Autonomous sensor guided precision landing system for multicopters

H.A. Steenbeek

University of Twente, Faculty of Engineering Technology, Drienerlolaan 5, 7522 NB, Enschede, The Netherlands

ABSTRACT: In this paper a precision landing system aimed at large scale agricultural multicopters is developed and tested. Using external sensors, an onboard computer, custom developed software and quadcopter desirable results were achieved but reliable centimeter accurate positioning still requires further development. This is mainly due to undesirable behavior of the RTK GPS and the fact that additional tuning is needed on the quadcopter that was used for testing.

Key words: UAV; multirotor; multicopter; autonomous landing; RTK GPS; agriculture; autonomisation

1 INTRODUCTION

1.1 Motivation

For companies to be able to compete in the current high tech climate they need to perform more efficiently then ever before. This has caused a huge demand for increased autonomisation. This is also true for the current agricultural sector where autonomisation is not only helpful, but required to let the farmers keep up with the current required company growth. To be able to manage and inspect vast amounts of land in a short time span, drones are the perfect tool. It is even estimated that the agricultural drone marked will reach up to $480 million in 2026 [1].
Currently one of their main disadvantages is their inability to operate for long time periods without recharging. This problem can be solved if the drone could autonomously dock and recharge itself, after which they can resume their mission.

1.2 Problem description

For drones to perform autonomous tasks, especially targeted landing, high precision positioning is required. Currently commercially available flight controllers and autopilots are not designed to perform high accuracy maneuvers. These drones determine their position using GPS and IMU (Inertial Measurement Unit). GPS is only accurate up to 5 meters [2] and IMUs use acceleration to calculate their orientation instead of position and thus they suffer from accumulated error due to integration over time, making them unsuitable for position estimation. Therefore the system needs additional sensory components and control systems to be able to perform these high precision tasks.
The goal of this paper is to develop a autonomous landing system that is to be used on a large scale multicopter that is currently still in development. The copter flies over large areas of land and sprays them. When the batteries need charging or the spray tank need to be refueled, the copter needs to return to a docking station where this can be done.

1.3 Requirements

The system needs to be able to land within a 10cm radius of a desired landing position and vary no more than 10 degrees in the intended yaw angle and should function in fog and nighttime environments and be able to mitigate wind gusts. Furthermore the system needs to be extendable so that further redundancy can be included in the future. The design of the docking station itself is outside the scope of this paper.

1.4 Outline

First an overview of the currently available systems are going to be discussed. Then the hardware components and software components that were selected and developed for the landing systems are going to be described after which the complete setup is going to be described, showing how all the hardware and software components fit together. This is followed up by the test results.

2 STATE OF THE ART

2.1 Navigation

The problem of navigation can be divided into two sections, state estimation and perception. In the section below multiple sensor systems that can be used for navigation are discussed, separated into these two sub-groups.

2.1.a State estimation

State estimation consists mainly of simply converting the raw sensor data into information about the state, position and attitude of the vehicle.

GPS GPS, or Global Positioning System, is a space-based navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. GPS has a vertical position error of 4.7m and 3.4m horizontal [2]. This is under ideal circumstances. The actual accuracy could be lower because of external factors like atmospheric effects, sky blockage and receiver quality. The main advantage of GPS is that it has unlimited range. As long as the receiver is connected to the satellites it is able to determine its position anywhere in the world. However, it is susceptible to blocking, signal jamming and other signal interruptions. The relatively low accuracy also make it unsuitable for high precision applications.

Augmented GPS Augmented GPS is a system that aids the default GPS by increasing accuracy, reliability or any other function of the GPS. There are multiple systems that bring the accuracy of the GPS into the centimeter range. For drone usage RTK (Real Time Kinematic) systems are the most common way to achieve centimeter level accuracy since recent advances in small scale electronics have made these systems more affordable. RTK GPS systems use phase measurements of the waves from the GPS satellite signals to calculate the relative distance between two RTK GPS stations (a static base station and a so called rover unit). This allows for relative centimeter accuracy between the two stations. RTK GPS systems have been used in autonomous precision landing with varying degrees of success and reliability [3], noting that the system that was used was just recently released and could still be improved upon.

Optical Optical state estimation utilizes cameras and markers with pre known properties and patterns [4] [5] [6] to calculate the relative position of the vehicle. The main disadvantage is the required visibility of the marker. In [7], this is solved by backlighting the marker with LEDs at night with promising results. This however would still be difficult in other situations where vision is impaired, like rain or fog. Some aerial systems also incorporate optical flow sensors, using series of camera images to estimate traveled distance. This system is used for stabilization and preventing position drift. However, this system does suffer from error accumulation over time, making it too unreliable to be used for docking.

Ultrasound Another method of absolute localization is using ultrasonic sensors. These systems use multiple ultrasonic nodes that transmit ultrasonic signals which are registered by a single receiver [8] [9]. Using the known locations of these nodes, the position of the receiver can be calculated.

These systems are however very susceptible to noise and are a long way off from reliable outdoor centimeter positioning.

Infrared A very reliable method for position tracking is infrared technology. Using infrared cameras and filters, computer vision is made less demanding compared to traditional optical systems. Their main disadvantage is their sensitivity to distortion from sunlight. This is usually circumvented by using beacons that emit signals at a specific frequency. It is possible to calculate 3d position using perspective-n-point techniques [10], but for precision landing systems only accurate x and y coordinates need to be known since GPS, IMUs and sonar sensors already provide information about the global position, attitude and height respectively.

Ultra Wideband UWB (Ultra Wideband) positioning is another promising technique for position tracking. Utilizing multiple tags that emit a very short, low power pulse on a high bandwidth and

using a single receiver, centimeter accurate positioning can be achieved [11]. Though professional UWB systems are relatively expensive, a startup [12] is currently providing more affordable UWB systems, aimed for indoor usage, ready for implementation.

2.1.b Perception

For drones to achieve real world autonomous behavior they need perception capabilities. These may include detecting and avoiding obstacles in real-time, recognizing targets, mapping the environment or detecting safe landing areas. For localization a technique called simultaneous localization and mapping (SLAM) is used in most cases. Using sensor data, an internal map of the scanned area is created while simultaneously localizing the robot in this map allowing it, once the map is created, to navigate around obstacles and choose routes.

Vision based perception Vision based perception utilizes optical sensors for perception tasks. Most systems use stereo cameras for depth perception and building SLAM maps, although it is also possible to use a single camera and utilize the movement of the drone to simulate stereo cameras [13]. Most vision based systems although are still very experimental [14].

LIDAR Based perception Light Detection and Ranging (LIDAR) uses a laser to measure distance in rapid succession to generate high precision maps of areas. It can also be used to detect crop growth or measure water depth. It is already being used successfully in multiple drone projects [15] [16] and is also production ready. The main disadvantage of LIDAR systems is their price range.

2.2 Conclusion

Currently there are a lot of affordable systems available. Some have been used and developed for a longer time, often making them more reliable. Other, also very promising systems, are being developed by new startups. Though these sometimes tend to be less reliable and often lack results to back their claims. Another noteworthy market trend is the development of currently very expensive systems and using new technology to redevelop them into more affordable systems that can be used by hobbyists and smaller scale businesses that could otherwise never afford these systems. Examples of these are LIDAR and RTK GPS.

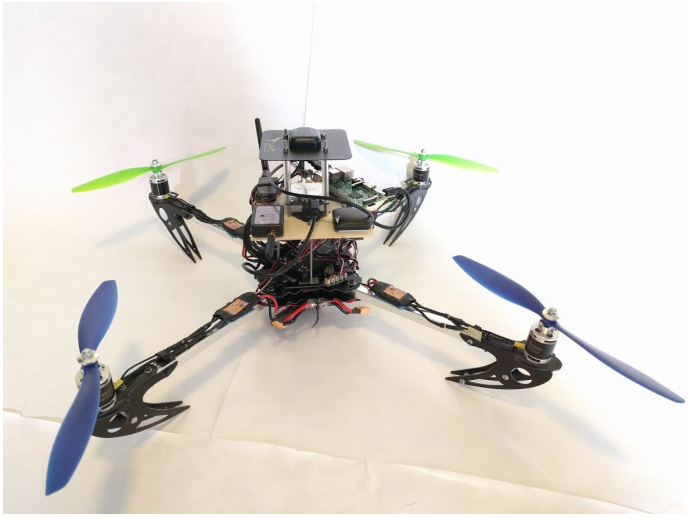

Figure 1: Quadcopter that was used for testing

Taking into account budget constraints and readily available systems, a combination of two systems was chosen. The first is a new RTK GPS system developed by Swiftnav, the second is a infrared system developed by IR-Lock.

3 HARDWARE

In this section the hardware components that were selected for the automated precision landing are described and discussed.

3.1 Copter

3.1.a Test copter

The actual code was tested on a small scale quadcopter specifically built for testing the landing procedure since testing with the large scale model would be too dangerous and would require considerable more time and precautions to set up.
The copter is a simple X frame setup with a wooden stack on top for additional sensors and has a diameter of 60cm. The RTK GPS was mounted on top of the wooden frame so it would be far away from the motors. The copter can be seen in figure 1.

3.1.b Large scale copter

The end result of the landing setup is meant to be implemented on a large scale coaxial quadcopter. This copter has a width of 2m, a length of 2,5m and can lift

up to 150kg using 8 brushless DC motors. No tests have been conducted using this machine but all systems should also be transferable to this system.

3.2 Autopilot

A Pixhawk was selected as the flight controller. It is an open-source hardware project, developed by 3D robotics [17]. It has an embedded 9DOF IMU, barometer and an external uBlox GPS module. The Pixhawk is responsible for managing the copter, including controlling the ESCs (Electronic Speed Controllers) which in turn control the motors, reading and processing sensor data, communication and logging flight data. On the Pixhawk itself runs the autopilot software. This, by default, is the PX4-firmware but the Pixhawk also allows for the Arducopter software to be run on top of the PX4-firmware. Arducopter is an alternative, very advanced flight stack which is also very popular but less suited for autonomous applications.

3.3 Companion Computer

A Raspberry-Pi (RPi) is selected as the companion computer. Since the Pixhawk does not have enough computing power to do heavy calculations, a companion computer is needed. This computer can do the more intensive calculations, image processing for example and is able to communicate with autopilot directly via mavlink messages using a specific port on the Pixhawk. The RPi is a small form factor, low power computer, about the size of pack of cards and runs Linux. Communication with the RPi is done via built-in WiFi over an SSH connection.

3.4 Sensors

3.4.a Swiftnav Piksi RTK GPS

The Swiftnav Piksi is a low power GPS receiver with Real Time Kinematic (RTK) functionality. In the case of Piksi the correctional data is send trough radio signals from the base to the rovers using the Swift Binary Protocol (SBP) which is a proprietary communication protocol.

The Piksi has three states, single point precision (SPP), RTK float and RTK fixed. In SPP mode the Piksi rover is just as accurate as a normal GPS, once it enters float mode it already has accurate relative positioning with an accuracy between normal GPS and RTK precision but still continues to search for a higher precision fixed solution (fixed RTK or RTK lock). Once in fixed RTK mode Piksi is accurate to 2cm horizontally on short baselines with good sky view. Vertical precision is typically 2-3 times worse than horizontal precision for GPS receivers, so we expect 6cm accuracy on short baselines. Due to the nature of RTK, accuracy degrades at a rate of 1mm horizontal and 3mm vertical for each km between the base and rover. [18] It takes about between 5 to 15 minutes for the Piksi rover to enter fixed RTK mode. The Piksi can fall back to a less accurate mode if the connection with a satellite or the base is lost or deteriorated, upon which it will try to restore it's connection. This generally takes less time than the initial RTK lock. The Piksi does require a clear and wide view of the sky to acquire RTK lock, not only does this prevent the docking station from being placed near tree lines or other large structures, it can also cause troubles with large roll or pitch excursions that momentarily obscure the GPS antenna of the copter. The Piksi rover transmits location data at a frequency of 10Hz, in combination with a co-variance matrix indicating the measurement accuracy.

In previous applications of Piksi RTK GPS on copters [19] [3] [20] promising results were achieved, although it is noted that the Piksi is very susceptible to signal noise from the DC motors on the copters. This could cause problems when the Piksi is connected to the larger scale model.

3.4.b IR-lock

The IR-lock consists of a small smart camera and a beacon. The camera filters out Infrared light and uses object recognition software built into the camera to return the position of the beacon in pixels on the screen. The beacon itself uses multiple infrared LEDs that flicker at a specific frequency as to prevent false positives due to sunlight or any other source. The pixel position in combination with the height of the copter is used to determine the X and Y distance from the beacon. Since all the object recognition is done on the camera, it can be used on relatively low powered systems.

4 SOFTWARE

In this section the software components of the setup are described and discussed.

4.1 MAVlink

MAVlink (Micro Air Vehicle Link) is a communication protocol for remote controlled vehicles and is the default protocol for the PX4 autopilot. MAVlink abstracts the communication, allowing multiple programming languages and software packages to communicate over a single protocol [21]. It is used in the setup to communicate between the PX4 and the RPi, and also for the PX4 telemetry to the ground station.

4.2 ROS

ROS (Robotic Operating System) is a framework for robot development. It uses a system of topics with subscribers and publishers for communication. A publisher sends out data to a topic that can be read by any subscriber on the network connected to that topic. ROS uses nodes to distribute functionality. Each node has a specific purpose and communicates using topics.
In the current setup the following 5 main nodes are implemented, each taking care of a specific component of the landing procedure.

4.2.a Mavros node

Mavros is a node that facilitates communication between ROS and the MAVlink protocol. It publishes sensor data into topics and subscribes to multiple topics that allow for vehicle control. Mavros also takes care of certain frame conversions. The PX4 and Arducopter firmware both operate in the NED (North, East, Down) frame, whereas ROS normally operates in the ENU (East, North, Up) frame.

4.2.b RTK node

The Piksi node publishes data collected from the RTK GPS and publishes it in combination with a covariance matrix representing the horizontal accuracy of the RTK lock. All the data send from this node is in the ENU frame.

4.2.c IR-lock node

The IR-lock node transmits the location of the IR beacon in pixel position on the camera.

4.2.d External position node

This node was specifically developed for the landing of this quadcopter, it uses external positioning node collects the data from the RTK node and IR-lock node and combines them to send one coordinate set of the landing position in the local ENU frame.

4.2.e Autoland node

Just like the External position node, this component of the software was developed specifically for the precision landing procedure. The autoland node controls the landing procedure by sending control messages to the mavros node. It Subscribes to the external positioning node to determine and correct the copter position.

4.3 PX4 Firmware

The Pixhawk is the autopilot of choice By default it runs the PX4 firmware. There is a compatibility layer that allows the Ardupilot autopilot to run on the Pixhawk. The autopilot firmware controls the copter directly, it contains all the control systems, takes care of communication, allows for easy calibration systems and sends data to the ESC's (Electronic Speed Controllers) that control the motors.
The PX4 autopilot software allows control of the copter in different flight modes. Each flight modes has a specific function and flight behavior. For the copter to accept velocity or position commands from the board computer it needs to be in offboard mode. This is done using a switch on the remote so that if something goes wrong, it can be switched back to stabilize mode in which the copter is controlled using the sticks of the remote control.

5 SETUP

An overview of the complete hardware setup is shown in figure 2. In this overview the default sensors connected to the Pixhawk have been ignored, these include the default 3DR UBlox GPS and Compass module, the embedded 6DOF IMU and the barometer. The IR-Lock component of the system is displayed in dotted lines since the actual implementation of was unable to be completed due to time constraints. It was included to illustrate how additional sensors can be added.

5.1 Sensors

The Piksi is connected to both the Raspberry-Pi and the Pixhawk. It allows the Raspberry-Pi to interpret the data, the Pixhawk can use the Piksi as main, higher precision, GPS if the Piksi is in RTK or float mode, otherwise it uses the default 3DR UBlox GPS.
All external position estimation sensors are connected

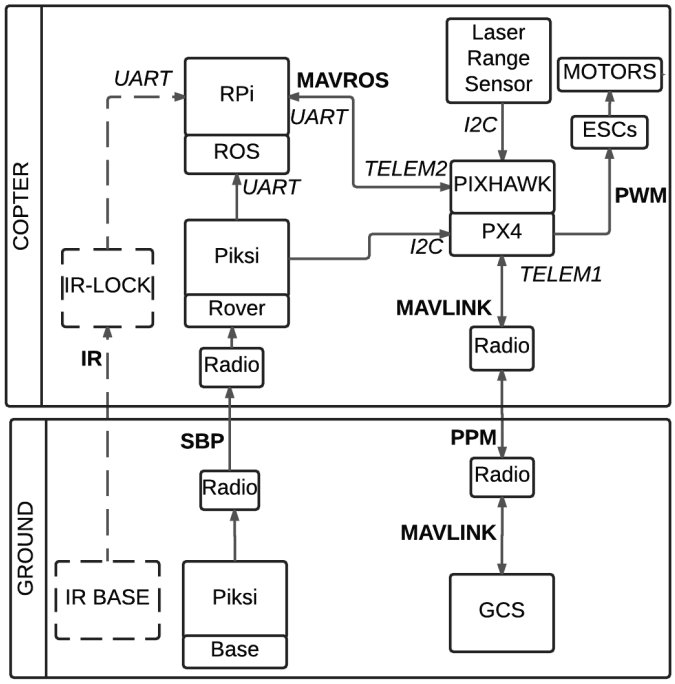

Figure 2: Hardware setup overview

to the Raspberry-Pi, allowing the sensor data to be filtered and combined in ROS. The laser range sensor data is processed by the Pixhawk to allow the PX4 firmware to benefit from the improved height accuracy and relieves the Raspberry-Pi from having to compensate for roll and pitch angles. The Raspberry-Pi retrieves all the Pixhawk sensor data using mavros topics.

5.2 ROS mavros GCS bridge

ROS is responsible for combining and filtering all the sensor data. All communication between ROS and PX4-Firmware is done using mavros, which translates the MAVlink protocol to ROS topics. During the tests everything was run on a Raspberry-Pi, but in the future, this could be replaced by any computer that can run ROS. The Raspberry-Pi is connected to the TELEM2 port on the Pixhawk which is meant to be used for companion computers.
The quadcopter is still controlled by the pilot on the ground using a GCS (Ground Control Station) or a remote control radio, both communicating using radio connected to the TELEM1 port of the Pixhawk.

5.3 Landing procedure

A simplified overview of the implemented landing procedure is shown in figure 3. In test scenarios the copter would be flown out into an arbitrary direction, then a button on the remote control would be switched that put the copter into offboard mode. Once in offboard mode the landing procedure would begin.

5.4 control

In figure 4 an overview of the control schematic of the landing procedure is shown. The top part of the schematic represents the control components that are ran on the companion computer, the bottom part contains the control components ran on the Pixhawk.
The land position is set when the copter is armed. When it is descending it uses a P filters to control the x and the y velocity's in the ENU frame by using the position error calculated from the external position estimator as setpoint. This velocity information is integrated by mavros and converted to a position in the NED frame that is then send to the Pixhawk.
The Pixhawk passes the position setpoint to the position controller which then uses this information to set a angular rate setpoint. This angular rate is then converted to force vectors that are passed to the actuator drivers that control the motors. The onboard sensors then provide feedback to the loop by sending information to the attitude and position estimators.

6 TESTS

6.1 Simulator

Before implementation, all the code was tested in a simulator to make sure the drone would not show any erratic behavior once in the air. The simulator used was the PX4 SITL simulator, interfaced with gazebo, a robotics simulation environment. In the simulator the flight dynamics of the Iris+ quadcopter were used since this is the only copter available in the simulation. In the simulator near centimeter accuracy was obtained but since the quadcopter model in the simulator is not modeled after the real world test copter, the obtained results only function as a test for a correct implementation of the landing procedure.

6.2 Piksi

To determine the real world accuracy of the RTK system, tests were conducted. The GPS was mounted on the test copter. It was placed on a level marked position on the ground, manually moved around the area between 1 and 2 minutes and then placed back on the

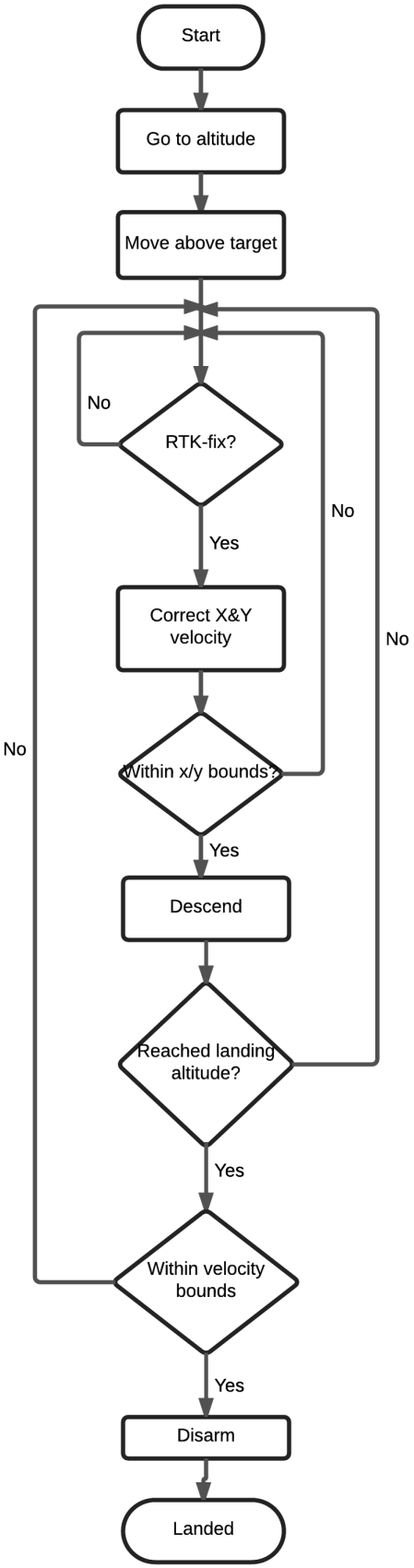

Figure 3: Flow chart of the landing procedure.

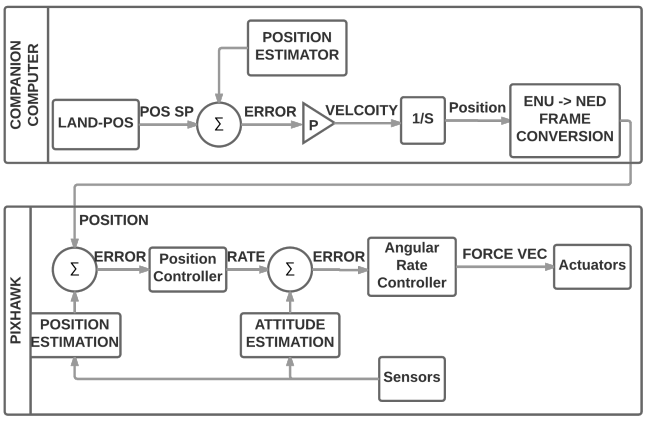

Figure 4: Complete control system overview

#	△ *x [cm]*	△ *y [cm]*	*Distance [cm]*
1	0.74	1.97	2.11
2	1.14	1.97	2.28
3	1.35	0.80	1.57
4	1.17	0.81	1.42
5	1.47	3.78	4.05
6	0.38	3.38	3.40
7	0.47	1.24	1.33
8	0.17	0.69	0.71
avg	0.94	1.61	1.84

Table 1: Piksi RTK results

#	△ *x [cm]*	△ *y [cm]*	*Distance [cm]*
1	103.99	5.12	104.12
2	200.42	0.27	200.42
3	155.69	11.71	156.13
4	65.54	32.09	72.98
5	10.51	139.75	140.14
6	55.93	17.22	58.52
7	156.53	166.24	228.33
8	77.92	11.02	78.70
avg	90.96	14.47	122.13

Table 2: PX4 local position estimator results

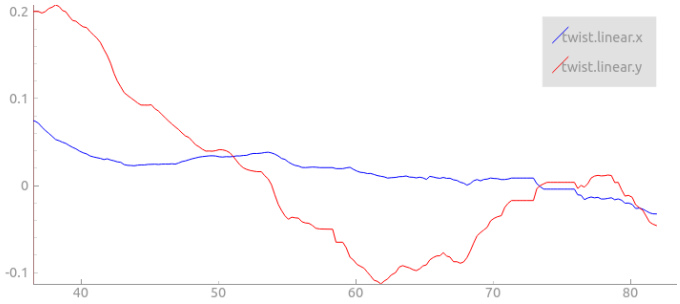

Figure 5: Velocity commands during landing where X velocity is marked blue and Y velocity is marked red.

marked position. The results can be seen in table 1. In these results the displacement between the start and land position in both the x and y direction are shown as well as the total distance from the starting point. It can be seen that the shift in x direction is within the 2cm margin but the results in the y direction tend to be less accurate. The end results still show that the RTK GPS is able to reliably locate the start position within the 10cm radius that is required for precision landing. This compared to the same tests conducted simultaneously using only the default internal position estimator of the PX4, as seen in table 2, which shows less then ideal results. It should be noted that the communication of the RTK base station data is transmitted to the rover on the same radio frequency as the PX4 telemetry. This causes signal packet loss on the Piksi side and also causes communication losses of the telemetry. It is possible to use the PX4 telemetry to carry the Piksi base station data to prevent interference, however, for the tests the telemetry was simply disconnected.

6.3 Real world tests

Multiple tests with the test copter have been conducted. For these test the P value of the velocity controller was set to 0.1. Small values would cause less overshoot but it did take more time for the copter to move to the correct position. Landing accuracy was achieved up to 40cm. Setting a lower acceptance radius would cause better landing precision but due to the copter instability it would move around too much, making the landing procedure take too long or due to the unreliable altitude controller, the x and y directional movements of the copter would cause it to descend too close to the ground, also causing a failed landing. The velocity command results of the controller can be seen in figure 5. It can be seen that the large error in the Y position causes an overshoot even with a low P value. This can be accounted to the slow response time of the copter itself. The RTK GPS would also cause failed landings, sometimes it would lose RTK lock, causing position shifts of up to 2m, other times it would provide a constant position after which it would suddenly jump to the correct position all without noting a loss in accuracy and causing erratic behavior in the landing procedure.

7 CONCLUSION

The landing of the drone was successful but a larger radius was needed for adequate performance. This was mainly due to the fact that the test drone was unable to hold position reliably enough. On a more stable copter more accurate landing results could be achieved. Furthermore, the RTK GPS proved to be accurate enough to determine the exact position required for the landing but does have reliability issues as described in section 6.3, making it unsuitable for reliable position estimation on it's own.

7.1 Recommendations

The unreliability of the RTK GPS could be solved by fusing the RTK sensor data with the onboard IMU data and additional external sensors like the IR-lock or LIDAR sensors which would also provide redundancy. All these systems would allow the copter to perform reliable precision landings. For real autonomous, reliable and safe landing however, the copter would require perception sensors and be able to perform autonomous actions like safe landing area detection and object avoidance.
For testing purposes it would be desirable to use an existing copter with known behavior, an off the shelve copter like the Iris+ would be very suitable since it is also used in most simulators. Alternatively a more realistic simulation model of the large scale copter could be developed and implemented. This would also allow for more reliable testing and speed up the process of implementing additional features.

7.2 Acknowledgments

I want to thank my supervisor R. Aarts for all the help and guidance during this project and I would also like to thank W. Rijssenbeek for his help and the trust he placed in me working on this project.

REFERENCES

1. "Agricultural Robots and Drones 2016-2026: Technologies, Markets, Players: IDTechEx." [Online]. Available: http://www.idtechex.com/research/reports/agricultural-robots-and-drones-2016-2026-technologies-markets-players-000491.asp
2. "GPS.gov GPS Accuracy." [Online]. Available: http://www.gps.gov/systems/gps/performance/accuracy/
3. N. Lu Arraga and R. Mcdonnell, "Autonomous Landing of a Quadcopter Using Real Time Kinematic GPS," 2015.
4. M. Verbandt, B. Theys, and J. De Schutter, "Robust marker-tracking system for vision-based autonomous landing of VTOL UAVs."
5. C. S. Sharp, O. Shakernia, and S. S. Sastry, "A Vision System for Landing an Unmanned Aerial Vehicle."
6. D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 5 2012, pp. 971–976.
7. T. Gomes Carreira, "Quadcopter Automatic Landing on a Docking Station," 2013.
8. W. Wang, Z. Li, W. Yu, and J. Zhang, "An autonomous docking method based on ultrasonic sensors for self-reconfigurable mobile robot," in *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 12 2009, pp. 1744–1749.
9. A. Saxena, "Ultrasonic Sensor Network: Target Localization with Passive SelfLocalization," 2015.
10. K. E. Wenzel, A. Masselli, and A. Zell, "Automatic Take Off, Tracking and Landing of a Miniature UAV on a Moving Carrier Vehicle," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1-4, pp. 221–238, 1 2011.
11. A. Alarifi, A. Al-Salman, M. Alsaleh, A. Alnafessah, S. Al-Hadhrami, M. Al-Ammar, and H. Al-Khalifa, "Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances," *Sensors*, vol. 16, no. 5, p. 707, 5 2016.
12. "Pozyx - centimeter positioning for arduino." [Online]. Available: https://www.pozyx.io/
13. J. J. Engel, "Autonomous Camera-Based Navigation of a Quadrocopter," 2011.
14. Kendoul F, "Survey of advances in Guidance, Navigation, and control of Unmanned Rotorcraft Systems."
15. P. Tsenkov, J. K. Howlett, M. Whalley, G. Schulein, M. Takahashi, M. H. Rhinehart, and B. Mettler, "A System for 3D Autonomous Rotorcraft Navigation in Urban Environments."
16. T. Merz and F. Kendoul, "Beyond visual range obstacle avoidance and infrastructure inspection by an autonomous helicopter," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 9 2011, pp. 4953–4960.
17. "3DR - Drone & UAV Technology." [Online]. Available: https://3dr.com/
18. "Swift Navigation Wiki." [Online]. Available: http://docs.swiftnav.com/wiki/Main_Page
19. R. Cabell, F. Grosveld, and R. McSwain, "Measured Noise from Small Unmanned Aerial Vehicles."
20. D. Zollo and R. Gohalwar, "Piksi ™ for UAV Aerial Surveying RTK Direct Georeferencing with Swift Navigation's Piksi GPS Receiver."
21. S. Balasubramanian, "MavLink Tutorial for Absolute Dummies (Part –I)," 2014.