Precision UAV Landing in Unstructured Environments

Kevin Pluckter

CMU-RI-TR-19-49

July 2019

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee:
Sebastian Scherer, Chair
Michael Kaess
Kumar Shaurya Shankar

Thesis proposal submitted in partial fulfillment of the requirements for the degree of Master's of Science in Robotics

Abstract

The autonomous landing of a drone is an important part of autonomous flight. One way to have a high certainty of safety in landing is to return the drone to the same location it took-off from. Current implementations of the return-to-home functionality fall short when relying solely on GPS or odometry as inaccuracies in the measurements and drift in the state estimate guides the drone to a position with a large offset from the initial position. This situation can be particularly dangerous if the drone took-off next to something like a body of water. Current work on precision landing relies on localizing to a known landing pattern, which requires the pilot to carry a landing pattern with them. We propose a method using a downward facing fisheye lens camera to accurately land a UAV from where it took off on an unstructured surface, without a landing pattern. Specifically, this approach uses a position estimate relative to the take-off path of the drone to guide the drone back to its original starting point. With the large field-of-view provided by the fisheye lens, our algorithm can provide visual feedback starting with a large position error at the beginning of the landing, until 1 m above the ground at the end of the landing. This algorithm empirically shows it can correct the drift error in the state estimation and land with an accuracy of 40 cm.

Contents

List of Figures

List of Tables

Acknowledgments

First, I want to give thanks to God for giving me the opportunity to study at Carnegie Mellon and for blessing my life with the people who have made it possible for me to get here. Next, I would like to thank Basti Scherer for giving me the opportunity to work in the AIR lab and for advising me these past two years. Also, thank you to Michael Kaess and Kumar Shaurya Shankar for being a part of my advising committee and for all the advice you have given me.

Thanks to Sanjiban Choudhury for introducing me to Basti and the AIR lab. Thank you Sankalp Arora, Rogerio Bonati, Ratnesh Madaan, Rohit Garg, Wenshan Wang, and Azarakhsh Keypour for welcoming me into the AIR lab. To Rogerio, thank you for always challenging me and for encouraging me to seek after a deeper understanding of my studies. To Ratnesh, thank you for all our conversations and for always being willing to talk through various bugs in my code. To Rohit Garg, thank you for all the times you debugged code alongside me and for being someone that I loved having philosophical conversations with. To Wenshan, thank you for helping me with AirSim and for sharing your culture with me by teaching me to make dumpling wrappers. To Azarakhsh, thank you for discussing my research with me and for helping review my thesis. To Sankalp, thank you for advising me throughout my research, particularly with my work pertaining to SubT.

Thanks to Team Explorer of the DARPA SubT challenge for a crazy last year of grad school. To the hardware team - thanks for making all our robots work. To the software team - thank you for working countless hours with me to get to the point that we are at in the project.

Thank you to my parents for the raising me to work diligently and to always challenge myself, as well as for talking with me through tough moments and for being my friends. Thank you to my girlfriend, Wendy, for sticking with me through two years of long distance and for putting up with my shenanigans - thank you for always supporting me, encouraging me, and challenging my views during my time in grad school. Thanks to my many Pittsburgh friends, including, but not limited to: Scott Moore, Adam Reedy, Josh, Elaina, Hunter Goforth, Leo Kesslemen, Karen Orton, Kevin Christensen, Mohammad Mousaie, Kevin LoGrande, Michael Tatum, Michael Kitcher, Dan Abad, Cherie Ho, Vai Viswanathan, Mike Lee, Joseph Aroh, Ada Taylor, Samuel Clarke, Rachel Kaufman, Tim and Mallory Onarecker, and Luke and Hannah Zappa - this experience wouldn't have been the same without you.

Chapter 1

Introduction

1.1 Motivation

Unmanned Aerial Vehicles (UAV) are becoming more capable with research efforts today and more useful in industry. Some use cases of them include warehouse inventory checking, package delivery, and building inspection. Even with various use cases, one of the difficulties for autonomous UAVs is landing. Many incidences of bad landings or landing in bad areas occur for these vehicles. For example, a drone hitting a tree while landing or landing into a lake. One way to safely land a quadrotor is to designate a safe landing zone. Much research has gone into landing a drone onto a designated landing pattern, such as a helipad that is level and sturdy for the UAV to land on. Another method is to have the drone return back to its initial position. It's a reasonable assumption that the spot the drone started at is safe to land back on, and that the environment has not changed much so it is still safe. Given this, an accurate and precise return to home method would be an effective means for safely landing the drone. Additionally, the UAV returning back to where it took off from is a desirable trait as it can make the process of retrieval and storage of the drone easier.

Majority of small low cost UAVs currently rely on GPS based state-estimation for return to home landing, but there is still fairly high error in the low cost GPS on light weight drones and the state-estimation systems are unreliable if in a GPS denied environment. Given this, a drone cannot reliably land in the same spot and could potentially attempt to land in an unforgiving location, such as a nearby tree. Given the small size and low price point of these UAVs, more powerful sensors like RTK GPS and 2-D or 3-D lidar are not an option for assisting in the return to home functionality. A sensor suite that is small enough and cheap enough to add in addition to the current setup is a camera and laser range finder. With the additional scene understanding provided by the camera the drone can return home safely. Specifically, this work addresses the challenge of precision landing using a downward facing fisheye camera and downward facing laser range finder to improve the

safety and robustness of landing. This is done by re-traversing the proven takeoff path back to the quadrotor's starting position. We are able to precisely land the drone with this method.

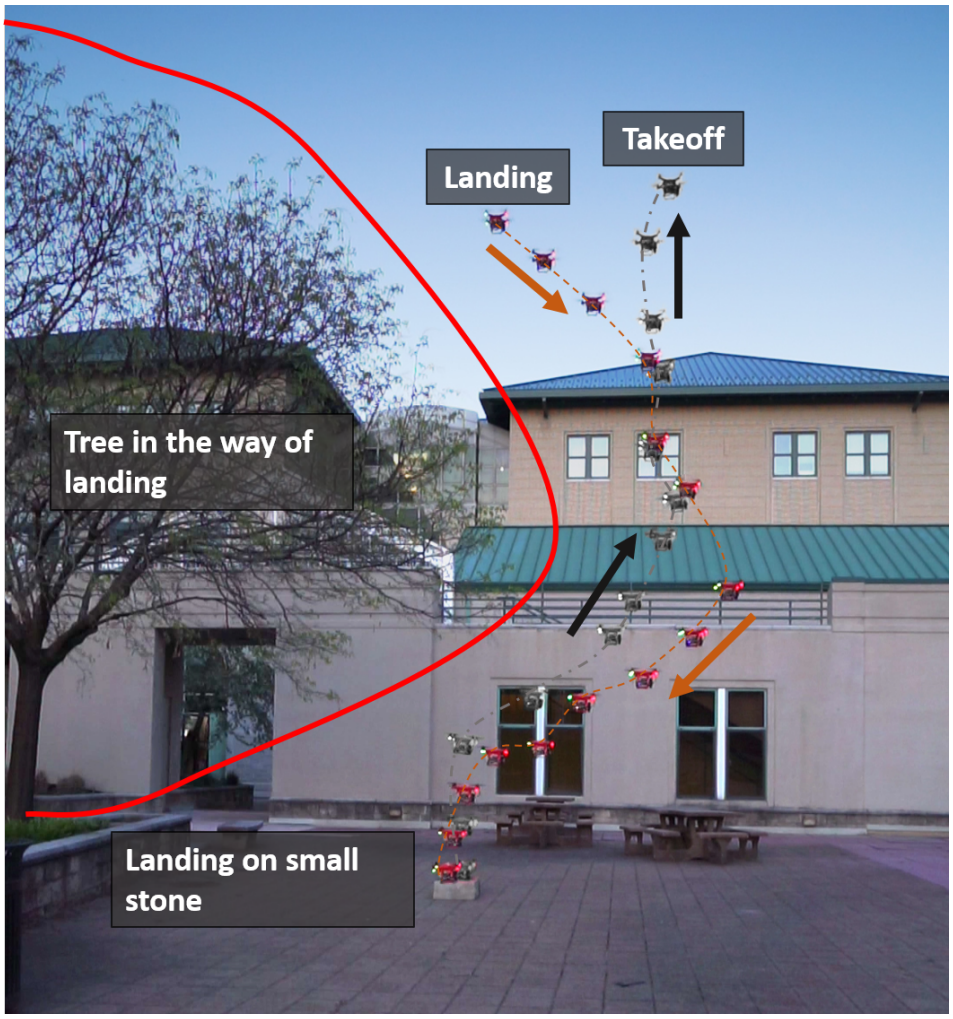

Figure 1.1: Precision Landing Demonstration: the takeoff path (black), and the landing path (orange). The key challenges are for the drone to safely and precisely land. This equates to the drone avoiding the tree that was avoided during takeoff, and landing on the stone the drone started on. Our algorithm successfully accomplishes both challenges.

As mentioned previously, majority of current work on precision landing focuses on landing in a known structured environment (helipad or runway), such as the one depicted in Fig. 1.2 [16]. Given a known landing pattern, these algorithms are able to identify the target and use its geometry to robustly estimate the relative state of the drone. Much work has gone into different landing pattern designs to enable accurate pose estimation [23,27,28]. While these methods result in accurate landings,

they require the drone to land on a specific type of landing pattern. This would require drone operators to physically carry a landing pattern with them in order to ensure the safe return and landing of their drone. Furthermore, there are problems with the field-of-view (FoV) for downward facing cameras, since not much is seen by the camera when closer to the ground. To overcome these issues these techniques use either more complex landing patterns, or attach additional cameras. Our work focuses on a new area, landing at the UAVs starting position in an unstructured and unknown environment. This enables drones to be autonomously deployed in the field and to return to their starting position while following the takeoff path in reverse. This allows UAVs to land in a wider variety of scenarios and does not require any structure (e.g. landing pad) to be put in place. This could assist with faster deployment of UAV systems in the field and increase their popularity and efficiency. Additionally, the fisheye camera assists in addressing the FoV problem faced by previous work, resulting in an overall simpler hardware solution.

Figure 1.2: Example Landing Pattern.

1.2 Challenges Contributions

In order to accurately land a quadrotor in an unstructured environment, without prior knowledge of the takeoff location several challenges must be addressed. First, the drone must be able to localize relative to where it has taken off from. Second, it must be able to guide itself to ensure a safe landing. Third, it must be able to continue to localize as the drone approaches the ground and the area seen by the

FoV decreases. This work addresses them with a light weight sensor suite meant for small UAVs that are meant to be sold at a lower price point.

This work addresses these problems with a method inspired by Visual Teach and Repeat (VTR) [8]. During the takeoff a set of images are recorded. During landing, the drone localizes to these images and descends along a similar path back to its initial position, as seen in Fig. 1.2. The approach improves the safety of landing twofold, by landing in the same starting position, there is a high likelihood of the location still being safe, and by taking a path similar to the one from takeoff, the drone can avoid obstacles that were avoided during takeoff. With a fisheye camera, the area seen when the drone is close to the ground is sufficient for landing, and is wide enough at the start of descent to correct for error from the GPS. To the best of our knowledge, we present the first algorithm for precision landing of a drone in unstructured environments, such as those in Fig. 1.3, with an average accuracy of 40cm. In summary, the contributions of this chapter are described below:

Figure 1.3: Landing locations: depicted are two of the landing locations used during testing.

- An algorithm for safe precision landing of quadrotors in unstructured environments.
- Experimental results on precision landing in various environments with baseline comparisons.
- Experimental comparison of a pinhole and fisheye lens based camera for the task.

Chapter 2

Related Work

2.1 Related Work

This section will go over work done in the area of precision landing and adjacent fields that have influenced the proposed method. Specifically, the related work covers precision landing in structured environments, visual servoing, visual teach and repeat, and image comparison techniqeus for structure.

2.2 Precision Landing

Within the area of precision landing for vertical takeoff and landing (VTOL) vehicles, much work has focused on using helipad design. Specifically, helipads with concentric circles and with an ”H” in the center are common along with other custom designed helipads [16]. Approaches using helipads with either an ”H” or T on them use pretrained neural networks to identify the letter and then using the known geometry of either the letter or the surrounding circle to estimate 6-DoF pose relative to the landing pattern [23, 27, 28]. A shortcoming of these methods is losing information on the landing pattern during the approach. Based on the size of the landing pattern and the fov of the camera, the quad-rotor will no longer be able to see the landing pattern and be unable to estimate it's pose.

Approaches using simpler geometric based methods also exist. These landing pads are generally made of varying concentric circles or combinations of unique shapes and color [2, 12, 15]. For these approaches, the method of landing pattern identification generally use Gaussian blurring and dilation to reduce noise, and then use tools such as Hough Transforms, Canny Edge Detection, or contour detection. The FoV issue still remains in these approaches and each has a different approach for handling it. Some of the landing patterns use unique circular patterning with varying size to ensure good recognition and pose estimation close up and far away [2, 5, 18]. Another approach is to use two different cameras, one for approaching from a distance and another once close up [1]. Finally, one method

has looked at using cadiotropic lens, such that the camera can see the landing pad even when close to it [15]. As mentioned before though, these methods require there to be a landing pattern of known size to be able to successfully perform an accurate landing. This is because they tend to rely on 6-degree-of-freedom (6-DoF) Perspective-n-Point (PnP) pose estimate. Without a landing pattern, other techniques and assumptions must be applied to recognize the area and land.

2.3 Visual Teach and Repeat

One method that has been used in similar scenarios to relatively localize and traverse long distances using a monocular camera is VTR. Primarily focusing on ground robots, this technique will "teach" a path to a robot via piloted traversal of the path, and then be able to robustly and accurately traverse the path using a monocular camera and other base level sensors for odometry [8, 9]. In order to estimate a 6-DoF pose estimate, the algorithm must first recognize the image from the teach pass that it is nearest to and then estimate the pose. The pose estimation is enabled with a monocular camera with the assumption of local ground planarity. After the robot re-localizes itself, it relies upon visual odometry to take it to the next keyframe in the path.

Recent work has begun to expand this concept to quad-rotors [19, 20]. The first of these papers shows a proof of concept for VTR with a drone equipped with a downward facing camera and laser range finder. This approach builds a local 3D map is during the teach pass that the drone then localizes to during the repeat pass. The approach shows promising results, but did not look into how the technique would be affected by varying altitudes. The second paper proposes a more fully developed VTR for drones, but uses a forward facing camera and a qualitative position estimation. From these various approaches to VTR, we have developed the proposed method for precision landing, using a similar architecture, but a different method for generation of motion command and transitioning between keyframes.

2.4 Visual Servoing

The premise of visual servoing is to to directly control the robot using vision [6, 7]. The two primary kinds of visual servoing are image-based visual servoing (IBVS) and position-based visual servoing (PBVS). For both techniques the goal is for a robot with a camera attached to it, or looking at it to move from a current position, to a new desired position. The new desired position is represented by an image that the camera can see if moved into the correct location.

For IBVS, the control inputs are generated by pixel the measured pixel error between a current image and a desired image. For PBVS, the control inputs are generated by 3-D points rather than the projected 2-D image points. This control

strategy can be represented in a similar manner and is shown below. PBVS is similar in nature to the methods used landing patterns where a model is known. In the proposed work, there is no model of the position from which the drone took off, but a planarity assumption for the ground from which the quad-rotor took off can be made. From this assumption, we can estimate pose from one camera frame to another if the height at each position is known. With the scale being known, a method similar to PBVS can be used for the controller input.

Both of these methods require some form of comparison between the two images in order to generate the command inputs. There are various image comparison methods for gaining information about the structure of the environment. The next portion of the related work goes over different major approaches to this problem.

2.5 Image Comparison for Structure

There are many ways to compare images to gain structural information about the environment. Two primary approaches are sparse and dense methods. Sparse methods use feature based techniques that extract a relatively small subset of key points from an image with descriptors describing them. These can then be compared and matched with key points and descriptors from another image and the point correspondences can be used to estimate structure between the images. For dense methods the pixel values are used rather than key points. All the pixels or a large portion of them are used to estimate some form of transformation between the two images, such as a homography.

For feature based image comparison strategies, there are various hand crafted and learned features that could be used. Popular techniques available on OpenCV include SIFT, SURF, FAST, and ORB.

Scale-Invariant Feature Transforms (SIFT) provides descriptors for features that are invariant to scale and rotation [17]. It finds features by finding the difference of Gaussians (DoG) several times. Specifically, a DoG is the difference of an image blurred with a Gaussian of a set σ by the same image blurred by a Gaussian with a different σ. It then looks for local maximas across the different DoGs to determine key points, rejecting some based on an intensity difference threshold and others based on an edge threshold. This ensures the features are scale invariant and not edges. Then the descriptors are extracted with 16x16 descriptor that includes an orientation histogram for rotational invariance. While SIFT provides highly robust features, the run-time to extract features and match their descriptors with that of another image can be computationally expensive, making SIFT often less ideal for real-time algorithms. Speeded-up Robust Features (SURF) looks to achieve similar robustness to SIFT, with scale and rotational invariant features, but at a higher speed [4]. SURF performs similar operations to SIFT, but rather than using a DoG, box filters are applied. Additionally, wavelet transforms are used to find a domi-

nant direction to add rotational invariance. Overall, SURF has similar robustness to the features, but a faster run time.

Even with SURF's faster run time, it is not always the best option. Features from Accelerated Segment Test (FAST) is another quick method to extract features using a decision tree based method that looks at a ring of 16 surrounding pixels thresholds if they are darker, similar, or brighter than the center pixel and uses this information with the decision tree to quickly compute key points [21]. This method is less robust than the SURF and SIFT, but much faster. Oriented FAST and Rotated BRIEF (ORB) is another fast feature extraction method [22]. ORB features, as the full name implies uses a modified version of FAST features that incorporates rotational invariance, and BRIEF descriptors that allow for rotation to be incorporated into the descriptor. ORB claims to have faster run-time than SIFT and SURF and better performance than SURF.

For dense methods one of the popular dense methods available for image comparison is the inverse compositional Lucas-Kanade [3]. In the original Lucas-Kanade method, a warp is estimated to transform an image to match a template image. This is accomplished by reducing the sum of squared errors of the difference between the warped image and the template image.

$$\sum_{x}[I(W(x;p)) - T(x)]^2 \tag{2.1}$$

In order to do this the gradient of the image is warped and then the Jacobian and Hessian are evaluated from this the direction of steepest descent is found and an update is for the warp is found. This process is done iteratively, which is computationally expensive since the Hessian must be calculated each time. The inverse compositional Lucas-Kanade method reverses the roles of the image and template, allowing for the Hessian to be computed once at the start of the process and held constant for each iteration. This method has proven to be very useful and performs well in estimating homographies and other warps between images. One of the downsides of Lucas-Kanade and other dense methods is that they need to be seeded well in order to converge to the correct minima since these methods rely on gradient descent and the problem is non-convex.

Chapter 3

Approach

The approach is inspired by VTR [8], where the takeoff is the teach pass and the landing is the repeat pass. In this section, we will discuss the rationale and details of this method: the teach pass (takeoff), repeat pass (landing), position estimation, recovery, and control will be explained. The entire pipeline can be seen in Fig. 3.1, which runs at 15 Hz onboard on an NVIDIA Jetson TX2.

3.1 Design Rationale

There were many critical design decisions in this method's approach which this section will go over. In Section 1.2, it stated that some of the main challenges of precision landing in an unstructured environment include determining the starting position, estimating the position error between the current position and the starting position, handling the FoV challenge, and controlling the drone back to its initial position. The primary design decisions are made by determining the requirements, creating a morphological chart and performing trade studies on the various items in the morphological chart. The requirements of the system can be classified into two categories: functional requirements, which are actions that the system must be able to perform, and non-functional requirements, which the system does not perform but must satisfy. Below are the functional and non-functional requirements on the system.

Functional Requirements:

- Sense the environment
- Collect data on takeoff position
- Navigate back to takeoff position consistently given the drone starts within GPS error of the takeoff location
- Give commands to control drone to takeoff position

- Recover if blown off course

Non-Functional Requirements:

- Low cost
- Light weight
- Simple hardware setup
- Seen does not contain planted structure
- Low compute

Given these requirements, a morphological chart can be created that gives options for the various functional requirements. The morphological chart can be seen in Table 3.1. The largest amount of options are contained in the "Sense" category. For sensing, all items are assumed to have a IMU since the drones need to have an IMU attached. GPS was out-ruled for previously mentioned reasons, as GPS error is too large. The LIDAR option cannot be considered as LIDAR systems are too expensive and heavy for the system and would not satisfy the non-functional requirements. This leads to camera based options remaining. Monocular camera options were considered over the stereo based options as they have less hardware complexity, and monocular cameras are easier to calibrate than stereo cameras are, simplifying the setup and decreasing the cost. For the direction of the camera, the downward facing camera is selected as it provides the best information regarding the axes parallel to the ground. To assist with the vertical axes, pairing the camera with a laser range finder was selected as it provides useful additional information and doesn't increase cost or complexity by much. Finally, not shown in Table 3.1, the option of fisheye lens and pinhole lens for the monocular camera is considered. The fisheye camera was selected based on the wide FoV, but a comparative analysis is done between the two options to verify the choice 4.4.

The next column focuses how to collect data during the takeoff, which is heavily tied with the next column, how to navigate to the initial position. The three options can be summarized as relying on localization, relying on full SLAM with loop closure, or relying on relative localization to data from takeoff (VTR). In considering these three options, the first two rely on explicitly identifying the takeoff location, which is difficult to do as the downward facing camera does not provide much information until the drone has lifted off. Additionally, localization has drift build up that is undesirable. Full SLAM with loop closure can be computationally expensive. This leaves collecting sensor readings to be used later for relative localization. For relative localization, there are many ways to estimate the transformation between two images including the method of image comparison, and how the transformation is calculated. Many of the options for image comparison were presented in Section 2.5. The dense methods are not considered as the the initial

Sense	Collect Data at Takeoff	Navigate to Initial Position	Control	Recover
GPS	Record state-estimate at intial position	Compare state-estimate positions	PID	Fly up
Single Forward Facing Camera	Build map and record initial state-estimate at inital position	Perform loop-closure and compare state-estimate	Adaptive Control	Restart landing
Single Downward Facing Camera	Collect sensor data at variable points during takeoff	Compare current sensor readings to individual ones from takeoff		
Single Downward Facing Camera and Laser				
Stereo Camera				
LIDAR				

Table 3.1: Morphological chart comparing various approaches to functional architecture.

error between images can be quite high and the registration methods are likely to fall into a local minima. For sparse methods, an appropriate feature was needed to be selected. While SIFT is very robust, its computationally expensive. Other feature-based techniques are also computationally expensive, such as learned features or features for spherical images, which hinder the low compute requirement and slows the algorithm speed. ORB features are fast and robust and are selected for these reasons.

For control the two primary techniques looked into PID control and adaptive control strategies. Adaptive control strategies have been used in previous precision landing with structure research and show to be effective as the controller responds well to system noise and environmental disturbances, but is more complicated and requires a model of the UAV. Due to this additional complexity, a PID controller was initially used for this work to prove the validity of the approach. Future work will look into incorporating an adaptive control strategy.

Finally, to accomplish recovery two strategies were considered: flying upward or restarting the landing. The idea of recovery is that if the UAV is blown off-course by wind that it can still navigate back to the takeoff position. Flying upwards would increase the area seen by the camera potentially allowing the camera to overlap with an image from takeoff and begin landing again. Restarting the landing procedure by navigating with GPS back to the starting position of the landing sequence would be reliable, but would take more time. The strategy to fly upward was selected as the primary strategy as it has the possibility to recover faster and avoids using the GPS.

3.2 Takeoff & Landing

During takeoff, a set of images is recorded in regular intervals with a fisheye lens camera. The wide FoV of the fisheye provides key information at the start of the takeoff when the camera is very close to the ground and good information at higher altitudes. Images are recorded at a higher rate while the drone is close to the ground (every 0.1m in the first 2 meters) and at a slower rate as the drone ascends higher (every 0.25m until 7.5m). This is done to compensate for the fact that the ground seen by the UAV changes much more drastically when initially ascending. Details on the image recording rate and other parameters used for the method can be found here. Once all of the images are recorded, key points are found and ORB descriptors are extracted and saved to be used during the landing phase. This process only takes seconds. Furthermore, it allows for faster run-time during landing, since the features extracted do not need to be recalculated during the landing procedure. In addition to the images, a downward facing laser range finder is used to record the height of the drone and an on-board Inertial Measurement Unit (IMU) is used to estimate and record the current roll and pitch of the drone. These measurements are recorded for each image taken.

For landing, since this method is for the final portion of a return home function, the drone starts nearby the final position from takeoff at 8 m height. Then the current image is compared with the image taken at an altitude just below the current altitude of the drone, which comes from the laser range finder. This image is close enough in height such that the scale change does not negatively effect the ORB feature matching process. A relative position estimate to the image from takeoff is estimated in the form of a 2-D rigid body transform. The method to find the 2-D rigid body is explained in Section 3.3. This estimate is then used to command the drone towards that position via a position controller. While the drone approaches the position of the image from takeoff, it constantly descends. Once the drone goes below the height of the current takeoff image being compared to, the next closest image is used. A simplified version of this sequence can be seen in Fig. 3.1. This process commands the drone along a similar path from takeoff back to it's original position.

Once the drone is 1m above the ground it stops descending until the estimated position error with the image from 1m during the takeoff is below a certain threshold. Once this occurs the UAV will enter into a final descent mode and rely on state estimation to finish the landing going straight down.

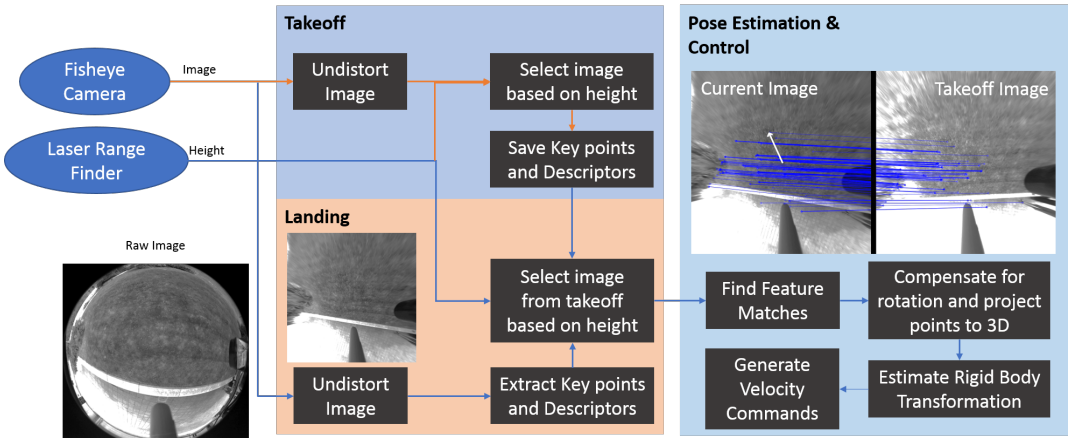

Figure 3.1: Pipeline of algorithm with steps during takeoff having orange arrows and steps during landing having blue arrows. Additionally, a sample of a raw image, undistorted image, and matched images are shown. In the matched image sample above the Pose Estimation and Control section shows how matches are found on the image and a resultant white arrow displays the motion required to minimize the position error.

3.3 Position Error Estimation

To initialize the landing process, the drone uses its state-estimate to fly above where it started. Once the drone is in position it needs to be able to localize. In order to accomplish this task we used a downward facing fisheye camera, a set of images as well as height and angle measurements collected during takeoff. A $180°$ FoV fisheye lens was selected as it provides much more information about the ground below the drone than a pinhole lens. This is particularly beneficial at the start and end of the landing, allowing for a larger offset at the start, 8m above ground, and for more reliable motion commands close to the ground. The primary steps for the position error estimation pipeline include: image undistortion, key point and descriptor extraction, key point matching, angle correction and scale recovery, and error estimation, which can be seen in Fig. 3.1.

Due to the large radial distortion of a fisheye lens, standard feature extraction methods will not work well across the image. There are feature extraction based methods that are invariant to this radial distortion, but have slower run times [29]. The camera can be calibrated with the radial distortion being taken into account with the following equation

$$\begin{bmatrix} \mathbf{P}_x \\ \mathbf{P}_y \\ \mathbf{P}_z \end{bmatrix} = \lambda \begin{bmatrix} p'_x \\ p'_y \\ a_0 + a_2\rho^2 + a_3\rho^3 + a_4\rho^4 \end{bmatrix} \tag{3.1}$$

Where $\mathbf{P}$ is the point in 3-D space, p' is the point in the fisheye image, $\rho = \sqrt{{p'_x}^2 + {p'_y}^2}$, and a_{0-4} are constants estimated during the camera calibration [24]. Given this model for the camera, the image can be undistorted and have estimated pinhole camera parameters associated with the new image. Given this, the images are projected to a flat image plane [13,14] and ORB features are extracted from the drone's current image and matched the descriptors from the corresponding image from takeoff in order to localize during the descent [22]. ORB features were selected over learned neural network features and features made for spherical images as literature shows that they are much slower [26,29], which would hinder the speed of the control loop.

Once features are matched, the pose is estimated relative to an image from takeoff. This could be done in many ways including estimating the homography of the ground plane or finding the fundamental or essential matrix for the image pair and estimating pose from these matrices, but since most drones are equipped with an IMU, less parameters need to be estimated. In fact, with the roll and pitch of each image known, there are two fewer parameters to estimate. Additionally, with the assumption of a flat ground plane and the measured height of the drone, the height is known removing the scale ambiguity. This results in only needing to estimate the horizontal position and yaw. Given this information the following method is used to estimate the position of the drone. First, the feature points' position is corrected using the estimated roll and pitch. Then they are projected into 3-D space using a planar assumption, the measured height, and intrinsic camera parameters. These two steps can be accomplished by rearranging the rotational flow correction equation from [11] and use it for correcting the rotation of an individual point and project it into 3-D space:

$$\mathbf{P}_z = \mathbf{Z}\cos(\theta_x)\cos(\theta_y) \tag{3.2}$$

$$\mathbf{P}_x = \frac{(-p_x - \sin(\theta_y)f)\mathbf{P}_z}{f} \tag{3.3}$$

$$\mathbf{P}_y = \frac{(-p_y + sin(\theta_x)f)\mathbf{P}_z}{f} \tag{3.4}$$

Where $\mathbf{Z}$ is the measured height from the laser range finder, θ is the current estimated roll and pitch of the drone, $\mathbf{P}$ is a feature point's metric coordinates in 3D with respect to the drone's current position, p is the feature point's position in the camera frame, and f is the estimated focal length of the camera. Once both sets of points are projected onto the ground plane, a 2-D rigid body transformation can be calculated and give a metric relative position error. The 2-D rigid body transformation can be calculated in the following manner. First, the centroid of the each image features sets are found.

$$\mathbf{p}_{c,centroid} = \frac{1}{n}\sum_{i=0}^{n}\mathbf{p}_{c,i} \tag{3.5}$$

$$\mathbf{p}_{t,centroid} = \frac{1}{n}\sum_{i=0}^{n}\mathbf{p}_{t,i} \tag{3.6}$$

Where $\mathbf{p}_c$ represent points from the current image and $\mathbf{p}_t$ represent points from the image from takeoff. Next, the rotation between the two sets of points can be determined with the orthogonal Procrustes Problem [25]. This method finds the covariance between the two sets of points in order to determine the rotation between them in the following manner.

$$\mathbf{p}_{c,o} = \mathbf{p}_c - \mathbf{p}_{c,centroid} \tag{3.7}$$

$$\mathbf{p}_{t,o} = \mathbf{p}_t - \mathbf{p}_{t,centroid} \tag{3.8}$$

$$\mathbf{M} = \mathbf{p}_{t,o}\mathbf{p}_{c,o}^T \tag{3.9}$$

$$[\mathbf{U}, \Sigma, \mathbf{V}] = SVD(\mathbf{M}) \tag{3.10}$$

$$\mathbf{R} = \mathbf{V}\mathbf{U}^T \tag{3.11}$$

Finally, the translation can be found by applying the rotation to the current set of points and then taking the difference between the centroid of the rotated current set of points to the set of points from takeoff.

$$\mathbf{p}_{rc,centroid} = \frac{1}{n}\sum_{i=0}^{n}R\mathbf{p}_{c,i} \tag{3.12}$$

$$\mathbf{T} = \mathbf{p}_{t,centroid} - \mathbf{p}_{rc,centroid} \tag{3.13}$$

Given this estimation method, random sample consensus (RANSAC) is used to reject outliers while finding the rigid body transform. Since the rigid body transformation estimation only has three parameters that it is estimating for, fewer iterations of RANSAC are needed to be performed in order to have a confident estimation. The equation for this is seen below [10].

$$k = \frac{log(1-P)}{log(1-w^n)} \tag{3.14}$$

Where k is the number of iterations, P is the probability that one of the iterations does not contain an outlier, w is the expected percentage of inliers, and n is the number of samples per iteration. Given, $w = 60\%$, $P = 99\%$, and $n = 3$, only 18

iterations are necessary for a confident estimation. Given a lower number of inliers in the data set such as $w = 30\%$, only 169 iterations are necessary. This is far less than just estimating the essential matrix since 5 points are needed, which would be 57 or 1893 iterations respectively. Overall, this pipeline runs at 15 Hz on-board the a NVIDIA Jetson TX2 on the drone.

3.4 Recovery

If the drone is descending and is pushed off course by a gust of wind, there is a chance that the current image may no longer align with the image from takeoff. If this happens then the position estimation will no longer be valid. This scenario can be detected through the position estimation method though. When two images are compared from differing scenes the feature matches between them tend to be poor sporadic matches. Given this, there will be a low consensus during RANSAC ($< 30\%$), which can be used to detect this scenario. If this occurs, the drone ascends to increase its FoV until it can again match with the image from takeoff.

3.5 Control

Once the position error is estimated, it is used as an input to a PI position controller. This generates a velocity command that is sent to an attitude controller on-board the X-Star Premium Quadcopter using the drone's on-board state-estimate. The drone receives velocity commands in the horizontal axes from the position controller during the descent and a constant descent speed at 15Hz until it is 1m above the ground. Here the horizontal velocity commands are sent from the position controller while the drone hovers at a constant altitude. Once the drone aligns with the concurring image from takeoff it performs a final descent only receiving a downward velocity command. The control strategy uses this simple PI position control as a baseline approach as it is very simple to implement and fairly effective. No derivative gain was used as the derivative tended to be quite noisy. Future work will focus on improving this aspect of the project by looking into adaptive control methods that can better take environmental errors into account.

Chapter 4

Results

In this section, we discuss the experimental setup, experiments, and results. Initial experiments were done in simulation to initially test the algorithm. Next, we perform a drone landing experiment to demonstrate the performance and robustness of our precision landing algorithm, as well as present a baseline comparison to landing using only GPS-IMU based state estimate. Additionally, data is collected and presented for comparison of the efficacy of both fisheye and pinhole camera lens.

A second experiment was performed to test the algorithm's robustness to scene change. Landing tests were performed with a van next to the drone for takeoff, and before landing, the van was moved to a set further distances as shown in Fig. 4.4. The setup for our experiments can be seen in Fig. 4.1 and all the data collected can be found here.

4.1 Simulation Experimental Setup

For the simulation experiments used AirSim by Microsoft, which uses Unreal Engine for photo-realistic images. A simulated quadrotor with a downward facing pinhole camera was used for the experiment. The ground truth of the altitude, roll, and pitch of the simulated quadrotor were used for position estimation and a PID position controller sent velocity commands to a built in velocity controller that comes with the package.

4.2 Simulation Results

The experiments were performed in a free simulation environment provided by Unreal Engine in a mountain area. A video displaying the environment and example of the experiment can be found here. The drone would be commanded along

X_{init} (m)	Y_{init} (m)	$X_{takeoff}$ (m)	$Y_{takeoff}$ (m)	$X_{landing}$ (m)	$Y_{landing}$ (m)	X_{final} (m)	Y_{final} (m)	Final Error (m)
0.054	-0.001	2.400	1.600	3.00	2.00	0.225	0.131	**0.216**
0.024	0.002	2.400	1.600	1.00	0.00	0.053	0.139	**0.139**
0.014	0.029	4.000	4.000	5.00	5.00	-0.104	0.277	**0.275**
-0.007	0.008	2.400	3.200	3.00	4.00	0.102	-0.063	**0.130**
0.000	0.000	2.400	3.200	2.00	2.00	0.075	-0.064	**0.099**

Table 4.1: Precision-Landing Simulation Results

a diagonal path to a position 10 m above the ground. The final image to be used during landing was recorded at 8 m, which means there was a horizontal and vertical offset between the final image recorded during takeoff and the starting position for landing. This was done purposely to help simulate the potential GPS error that would occur when the drone would return to the final takeoff position. Once the quadrotor finished the ascent, the landing was started. After the landing the final position was recorded. The results can be seen in Table 4.1. The average accuracy was 17 cm, which gave us confidence in moving forward to experimenting on the hardware system.

4.3 Hardware Experimental Setup

Experiments were conducted with the X-Star Premium Quadcopter by Autel Robotics, seen in 4.1. For sensing, an embedded computing device (NVIDA Jetson TX2), ueye camera equipped with a Lenssagon BF10M19828S118 C fisheye camera lens, and SF30 laser range finder are equipped to the drone. A ueye camera with a S-Mount lens Lensagon B10M5022S12 pinhole camera is attached for the comparison to the fisheye. The Jetson TX2 has Ubuntu 16.02 installed on it with ROS Kinetic, OpenCV, and Eigen libraries. The Jetson TX2 is connected to the X-Star Premium's embedded computing device in order send velocity commands to the drone. A second ueye camera is attached with a pinhole lens for offline comparison against the fisheye lens. The fisheye camera records monochrome images at a rate of 15 Hz, while the pinhole records them at 10 Hz. The recording difference was due to the bandwidth available across the USB connection to the computer. The fisheye camera has a resolution of 1100x1100, and the pinhole camera has a resolution of 1280x1024. There are a few additional tunable parameters for the algorithm. They can be seen here in the Experiment Parameters sheet and include: Image recording threshold (inital rate and final rate), ORB feature count, image size, and ratio test setting. These parameters were selected prior to experimenting and held constant throughout the various scenes. The primary trade-off in selecting the parmeters is run-time speed

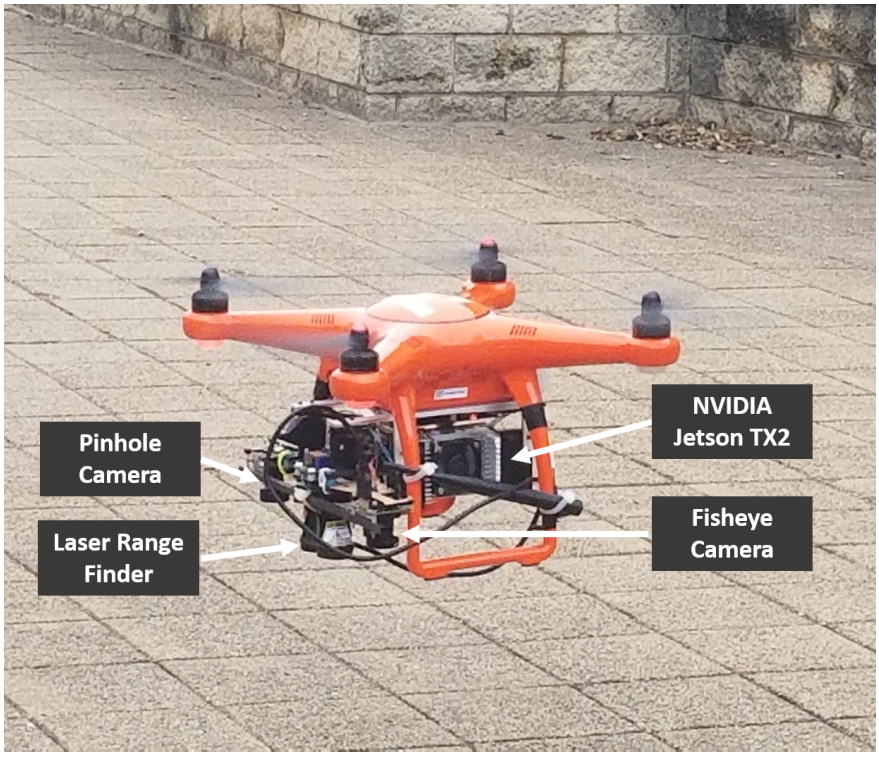

Figure 4.1: Hardware experimental setup with sensors equipped.

and accuracy in position estimation. For example, the number of ORB features will increase the time to find and match features, but provide greater robustness in position estimation. The numbers were selected based on prior experimental trials.

4.4 Landing Experiments and Comparisons

The objective of the landing experiments was to verify the accuracy and precision of the landing algorithm and compare it against the UAV's GPS-IMU based state

Method	Environment	Number of Trials	Accuracy (cm)	Standard Deviation (cm)	Wind Speed (m/s)	Max Gust (m/s)
Precision Landing	Patio	18	38	23	3	5
Precision Landing	Grass 1	17	44	42	3	5
Precision Landing	Turf	10	34	17	7	12
Precision Landing	Grass 2	10	45	24	8	15
State Estimate	Patio	10	183	74	2	0
State Estimate	Grass 1	10	359	205	2	0

Table 4.2: Precision-Landing and State-Estimate Landing Results

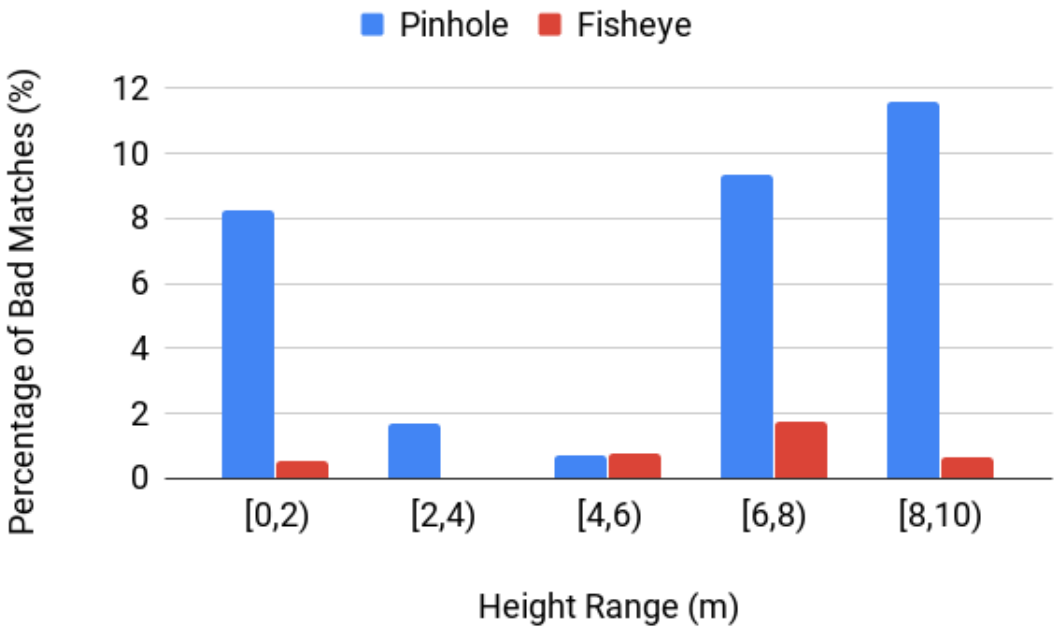

Figure 4.2: Comparison of Fisheye and Pinhole camera lens for the task of localization. Each bar displays the percentage of bad position estimations during landing across various trials in the different environments. The fisheye camera has fewer bad matches than pinhole camera.

estimate used for landing. Additionally, data for a comparison between the performance of the fisheye lens and a pinhole lens was collected during these experiments. The experimental procedure is as follows. First, an environment is selected and the drone is placed in its initial position. For measurement purposes, the drone is placed next to either a natural landmark, such as a corner of a tile, or an artificial landmark, such as a golf tee in the grass. Second, the drone is manually piloted with varying paths taken (i.e. straight, diagonally, curved, etc.). Once the take-off is completed, it is flown in a random direction in varying distances (approximately 1-3m). This is to simulate the error from GPS that may occur from a drone's state-estimate returning to the take-off position. Finally, the landing procedure is initiated and the drone autonomously lands. After the drone has landed, measurements are taken for the drone's displacement from the initial position. This test was performed both with the precision landing algorithm using a fisheye lens, and using a GPS & IMU based Extended Kalman Filter (EKF) state-estimation running on-board the X-Star Premium Quadcopter. This test has been performed in a wide range of environments, grass, turf, and stone tiling. Examples of the experiment and a short summary of the method can be seen here. Wind speeds are recorded from METAR report from the Allegheny County Airport (KAGC) to observe the

algorithms robustness to high wind speeds and gusty weather. The performance for the proposed method and the GPS-IMU based EKF can be seen in Table 4.2.

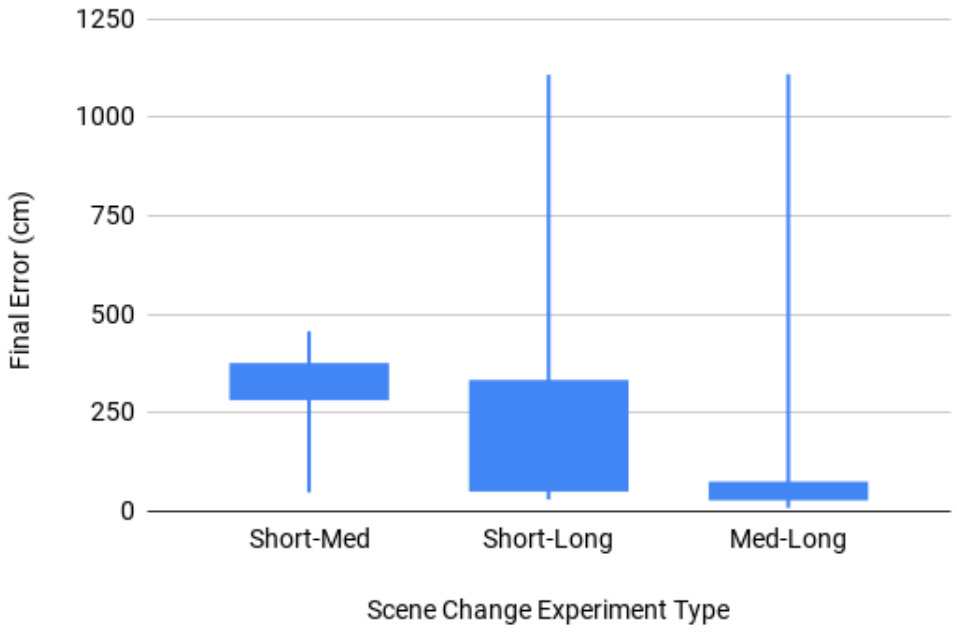

Figure 4.3: Results from scene change experiments. Each box-and-whisker plot shows the resulting accuracy across the three scene change tests. Short-Med means the van moved from 4ft from the drone to 16ft from the starting point. For Short-Long, the van moves from 4ft away to 41ft. For Med-Long the van moves from 16ft to 41ft.

Data is recorded from both the fisheye and pinhole cameras and analyzed offline to determine the reliability of position estimation for each lens type. The algorithm is ran on the data and if RANSAC cannot determine a consensus on a rigid body transform, the image instance is considered a failed localization. The percentage of failed localizations across multiple trials can be seen in Fig. 4.2. This shows that the fisheye lens can localize well even when close to the ground and has a lower failure rate throughout the landing than the pinhole.

4.5 Scene Change Experiments

The experimental procedures for the scene change experiments are the same as the landing with the addition that a van moves from a set initial position to a new set position further away. Three sets of five experiments were ran with each set having a different starting and ending offset of the van from the drone's initial position.

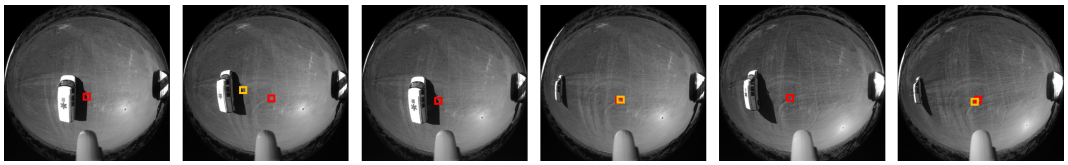

Figure 4.4: Displays three scene change experiments. From left to right in image pairs is each experiment: Short-Med, Short-Long, and Med-Long, with the take-off location in red and landing in yellow.

Specifically, the three sets of positions are defined as short-medium (the van starts 4ft away from the drone and moves to 16ft after take-off), short-long (the van starts 4 ft away and moves to 41 ft away after take-off), and medium-long (the van starts 16 ft away and moves to 41 ft away). The objective of these three distances was to put the van either within the drone's central view the entire time, the edge view throughout the take-off, or mostly out of view for the full take-off and see how moving the van to these various positions affects the performance of the algorithm. The results can be seen in Fig. 4.3.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

The results from the experiments show that our algorithm can safely and accurately land a quadrotor to its original takeoff position in various environments, and that it outperforms the IMU-GPS EKF state-estimate by a statistically significant margin. This is likely due to taking more rich data from the camera into account in measuring the position of the drone during landing. Additionally, our algorithm proves to be robust to wind as the results do not vary with the wind conditions. This is likely due to the wide FoV of the fisheye lens as well as the recovery behavior of the algorithm allowing wind to push the drone off-course, but still have the view from takeoff in view or be able to gain more FoV quickly in the recovery behavior mode. There have been three observed failure cases, if the ground plane has a sudden large change in height, when there is an aggressive roll or pitch during the takeoff and an image is recorded, and when the scene has a large amount of change. When there is a large change in height the planar assumption is no longer valid and an accurate rigid body transform cannot be calculated. Additionally, a large amount of data from the takeoff is missed because the key frames recorded during the takeoff is based on change in height. When there is an aggressive roll or pitch during takeoff, images recorded may be blurred, causing the feature matching to fail. For scene change the effects are further explored with the scene change experiment.

From the fisheye and pinhole camera lens comparisons, we conclude that the fisheye lens, although having high radial distortion, provides more information for the task of autonomous precision landing. This is particularly useful in two scenarios. Firstly, when a drone initially starts its descent. This can be seen in the results when there is a large offset from the end of the takeoff to the beginning of the landing. Overall, this results in a more robust landing capability as the drone relies less on the state-estimate since it can have a larger position error when starting the landing algorithm and still have overlap with the final image from takeoff, whereas the pinhole camera may not have enough initial overlap to start the landing given

a starting position with a large offset to the final takeoff position. Secondly, if the drone is pushed off course by wind when it is closer to the ground, the fisheye will likely have enough overlap to maintain a position estimate, but the pinhole lens may have no overlap at all. In essence, due to the fisheye lens' greater FoV, even if pushed off-course by wind when close to the ground, it maintains visual of the image from takeoff and can recover, whereas the pinhole camera will loose overlap between the images and be unable to estimate position error. This shows that the fisheye lens based cameras can help solve the FoV issue seen in many other precision landing based works and reduce the necessary hardware to an individual camera instead of two cameras.

The scene change experiment demonstrated the effects of a change in the environment for the landing algorithm with three different scenarios. The first where a large object that provides many of the features in the images moves to a new position relatively close to where it started. This causes the landing algorithm to be drawn off course and land close to the area with the highest number of the original features. This short coming gives interesting insight on potential other uses for the proposed method. One, landing on a moving landing pattern. If the landing pattern is highly textured, even if the landing pad were moving in the environment, the drone would be drawn towards it. Second, if a highly textured landing pattern was used, the drone could change where is lands relatively easily. This shows potential for this algorithm to not only work for precision landing in unstructured environments, but structured ones as well. The other two scenarios where the van moved far from its initial position resulted in accurate landings. Each of these scenarios had one edge case of the drone following the van, likely because the van was not far enough away to be out of the drone's FoV. From these two results it can be seen that if strong features are removed, the drone can still land accurately since there will still be features in the area from takeoff that the landing algorithm can match to.

5.2 Future Work

There are several interesting areas to be explored for future work including further exploring the capabilities of the proposed method, improving the controls of the algorithm, completely removing dependence of GPS, and developing methods to further improve the robustness of the algorithm. One way to further explore the capabilities of this algorithm is landing on a highly textured surface, both stationary and moving. This would be interesting to observe to see if the current algorithm can be used in various scenarios for landing. Additionally, if the added structure in the environment would assist in the overall accuracy of the landing as there would be good features for matching during the final portion of the landing. For controls work, several authors of previous precision landing papers for precision landing in

structured environments use an adaptive controller [2, 15] rather than a standard PID controller. Particularly something like Adaptive Disturbance Rejection Controller (ADRC) that can take into account disturbances such as wind or noise in state-estimation could be used to improve the overall landing and the final portion of the landing where images are no longer used for the position estimation. For removing the need for GPS in the algorithm switching to a visual or visual-inertial based state estimate would be useful. This would be beneficial for the algorithm as it could then work in GPS denied environments and potentially have better state-estimate while close to the ground.

For improving robustness of the algorithm there are several avenues that can be taken. The change in the control strategy and state-estimation strategy previously mentioned would both likely improve the performance of the algorithm. Two methods to improve the takeoff procedure of the algorithm include using a change in FoV calculation to determine when to record the data to be used for landing, as well as using the angular rate from the IMU to filter out blurry images. Both of these should result in the algorithm being more robust to more aggressive takeoffs as more useful data will be collected during the takeoff. Particularly in the case of takeoffs with more horizontal motion, as the change FoV would incorporate that. Additionally, in takeoffs with more aggressive horizontal motions as those images that are blurry would not be recorded. For improving the robustness of the landing portion of the algorithm potentially including a machine learning based element to detect objects and use those for estimating position error, as well as using those to identify if the objects in the scene have moved. This could be useful to prevent the drone from being drawn off-course as seen in the scene change experiment as the object of the van could be detected as moving in relation to other features or objects and therefore ignored during the relative position error estimation.

Finally, the last major piece of future work is to incorporate this algorithm into a larger intelligence stack to create a fully autonomous drone. For this to happen there would need to be a way to combine the precision landing algorithm with other various algorithms such as exploration, go-to functionalities, etc.

Bibliography

[1] S. Arora, S. Jain, S. Scherer, S. Nuske, L. Chamberlain, and S. Singh. Infrastructure-free shipdeck tracking for autonomous landing. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 323–330. IEEE, 2013.

[2] W. Bai, F. Pan, B. Y. Xing, C. Pan, and M. X. Pei. Visual landing system of uav based on adrc. In *Control And Decision Conference (CCDC), 2017 29th Chinese*, pages 7509–7514. IEEE, 2017.

[3] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

[4] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[5] A. Benini, M. J. Rutherford, and K. P. Valavanis. Real-time, gpu-based pose estimation of a uav for autonomous takeoff and landing. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3463–3470. IEEE, 2016.

[6] F. Chaumette and S. Hutchinson. Visual servo control. i. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.

[7] F. Chaumette and S. Hutchinson. Visual servo control. ii. advanced approaches [tutorial]. *IEEE Robotics & Automation Magazine*, 14(1):109–118, 2007.

[8] L. Clement, J. Kelly, and T. D. Barfoot. Monocular visual teach and repeat aided by local ground planarity. In *Field and Service Robotics*, pages 547–561. Springer, 2016.

[9] L. Clement, J. Kelly, and T. D. Barfoot. Robust monocular visual teach and repeat aided by local ground planarity and color-constant imagery. *Journal of Field Robotics*, 34(1):74–97, 2017.

[10] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[11] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys. An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1736–1741. IEEE, 2013.

[12] C. Hui, C. Yousheng, L. Xiaokun, and W. W. Shing. Autonomous takeoff, tracking and landing of a uav on a moving ugv using onboard monocular vision. In *Proceedings of the 32nd Chinese Control Conference*, pages 5895–5901. IEEE, 2013.

[13] J. Kannala and S. S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE transactions on pattern analysis and machine intelligence*, 28(8):1335–1340, 2006.

[14] J. Kannala, J. Heikkilä, and S. S. Brandt. Geometric camera calibration. *Wiley Encyclopedia of Computer Science and Engineering*, 2008.

[15] J. Kim, Y. Jung, D. Lee, and D. H. Shim. Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1243–1252. IEEE, 2014.

[16] W. Kong, D. Zhou, D. Zhang, and J. Zhang. Vision-based autonomous landing system for unmanned aerial vehicle: A survey. In *Multisensor Fusion and Information Integration for Intelligent Systems (MFI), 2014 International Conference on*, pages 1–8. IEEE, 2014.

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

[18] T. Merz, S. Duranti, and G. Conte. Autonomous landing of an unmanned helicopter based on vision and inertial sensing. In *Experimental Robotics IX*, pages 343–352. Springer, 2006.

[19] T. Nguyen, G. K. Mann, R. G. Gosine, and A. Vardy. Appearance-based visual-teach-and-repeat navigation technique for micro aerial vehicle. *Journal of Intelligent & Robotic Systems*, 84(1-4):217–240, 2016.

[20] A. Pfrunder, A. P. Schoellig, and T. D. Barfoot. A proof-of-concept demonstration of visual teach and repeat on a quadrocopter using an altitude sensor and a monocular camera. In *Computer and Robot Vision (CRV), 2014 Canadian Conference on*, pages 238–245. IEEE, 2014.

[21] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[22] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011.

[23] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme. Visually guided landing of an unmanned aerial vehicle. *IEEE transactions on robotics and automation*, 19(3):371–380, 2003.

[24] D. Scaramuzza, A. Martinelli, and R. Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701. IEEE, 2006.

[25] P. H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[26] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 118–126, 2015.

[27] G. Xu, Y. Zhang, S. Ji, Y. Cheng, and Y. Tian. Research on computer vision-based for uav autonomous landing on a ship. *Pattern Recognition Letters*, 30(6):600–605, 2009.

[28] S. Yang, S. A. Scherer, and A. Zell. An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. *Journal of Intelligent & Robotic Systems*, 69(1-4):499–515, 2013.

[29] Q. Zhao, W. Feng, L. Wan, and J. Zhang. Sphorb: A fast and robust binary feature on the sphere. *International Journal of Computer Vision*, 113(2):143–159, 2015.