integers, and $Z^+$ be a set of positive integers. Each event $\sigma \in \sum$ will be equipped with a lower time bound $l_\sigma \in$ ℕ and an upper time bound $u_\sigma \in Z^+ \cup \infty$ such that $l_\sigma \le u_\sigma$. The lower bound is considered to be a delay, and the upper bound is considered to be a deadline. Let $l, u \in Z^+$. Possible types of events according to these semantics are:

1) $\sigma[0; u]$, which has only a deadline;

2) $\sigma[l; \infty)$, which has only a delay;

3) $\sigma[l; u]$, which has both a delay and a deadline; and

4) $\sigma[0; \infty)$, which has neither a delay nor a deadline, and is considered to be untimed.

The passage of time is measured by counting the tick of a global clock which updates every $\tau \in \Re^+$. The lower time bound dictates that the system must wait in a state until $l_\sigma$ ticks have been counted before the event can occur; event $\sigma$ must occur before $(u_\sigma + 1)$ ticks, unless another event $\beta$ occurs before it.

Timing constraints are able to persist over the state transition (the timing constraints of event $\sigma$ may not be affected by the occurrence of $\beta$). Such a persistency rule helps to describe an evident timing constraint such as a battery discharge time over the sequence of states during an automated landing. The timer is assigned to each event label, and the enablement of the counting coincides with the enablement of the corresponding event.

Let $r_\sigma := \begin{cases} u_\sigma, u_\sigma \neq \infty; \\ l_\sigma, otherwise. \end{cases}$ Then, the integer range $[0, r_\sigma]$ will represent all possible assignments of local timers associated with the transition $\sigma$. Let $L := (\sigma_1, \sigma_2, ..\sigma_n)$, where $n$ - the cardinality of $\sum$. Thus, timed DES can be represented as:

$$P = (Y, \textstyle\sum_p, \delta, y_0, Y_m); \qquad (4.2)$$

where $Y = A \times \prod_{i=1}^{n} [0, r_\sigma]$ is a finite set of timed states capable of expressing infinite upper time bounds; $\sum_p$ - alphabet and $\delta$ - transition function, so that $\delta : Y \times \sum_p \rightarrow Y$ with $y_0 \in Y$ as an initial state, and $Y_m \subseteq Y$ - the subset of marker states (states that complete a task or a sub-task).

Let UAV states during automatic alignment be partitioned using the method of Millan (Millan, 2006) with functional $F = D - l$. This partitioning allows the alignment stage to be divided into two fields – "near field" $(F \leq 0)$ and "far field" $(F > 0)$, where $D$ - distance from the UAV to the desired point of touchdown in meters. This distance $D$ can be estimated from the GPS location of the UAV relative to the moving platform position.

Considering the landing phase starting at the distance of 60 m away from the moving platform, it is reasonable to divide "far field" and "near field" with the proportion of 5/6 and 1/6, respectively, while "near field" is separated from "far field" with a distance $l = 10\,m$. For a small multi-rotor UAV moving with a speed of 3 m/s during its automatic alignment, the update time is chosen to be 0.5 s.

Since in the case of large distance $D$ the targets appear close to each other on the image, the algorithm at "far field" can either detect the whole pattern, or cannot detect the pattern at all. This situation can be short-term (due to an image processing error, blur, etc.) or long-term (due to a change in the lighting conditions).

The DES states are defined:

**Forbidden (Blocking):** Let $\Sigma_p$ be a finite event set, $Y$ - a set of states, $Y_m$ - set of marker states. State $y_f \in Y$ is called *forbidden* if there is no marker state reachable from $y_f$.

Since a forbidden state is a blocking state, this state will be avoided in the synthesis of a non-blocking DES supervisor.

**Normal:** Let $\Psi$ be a set of detected red targets on a particular virtual orthogonal camera image. The state is called *normal* if the cardinality of $\Psi$ is equal to 5.

*Note.* The state is called *abnormal* if the cardinality of $\Psi$ is greater or lower than 5. If the cardinality of $\Psi$ is greater than 5, this infers that the detection threshold is too low for current lighting conditions ("false positive"); if the cardinality of $\Psi$ is lower than 5, this infers that the detection threshold is too high for current lighting conditions ("false negative").

The activity transition graph for "far field" is given in Fig. 4.2.

Fig. 4.2 represents the State “normal” (pattern detected), “abnormal”- cannot detect the pattern; State “reset” – when the detection threshold is tuned by the AVO, and State “forbidden” - crashed; Events “lose” targets, “regain” targets, “stop_mission”, “restart_mission”, and “crash”. Markings “×” and “/” are related to the control mechanism, which will be explained in Chapter 4.3.

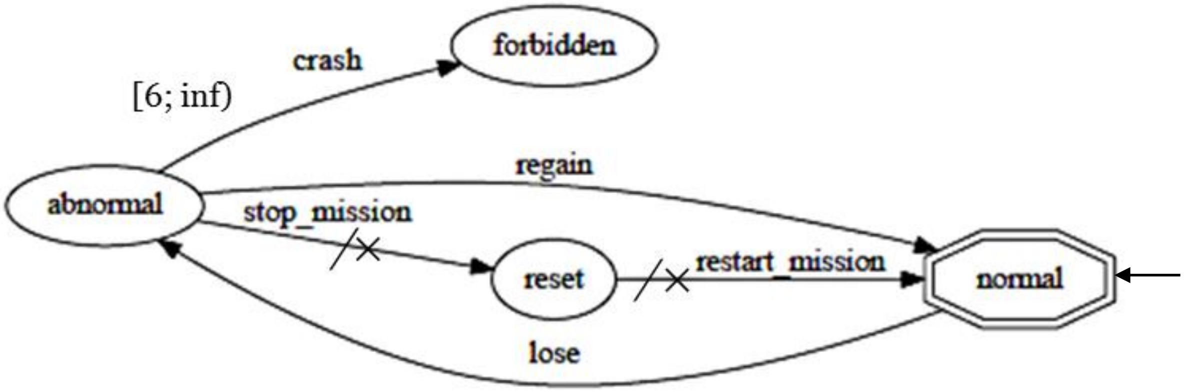


Figure 4. 2 - Activity transition graph for “far field”

The UAV starts the landing mission in the initial state “normal” (marked with an arrow in Fig. 4.2), which is also a marker state. If Event “lose” occurs, and the pattern cannot be detected (State “abnormal”) the UAV will crash due to the absence of control input to the autopilot. This Event “crash” will occur if the control input stays undefined longer than 6 ticks. Event “stop_mission” occurs when the UAV receives a command from the AVO to change “landing” mode into “hovering” mode. The detection threshold is updated in the State “reset”, after which the alignment with the moving platform is set to start again (Event “restart_mission”).

**Aligned:** Let $m(x_{o_m}, y_{o_m})$ be the fifth blob and $n(x_{o_n}, y_{o_n})$ - aiming point on the virtual orthogonal image plane. The state is called *aligned*, if $\begin{cases} \left|x_{o_m} - x_{o_0}\right| < \varepsilon, \\ \left|x_{o_n} - x_{o_0}\right| < \varepsilon, \\ \left|y_{o_n} - y_{o_0}\right| < \varepsilon; \end{cases}$

where $(x_{o_0}, y_{o_0})$ - actual center of the orthogonal image, $\varepsilon$ - acceptable pixel error specific to the configuration chosen, e.g. a 10-pixel tolerance was used in the “near field” implementation.

*Note*. The state is called *not aligned* if $\begin{cases} \left|x_{o_m} - x_{o_0}\right| > \varepsilon, \\ \left|x_{o_n} - x_{o_0}\right| > \varepsilon; \\ \left|y_{o_n} - y_{o_0}\right| > \varepsilon. \end{cases}$

When aligned with the pattern, the projection of the centerline of the pattern on the orthogonal image would coincide with the vertical centerline of the orthogonal image (property of perspective projection). Also, the aiming point would be located at the actual center of the orthogonal image to ensure correct slopes. To avoid the situation when, for example, the error of 1 pixel marks the camera state as “not aligned”, some acceptable error level $\varepsilon$ is allowed as described above.

**Dangerous:** Let $\Psi$ be a set of detected red targets on a particular virtual orthogonal camera image. The state is called *dangerous*, if cardinality $\Psi < 3$.

State “dangerous” can be entered only if the UAV is following the wrong trajectory (controller is not tuned to wind/gust). It is assumed if the “near field” zone is entered, the detection threshold is tuned to the lighting conditions.

An activity transition graph to illustrate the discrete-event UAV behavior in “near field” is given in Fig. 4.3.

Fig. 4.3 represents the States “aligned”, “not_aligned”, “dangerous”, “forbidden” (crashed), State “EP_control”, when the UAV is manually controlled by the external pilot, and State “landed” – marker state; Events “lose” targets from FoV, “regain” targets, “misalign”, “come_back” to aligned, “crash”, “hit_the_ship_gear”, “auto_finished” – which indicates the alignment stage is over, “abort” – switching to manual control, and “land”. The markings “×” and “/” are related to the control mechanism, which will be explained in Chapter 4.3.

The UAV can enter the “near field” zone in either of the initial states (marked with an arrow in Fig. 4.3): “aligned”, or “not_aligned” (when the controller is not tuned to the wind). If the mission is started in the State “aligned”, the UAV can transit to the State “not_aligned” (Event “misalign”) if an unexpected gust affects the UAV trajectory. If there is enough time/gain for the controller to align the UAV with the moving platform, the UAV will return to the State “aligned” (Event “come_back”). The UAV can also lose targets from the FoV when following the wrong path, and transit to the State “dangerous” (Event “lose”). If the targets are not regained within 1.5 s (3 ticks considering an update time of 0.5 s), the UAV will crash due to the absence of control input to the autopilot.

Considering a distance to the DPT of 10 m when entering the “near field” zone, the multi-rotor’s velocity of 3 m/s, update time of 0.5 s, and the velocity of a moving platform of 1 m/s (as was implemented in the set of experiments

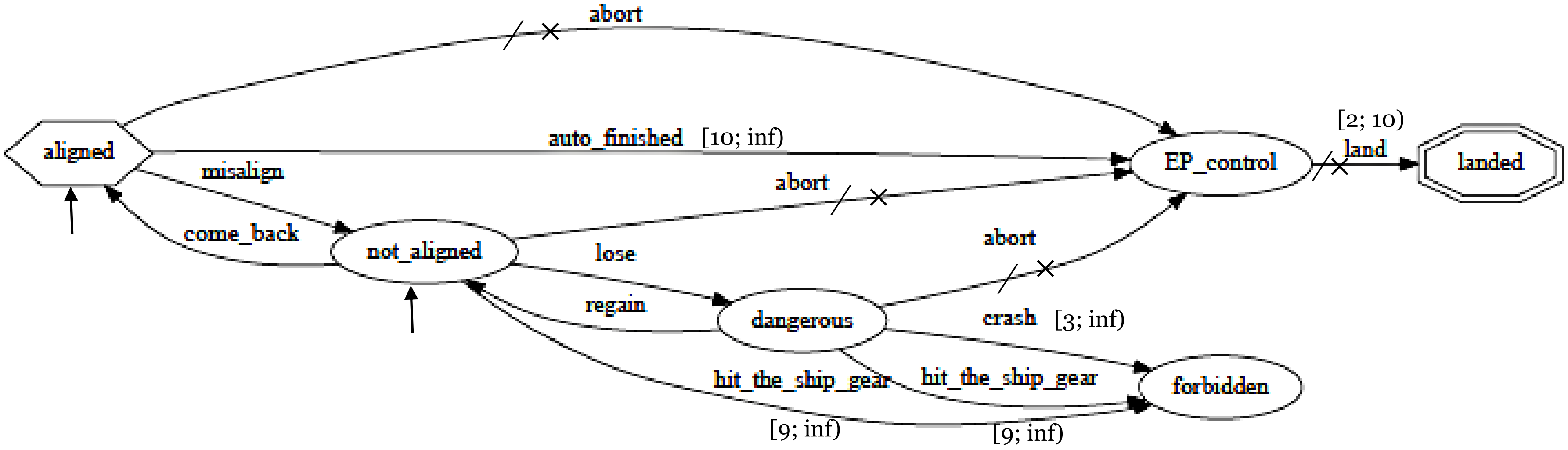


Figure 4.3 - Activity transition graph for “near field”

described in Chapter 5), the automatic alignment in “near field” is assumed to last for 10 ticks.

If the controller gains are not tuned to the wind, and the UAV remains in the State “not_aligned” when being close to the ship (assume 9 ticks for the current configurational setup), the UAV might hit the ship’s gear and crash into it. Event “auto_finished” can occur after 10 ticks, which is the time when the UAV will be located over the DPT. The pilot can abort the auto-landing mission and take over the control of the UAV during the “near field” phase at any state. Event “land” can occur from [2; 10] ticks, which depends on the location of the UAV when exceptions happen.

## 4. 3 Supervisory control strategy

### 4.3.1 System specification

Let $S$ be a DES that represents a set of dynamical constraints on a plant $P$, whose dynamics are represented as TDES. This set of constraints is called specification. In this case, $S$ constrains plant $P$ to synchronize all transitions labeled by a shared event in $\sum_s \bigcap \sum_p$. The synchronous product of $\Theta = P \parallel S$ is called a specified system.

The specification for the landing system in the “far field” is given in Fig. 4.4. The main requirement of the specification is to allow the UAV enter any state except for forbidden ("crashed"). Technically, this "crash" transition is not needed since the "forbidden" state will be avoided automatically by the non-blocking supervisor; it is added here for clarity.

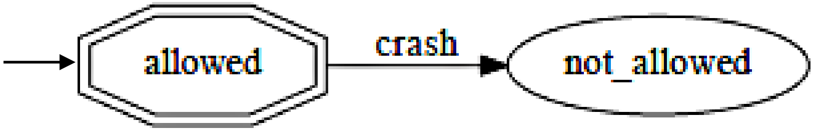


Figure 4.4 - Specification for the system in "far field"

The specification for the system in "near field" is given in Fig. 4.5. The requirement of the flight in "near field" is to ensure the UAV is coming to the marker state "landed" considering the battery life limitations (assume 60 s (120 ticks) for the current configurational setup).

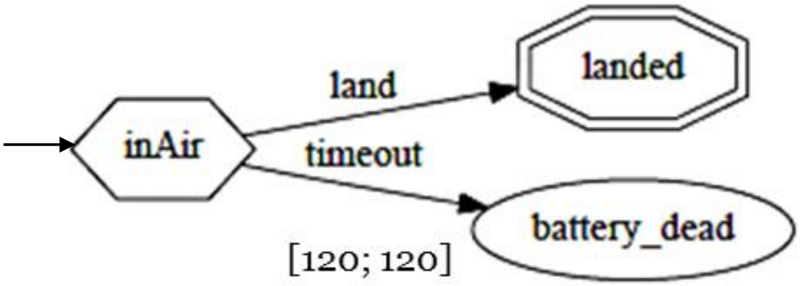


Figure 4.5 - Specification for the system in "near field"

*Note* that the dynamics of the systems under design (Fig. 4.2 and 4.3) have been abstracted in the formulation of the specifications (Fig. 4.4 and 4.5), making it possible for another person, e.g. a client, with no or little knowledge of the system to specify, declaratively, the desirable outcomes.

### 4.3.2 Supervisor synthesis

The supervisory control strategy follows the semantics of O'Young (1991), and differs from the control strategy of Brandin and Wonham (1994). The supervisor in this work is state-based, and takes an active role in sending timely warnings to the system for expediting and/or delaying certain control actions. This way, the supervisor can ensure that specified tasks are performed within a specified time. This expediting mechanism can be regarded as a timed version of the forcing mechanism developed by Golaszewski and Ramadge (1987). The idea behind expediting events is to avoid an unrecoverable error.

Let $P$ be a timed DES, which is synchronized with constraints $S$, so that a specified system is given as $\Theta = P \| S$. Let $\tilde{\Theta}$ be a reachable discrete-event subsystem of a specified system $\Theta$, and $y_{\tilde{\Theta}} \in Y_{\tilde{\Theta}}, y_p \in Y_p$. The supervisory control pattern is defined as:

$$\Delta := \begin{cases} \{\sigma : \sigma \in \Sigma_p, \delta(\sigma, y_p)!, \delta(\sigma, y_{\tilde{\Theta}})\neg!\}, \\ \{tick : \delta(tick, y_p)!, \delta(tick, y_{\tilde{\Theta}})\neg!\}; \\ \qquad \{\} otherwise. \end{cases} \tag{4.3}$$

The pattern $\Delta$ is *controllable* if $\sigma$ is a *delayable* event (an event that can be delayed until at least next clock tick). At "far field" *delayable* events are the

Events “stop_mission” and “restart_mission”, at “near field” *delayable* events are the Events “abort” and “land”. These labelled transitions are marked as “/” in Fig. 4.2 and 4.3.

If the tick is $\Delta$ , there must exist an *expeditable* event $\sigma_e$ defined in $y_{\tilde{\Theta}}$ that can be forced to occur before the next clock tick. At “far field”, *expeditable* events are Events “stop_mission” and “restart_mission”, at “near field” *expeditable* events are the Events “abort” and “land”. These labelled transitions are marked as “×” in Fig. 4.2 and 4.3.

An empty control pattern is trivially controllable because no control action is needed. A reachable discrete-event sub-system of a specified system $\Theta$ is controllable if the control pattern $\Delta$ at each reachable state of the sub-system. The largest controllable and non-blocking sub-system exists and is unique; it is called the least-restrictive and non-blocking supervisor as in (Brandin & Wonham, 1994).

The least-restrictive supervisor synthesis for the models developed in Chapter 4.2 was done in OTCT software (O'Young, 1992). OTCT is a suite of C++ routines, implemented in an object-oriented style, for the design and simulation of logical and timed discrete-event systems. OTCT handles both deterministic and non-deterministic structures, and accepts composite transition structures built from a mixture of timed and untimed components. Control data in OTCT after synthesis by the specification are displayed in a *.txt file where delaying and/or expediting occur together with the events which must be delayed or expedited there.

The supervisor in “far field” automatically designed by OTCT software has 21 transitions and 8 states. The control data is given in Table 4.1.

Table 4.1 - Supervisor’s control data for “far field”

PLANT: [abnormal,[crash,5]]
SUPER: [[abnormal,[crash,5]],allowed]
EXPEDITE: stop_mission

The object-oriented structure of OTCT lists the plant state ([abnormal,[crash,5]]), followed by the supervisor’s state ([[abnormal,[crash,5]],allowed]). The plant state listed by the software implies that the UAV will be in the State “abnormal” with the timer on the Event “crash” having counted five ticks from the global clock. The supervisor’s state is a synchronization of the state of the plant ([abnormal,[crash,5]]) and specification ([allowed]). When coming to such a state, the supervisor will give an audio warning to the AVO to expedite the Event “stop_mission”. The input and output files to the OTCT software for “far field” are given in Appendix B. Timed transition graphs for both UAV and a supervisor are given in Appendix C.

The conclusion, made from the data displayed by the optimal supervisor, infers that AVO has to stop the landing mission and tune the detection threshold at the last moment to prevent the UAV crashing.

The supervisor designed in OTCT for “near field” has 6,312 states and 17,283 transitions, which takes nearly 2 hours for a (Intel Core i7, 2 GHz, 6Gb RAM) PC to calculate. Since it is not possible to present all the control data due to space limitations, the input and output files to the software are given in Appendix D, so that the user can reproduce the result. Several examples of the required control actions are shown in Table 4.2.

Table 4.2 - Supervisor’s control data for “near field”

1. PLANT: [EP_control,[land,3]]

   SUPER: [[EP_control,[land,3]],[inAir,[timeout,120]]]

   EXPEDITE: land

2. PLANT: [aligned,[auto_finished,5]]

   SUPER: [[aligned,[auto_finished,5]],[inAir,[timeout,118]]]

   EXPEDITE: abort

3. PLANT: [not_aligned,[hit_the_ship_gear,8]]

   SUPER: [[not_aligned,[hit_the_ship_gear,8]],[inAir,[timeout,116]]]

   EXPEDITE: abort

4. PLANT: [dangerous,[crash,0],[hit_the_ship_gear,8]]

   SUPER: [[dangerous,[crash,0],[hit_the_ship_gear,8]],[inAir,[timeout,57]]]

   EXPEDITE: abort

5. PLANT: [dangerous,[crash,2],[hit_the_ship_gear,2]]

   SUPER: [[dangerous,[crash,2],[hit_the_ship_gear,2]],[inAir,[timeout,94]]]

   EXPEDITE: abort

6. PLANT: [dangerous,[crash,0],[hit_the_ship_gear,2]]

   SUPER: [[dangerous,[crash,0],[hit_the_ship_gear,2]],[inAir,[timeout,118]]]

   EXPEDITE: abort

Control data displays that the supervisor will expedite the Events “abort” or “land” at the last moment to prevent the UAV crashing (Example 5), hitting the ship’s gear (Examples 3 and 4), or violating the restrictions of battery life (Examples 1, 2, and 6). The persistency of the time, previously mentioned in Chapter 4.2, can be traced through the event “hit_the_ship_gear”, which can occur after coming to the States “not_aligned” and “dangerous”, and the Event “timeout”, which can occur at any state. This time persistency plays a key role in the suggested timing of expediting events in least-restrictive supervisory control.

## 4.4 Conclusion

The vision-based automatic landing system was modeled as a timed discrete-event system. The states of the DES are derived from the continuous control system, and are synchronized with its timing. The landing phase is partitioned into two fields: “far” and “near”; the states for each field are defined according to continuous or discrete-event logics.

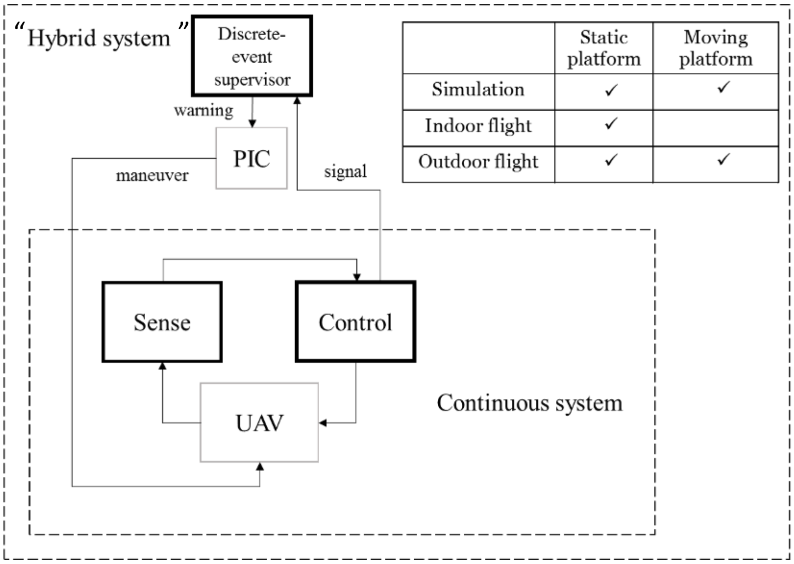
A supervisory control strategy for the automatic landing of a multi-rotor machine on a moving target is developed; the specifications for the system performance generalized the desired system behavior. The designed discrete-event supervisor that is implemented as a high-level controller prompts the human pilot-in-command to take over the control at the last moment after the exception occurs. If there is more than one eligible *expeditible* event, the warning sent from the supervisor to the PIC (Fig. 4.1) is non-deterministic in the sense that the supervisor is indifferent to a particular choice of an *expeditible* event as long as the plant is taken out of the unsuitable state before the pending tick. Thus, the least-restrictive solution allows the PIC to have maximum freedom in controlling the landing without the violation of safety constraints.

The discrete-event supervisor was synthesized for the system configuration as it was implemented in a real-time setup in Section 5, where the speed of the multi-rotor during landing was 3 m/s, and velocity of a moving target - 1 m/s. "Online" supervisor synthesis, accounting for the detailed dynamics in the continuous model, is considered as future work.

# Chapter 5

# Experimental results

The validation of the landing performance predicted by the theoretical design given in Chapters 2-4 is provided. The implementation was first in the simulation and followed by indoor and outdoor flight tests (Fig. 5.1). Indoor flight tests were done for landing on a static target; auto-landing on a moving platform was not conducted due to space limitations. The outdoor tests were conducted for both static and moving platforms under different lighting conditions and levels of wind gusts.



| | Static platform | Moving platform |
|---|---|---|
| Simulation | ✓ | ✓ |
| Indoor flight | ✓ | |
| Outdoor flight | ✓ | ✓ |

Figure 5.1 – Simulation and flight tests of the developed system

## 5.1 Simulation platform

### 5.1.1 Robot operating system

Robot operating system (ROS) is a group of software packages for the design and development of robot applications (Quigley et al., 2009). The main advantage of ROS is that it is open-source and provides the opportunity to use modeled sensors, as well as real sensors. The fundamental concepts of ROS are nodes, messages, topics, and services.

Another advantage of using ROS is that it provides operating system-like functionality. Communication between ROS processes is represented in an RQT graph (Qt-based framework for GUI development for ROS) that shows the connections between nodes that send or receive messages. Although ROS is not a real-time operating system, it is possible to integrate ROS with real-time code. This way, the same code that is used in the simulation could be applied for a real-time performance by modifying only the name of the topic, as was implemented in Chapters 5.2.1 and 5.5.1.

The main ROS libraries are C++ and Python. Altogether, ROS is built around a Linux for the reason that it has a large collection of open-source software.

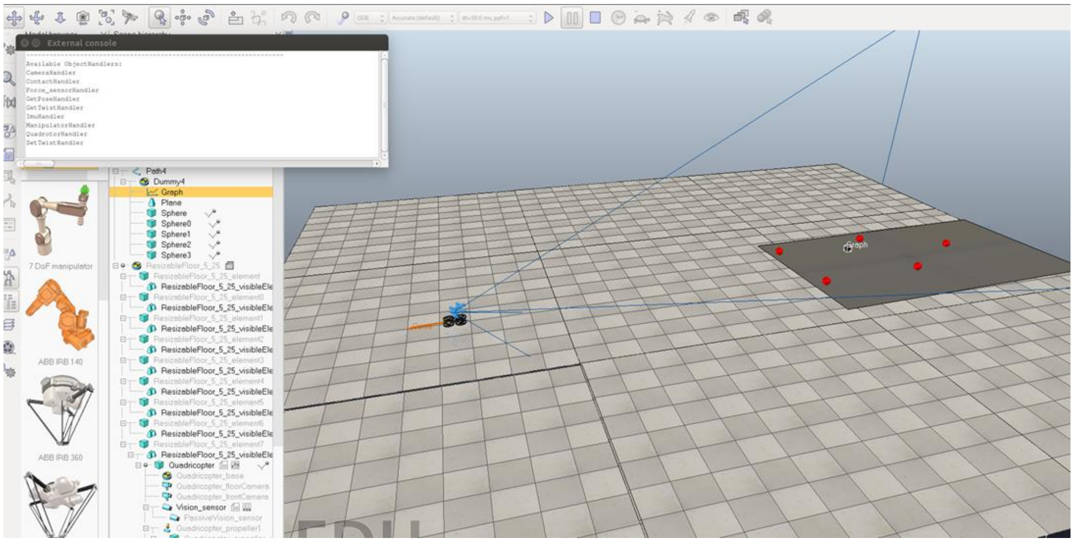
### 5.1.2 V-REP robotics simulator

V-REP (Virtual Robotics Experimentation Platform) is a robotics simulator which can be used to create applications for a robot without depending physically on the actual machine (Rohmer, Singh, & Freese, 2013). The main advantage of using such a simulator is that the applications can be transferred on the real multi-rotor without major modifications.

The robot simulator V-REP uses a distributed control architecture, such that each model can be individually controlled via a script or a ROS node. Controllers can be written in C, C++, LUA, Python, etc. In a lot of current research, V-REP is used for algorithm development, prototyping and method verification.

V-REP was chosen as a 3D robot simulator because it provides more realistic graphics than other available environments, e.g. Gazebo (Koenig & Howard, 2004). V-REP includes many different modules to model robots, their kinematics, dynamics, path planning, etc. Using V-REP, a developer can model a scene that approximates the real world by adding different objects to the simulation. V-REP is convenient to use as it allows the user to see all the interactions within the environment, the influence of disturbance, and robot position change during the algorithm performance.

The V-REP graphical interface and the scene developed for auto-landing simulation, including the moving platform, multi-rotor, vision sensor, and object pattern, are depicted in Fig. 5.2.



External console
Path4
Dummy4
Graph
Plane
Sphere
Sphere0
Sphere1
Sphere2
Sphere3
ResizableFloor_5_25
ResizableFloor_5_25_element
ResizableFloor_5_25_visibleEle
ResizableFloor_5_25_element0
ResizableFloor_5_25_element1
ResizableFloor_5_25_element2
ResizableFloor_5_25_element3
ResizableFloor_5_25_element4
ResizableFloor_5_25_element5
ResizableFloor_5_25_element6
ResizableFloor_5_25_element7
Quadricopter
Quadricopter_base
Quadricopter_floorCamera
Quadricopter_frontCamera
Vision_sensor
PassiveVision_sensor
Quadricopter_propeller1
7 DoF manipulator
ABB IRB 140
ABB IRB 360
Graph

```
------------------------------------------------------------------
Available ObjectHandlers:
CameraHandler
ContactHandler
Force_sensorHandler
GetPoseHandler
GetTwistHandler
ImuHandler
ManipulatorHandler
QuadrotorHandler
SetTwistHandler
```

Figure 5.2 – V-REP scene modeled for auto-landing

## 5.2 Simulation results

### 5.2.1 Simulation with static targets

The current simulation setup included a quadcopter model with a fixed front-facing vision sensor (512 x 256 pixels resolution), and 5 red visual markers, modeled in a V-REP scene. The visual markers were static in this experiment. The algorithm calculated corrections to control horizontal and vertical deviations based on visual data, and sent commands to the quadcopter to align with visual markers. The developed software included:

- LUA quadcopter script to receive ROS commands to control the velocity vector;
- LUA vision sensor script to send an image to the ROS node for further processing; and
- ROS node to process the image and send commands to a quadcopter.

The RQT-graph of the simulation software is depicted in Fig. 5.3.

The connections between different pieces of software are clearly seen from the graph:

1) V-REP accepts ("subscribes to") velocity commands from the main node "quad_simulation" through the ROS topic "quad_velocity_control".

2) V-REP sends ("publishes") the vision sensor data and the UAV/ platform roll and pitch angles to the main node "quad_simulation" through the topics "/vrep/visionSensorData", "/vrep/uav_pose", and "/vrep/ship_pose", respectively.

3) The main node accepts joystick commands for the manual quadcopter control through the built-in ROS "joy_node" and sends audio warnings through the "soundplay_node".

In the current implementation, the images from the camera were accepted in ROS "/sensor_msgs/Image" format. CvBridge module (Bradski & Kaehler, 2000) was chosen to pass the images to OpenCV library for further processing (Fig. 5.4).

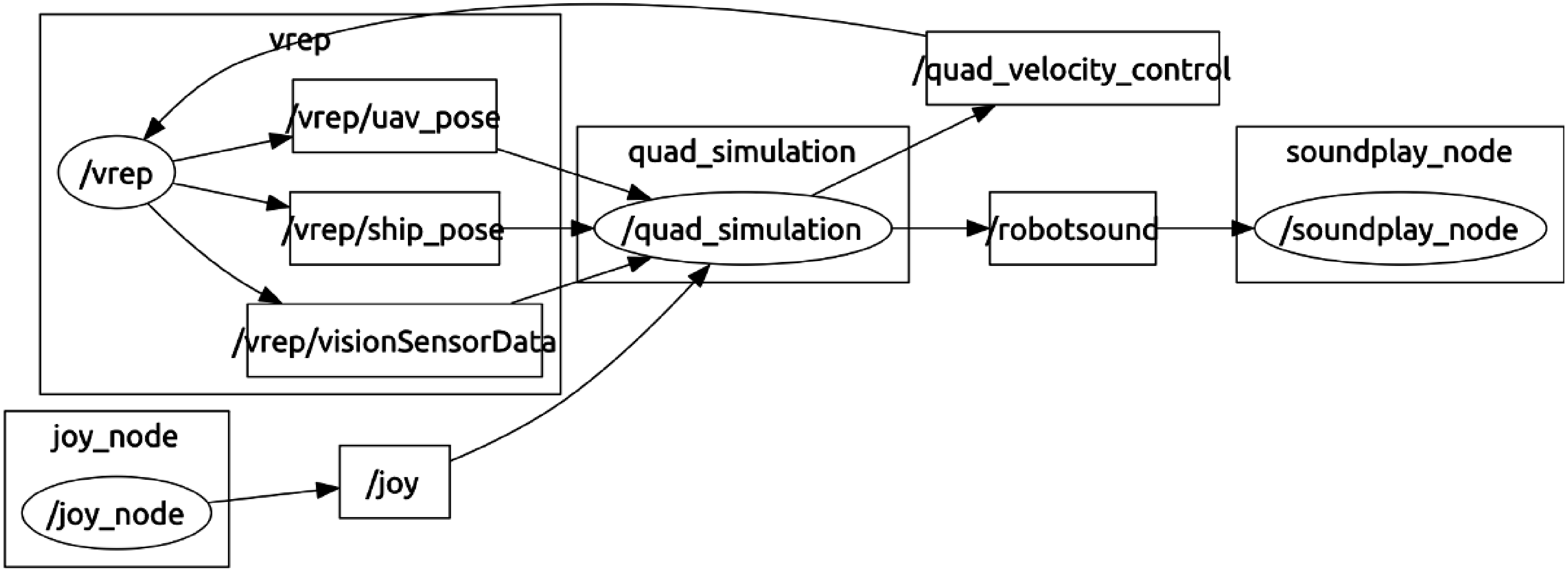


Figure 5.3 – RQT-graph of the simulation software

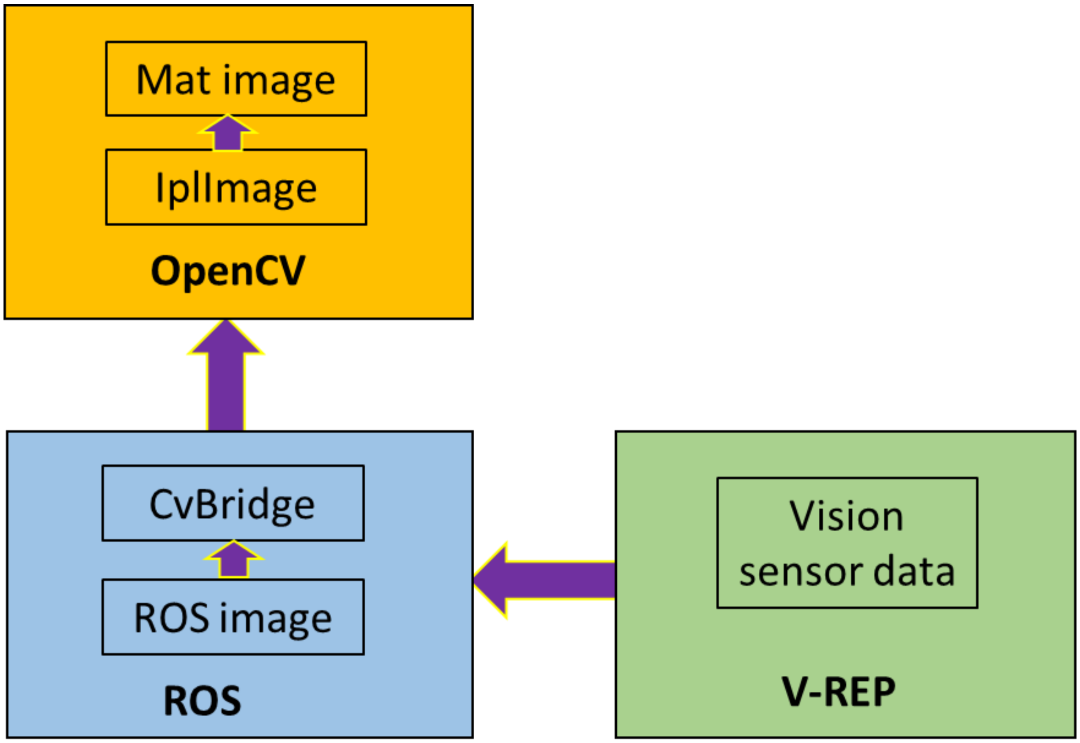


Figure 5.4 – Diagram of passing image from "vision sensor" to processing module

The V-REP simulation window is shown in Fig. 5.5. Since this simulation experiment was performed for landing on a static target without wind disturbance, the image processing was done as described in Chapter 2.2 without projecting the actual image to the virtual plane. The simulation results show that the algorithm correctly directs the quadcopter to align with visual markers based on the image data. From the following sequence of images it is seen as we pass from Image 1 to Image 5 that the quadcopter aligns with the intended static target both horizontally and vertically with an accuracy of 10 cm from the DPT. This accuracy was measured for the UAV location from the DPT, over which the UAV hovered before the external pilot performed the final touchdown.

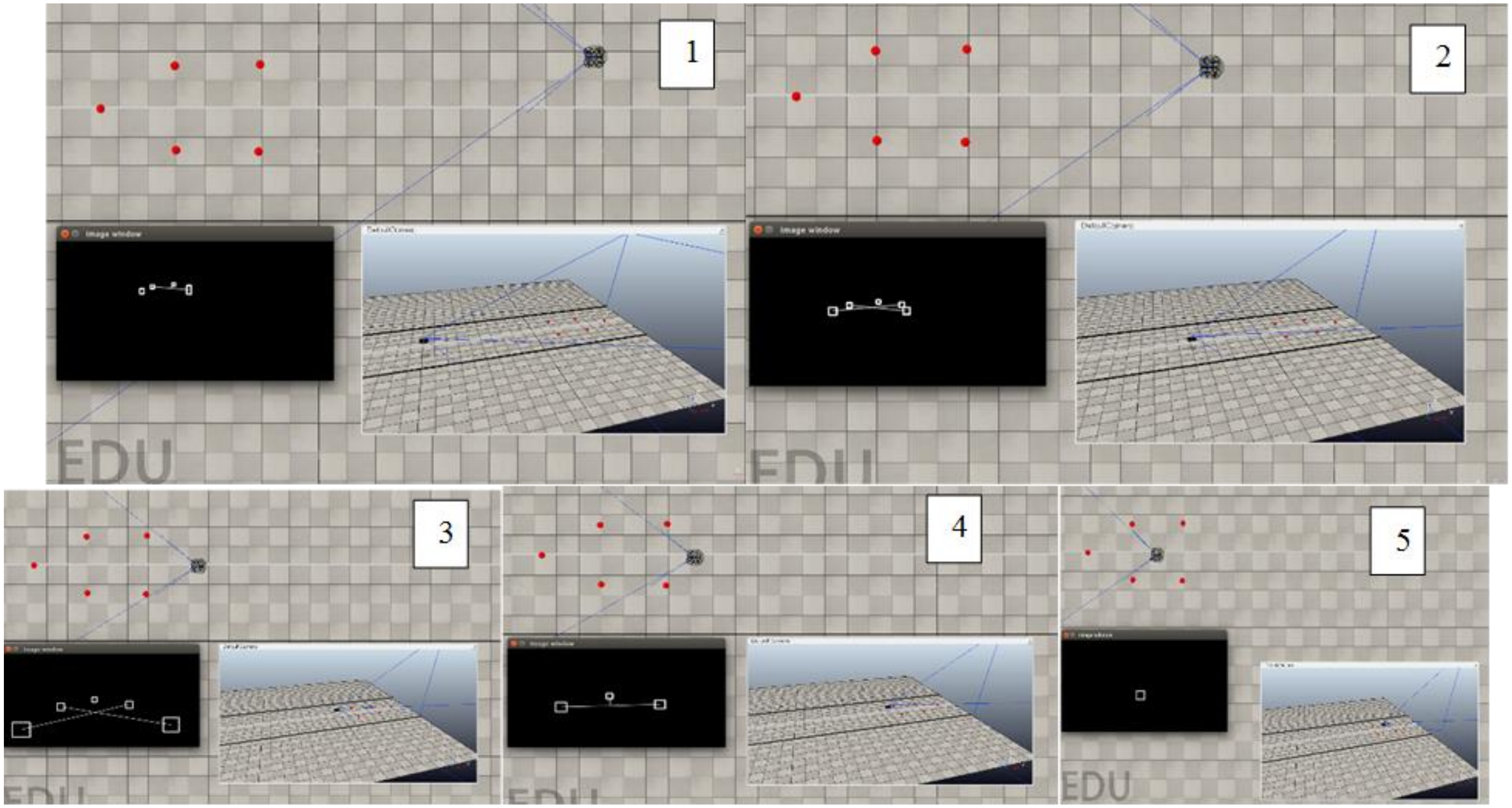


Figure 5.5 – V-REP simulation of the algorithm with static targets. Sequence of captured frames during simulation

### 5.2.2 Simulation with moving targets

In the following experiment, 5 red targets were located on the platform, which was moving forward as well as rolling/pitching and yawing with respect to the UAV. Roll and pitch angles of the moving platform, as well as the UAV roll and pitch were passed from V-REP scripts to the main ROS node for further usage by the image processing algorithm. The markers in this simulation were separated by a 3 m distance to make up a square and an equilateral triangle. The 3 m separation between the targets was chosen for simple visual estimation of the landing performance. Since the problem formulated in Chapter 1 stated that the system should come to the DPT with an accuracy of +/- 1.5 m or better, after separating the targets by 3 m from each other, one can visually identify if the system is capable of the minimum specified performance. If the UAV ends automatic alignment within the square formed by the red targets, then the minimum required performance is achieved; if the UAV ends automatic alignment outside of the square, then the minimum required performance is not achieved.

This simulation included wind and gust perturbations. Since V-REP does not simulate wind environment, wind/gust was artificially modeled as air particles and force vectors.

The image processing method in this simulation was implemented as described in Chapter 2.3, accounting for the UAV and a platform rolling, pitching, and yawing. It can be seen from Fig. 5.6 that if the controller is tuned to certain wind conditions, the system is capable of 10 cm error when coming to the DPT. This accuracy was measured for the UAV location from the DPT over which the UAV hovered before the external pilot performed the final touchdown. This accuracy was established as an average value after running about 200 simulations.

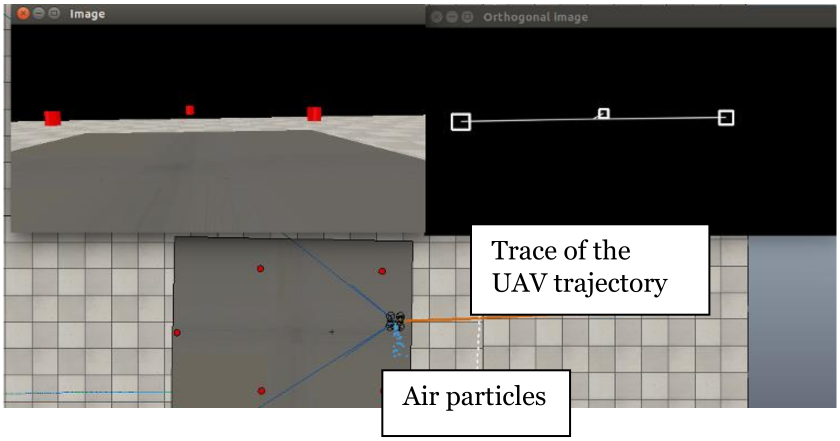


Figure 5.6 – Simulation of landing on a moving platform

The graph of the quadcopter's position change relative to the DPT is depicted in Fig. 5.7. In this figure, Z  indicates forward direction, X – horizontal direction, and Y – vertical direction. From this graph it is seen that the quadcopter aligns with the targets horizontally and vertically and the alignment errors go to zero.

The discrete-event supervisor for both the "far" and "near" fields was tested in this simulation. In the experiment for "far field", the detectability of the pattern was affected by fog which was added to the scene. In this case, the image processing algorithm could not detect a pattern (Fig. 5.8).

The system sent an audio message using a ROS “sound_play” node that indicated when to restart the mission.

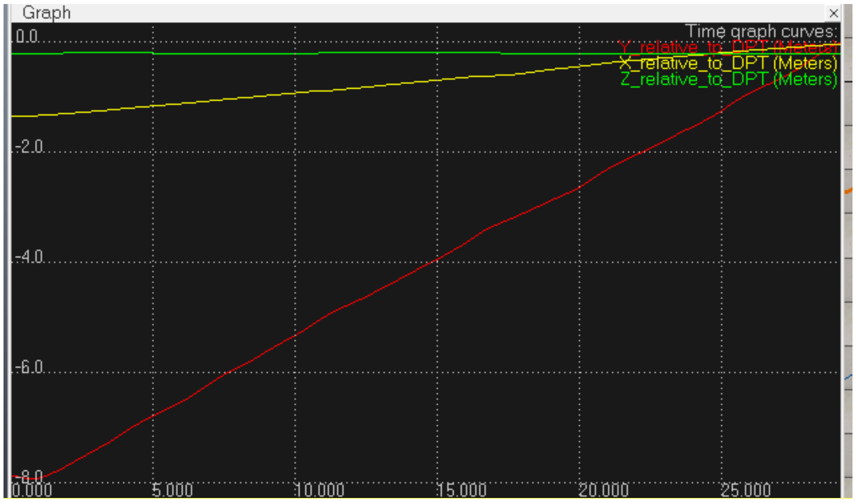


Figure 5.7 – Quadcopter’s position change relative to the DPT

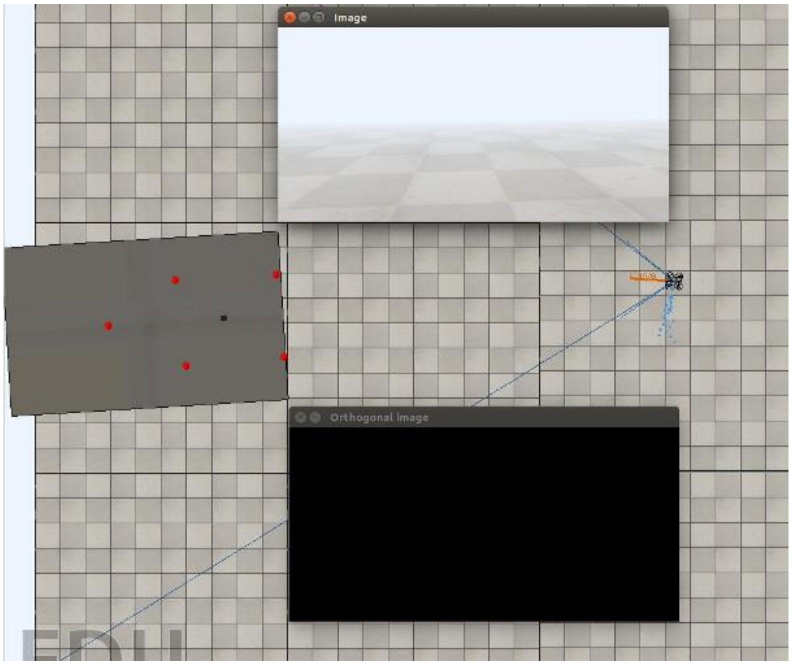


Figure 5.8 – Fog affecting pattern detection

The supervisor for “near field” was tested in the simulated scenarios when the controller was not tuned to wind/gust. The system sent audio messages to the pilot indicating when to abort the mission using the ROS “sound_play” node in the timings suggested by the discrete-event supervisor. The pilot controlled the multi-rotor model using a joystick after the exception occurred. A practical example of these exceptions is shown in Fig. 5.9.

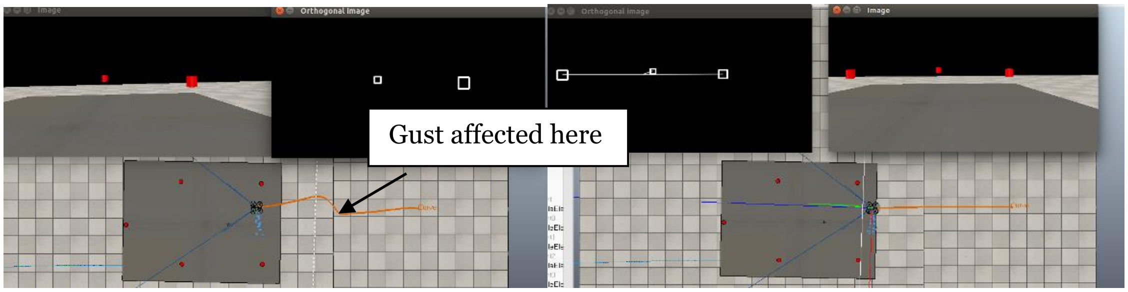


Figure 5.9 – Perturbations during landing mission:
a) gust affecting the UAV trajectory; b) wind affecting the UAV trajectory

## 5.3 Data gathering test with EPP-FPV

The experiment for data gathering and analysis was held at Argentia Airport, Placentia, NL. These experiments were held under Special Flight Operations Certificate (SFOC) for project RAVEN II. For the current experiment, five red targets were set on the runway. Red safety cones of 12” height were used as visual markers. EPP-FPV fixed-wing UAV was manually controlled by the pilot to record image data for further processing. This setup is depicted in Fig. 5.10.

During the flight tests, six manual RC flights were held, and 15 GB of image data were collected. The first prototype landing system was developed using a Raspberry Pi camera-board and Raspberry Pi (model B) processor (Richardson & Wallace, 2012). The set of images taken during the day-time data gathering experiment is depicted in Fig. 5.11. This experiment was conducted at 1 pm in August 2013.

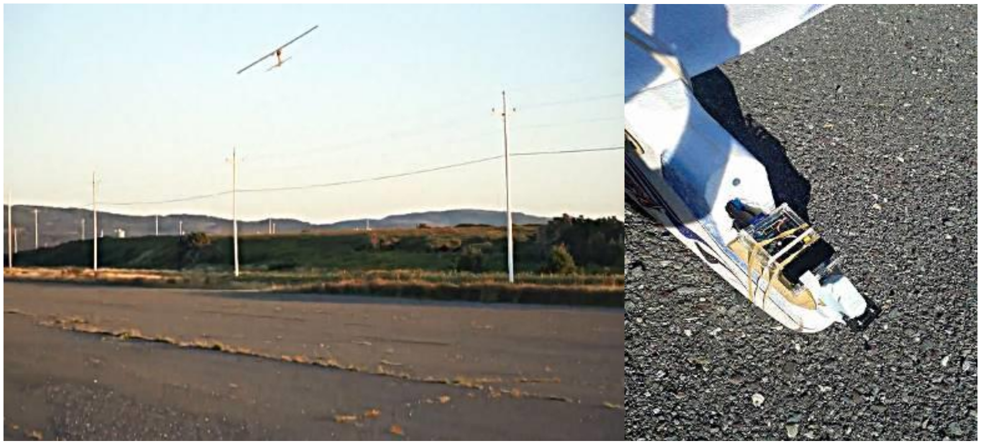


Figure 5.10 – Experimental setup, EPP-FPV UAV

The images collected in this experiment were 1280 x 960 resolution and were saved in BMP format to obtain the best processing result. The developed algorithm and processing software were able to detect the visual markers from a distance of 150 m.

Another set of experiments was done at 7 pm to collect night-time data. The red targets were equipped with red LEDs on top to ensure pattern detection in the dark.  The current setup used a Flight Site Scene Safety kit as a prototype, which was tested for helicopter landings. The experimental results are depicted in Fig. 5.12.

Analysis of the onboard data can be summarized:

1) The image processing system was not fast enough to provide the necessary speed for the data to stream to the autopilot (only 3 frames per second was achieved due to the delay for image processing and additional time to save an image).

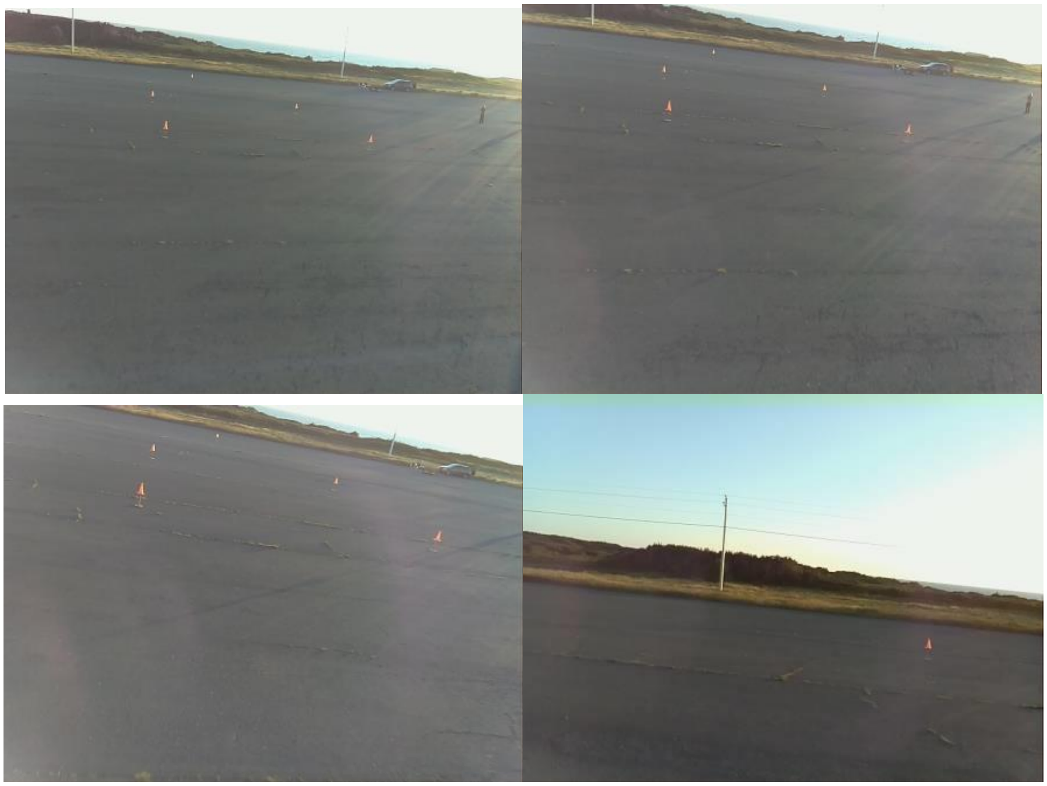


Figure 5.11 – Day-time image data from Raspberry Pi

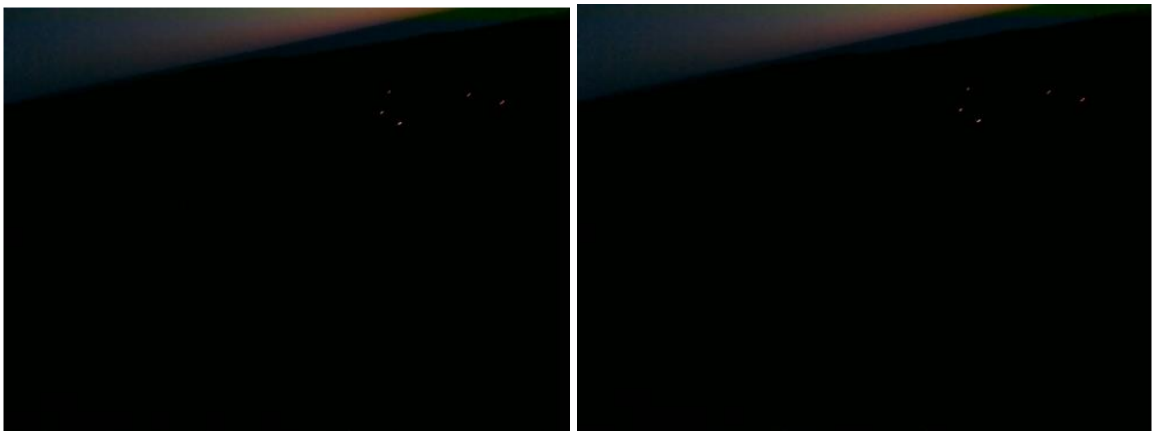


Figure 5.12 – Night-time image data from Raspberry Pi

2) The visual markers were too small to be reliably detected on the image; the detection threshold had to be tuned depending on the distance to the targets.

3) The LEDs used in the experiment were not powerful enough to allow a successful night-time landing.

4) After improving 1-2, the system could potentially provide successful pattern detection for a day-time landing from a distance of 150 m.

Based on the data gathering results, the decision was made to use red targets that are 28” that could be reliably detected on the image. It was decided that the Raspberry Pi processor would be replaced with a faster onboard computer – Odroid U3 (further described in Chapter 5.4.3).

## 5.4 Experimental platform

### 5.4.1 AR Drone 2.0

The AR Drone 2.0 (Fig. 5.13) is a multi-rotor UAV designed by Parrot, which is 23 x 0.5 x 23 inches, and weighs 4 lbs (Krajník, Vonásek, Fišer, & Faigl, 2011). In current research it is used as an indoor UAV; however, it could be flown outside in low-wind conditions. The flight time of the battery is about twelve minutes, which is acceptable for testing the auto-landing strategy indoors.

The AR Drone 2.0 is equipped with a front-facing camera with a 640 x 360 pixels sensor resolution capable of 30 frames per second. The advantage of using the AR Drone 2.0 is that it has a built-in ROS driver called “ardrone_autonomy” (Hamer, 2012); this allowed the use of the same code for both the AR Drone 2.0 landing and simulated landing as was described in Chapter 5.2.1.

Similar to the simulation setup described in Chapter 5.2.1, the AR Drone 2.0 can be controlled with a joystick connected to the Ground Control Station computer that runs the ROS driver. The image taken by the front-facing camera was transmitted from the UAV to the Ground Control Station, after which the designed ROS node sent commands to the AR Drone 2.0 through Wi-Fi from the Ground Control Station.

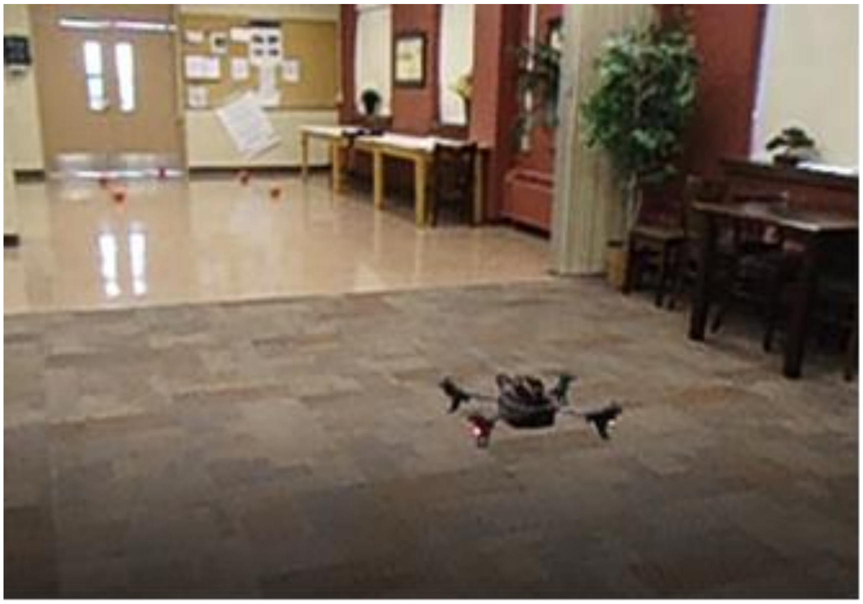


Figure 5.12 – AR Drone 2.0

### 5.4.2 S500 Hobbyking UAV

The S500 Hobbyking multi-rotor UAV is built from glass fiber and polyamide nylon. The frame weighs 405 g, and is 170 mm in height. Another advantage of the S500 is that the arms have a slight up sweep, which gives the S500 a dihedral effect. This dihedral effect helps the S500 to stabilize when descending from altitude, which is important while testing an auto-landing system. The S500 has an adjustable battery mount to achieve the desired weight distribution when installing auto-landing hardware on board. The arms of the S500 are ridged and have a carbon fibre rod through the center, making it resilient to shock and crash due to landing experiments.

The S500 Hobbyking UAV together with the auto-landing gear is depicted in Fig. 5.14.

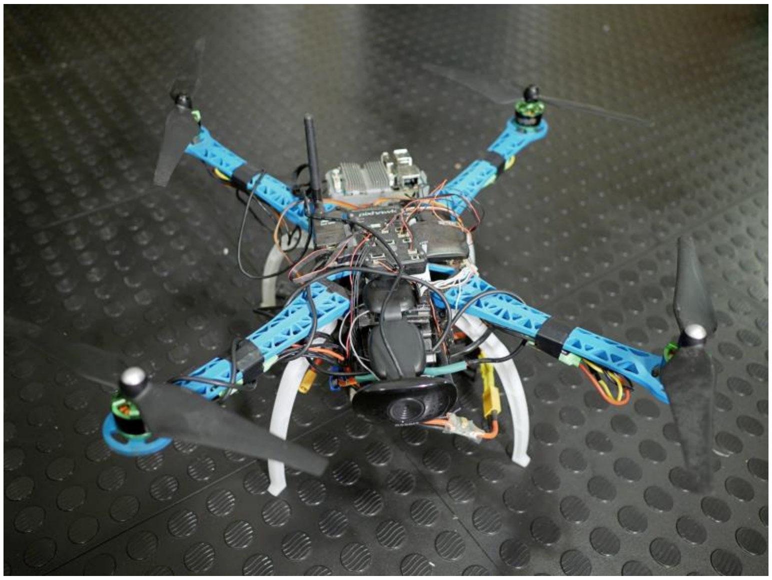


Figure 5.14 – S500 Hobbyking UAV with a developed auto-landing system

### 5.4.3 Odroid U3 setup

The Odroid U3 was chosen as it is a relatively powerful onboard computer (1.7 GHz Quad-Core processor and 2 GB RAM), that is convenient to use for programming, as it runs on XUbuntu 13.10. The Odroid U3 has 3 high speed Host USB-ports that were used to connect the camera and send commands to the autopilot. It is light-weight and small in size (83 x 48 mm, 48 g). The Odroid web-camera has a high-speed CMOS sensor, 1.0 Megapixel (1280 x 720 HD resolution), capable of capturing 30 frames/second with an FOV of 68 degrees. All together, the Odroid camera and processor make a viable setup for the prototype development of a vision-based auto-landing system that meets the size, weight, and power restrictions for operating a small multi-rotor UAV under a 2 kg maximum take-off weight under exemption rules (Transport Canada, 2014). Considering the delay due to image processing, the Odroid camera system is capable of capturing 10 frames per second, which is enough for efficient multi-rotor control.

The Odroid U3 processor and web-camera used in the auto-landing experiments are depicted in Fig. 5.15.

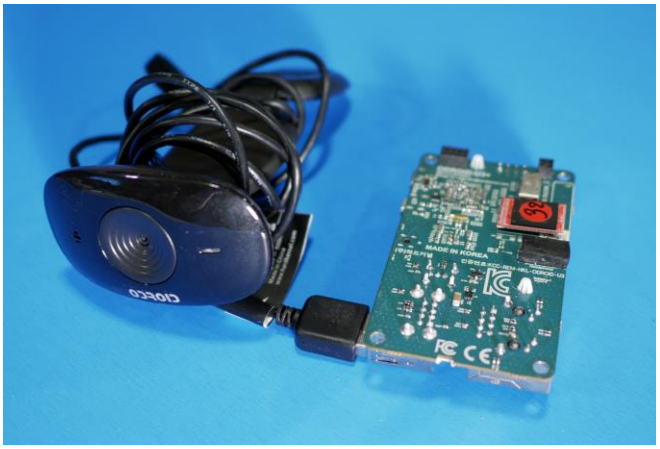


Figure 5.15 – Odroid image processing unit

### 5.4.4 Pixhawk autopilot

The Pixhawk is a high-performance autopilot suitable for fixed wing, multi rotors, helicopters, and any other robotic platforms. It consists of a PX4-FMU controller and a PX4-IO integrated on a single board. It has a safety switch, audio indicator, and multi-color LED module, which increase the reliability of the performance and allow easy understanding of any potential issue.  The Pixhawk can use both Ardupilot and PX4 firmware, providing a lot of options for the user.

The Pixhawk autopilot used for the landing experiments is depicted in Fig. 5.16.

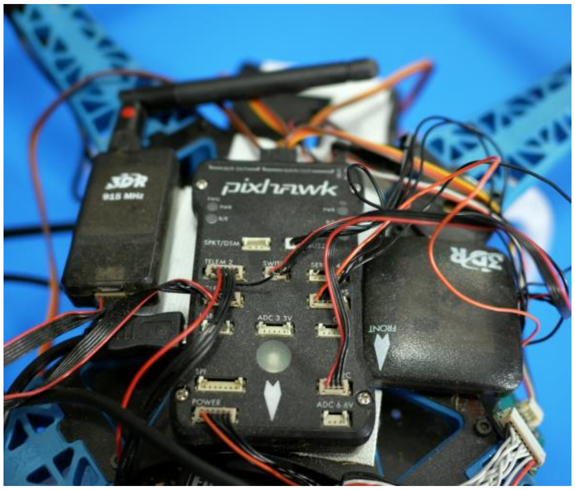


Figure 5.16 – Pixhawk autopilot

Another advantage of the Pixhawk is that it allows easy communication from the processor through Mavlink protocol (Meier et al., 2013). Together with QGroundControl (Ground Control Station software, Meier et al., 2013) the control algorithms for multi-rotors could be tested in hardware-in-the-loop (HIL) simulations with a jMAVsim library (Fig. 5.17). This HIL test was conducted prior to an actual flight to ensure the commanded maneuver was performed correctly. During the HIL experiment, the developed Odroid software sent velocity set-point commands of a different value range to the Pixhawk autopilot. The developed test-cases indicated that when the autopilot is tuned, the multi-rotor's velocity goes to the desired value, proving the possibility of effective control for visual-based automatic landing according to the developed approach.

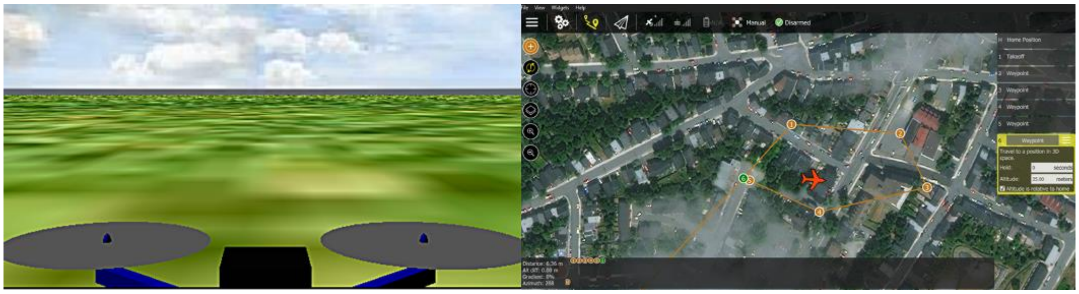


Figure 5.17 – Pixhawk in HIL with jMAVsim and QGroundcontrol

## 5.5 Real-time landing experiments

### 5.5.1 Static platform experiments with AR Drone 2.0

The developed algorithm was tested indoors with an AR Drone 2.0. The test was conducted with static targets on the ground. The image processing software was running on the Ground Station computer, which was connected to the quadcopter through Wi-Fi; an AR Drone front-facing camera was used for taking images. Since the platform was static, and the experiment was held indoors in no-wind conditions, this experiment did not use virtual projection of the image.

The results of the successful implementation of the algorithm with the AR Drone 2.0 are depicted in Fig. 5.18. In this set of experiments, the red, 10-cm-in-diameter spheres were chosen as targets in the pattern. Due to the lack of space indoors, the targets were separated with 1.5 m distance from each other to make up a square and an equilateral triangle. This lack of space could create difficulties in object pattern detection when coming close to the targets. This is why the small and low-height targets were chosen for this set of experiments.

The deviation from the desired point of touchdown was about 15 cm in both the horizontal and vertical directions. This accuracy was measured for the UAV location from the DPT, over which it would hover before the external pilot performed the final touchdown. The average accuracy value was established among 35 indoor trials.

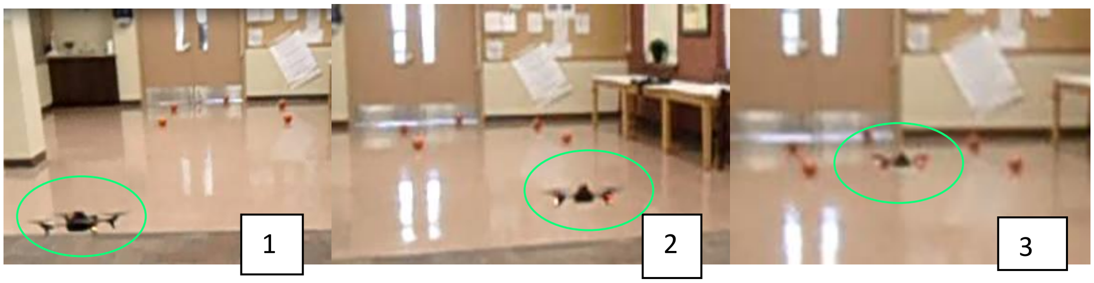


Figure 5.18 – AR Drone 2.0 correctly aligns with the static targets

The discrete-event supervisory control strategy for “far field” was tested with the AR Drone 2.0 a during set of experiments for landing on a static target. The AVO was obtaining audio warnings from the Ground Control Station computer to stop the mission when the pattern could not be detected for longer than 5 ticks (2.5 seconds). These audio messages were sent using the ROS “sound_play” node at the timing suggested by the supervisor.

### 5.5.2 Moving platform experiments with S500

The experiment for auto-landing on a moving platform was held at Witless Bay Line, St. John's, NL at around 1 pm with wind conditions of approximately 10 km/hour. The concept of operation for this set of experiments met the Transport Canada Exemptions rules (Transport Canada, 2014) for operating a UAV under 2 kg maximum take-off weight. The S500 multirotor was set to take off to Waypoint 1 at an altitude of 8 m (Fig. 5.19), and to continue to Waypoint 2 at the same altitude and speed of 3 m/s.

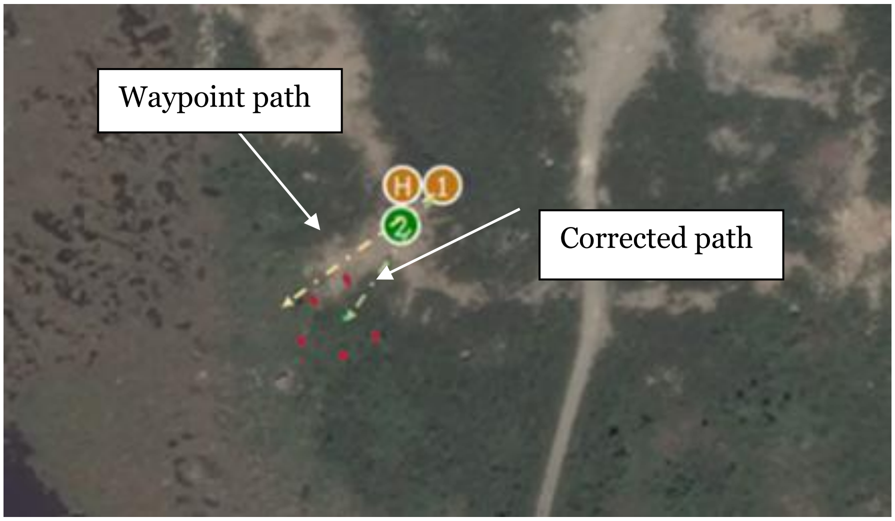


Figure 5.19 - Map of the experiment

Between these waypoints, the system was switched to the "Offboard" autopilot control for vision-based automatic alignment with the platform. The red targets (28" red safety cones) were separated by a distance of 3 m from each other, and were attached to a wooden frame that was moving with a walking speed of approximately 1 m/s. The 3 m separation between the targets was chosen for simple visual estimation of the landing performance. Since the problem formulated in Chapter 1 stated that the system should come to the DPT with an accuracy of +/- 1.5 m or better, after separating the targets by 3 m from each other, one can visually identify if the system is capable of the minimum specified performance. If the UAV ends automatic alignment within the square formed by the red targets, then the minimum required performance is achieved; if the UAV ends automatic alignment outside  of the square, then the minimum required performance is not achieved.

The vision feedback system was correcting the horizontal and vertical errors between initial path and desired path of the multirotor to align with the pattern.

The current experiment used the auto-start C++ script, which executed the image processing application after powering the Odroid board. When the targets were detected, Odroid streamed velocity set-point commands to the Pixhawk; the Pixhawk accepted the commands when the “Offboard” mode was enabled from the Ground Control Station.

After the UAV came to the DPT, all the targets (except the fifth point) were lost from the FoV. In this case, no set-point commands were streaming from Odroid; the Pixhawk went to the Position Control mode and the human pilot took over the control.

Onboard data recorded by the Odroid processor shows that the UAV was following the instructions correctly (Fig. 5.20). Since the platform was moving

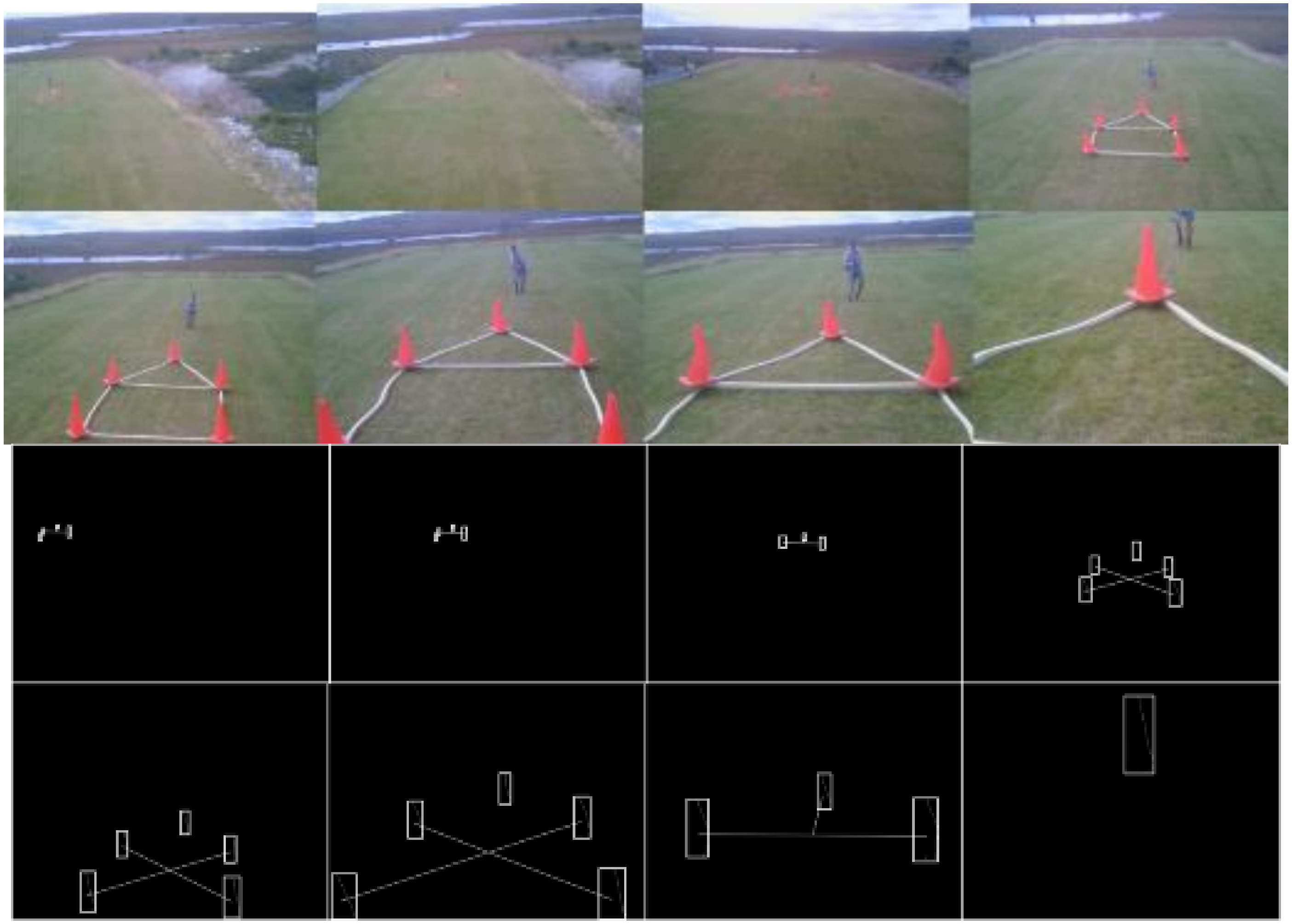


Figure 5.20 - Onboard images: a) actual; b) processed

only in the forward direction, this experiment did not use projection of the image to the virtual plane; image processing was done as described in Chapter 2.2. The first image indicates the offset to the left between the desired and initial path. However, the controller corrected the horizontal and vertical error and ensured accurate alignment with the pattern, which is seen in the sequence of images depicted in Fig. 5.20. The intersection of the lines on the processed images indicated the targeting point. The view of the landing experiment taken by the side and front cameras is depicted in Fig. 5.21. The connections between hardware components for the developed auto-landing system are given in Appendix E.

The system was tested with different wind and lighting conditions to verify the performance of the "near field" supervisory control strategy. When the targets were lost from the FoV due to inner controller detuning, the UAV would go to the Position Control mode and the system would send audio warnings instructing the pilot to take over the control at the timings suggested by the supervisor. The example of the exception when the targets are lost from the FoV is depicted in Fig. 5.22.

The experimental results show, however, that if the image processing algorithm and controller are tuned, the prototype system can provide an accuracy of about 25 cm from the location of the DPT. This accuracy was measured from the UAV location to the DPT, over which the UAV hovered before final touchdown was performed by the external pilot. The average accuracy was established among 16 outdoor trials.

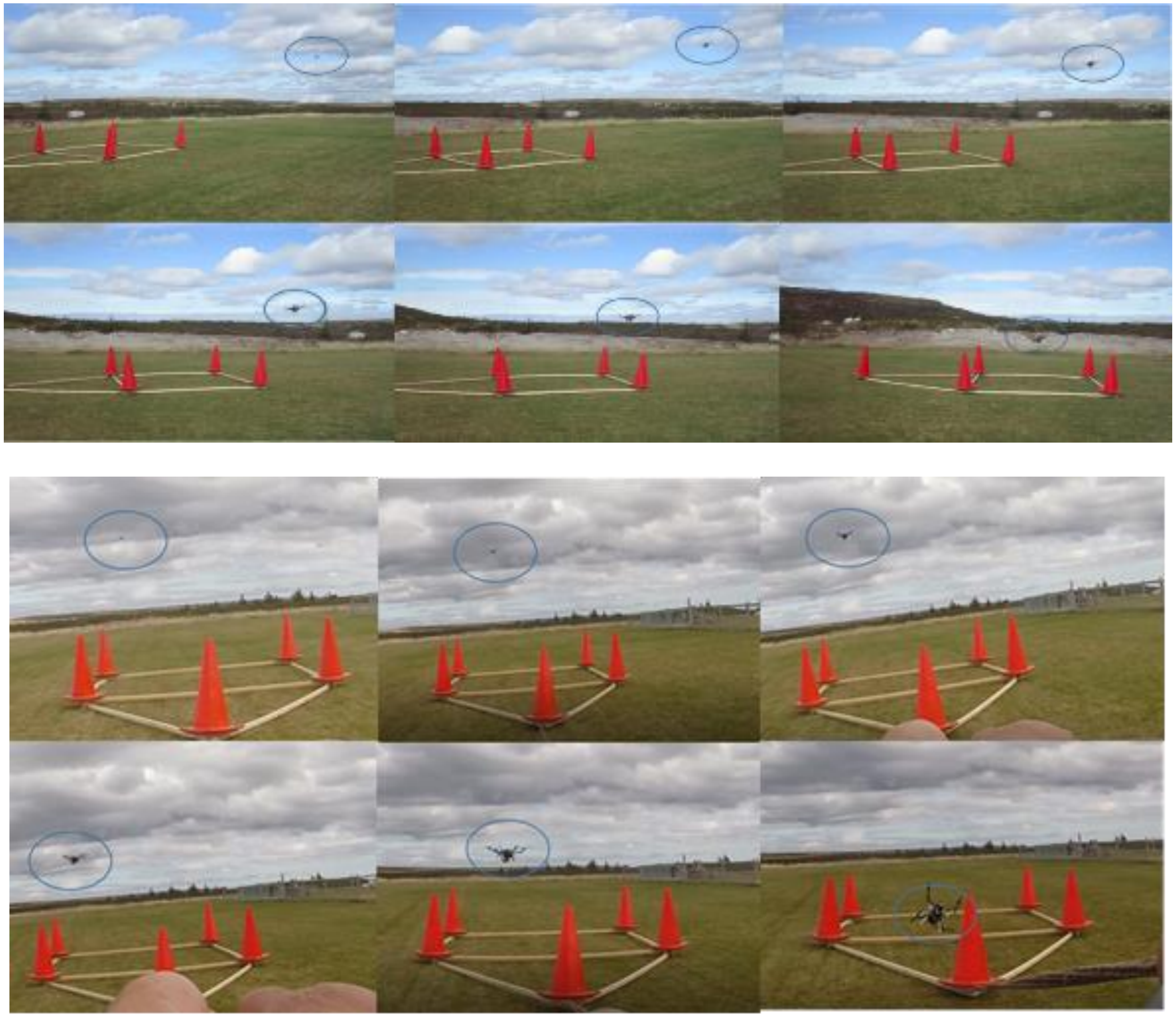


Figure 5.21 - Experimental results: a) view from the side; b) view from the front

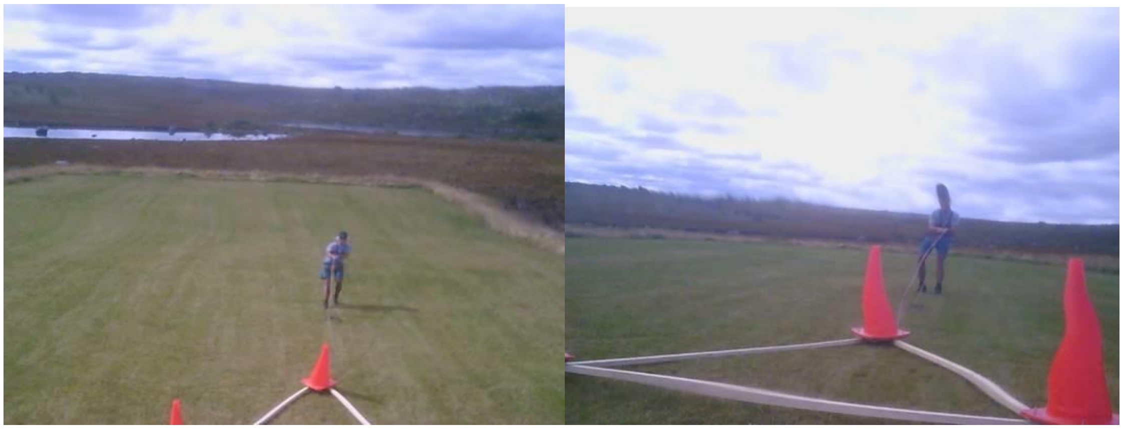


Figure 5.22 - Example of exceptions during “near field” flight

## 5.6 Conclusion

As shown by the experimental results, the suggested visual servoing approach allows an accurate landing of a multi-rotor on a moving target. Simulation experiments of landing on a moving target demonstrated an average accuracy of 10 cm from the desired point of touchdown; indoor flight-tests with no wind conditions ensured 15 cm error on average; and outdoor field tests demonstrated an accuracy of the prototype landing system of 25 cm from the desired point of touchdown on average. The errors were measured after performing 35 indoor trials, 16 outdoor tests, and nearly 200 simulations. The experimental results show that this system is robust to controller detuning, wind (gust) perturbations, and changes in lighting conditions.

# Chapter 6

# Summary and future work

## 6.1 Summary of results

A novel approach has been presented to solve the problem of an automatic landing of a multi-rotor UAV on a moving target (Fig. 6.1). Compared to existing methods the developed approach has low computational complexity. The designed IBVS scheme allows the multi-rotor to join the glide-slope and accurately align with the moving platform in a horizontal direction. Approaching the moving ship on a certain angle is important, since vertical landing of the multi-rotor can lead to hitting the ship's superstructure (e.g. masts and wires).

The developed method was tested in a series of simulations, indoor and outdoor flight tests. The prototype landing system developed according to the suggested method provided an error in excellent agreement (25 cm from the DPT). The verified method is robust to wind/gust, controller detuning, changes in lighting conditions, and battery life limitations.

The problem of automatic landing is challenging due to the following constraints:

- Possible rolling/pitching/yawing of the UAV and the platform (Technical Challenge 2 described in Chapter 1.3);

- High complexity of existent image processing techniques, which are almost impossible to implement in real-time due to SWaP restrictions for a small UAV (Technical Challenge 1);

- Requirement for the landing system to be controlled in such a way that the multi-rotor joins the glide-slope, and accurately aligns with the moving platform in a horizontal direction (Technical Challenge 3); and

- Possible exceptions that can occur during a landing mission, e.g. wind/gust perturbations, battery life limitations, and losing targets from the FoV (Technical Challenge 4).

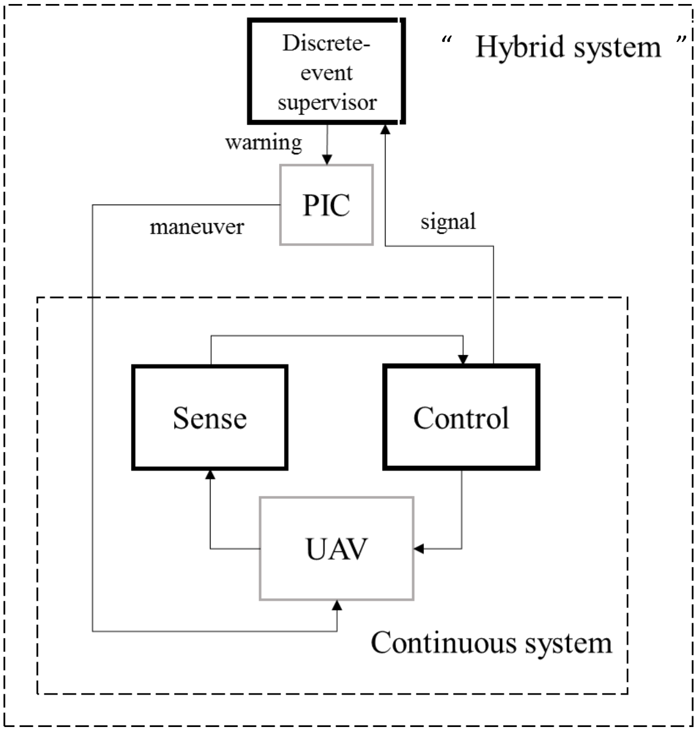


Figure 6.1 – Overall approach to the automatic landing system

A new image processing approach has been developed in Chapter 2 to overcome Technical Challenges 1 and 2. Compared to other known methods, the suggested image processing approach has low computational complexity $(\Theta(N \times M))$, which makes it possible to implement using low-cost general-purpose hardware. This algorithm also accounts for the roll, pitch, and yaw of the UAV and the moving vessel, which is essential for actual deployment.

Chapter 3 pertains to solving Technical Challenge 3. Compared to other known methods that use a down-facing camera, the current research uses a front-facing camera. The IBVS scheme is based on correction angles to ensure that the UAV follows the slopes; this method does not require 3D pose reconstruction or depth estimation, and is simple in implementation (the option of sending velocity set-point commands is available in the majority of current autopilots). Since the control laws are derived in the image-space, the developed IBVS scheme is less sensitive to camera calibration errors and is robust to the variations in cameras.

Kinematic and dynamic effects on the vision-based closed-loop control system, considering the influence of target motion and delay due to image processing, are analyzed in Chapter 3. It is shown that tracking without a steady-state error and closed-loop stability for the current vision-based system design is viable. As is shown by the experimental results in Chapter 5, the prototype system, based on such a control approach, provided 25 cm of an average error for landing on a large moving target.

In Chapter 4, the solution to Technical Challenge 4 by adding a discrete-event layer on top of the continuous control system is developed. A vision-based landing system is modeled as a timed DES; it accounts for unpredictable situations like wind, gust, controller detuning, changes in lighting conditions, and battery life limitations. After an exception occurs, the developed discrete-event supervisor will give instructions to the AVO and EP about when to abort or restart the landing mission, providing maximum freedom in performing the desirable maneuvers.

The theoretical results developed in Chapters 2-4 are validated in Chapter 5. The details of the simulation software to test vision-based landing, using V-REP and ROS, are described. This simulator can be modified to verify the landing performance of the variety of multi-rotor machines; it models wind, gust, changes in lighting conditions, and target rolling/pitching/yawing. The experimental work includes landings on both static and moving platforms. The developed method was validated using an AR Drone 2.0 indoors; the prototype landing system was developed for the landing of an outdoor quadcopter.

It is clear from Chapter 5 that the simulation results provided better accuracy than the prototype landing system. The simulation results indicated an average error of 10 cm from the DPT in both the horizontal and vertical directions after running 200 simulations. The deviation from the desired point of touchdown was 15 cm on average for the indoor static-target test with the AR Drone 2.0 in no-wind conditions after 35 indoor trials. An average error of 25 cm was obtained during the outdoor moving-platform flight test after performing 16 experiments. During these experiments, the accuracy was measured as the deviation from the DPT, over which the UAV would hover before the external pilot performed the final touchdown.

The difference in performance between the simulations and real-time experiments is connected with the idealized UAV and sensor models used in the simulation. For example, in the simulated scenario the intensity of the targets does not change with distance. This is why in the scenario of full visibility during the simulated landing, all the targets were reliably recognized on the image without detecting false alarms. Since the simulation described in Chapter 5 used idealized equations to describe the UAV motion, the performance of the simulated landing was of higher accuracy.

Chapter 5 experimentally validates that a solution to the problem stated in Chapter 1.3 has been achieved. The developed prototype landing system is low-cost ($60), small (100 x 100 mm), light-weight (under 100 g), and highly accurate (25 cm from DPT). It is robust to wind/gust perturbations, changes in lighting conditions, and inner controller detuning. The implementation of discrete-event logic in real life helps the crew to make "high-level" decisions without being burdened with lower-level system performance limitations, and reduces the impact of the human factor in making errors during a critical landing mission.

The novelty and the main contribution of this thesis is a creative combination of incremental advances in three fields: image processing, controls, and discrete-event supervisory, as applied to the UAV automatic landing.

The field-specific advances are summarized:

**Image processing:**

- The image processing method was developed to estimate the aiming point on the image using color-based detection applied to a geometric pattern. This method has lower complexity compared to other known methods. It accounts for losing targets from the field of view (FoV) when coming close to the moving platform, as well as the rolling, pitching, and yawing of the UAV and the platform.

**Control:**

- An IBVS scheme to control a UAV during landing using correction angles was developed. This approach does not require estimation of the 3D UAV position relative to the pattern, or calculation of the distance to the objects. This method is simple to implement; it provides satisfactory performance for landing on both moving and static platforms.

**Discrete-event supervisory control:**

- The landing phase was modeled as a timed discrete-event system (TDES), considering possible exceptions and flight time limitations. This modeling framework allows the modeling of the battery discharge over the sequence of states, since the events are able to persist over the state transitions.

- An optimal discrete-event supervisor was synthesized for the developed timed discrete-event landing system. The applied control mechanism is based on expediting certain control actions to avoid an unrecoverable error. According to the supervisor's control data, the EP and AVO have to take over the UAV control at the last moment after the exception occurred; this approach gives maximum freedom to a PIC in performing desirable maneuvers.

- The discrete-event layer on top of the continuous controller forms a hybrid-like system. This system is robust to wind/gust, controller detuning, and changes in lighting conditions.

**Validation:**

- A simulator for a vision-based multi-rotor landing on a static and a moving platform was developed. This simulator allows to test the landing performance of  a wide range of different multi-rotor platforms after adjusting model parameters. The simulator models wind and gust perturbations, changes in lighting conditions, and visibility constraints. In this simulator, the platform is set to roll, pitch and yaw relative to the UAV to approximate a real-case landing scenario.

- The prototype vision-based landing system was developed; it is small (100 x 100 mm), light-weight (100 g), and highly accurate (25 cm from the DPT). The auto-landing method was tested indoors and outdoors for landings on both static and moving platforms.

A full design cycle has been completed: starting from theoretical design, to modeling and simulation (M&S), then to implementation over two platforms and ending by validating the M&S predictions through first indoor and then outdoor field trials.

The results of this thesis were presented at two conferences (Borshchova, 2014; Borshchova, 2015), and documented in three journal papers (Borshchova and O'Young, 2017 a; Borshchova and O'Young, 2017 b; Borshchova and O'Young, 2017 c). This work was presented at the Student Paper Competition Unmanned Systems Canada, 2014, where it was named one of the top three student papers across Canada.

## 6.2 Future work

The possible future extensions of the current work are:

1) Improvements of the image processing method to eliminate false alarms.

A potential solution could be to use lights beyond human vision to provide reliable pattern recognition. The lights could be set to flash with a certain frequency, which can help to automatically distinguish between the "target" and the "false alarm". The detection part of the image processing algorithm should be further modified according to this proposition.

2) Developing a night-time vision-based automatic landing system.

The first experiment was held at Argentia Airport, and is described in Chapter 5.3. This idea could be extended to using powerful LEDs, compared to the LEDs used during the data gathering experiment. To complete night-time landings, further improvements of the image processing method are required.

3) Fixed-wing UAV landing.

The developed image processing method could be applied to land any rotary UAV or a fixed-wing UAV. Since fixed-wings are less stable compared to rotary wings, a different control strategy, rather than controlling the velocity vector, might be essential to compete the control part of landing. To land a fixed-wing, the control approach should consider inner autopilot loop (roll, pitch) and outer autopilot loop (bearing, altitude) interactions while landing the UAV.

4) Modification of the landing system for a GPS-denied environment.

Current research uses a GPS for the initial alignment. However, recognition of the asymmetric pattern of objects, such as a suggested pattern, can be potentially used to land the UAV in a GPS-denied environment, giving full control to visual servoing. The asymmetric structure of the suggested pattern allows to find a unique solution to estimate the UAV pose and orientation.

5) Comparison of the landing behaviour using different control strategies.

The current approach uses a proportional controller; however, it is useful to research and compare the reliability of landing using different control strategies. The controller tuning during flight tests of the prototype system was done manually. Future work should conduct the theoretical design of an optimal controller that provides the desired behavior under a wide range of disturbances, considering the nonlinear dynamics of a multi-rotor and couplings within IBVS control laws.

6) Extend the developed landing approach to use a gimbaled camera.

The developed methods use a fixed, nose-mounted single camera. To improve the continuous system performance during landing in high-wind conditions, a gimbaled camera could be considered to correct for large roll/pitch deviations to prevent losing targets from the field of view.

7) Flight test of landing on a moving boat.

The developed prototype system was tested for automatic landing on a platform that was only moving forward. Future work should extend to automatic landing on an actual moving ship, while accounting for the platform rolling, pitching, and yawing using the virtual orthogonal camera approach described in Chapter 2.3.

8) Fixed-wing UAV and PIC modeling.

A general solution to the problem of automatic landing is given in Fig. 6.1. This thesis included a theoretical design and practical results in the areas “Sense”, “Control”, and “Discrete-event supervision”. Future work should cover the remaining blocks, which are fixed-wing UAV modeling, and modeling of human factor (PIC).

# References

Agin, G. J. (1979). Real time control of a robot with a mobile camera: SRI International.

Asl, H. J., & Bolandi, H. (2014). Robust vision-based control of an underactuated flying robot tracking a moving target. Transactions of the Institute of Measurement and Control, 36(3), 411-424.

Asl, H. J., Oriolo, G., & Bolandi, H. (2014). An adaptive scheme for image-based visual servoing of an underactuated UAV. International Journal of Robotics and Automation, 29(1), 92-104.

Azrad, S., Kendoul, F., & Nonami, K. (2010). Visual servoing of quadrotor micro-air vehicle using color-based tracking algorithm. Journal of System Design and Dynamics, 4(2), 255-268.

Baggio, D. L. (2012). Mastering OpenCV with practical computer vision projects: Packt Publishing Ltd.

Bird, I. (2013). Hydrodynamic Impact Analysis and Testing of an Unmanned Aerial Vehicle.

Borshchova, I. (2014). A vision based autonomous landing system for mini UAVs, Unmanned Systems Canada, Montreal, Nov. 2014.

Borshchova, I. (2015). Visual servoing for a uav autonomous landing, Unmanned Systems Canada, Halifax, Nov. 2015.

Borshchova, I., O'Young, S. (2017 a). Visual Servoing for Autonomous Landing of a Multi-rotor UAS on a Moving Platform, Journal of Unmanned Vehicle Systems, 5(1): 13-26, https://doi.org/10.1139/juvs-2015-0044.

Borshchova, I., O'Young, S. (2017 b). Marker-guided auto-landing on a moving platform, International Journal of Intelligent Unmanned Systems, 5(1): pp.28-42, https://doi.org/10.1108/IJIUS-10-2016-0008.

Borshchova, I., O'Young, S. (2017 c). Discrete-event system for a UAV automated landing, IEEE TCST (under review).

Bourquardez, O., & Chaumette, F. (2007). Visual servoing of an airplane for auto-landing. Paper presented at the Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on.

Bradski, G., & Kaehler, A. (2000). OpenCV. Dr. Dobb's Journal of Software Tools.

Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library: " O'Reilly Media, Inc.".

Brandin, B. A., & Wonham, W. M. (1994). Supervisory control of timed discrete-event systems. IEEE Transactions on Automatic Control, 39(2), 329-342.

Brave, Y., & Heymann, M. (1988, December). Formulation and control of real time discrete event processes. In Decision and Control, 1988., Proceedings of the 27th IEEE Conference on (pp. 1131-1132). IEEE.

Canny, J. (1986). A computational approach to edge detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on(6), 679-698.

Chamberlain, L., Scherer, S., & Singh, S. (2011). Self-aware helicopters: Full-scale automated landing and obstacle avoidance in unmapped environments.

Paper presented at the Proceedings of the 67th Annual Forum of the American Helicopter Society, Virginia Beach, VA.

Chaumette, F., & Hutchinson, S. (2006). Visual servo control. I. Basic approaches. IEEE Robotics & Automation Magazine, 13(4), 82-90.

Corke, P. I. (1994). High-performance visual closed-loop robot control. University of Melbourne.

Corke, P. I., & Hutchinson, S. A. (2000). Real-time vision, tracking and control. Paper presented at the Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on.

Corke, P. I., & Hutchinson, S. A. (2001). A new partitioned approach to image-based visual servo control. IEEE Transactions on Robotics and Automation, 17(4), 507-515.

De Luca, A., Oriolo, G., & Giordano, P. R. (2008). Feature depth observation for image-based visual servoing: Theory and experiments. The International Journal of Robotics Research, 27(10), 1093-1116.

Ding, W., Gong, Z., Xie, S., & Zou, H. (2006). Real-time vision-based object tracking from a moving platform in the air. Paper presented at the Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on.

Dusha, D., Boles, W. W., & Walker, R. (2007). Fixed-wing attitude estimation using computer vision based horizon detection.

Elfes, A., Bueno, S. S., Bergerman, M., & Ramos, J. G. (1998). A semi-autonomous robotic airship for environmental monitoring missions. Paper presented at the Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on.

Fitzgerald, D., Walker, R., & Campbell, D. (2005). A vision based forced landing site selection system for an autonomous UAV. Paper presented at the Intelligent Sensors, Sensor Networks and Information Processing Conference, 2005. Proceedings of the 2005 International Conference on.

Freeman, W. T., & Adelson, E. H. (1991). The design and use of steerable filters. IEEE Transactions on Pattern analysis and machine intelligence, 13(9), 891-906.

García Carrillo, L.-R., Rondon, E., Sanchez, A., Dzul, A., & Lozano, R. (2011). Stabilization and trajectory tracking of a quad-rotor using vision. Journal of Intelligent & Robotic Systems, 61(1), 103-118.

Geetha, A., & Murali, S. (2013). Automatic rectification of perspective distortion from a single image using plane homography. IJCSA, 3(5), 47-58.

Golaszewski, C., & Ramadge, P. J. (1987). Control of discrete event processes with forced events. Paper presented at the Decision and Control, 1987. 26th IEEE Conference on.

Gonzalez, R. C., & Woods, R. E. (2007). Image processing. Digital image processing, 2.

Hamel, T., & Mahony, R. (2002). Visual servoing of an under-actuated dynamic rigid-body system: an image-based approach. Robotics and Automation, IEEE Transactions on, 18(2), 187-198.

Hamer, M. (2012). AutonomyLab/ardrone_autonomy· GitHub.

Hashimoto, K., Kimoto, T., Ebine, T., & Kimura, H. (1991). Manipulator control with image-based visual servo. Paper presented at the Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on.

He, Z., Iyer, R. V., & Chandler, P. R. (2006). Vision-based UAV flight control and obstacle avoidance. Paper presented at the American Control Conference, 2006.

Heikkila, J., & Silvén, O. (1997). A four-step camera calibration procedure with implicit image correction. Paper presented at the Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on.

Jabbari Asl, H., Oriolo, G., & Bolandi, H. (2014). Output feedback image-based visual servoing control of an underactuated unmanned aerial vehicle. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 228(7), 435-448.

Kamate, S., & Yilmazer, N. (2015). Application Of Object Detection And Tracking Techniques For Unmanned Aerial Vehicles. Procedia Computer Science, 61, 436-441.

Kannala, J., Heikkilä, J., & Brandt, S. S. (2008). Geometric camera calibration. Wiley Encyclopedia of Computer Science and Engineering.

Kendoul, F. (2012). Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. Journal of Field Robotics, 29(2), 315-378.

Koenig, N., & Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. Paper presented at the Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on.

Krajník, T., Vonásek, V., Fišer, D., & Faigl, J. (2011). AR-drone as a platform for robotic research and education. Paper presented at the International Conference on Research and Education in Robotics.

Kumar, R., Chung, H. M. and Marcus, S. I. (1998). Extension based limited lookahead supervision of discrete event systems, Automatica 34(11), 1327-1344.

Kumar, R. and Garg, V. K. (1995). Modeling and Control of Logical Discrete Event Systems, Kluwer Academic Publishers.

Laganière, R. (2014). OpenCV Computer Vision Application Programming Cookbook Second Edition: Packt Publishing Ltd.

Lee, D., Ryan, T., & Kim, H. J. (2012). Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing. Paper presented at the Robotics and Automation (ICRA), 2012 IEEE International Conference on.

Lin, F., Vaz, A., & Wonham, W. M. (1988). Supervisor specification and synthesis for discrete event systems. International Journal of Control, 48(1), 321-332.

Lippiello, V., Siciliano, B., & Villani, L. (2007). Position-based visual servoing in industrial multirobot cells using a hybrid camera configuration. IEEE Transactions on Robotics, 23(1), 73-86.

Liu, W., & Li, S. (2012). An effective lane detection algorithm for structured road in urban. Paper presented at the International Conference on Intelligent Science and Intelligent Data Engineering.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), 91-110.

Malis, E., & Rives, P. (2003). Robustness of image-based visual servoing with respect to depth distribution errors. Paper presented at the Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on.

Mariottini, G. L., Oriolo, G., & Prattichizzo, D. (2007). Image-based visual servoing for nonholonomic mobile robots using epipolar geometry. IEEE Transactions on Robotics, 23(1), 87-100.

Meier, L., Camacho, J., Godbolt, B., Goppert, J., Heng, L., & Lizarraga, M. (2013). Mavlink: Micro air vehicle communication protocol. Online]. Tillgänglig: http://qgroundcontrol. org/mavlink/start.[Hämtad 2014-05-22].

Meng, D., Yun-feng, C., & Lin, G. (2006). A method to recognize and track runway in the image sequences based on template matching. Paper presented at the Systems and Control in Aerospace and Astronautics, 2006. ISSCAA 2006. 1st International Symposium on.

Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors. IEEE transactions on pattern analysis and machine intelligence, 27(10), 1615-1630.

Millan, J. P. (2006). Online discrete event control of hybrid systems. Memorial University of Newfoundland.

Millan, J. and O'Young, S. (2000). Hybrid modeling of tandem dynamically positioned vessels, Proceedings of the 39th IEEE Conference on Decision and Control.

Millan, J. and O'Young, S. (2006). Hybrid system control using an online discrete event supervisory strategy, IFAC Conference on Analysis and Design of Hybrid Systems, IFAC.

Millan, J. and O'Young, S. (2006). On-line supervisory control of hybrid systems using embedded simulations, Proceedings of the 8th International Workshop on Discrete Event Systems WODES06.

Millan, J., Smith, L. and O'Young, S. (2002). Coordination of FPSO and tanker offloading operations, Proceedings of the MTS Dynamic Positioning Conference 2002.

Mondragon, I. F., Campoy, P., Correa, J. F., & Mejias, L. (2007). Visual model feature tracking for UAV control. Paper presented at the Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on.

Nex, F., & Remondino, F. (2014). UAV for 3D mapping applications: a review. Applied Geomatics, 6(1), 1-15.

Nonami, K., Kendoul, F., Suzuki, S., Wang, W., & Nakazawa, D. (2010). Autonomous flying robots: unmanned aerial vehicles and micro aerial vehicles: Springer Science & Business Media.

Ogata, K. (1995). Discrete-time control systems. Prentice Hall International, Inc.

Ogata, K. (1970). Modern control engineering. Prentice Hall.

Olivares Mendez, M. A., Kannan, S., & Voos, H. (2014). V-REP & ROS Testbed for Design, Test, and Tuning of a Quadrotor Vision Based Fuzzy Control System for Autonomous Landing. Paper presented at the Porceedings of The International Micro Air Vehicle Conference and Competition 2014.

O'Young, S. (1991). On the synthesis of the supervisors for timed discrete event processes, Technical Report 9107, Department of Electrical and Computer Engineering, University of Toronto.

O'Young, S. D. (1992). Systems control group report, Object TCT: Users guide, Technical report, Department of Electrical Engineering, University of Toronto.

Palmieri, G., Palpacelli, M., Battistelli, M., & Callegari, M. (2012). A comparison between position-based and image-based dynamic visual servoings in the control of a translating parallel manipulator. Journal of Robotics, 2012.

Park, D.-H., Kwon, J.-H., & Ha, I.-J. (2012). Novel position-based visual servoing approach to robust global stability under field-of-view constraint. IEEE Transactions on Industrial Electronics, 59(12), 4735-4752.

Park, J. S., & Chung, M. J. (2003). Path planning with uncalibrated stereo rig for image-based visual servoing under large pose discrepancy. IEEE Transactions on Robotics and Automation, 19(2), 250-258.

Pérez, P., Hue, C., Vermaak, J., & Gangnet, M. (2002). Color-based probabilistic tracking. Paper presented at the European Conference on Computer Vision.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., . . . Ng, A. Y. (2009). ROS: an open-source Robot Operating System. Paper presented at the ICRA workshop on open source software.

Ramadge, P. J., & Wonham, W. M. (1989). The control of discrete event systems. Proceedings of the IEEE, 77(1), 81-98.

Richardson, M., & Wallace, S. (2012). Getting started with raspberry PI: " O'Reilly Media, Inc.".

Rohmer, E., Singh, S. P., & Freese, M. (2013). V-REP: A versatile and scalable robot simulation framework. Paper presented at the Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on.

Ruffier, F., & Franceschini, N. (2004). Visually guided micro-aerial vehicle: automatic take off, terrain following, landing and wind reaction. Paper presented at the Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on.

Saggiani, G., & Teodorani, B. (2004). Rotary wing UAV potential applications: an analytical study through a matrix method. Aircraft Engineering and Aerospace Technology, 76(1), 6-14.

Sanchez-Lopez, J. L., Pestana, J., Saripalli, S., & Campoy, P. (2014). An approach toward visual autonomous ship board landing of a VTOL UAV. Journal of Intelligent & Robotic Systems, 74(1-2), 113-127.

Se, S., Lowe, D., & Little, J. (2001). Local and global localization for mobile robots using visual landmarks. In Intelligent Robots and Systems, 2001.

Proceedings. 2001 IEEE/RSJ International Conference on (Vol. 1, pp. 414-420). IEEE.

Shademan, A., & Janabi-Sharifi, F. (2005). Sensitivity analysis of EKF and iterated EKF pose estimation for position-based visual servoing. Paper presented at the Control Applications, 2005. CCA 2005. Proceedings of 2005 IEEE Conference on.

Silveira, G. F., Azinheira, J. R., Rives, P., & Bueno, S. S. (2003). Line following visual servoing for aerial robots combined with complementary sensors. Paper presented at the Proc. IEEE International Conference on Advanced Robotics. Coimbra, Portugal.

Silveira, G. F., Carvalho, J. R. H., Madrid, M. K., Bueno, S. S., & Rives, P. (2001). Towards vision guided navigation of autonomous aerial robots. Paper presented at the Proceedings of the IV Brazilian Symposium on Intelligent Automation.

Stevenson, J. D., O'Young, S., & Rolland, L. (2015). Beyond line of sight control of small unmanned aerial vehicles using a synthetic environment to augment first person video. Procedia Manufacturing, 3, 960-967.

Teuliere, C., Eck, L., & Marchand, E. (2011). Chasing a moving target from a flying UAV. Paper presented at the Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on.

Thuilot, B., Martinet, P., Cordesses, L., & Gallice, J. (2002). Position based visual servoing: keeping the object in the field of vision. Paper presented at the

Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on.

Transport Canada (2014), “Guidance material for operating unmanned air vehicle systems under exemptions”, Advisory Circular, No. 600-004.

Valavanis, K. P., & Vachtsevanos, G. J. (2014). Handbook of unmanned aerial vehicles: Springer Publishing Company, Incorporated.

Vinukonda, P. (2011). A study of the scale-invariant feature transform on a parallel pipeline. Louisiana State University.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Paper presented at the Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on.

Watanabe, Y., Lesire, C., Piquereau, A., Fabiani, P., Sanfourche, M., & Le Besnerais, G. (2010, May). The onera ressac unmanned autonomous helicopter: Visual air-to-ground target tracking in an urban environment. In American Helicopter Society 66th Annual Forum (AHS 2010).

Whitworth, C., Duller, A., Jones, D., & Earp, G. (2001). Aerial video inspection of overhead power lines. Power Engineering Journal, 15(1), 25-32.

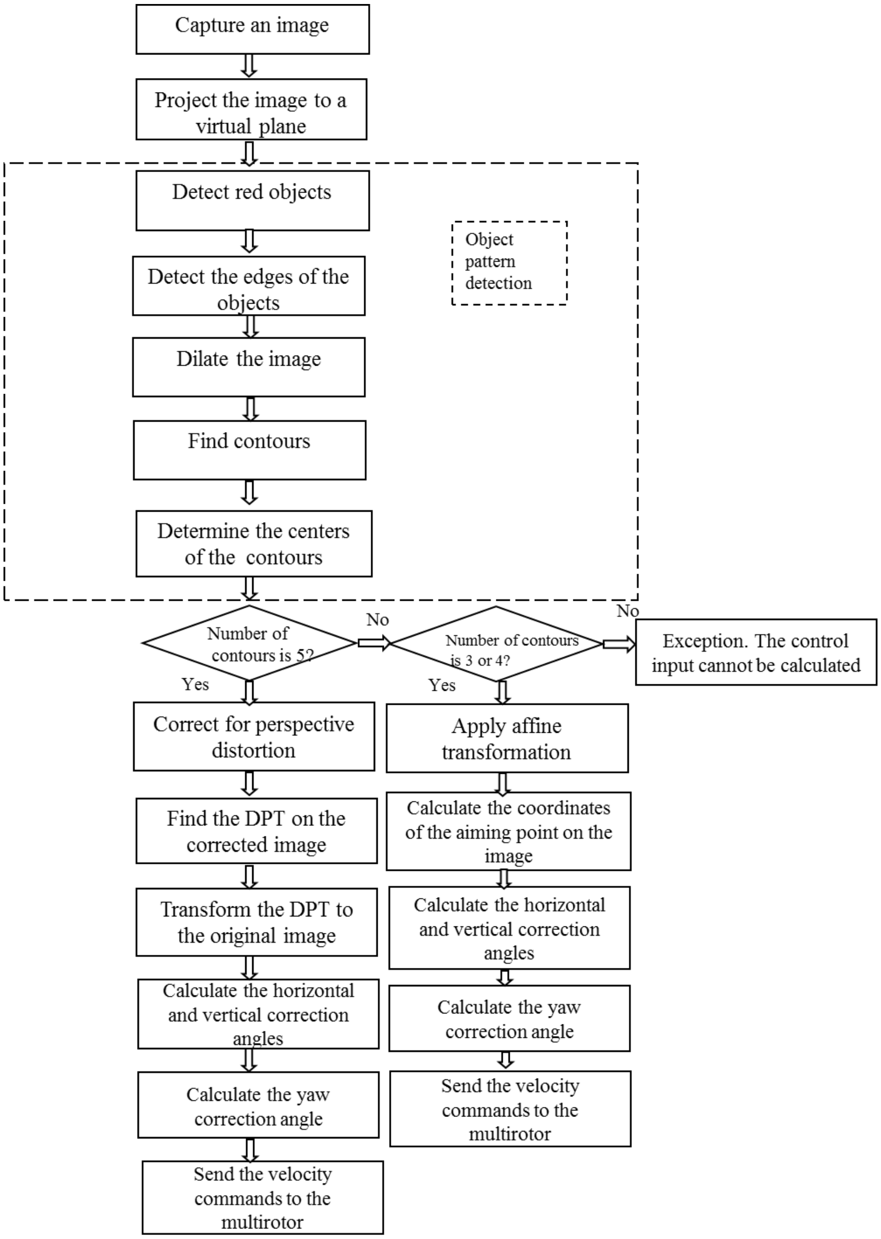
Wilson, W. J., Hulls, C. W., & Bell, G. S. (1996). Relative end-effector control using cartesian position based visual servoing. IEEE Transactions on Robotics and Automation, 12(5), 684-696.

Wonham, W. M., & Ramadge, P. J. (1987). On the supremal controllable sublanguage of a given language. SIAM Journal on Control and Optimization, 25(3), 637-659.

Zhou, G., & Zhou, B. (2015). Forecasting Method for Ship-borne Helicopter deck-landing. Paper presented at the International Conference on Education, Management, Commerce and Society.

# Appendix A

## Flow chart of the software


Capture an image
Project the image to a virtual plane
Detect red objects
Detect the edges of the objects
Dilate the image
Find contours
Determine the centers of the contours
Object pattern detection
Number of contours is 5?
No
Yes
Number of contours is 3 or 4?
No
Yes
Exception. The control input cannot be calculated
Correct for perspective distortion
Find the DPT on the corrected image
Transform the DPT to the original image
Calculate the horizontal and vertical correction angles
Calculate the yaw correction angle
Send the velocity commands to the multirotor
Apply affine transformation
Calculate the coordinates of the aiming point on the image
Calculate the horizontal and vertical correction angles
Calculate the yaw correction angle
Send the velocity commands to the multirotor

# Appendix B

## Input and output files for “far field”*

**UAV.ttm**

```
initial.
[normal].
marker.
[normal].
tran.
[[normal,  lose, abnormal],
[abnormal, regain, normal],
[abnormal, stop_mission, reset],
[abnormal,  crash, forbidden],
[reset,  restart_mission, normal]].
timer.
[crash, 6, inf].
forcible.
[[stop_mission],
[restart_mission]].
controllable.
[[stop_mission],
[restart_mission].
```

**Spec.fsm**

```
initial .
[allowed].
marker.
[allowed].
tran.
[allowed,  crash, not_allowed].
```

**Run.txt**

```
plant = ttm(uav.ttm)
spec = fsm(Spec.fsm)
sup = supfcBySync(plant,spec)
condat(plant,sup,control.txt)
printWithMap(spec,spec.pri)
printWithMap(sup,sup.pri)
printWithMap(plant,plant.pri)
printAsPDF(spec, spec.pdf)
printAsPDF(plant, uav.pdf)
printAsPDF(sup, sup.pdf)
```

**Control.txt**

```
PLANT: [abnormal,[crash,5]]
SUPER: [[abnormal,[crash,5]],allowed]
EXPEDITE:  stop_mission
```

**tct.log**

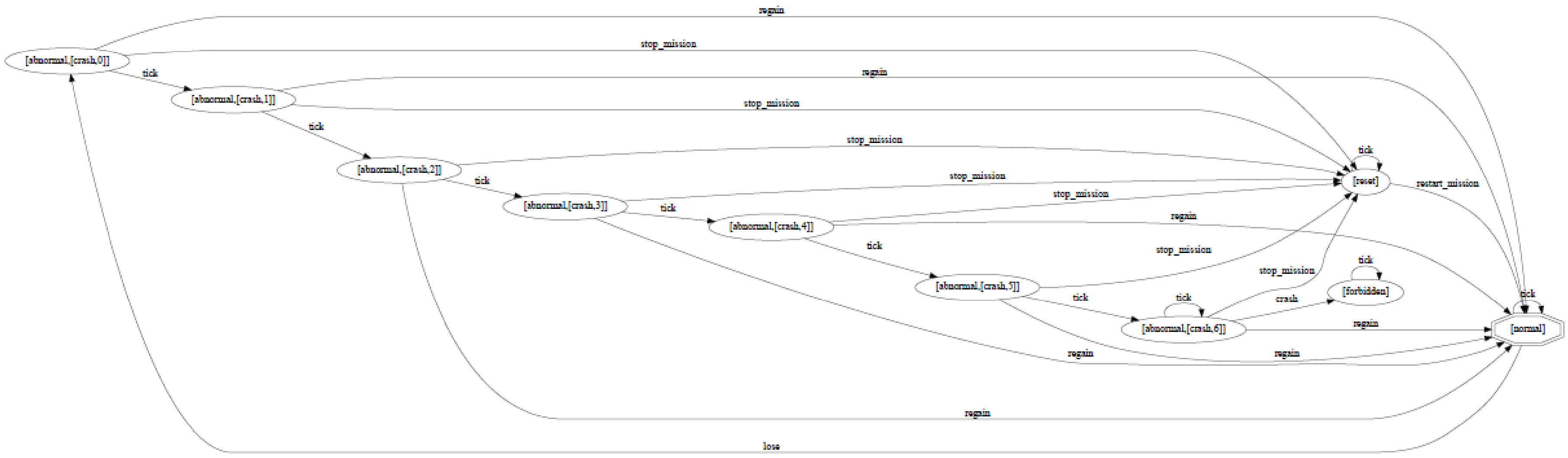
```
EXECUTING: plant = ttm(uav.ttm)
Started at: Sun Aug 13 10:32:45 2017
Working Set Size: 4055040
EXECUTING: uav1 = fsm(uav.fsm)
Started at: Sun Aug 13 10:32:45 2017
Working Set Size: 4079616
EXECUTING: printWithMap(plant, plant.pri)
Started at: Sun Aug 13 10:32:45 2017
Working Set Size: 4128768
EXECUTING: spec = fsm(Spec.fsm)
Started at: Sun Aug 13 10:32:45 2017
Working Set Size: 4136960
EXECUTING: printWithMap(spec, spec.pri)
Started at: Sun Aug 13 10:32:46 2017
Working Set Size: 4145152
EXECUTING: condat(plant, sup, control.txt)
Started at: Sun Aug 13 10:32:46 2017
Working Set Size: 4354048
EXECUTING: printWithMap(sup, sup.pri)
Started at: Sun Aug 13 10:32:46 2017
Working Set Size: 4370432
EXECUTING: printAsPDF(uav1, uav1.pdf)
Started at: Sun Aug 13 10:32:46 2017
Working Set Size: 5017600
EXECUTING: printAsPDF(spec, spec.pdf)
Started at: Sun Aug 13 10:32:48 2017
Working Set Size: 5029888
EXECUTING: printAsPDF(plant, uav.pdf)
Started at: Sun Aug 13 10:32:49 2017
Working Set Size: 5066752
EXECUTING: printAsPDF(sup, sup.pdf)
Started at: Sun Aug 13 10:32:49 2017
Working Set Size: 5111808
```

* *Run.txt* is a batch file used by OTCT to obtain the least-restrictive non-blocking supervisory solution for a modeled system. It specifies the input files:
1 ) model file (*UAV.ttm*);
2) specification file (*Spec.fsm*).
*Run.txt* executes functions for obtaining the largest controllable sublanguage of the plant according to the specification (*supfcBySync*), printing the control action of the supervisor (*condat*), and printing the transition structure (*printWithMap, printAsPDF*).
Each of the input file consists of the initial and marker states, set of transitions between the states, labeled by an event, timers for each event, and set of forcible and controllable events. The output file *control.txt* lists the plant’s state followed by the supervisor’s state where disabling and/or forcing occur, together with the events which must be disabled or forced there.

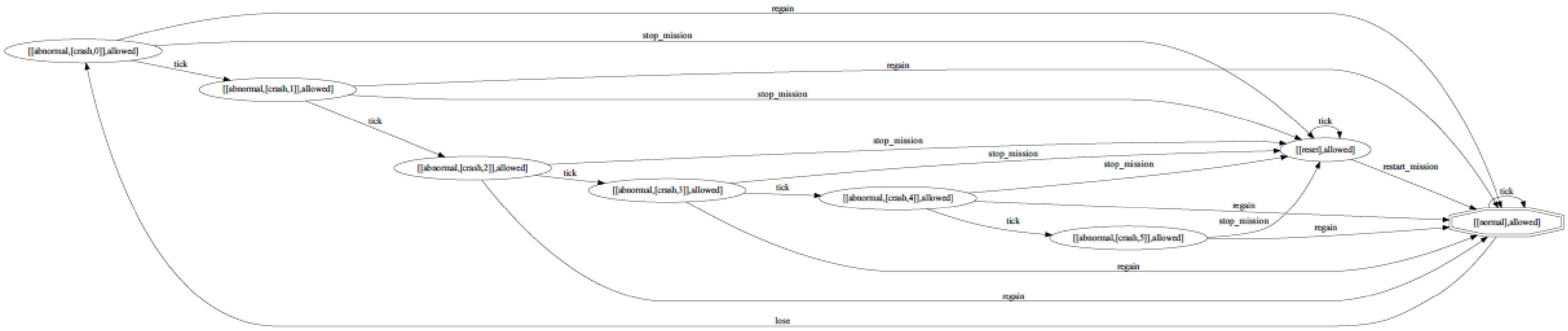
# Appendix C

## Timed transition graphs for “far field”

a) UAV



b) Supervisor



# Appendix D

## Input and output files for “near field”*

**Uav.ttm**

```
initial.
[[aligned],
[not_aligned]].
marker.
[landed].
tran.
[[aligned,  misalign, not_aligned],
[not_aligned, lose, dangerous],
[dangerous, crash, forbidden],
[not_aligned, come_back, aligned],
[dangerous,  regain, not_aligned],
[dangerous,  abort, EP_control],
[not_aligned, abort, EP_control],
[not_aligned, hit_the_ship_gear, forbidden],
[dangerous, hit_the_ship_gear, forbidden],
[aligned, abort, EP_control],
[aligned, auto_finished, EP_control],
[EP_control, land, landed]].
timer.
[[crash, 3, inf],
[hit_the_ship_gear, 9, inf],
[auto_finished, 10, inf],
[land, 2, 10]].
forcible.
[[abort],
[land]].
controllable.
[[abort],
[land]].
```

**Spec.ttm**

```
initial.
[[inAir]].
marker.
[landed].
tran.
[[inAir,  land , landed],
[inAir, timeout, battery_dead]].
timer.
[timeout, 120, 120].
```

**Run1.txt**

```
uav1 = fsm(uav.fsm)
spec1 = fsm(Spec.fsm)
printAsPDF(uav1, uav.pdf)
printAsPDF(spec1, spec.pdf)
plant = ttm(uav.ttm)
spec = ttm(Spec.ttm)
sup = supfcBySync(plant,spec)
condat(plant,sup,control.txt)
printWithMap(sup,sup.pri)
```

**tct.log**

```
Batch Command File: run1.txt
EXECUTING: uav1 = fsm(uav.fsm)
Started at: Mon Mar 20 09:48:57 2017
Working Set Size: 3809280
EXECUTING: spec1 = fsm(Spec.fsm)
Started at: Mon Mar 20 09:48:57 2017
Working Set Size: 3829760
EXECUTING: printAsPDF(uav1, uav.pdf)
Started at: Mon Mar 20 09:48:57 2017
Working Set Size: 3792896
EXECUTING: printAsPDF(spec1, spec.pdf)
Started at: Mon Mar 20 09:48:59 2017
Working Set Size: 3805184
EXECUTING: plant = ttm(uav.ttm)
Started at: Mon Mar 20 09:49:00 2017
Working Set Size: 3817472
EXECUTING: spec = ttm(Spec.ttm)
Started at: Mon Mar 20 09:49:00 2017
Working Set Size: 3829760
EXECUTING: sup = supfcBySync(plant, spec)
Started at: Mon Mar 20 09:49:00 2017
Working Set Size: 83095552
EXECUTING: condat(plant, sup, control.txt)
Started at: Mon Mar 20 11:28:29 2017
Working Set Size: 77901824
EXECUTING: printWithMap(sup, sup.pri)
Started at: Mon Mar 20 11:29:08 2017
Working Set Size: 120504320
```

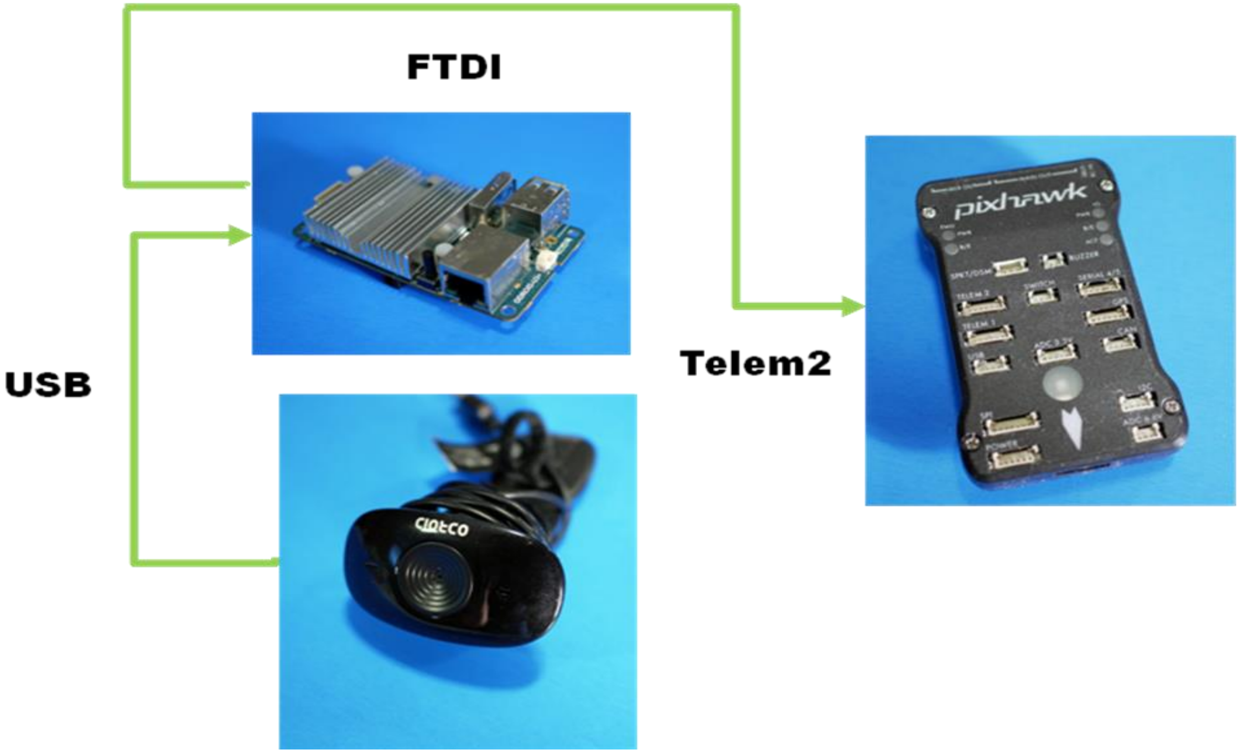
**Control.txt** (abridged )

```
PLANT: [EP_control,[land,3]]
   SUPER: [[EP_control,[land,3]],[inAir,[timeout,120]]]
   EXPEDITE:  land
 PLANT: [aligned,[auto_finished,5]]
   SUPER: [[aligned,[auto_finished,5]],[inAir,[timeout,118]]]
   EXPEDITE:  abort
 PLANT: [not_aligned,[hit_the_ship_gear,8]]
   SUPER: [[not_aligned,[hit_the_ship_gear,8]],[inAir,[timeout,116]]]
   EXPEDITE:  abort
PLANT: [dangerous,[crash,0],[hit_the_ship_gear,8]]
   SUPER: [[dangerous,[crash,0],[hit_the_ship_gear,8]],[inAir,[timeout,57]]]
   EXPEDITE:  abort
PLANT: [dangerous,[crash,2],[hit_the_ship_gear,2]]
   SUPER: [[dangerous,[crash,2],[hit_the_ship_gear,2]],[inAir,[timeout,94]]]
   EXPEDITE:  abort
PLANT: [dangerous,[crash,0],[hit_the_ship_gear,2]]
   SUPER: [[dangerous,[crash,0],[hit_the_ship_gear,2]],[inAir,[timeout,118]]]
    EXPEDITE:  abort
```

* Graphical files for the activity transitions at the “near field” are included in Chapter 4. Timed structures are not presented due to the limitation of space, but the result can be reproduced after executing the given input files.

# Appendix E

## Connections between hardware components


FTDI
USB
Telem2
pixhawk