DETECTING SLIP IN A VEHICLE PERCHED ON A DYNAMIC PERCH

by

Kyle L. Crandall

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

The University of Utah

December 2015

The University of Utah Graduate School

STATEMENT OF THESIS APPROVAL

The thesis of **Kyle L. Crandall**

has been approved by the following supervisory committee members:

Mark Minor	, Chair	**10/2/2015** Date Approved
Jake Abbott	, Member	**10/5/2015** Date Approved
John Hollerbach	, Member	**10/7/2015** Date Approved

and by **Tim Ameel** , Chair/Dean of

the Department/College/School of **Mechanical Engineering**

and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

There has been much research on how to get unmanned aerial vehicles (UAVs) to perch on many different types of surfaces and objects, including flat surfaces, ramps, tree branches, power lines, etc. Many of these surfaces are static and it is easy to detect falls using inertial sensors such as accelerometers or gyroscopes. However, some perches, such as tree branches and power lines, are not static. When the UAV is perched on these perches, it will move with them, making the detection of falls from such a perch much more difficult than simply trying to sense motion.

This thesis proposes two methods for fall detection of a UAV perched on such a dynamic perch. Computer vision is used on a feed from a camera mounted on the bottom of the UAV. Optical flow is used in conjunction with a filter that segments the perch in the image from the background to estimate the relative motion between the UAV and the perch. If the motion exceeds certain bounds, the UAV is considered falling.

The second method tries to find the instantaneous center of rotation (ICR) of the UAV utilizing accelerometers and a gyroscope mounted to the UAV frame. Two methods are proposed to do this, one based on integrating the accelerometers to find the velocity at a point, the other finds the distance between the ICR and three points on the rigid frame of the UAV. The ICR estimates from these two methods are compared to an ICR estimate derived from data from an external Vicon motion capture system. The estimated ICR is then compared to the ICR of the perch that the UAV has perched on, if the two diverge enough, the perch is considered to be falling.

The proposed methods were tested experimentally by placing a test quadrotor fitted with the appropriate sensors on three different test perches: a painted PVC pipe, a PVC pipe with a swirl pattern on it, and a tree branch. The quadrotor and perch are then actuated in three different tests. The first test has the quadrotor rotating about the perch while the perch is static. The second test has the quadrotor swinging on the perch without slipping. In the final test, the swing perch angle is increased until the quadrotor falls off. Each of the 9 tests is performed 5 times. Accelerometer, gyroscope, and vision data are gathered during these tests and analyzed using the methods described in this thesis. These experiments show that the vision method works fairly well, and that the ICR method works to a degree, but there is more work to be done in that area.

For Rachel

Without you, I never could have made it.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to express my gratitude to Mark Minor for the opportunity I have had working under him for the past five years and for supporting me and helping me reach this point. I would also like to thank my committee members, Jake Abbott and John Hollerbach, for their support and input. I would also like to thank these and all my other professors for the education they provided both in and out of the classroom.

I would like to thank my colleagues, Michael Ott, David Dunlop, Varun Cumbamangalam, Dustin Webb, Jim Dotterweich, John Miller, Babak Hejrati, and Owen Barnes for their help, support, and entertainment provided during my years at the University of Utah.

Lastly, I would like to thank my friends and family for their love and support. I would especially like to thank my Dad, who taught me the value of learning and scientific exploration, and my Mom, who taught me to keep an open mind. Finally I would like to express my special thanks to Rachel Ware for her support and advice.

Without any of you, I would not be here today. Thank you.

CHAPTER 1

INTRODUCTION

Unmanned aerial vehicles (UAVs) have been a very active and popular area of research in the past few years. UAVs that can land and perch on different surfaces are a growing area of interest within this field. While most of this perching work has been focusing on landing the aircraft on some sort of stable perch that does not move relative to the inertial frame, a much more general case of this problem is landing on something that does move, like a tree branch or power cable. A critical part of remaining perched on a dynamic surface like this is fall detection. The major aspect of this is detecting slip in the interface between the vehicle and its perch.

In this thesis, two main approaches are proposed to achieve this distinction using a combination of computer vision, kinematic modeling, and inertial sensor data. The first approach proposed does an optical flow analysis with a filter to segment the perch from the background of the image to detect relative motion between the perch and UAV. The second approach attempts to calculate the instantaneous center of rotation (ICR) of the UAV and compare it to the ICR of the perch. Two methods are proposed to estimate the ICR, one based on velocity estimates and one based on acceleration estimates. Both of these methods use an extended Kalman filter to reduce the noise in accelerometer and gyroscope readings. These methods are then compared to the ICR estimate from an external Vicon motion capture system.

In this chapter, the problem will be introduced by first discussing the approaches we are proposing and the challenges that need to be overcome to complete this task. Then the previous work in perching and slip detection is discussed. Lastly, the specific contributions of this thesis will be outlined.

This thesis has been divided into three main parts. Chapter 2 will discuss the computer vision approach. Next the kinematic modeling and inertial sensing will be discussed. The last chapter will discuss the conclusions drawn from this thesis and discuss future work.

Approach and Challenges

The motion of the UAV can be broken into two components. The first component of the motion is the result of the motion of the perch the UAV is sitting on. The second component is the relative motion between the perch and the UAV. The methods proposed in this thesis look to distinguish these two components, and use the magnitude of the second component to estimate if the UAV is falling from its perch. If the slip between the two bodies is significant, then a fall can be assumed. Two main approaches are proposed, the first being a computer vision approach utilizing a downward facing camera mounted to the belly of the vehicle. The second approach relies on detecting the instantaneous center of rotation (ICR) of the UAV and comparing it to the ICR of the perch, if known. These methods are proposed because they are able to be implemented with minimal additional hardware for the UAV. Almost every UAV will be equipped with an IMU as well as a camera. In helicopter UAVs, downward facing cameras are quite common for stabilization. These methods will limit the amount of additional hardware to a couple additional accelerometers, which are small, cheap, and easy to mount; and possibly an additional camera, which could serve other purposes in addition to fall detection. Tactile sensing in the gripping mechanism of the UAV was also considered, however, this would require a significant amount of additional hardware to implement.

The computer vision part is focused around using optical flow to detect relative motion between the UAV and its perch. The main challenge with this part is distinguishing the perch from the background in the image received from the camera. A k-means segmentation algorithm and a low-pass filter on the optical flow estimation from the camera feed as well as the image itself is used to address this issue. Once the perch is separated from the background, an optical flow analysis of the perch will help with addressing whether the perch is moving relative to the UAV or not.

The second method proposed attempts to use the ICR of the UAV and compare it to the ICR of the perch to detect slip between the two rigid bodies. One of the major challenges associated with this approach is the estimation of the ICR of the UAV itself. Without the use of an external sensor such as a Vicon motion capture system, this can be a very difficult task. Inertial sensors such as accelerometers and gyroscopes tend to be relatively noisy. To help account for this, an Extended Kalman Filter (EKF) is used to help clean up the signals, as well as estimate some of the values required to calculate ICR, namely the angular position and rate, using multiple sources of data. Two methods are proposed to calculate the ICR of the UAV based on the output of an EKF. These methods are compared to the method of calculating the ICR based on external sensors. Another major challenge associated with the ICR method is the fact that the perch's ICR is required to be known a priori. This is a very limiting factor, but if the general shape of the motion that the perch is undergoing is known (i.e., swinging, translation etc.), the ICR can still be used to detect a fall.

Prior Work

There are two main areas of prior work that should be considered, the first of which is perching mechanisms and methods. There are many different perching mechanisms proposed in the literature, both passive and active, for fixed-wing and helicopter-type UAVs. Kovac et al. propose a perching mechanism for micro aerial vehicles (MAVs) that allows the vehicle to perch on various surfaces [1]. Desbians et al propose a similar spine based perching method for landing on flat surfaces such as walls [2]. Daler et al. propose an adhesive-type perching mechanism with similar goals, but argue that the adhesive approach is superior because it does not require high precision control or a mechanism that is robust to high-energy impact [3]. The major problem with these proposals is that they focus on flat surface perching on walls. This thesis is focused on detecting slip in UAVs perched on dynamic surfaces that have a tendency to move. Mohta et al. land on less sturdy line perches using an adhesive-type perching mechanism, but they do not discuss the stability of the perch once it is achieved [4]. Doyle et al. designed an avian-inspired landing system that allows UAVs to passively perch on a wide variety of perches [5]. This perching mechanism proposed by Doyle et al. was selected for use in these studies because it has the most diverse range of possible perches that are likely to be dynamic. This particular mechanism was also available for use at the start of this research. It was the most practical and convenient option considered. The methods proposed in this thesis, however, would still be applicable to other perching mechanisms for other types of UAVs as well.

Now that the UAV can perch on a wide variety of perches, the question becomes whether it maintains its perch. A major part of this process is detecting slip between the perch and the vehicle. Most slip detection research has been based on tactile sensors in gripping robots, such as the method proposed by Melchiorri [6][7][8][9]. These methods tend to rely on arrays of tactile sensors to detect slip. While the focus of this research is not perching, but grasping, it is a very similar problem. Others have proposed various techniques to detect slip in wheeled, mobile robots. Reina et al. propose a vision-based method to detect slip in such a vehicle by analyzing the tread mark created by a wheel as it rolls through dirt [10]. Gonzalez et al. use vision based slip detection on an off road vehicle which utilizes a camera that is also used for visual odometry[11]. A more applicable method proposed by Ikeda et al. for detecting slip in a gripping robot by monitoring the contact area between a finger and the gripped surface [12].

Because most perching research has assumed stable perches, there has not been much research specifically into the problem of detecting a fall from the perch. It is generally assumed that if the vehicle can achieve a perch in the first place, since the surface it is perched upon does not move, it will be able to maintain that perch indefinitely.

Contributions

Current research on perching UAVs mostly focuses on the micro subset of these vehicles, small vehicles that can barely lift themselves. However, perching in larger robots is beginning to appear. The methods proposed in this paper will work regardless of size of the robot, provided they can be equipped with the required sensors. The sensors required, however, we believe are minimal for the task of detecting a fall from a dynamic perch.

Another major difference between current research and this thesis is the types of perches being considered. Most current research is looking at perching on walls, tree trunks, or other surfaces considered to be significantly more massive than the vehicle itself, such that motion of the perch is negligible. With the perches being looked at in this thesis, this assumption cannot be made. The real-world perches being anticipated are perches such as tree branches and power lines. These perches tend to move and sway in the presence of disturbances such as wind, birds landing on the same perch, other animals, small children, etc.

As discussed above, vision methods have been considered before as a method of detecting slip in several different types of systems. This thesis aims to propose another approach to the vision problem by utilizing optical flow algorithms and a segmentation algorithm to separate the perch from the background of the image. This method would not have the requirement of sensors inside the perch, as does the approach suggested by Ikeda et al. [12] and is more like the vision method proposed by Reina [10]. As discussed previously, one of the major challenges with this method is determining which parts of the image are the perch and which parts are the background. To do this, a novel filter is proposed that uses optical flow, pixel position, and pixel values to create a feature vector used with a k-means algorithm to segment the perch from the background.

The other slip detection method proposed in this thesis compares the instantaneous center of rotation (ICR) of the UAV to the ICR of the perch it has landed on. The major contribution from this method is an exploration of the possibility of using cheap, onboard inertial sensors such as accelerometers and gyroscopes to estimate the ICR of a rigid frame. At the time of writing, there has been very little research into using inertial sensors for this type of task.

The vision method works fairly well. The filter is able to locate the perch in the image and distinguish it from the background, and the optical flow can be used to detect a fall. Even in cases where the perch does not have features that can be tracked by the camera, the filter can be exploited to detect a fall. The ICR method works to a degree, the radius method of detecting the ICR shows the most promise, however there is still some work to be done on this method before it is completely viable.

CHAPTER 2

COMPUTER VISION APPROACH

The computer vision part of this thesis uses a downward facing camera mounted to the body of the perched aircraft. This camera will have part of the perch in its view as well as some of the background. The theory behind this approach is that, if there is relative motion between the UAV and its perch, this motion will result in some optical flow on the perch in the video stream from the camera.

Because computer vision algorithms can be very computationally expensive, and timing is something important to this project that should be considered since we want to eventually deploy these methods in real time, it was decided to use the open source libraries provided in the OpenCV software package. This C++ package provides a number of headers and libraries that are already optimized and perform many useful computer vision functions such as optical flow, feature detection, as well as a basic framework for easily reading images from a camera or file, and an intuitive method for handling and processing these images.

The process of analyzing a video feed from the mounted camera is broken into three major parts. First is the optical flow analysis on the image. Then the image is segmented into the part that contains the perch and the part that contains the background. Then the optical flow of the segment identified as the perch is considered. The average optical flow of this segment is calculated. Lastly, a filter is applied to the results of the segmentation as well as the optical flow to help eliminate noise. In this chapter, each of these three steps will be discussed in detail. After the methods have been established, the advantages and limitations of this approach will be discussed.

To help explain some of the decisions made and to aid in illustrating the process, Figure 1 is an example image taken from the downward facing camera while the UAV was perched on the branch that can be seen on the right of the image.

Optical Flow Analysis

The first main step is the calculation of the optical flow. An optical flow algorithm gives an approximation of how a pixel has moved across two frames of a video. In their paper, Horn and Schunck describe optical flow as "…the distribution of apparent velocities of movement of

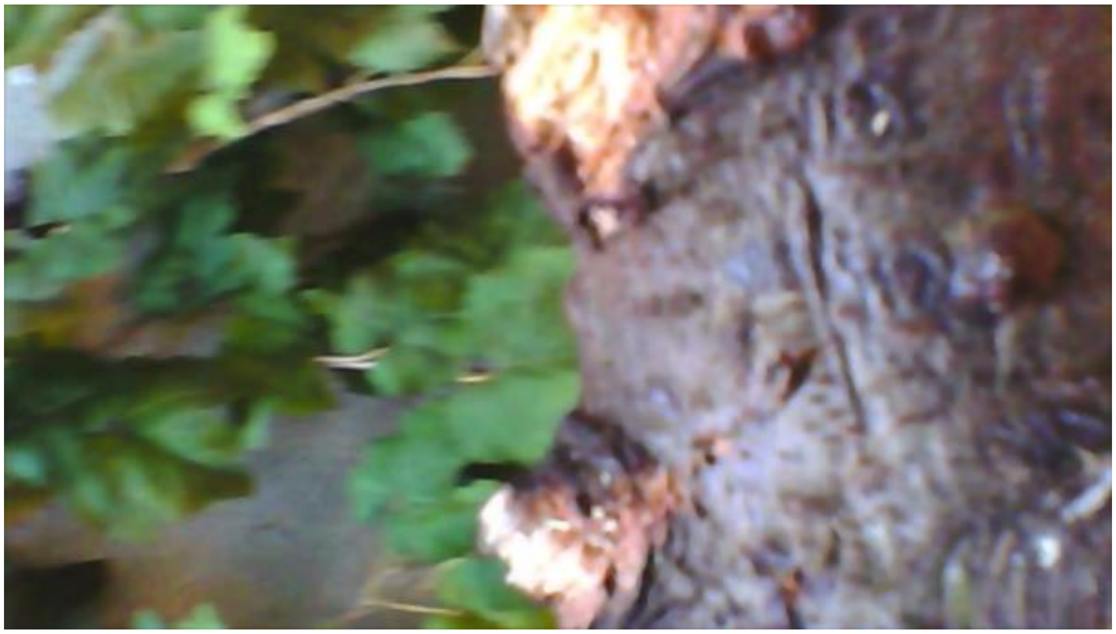

Figure 1: Sample frame from downward facing camera. For the remainder of this thesis, the picture coordinate system is defined with the positive u direction going from left to right and the v direction going from top to bottom. The origin of the coordinate system is in the top left corner of the image.

brightness patterns in an image." [13]. Optical flow provides an estimate of the relative motion between the perched UAV and a given point on the image by giving an indication of how objects in the video feed, including the perch, are moving.

The OpenCV libraries used provide two methods of calculating optical flow: sparse optical flow and dense optical flow. The sparse optical flow algorithm estimates the movement of features in an image, while the dense algorithm estimates the movement of a window of pixels around each pixel in the image. The latter method was chosen for a number of reasons. Because the optical flow will be used to estimate what pixels in the image make up the background and what pixels make up the perch, and the optical flow is a key part of determining this, it is important to have an optical flow estimate at each pixel. Also, once the segmentation of the image is completed, the dense optical flow method will give more points to average to estimate the optical flow of the perch.

The dense optical flow algorithm provided by the OpenCV libraries is based on the algorithm describe by Farnebäck [14]. This method involves taking a second-order polynomial expansion of two images and fitting a displacement between these two images. The OpenCV library provides a function that utilizes this algorithm [15].

While there are several parameters passed to the optical flow function, the most important of these parameters is the window size that corresponds to the size of the neighborhood discussed in [14]. This value determines the sharpness of the results, the maximum movement that can be detected by the algorithm, and the sensitivity of the algorithm to noise. The tradeoff with this parameter is that the larger it is, the less sensitive the algorithm is to noise and the faster the motion it can track, but the results are liable to be more blurry. The window size was tuned to be relatively large. This was decided because blurred results spread the perceived motion over more area and make the segmentation easier. The large window also allows the algorithm to be sensitive to higher velocity movements and less sensitive to noise in the camera's image. Figure 2 shows a visualization of the optical flow with a large window size of 40 pixels, while Figure 3 shows the same visualization for a window size of 10 pixels. The smaller window size has higher resolution, however it also has larger black patches (representing no movement) in the left half of the image, which in this example is the area that should register motion. As mentioned earlier, the larger window size also allows the system to register larger motions.

Another key aspect of this optical flow algorithm is that it operates on a single channel. A digital camera typically provides three channels of data: red, green, and blue (RGB). This RGB image is then converted to a hue, saturation, and value (HSV) image. From this HSV image, the hue channel is extracted and passed to the optical flow algorithm. The hue of an image is more robust to changes in lighting, as described by Finlayson and Schaefer, [16] and will provide improved results over using a single color channel or using a grayscale image.

In summary, this part of the computer vision algorithm applies a dense optical flow algorithm provided by the OpenCV library to the hue channel of an incoming video stream. This will provide an estimate of the relative motion between the camera, which is rigidly attached to the perched UAV, and every pixel in the image.

Figure 2 and Figure 3 are visualizations of the optical flow calculated on the image in Figure 1 and the previous image taken by the downward facing camera. In these images, the color of a pixel represents the direction of the optical flow (green/blue represent side to side motion) and the brightness of a pixel represents the magnitude of the optical flow. The background clearly has a significant amount of motion while the perch does not appear to have much motion at all. In the end, the motion associated with the perch itself will be considered to help determine if the UAV is slipping or not.

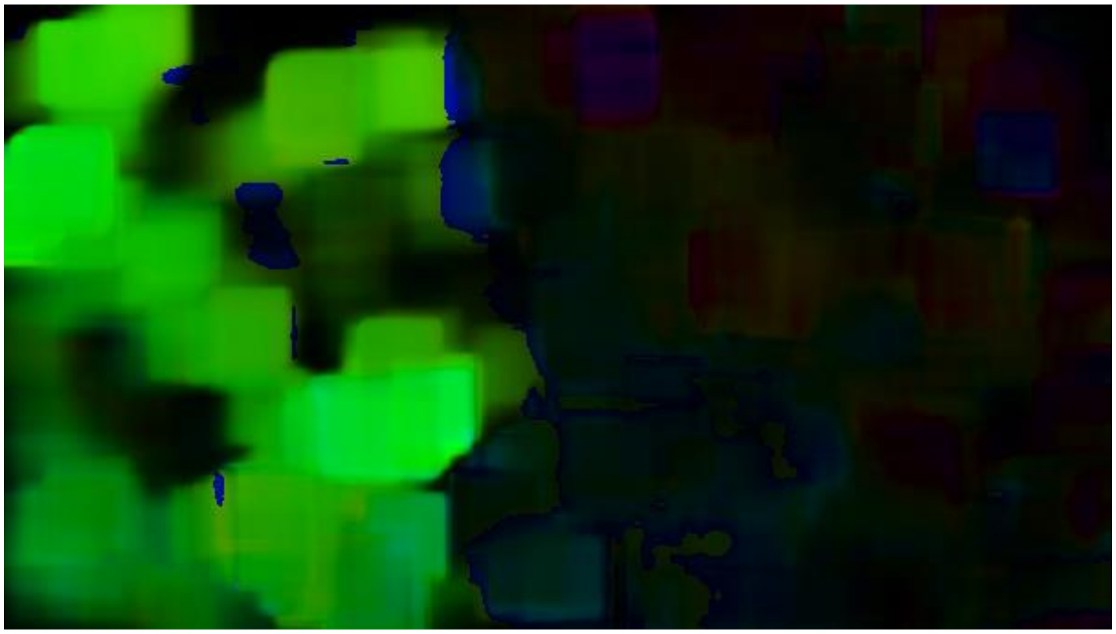

Figure 2: Visualization of optical flow with window size of 40 pixels.

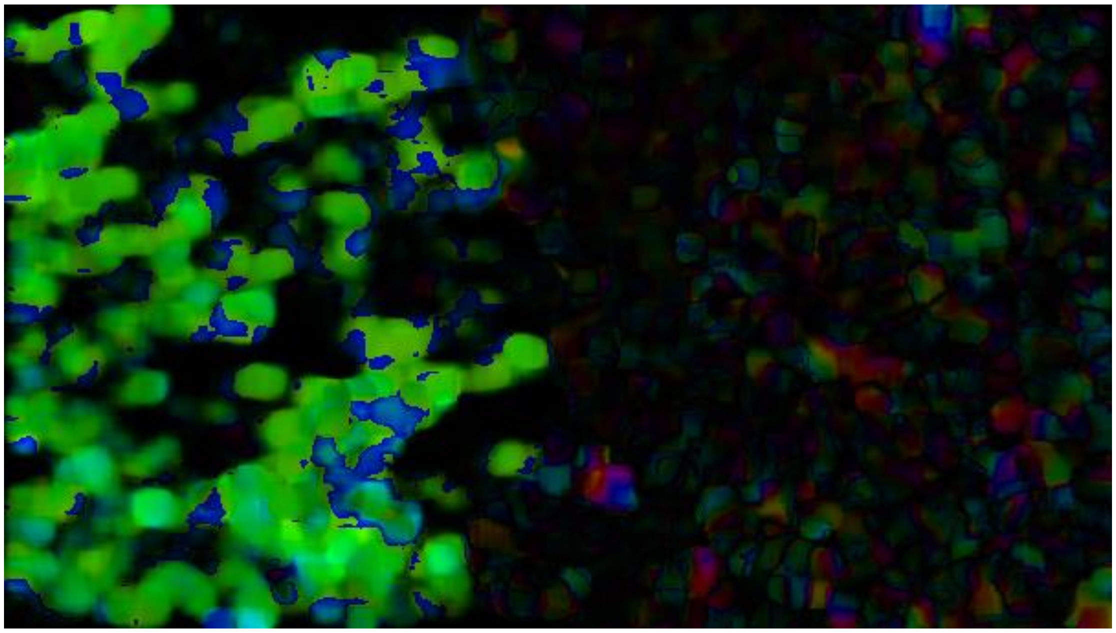

Figure 3: Visualization of optical flow with window size of 10 pixels.

Segmentation

The second major portion of this algorithm is the estimation of the segmentation of the perch from the background in the image. To do this, a K-means clustering algorithm is used on a feature vector defined in equation (1).

$$\boldsymbol{v}_v = \begin{bmatrix} \boldsymbol{u}' & \boldsymbol{v}' & \frac{1}{2}\boldsymbol{x} & \frac{1}{2}\boldsymbol{y} & \boldsymbol{r} & \boldsymbol{g} & \boldsymbol{b} \end{bmatrix} \tag{1}$$

This feature vector consists of three main components: the optical flow values, given by $\boldsymbol{u}'$ and $\boldsymbol{v}'$, the Cartesian coordinates in pixels of a pixel given by $\boldsymbol{x}$ and $\boldsymbol{y}$, and the RGB values of a blurred version of the latest frame from the camera. A value for this feature vector is generated for each pixel in the image.

Figure 1 is a frame taken from the downward facing camera while the UAV was perched on the branch visible in the right half of the image. As can be seen in this image, the perch dominates the right half of the image and does not split the image. The pixel positions are included in the feature vector to help generate two continuous areas that represent the perch and the background. The pixel positions are multiplied by 0.5 so that this effect does not dominate the clustering. This allows the edge between perch and the background to be more flexible and better fit the actual edge.

From Figure 1, it can also be seen that the line that separates the perch from the background is close to vertical. To aid in the clustering, the red, green, and blue values of a blurred version of the image are also used in the feature vector. The blur filter that is applied to the image blurs the image primarily in the vertical direction to help make the colors of the background and perch more uniform while limiting the amount of blur on the boundary between the two areas. This blur filter is a simple averaging filter as opposed to a Gaussian blur to reduce computation time.

After the feature vectors for each pixel are found, they are run through a K-Means clustering algorithm that will generate two clusters. One fault of the K-means algorithm provided by the OpenCV libraries is that it is not guaranteed which of these clusters is the perch and which is the background. As can be seen in Figure 1, the perch dominates the right half of the image and the background dominates the left. To determine which cluster is the perch, the centers of the two clusters are compared and the one with the greater value in the u direction is assumed to be the perch.

In summary, the segmentation process utilizes a K-Means clustering algorithm and a feature vector that consists of the optical flow, position, and color values of each pixel. The image is segmented into two parts, which are then categorized as either the perch or the background.

Filtering Segmented Image

While for most frames this method works well enough by itself to provide a decent estimation of the parts of the image that make up the perch and the parts that make up the background, there are typically clumps of frames that just do not segment well. To help with this and to help eliminate noise, a voting scheme is used that acts like a low-pass filter to reduce noise in the segmentation estimations.

This filter starts by creating a matrix with one entry for each pixel. The algorithm goes through each pixel, and if the pixel was estimated to be part of the perch and the number of votes for that pixel is less than some maximum number of votes, then a vote is cast for that pixel. If the pixel is estimated to be part of the background and the number of votes for that pixel is greater than 0, then a vote is taken away from that pixel. If the number of votes for a pixel is considered to be greater than some percentage of the maximum number of votes, then that pixel is considered to be part of the perch.

This filter effectively acts as a low-pass filter for each pixel. The two parameters that can be adjusted on the filter are the maximum number of votes and the threshold percentage of the maximum number of votes that are required for a pixel to be considered part of the perch. The maximum number of votes acts like a time constant for the filter. As the maximum votes increases, the distance into the past that filter looks increases. A higher max number of votes increases the lag induced by the filter, but results in a smoother filter overall. The threshold percentage can be adjusted to help increase the response time of the filter in one direction by telling the filter to give preference to one mode over the other. This can help offset some of the lag induced by a higher max number of votes, but it also adds hysteresis to the filter, effectively making it easier to convince the filter that a given pixel is part of one group than it to convince the filter that it is part of the other group.

Figure 4 shows the image from Figure 1 after it has been run through the optical flow algorithm, segmented, and filtered. In this image, the green area is the area estimated to be

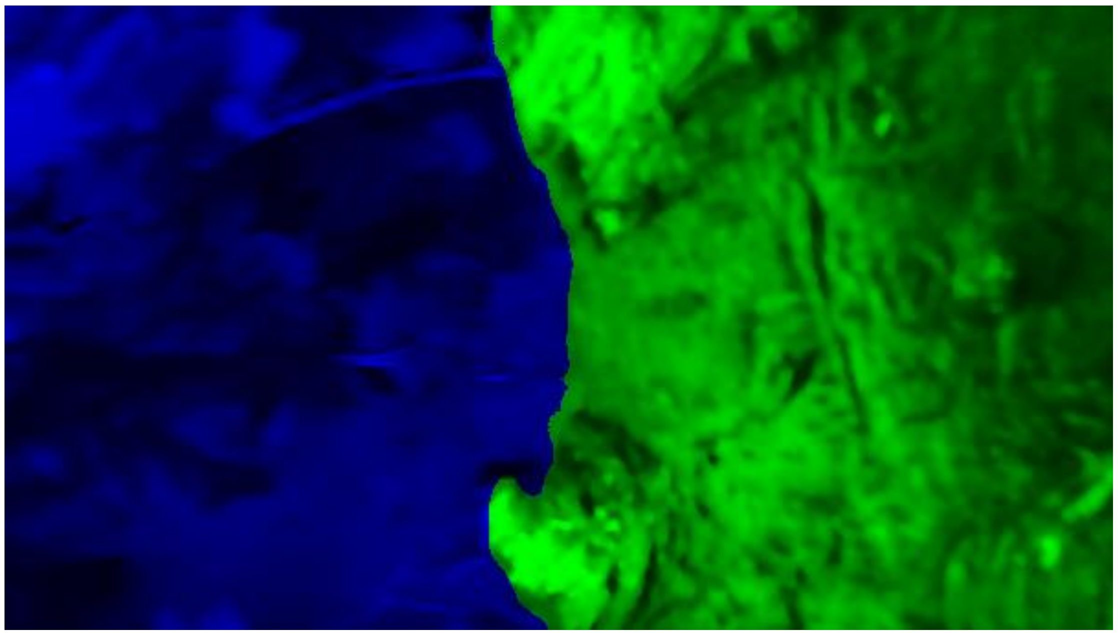

Figure 4: Segmented image after filtering.

part of the perch and the blue area is the area estimated to be part of the background. This figure was generated by taking the gray scale value of the image in Figure 1 and applying it to either the blue or green channel and zeroes to the other channels, depending on whether the filter identified the pixel as part of the perch or not.

Experimentation

The experimental setup used consisted of a test quadrotor equipped with the landing gear described by Doyle et al. [5], as well as a USB Creative Live! Webcam mounted to its belly and pointed at the perch. The camera was connected to an external laptop. The quadrotor was also outfitted with several markers for tracking with an external Vicon Motion Capture system. The quadrotor was also instrumented with a number of inertial sensors, however, the data from these sensors were only used for the fall detection methods described in the next chapter. The data from the camera, Vicon system, and inertial sensors were collected and recorded using the Robot Operating System (ROS). This allowed the data to be time stamped and synchronized. These data were later postprocessed with the algorithm described in this chapter.

A total of nine (9) tests were performed with five (5) trials per test. The nine (9) tests result from varying three (3) variables. The first is the type of motion the quadrotor undergoes. Three types of motion are considered: the quadrotor is pushed off of a static perch, the quadrotor is swinging on a swing with no slip, and the quadrotor is sitting on a swing while its angle is increased to the point where the quadrotor falls off. In all these tests the quadrotor is held still for the first ten seconds to get an initial estimate of the state and to let the filter initialize itself with the colors in the image. After this calibration period, the quadrotor undergoes its motion. These tests give data showing the quadrotor in a state where it is completely static, where it is moving independent of the perch, where it is moving with the perch, and where it falls off the perch, respectively. The second variable considered is the type of perch. Three perches are used: a PVC pipe painted white, a PVC pipe with a swirl patter painted onto it (like a barber shop pillar), and finally a tree branch. These perches will help us determine the effectiveness of this algorithm on various types of perches. We suspect that the feature-rich tree branch will give the best results, while the plain PVC pipe will give the worst.

Results

Figure 5 is a sample of the results of one set of trials of the optical flow method. The plotted values are the optical flow of the background in the u and v directions (blue and dark green) and the perch in the u and v directions (red and cyan). The u direction is from left to right and the v direction is from top to bottom. The independent axis of each plot is the time in seconds. The dependent axis is the average optical flow value of the area in pixels per frame.

Figure 6 is a plot of the optical flow of the perch in the u direction during the fall test on the branch. The three lines are when the optical flow filter was run with a maximum number of votes at 5, 10, and 20 for the blue, green, and red lines, respectively. The axes have the same units as Figure 5. However this is only a plot of 100 samples in the middle of the test. This range was chosen to outline how the maximum votes effects the results of the process.

Figure 7 shows the optical flow in the u direction during the fall test on the branch. The various lines again represent the algorithm run with different threshold values. It is apparent that the threshold does not really effect the results of the test. The few spots where there is a visible effect are at sharp peaks in motion where a lower threshold generally smooths the peak compared to higher threshold values.

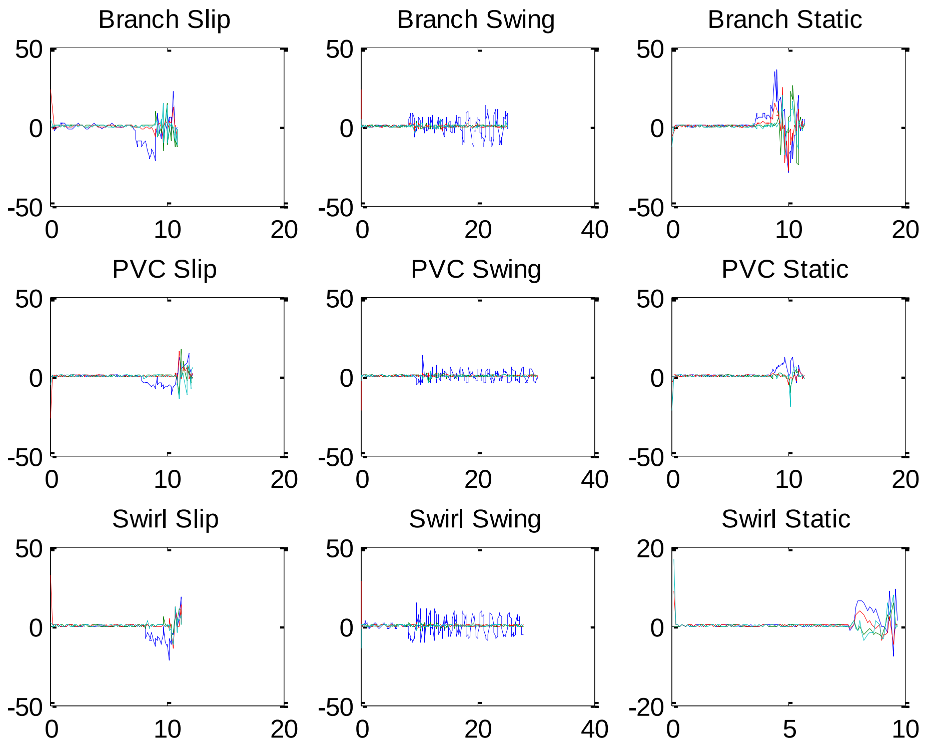

Figure 5: Sample results of optical flow process. In each of these plots, the independent axis is the time in seconds, and the dependent axis is the magnitude of the optical flow in pixels per second. In these plots, the blue and green lines are the optical flow of the background in the u and v directions, respectively, and the red and cyan lines are the optical flow of the perch in the u and v directions, respectively.

Analysis

Looking at the plots in Figure 5 and data from additional trials, it can be seen that the filter does do a good job at distinguishing the perch from the background. This can be seen particularly in the swing and slip examples, where the background can be seen to be moving while the perch stays relatively still. In these plots, the regular PVC pipe seems to be rejecting the noise the best of the three surfaces. However, when we look at the results of the static motion, the PVC pipe also shows little motion of the perch as well. This indicates that the vision

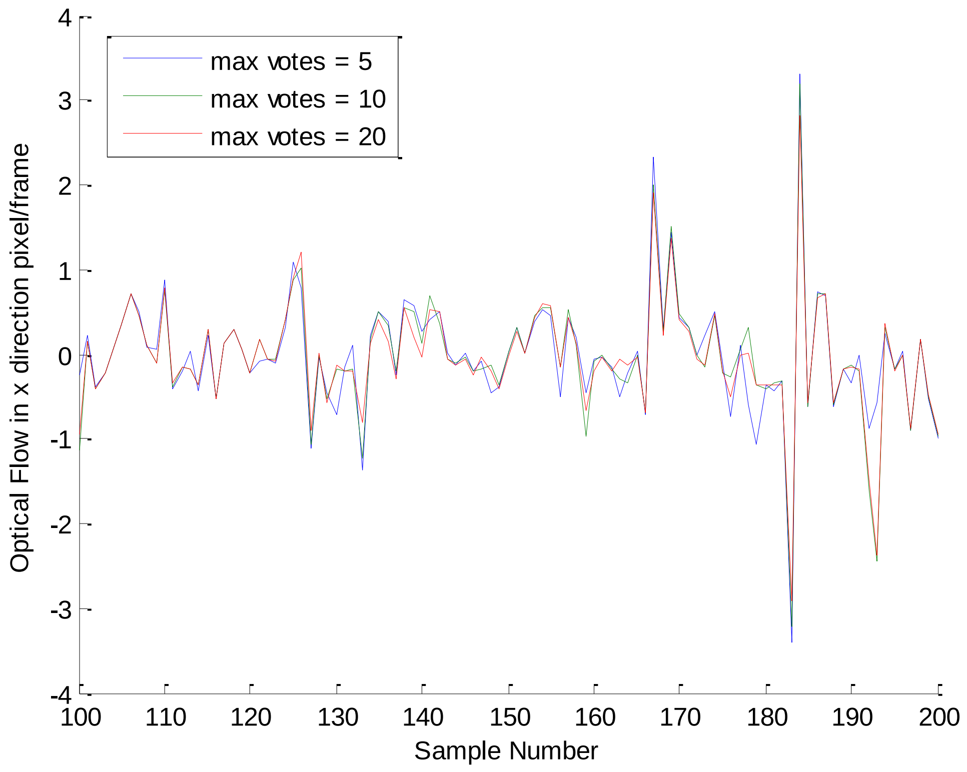

Figure 6: Comparison of optical flow process run with different maximum votes.

method cannot actually detect the motion in the pvc pipe painted white, which is to be expected as it has no features that can be tracked. The spikes in the slip test where the UAV falls actually take place slightly after the fall starts and the perch shifts its position in the frame of the video stream. The spike of motion is a result of the filter trying to catch up, because the perch has shifted in the frame, but the filter has not had time to adjust for this, some of the background is read as part of the perch and the optical flow is calculated as perch motion. This effect is in all of the tests, but it is more pronounced in the PVC perch. However, this does mean that even if the motion of the perch itself is difficult to track, the motion of the background and the lag in the filter can also work together to produce a signal that indicates a fall.

Looking at the results from the perch with the swirl pattern, there is more motion in the v direction than the other perches. This is an artifact of the pattern on the perch. As the perch

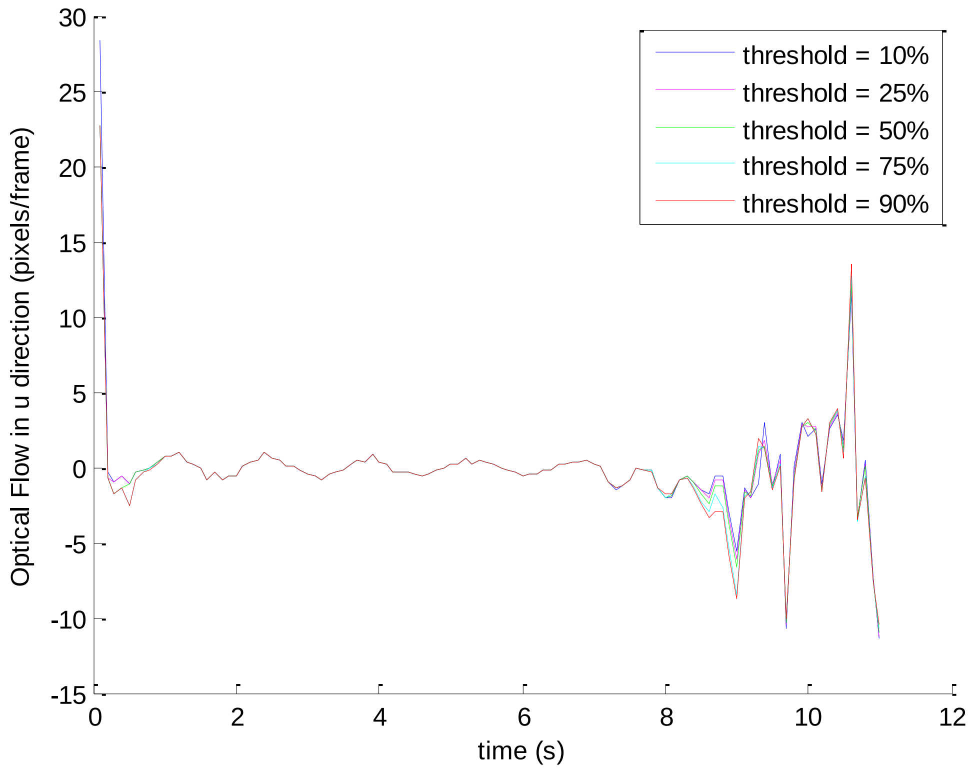

Figure 7: Plot of optical flow in the u direction during branch fall test with varying threshold values.

rotates, the pattern appears to move in the v direction as opposed to the u direction seen in the other perch types. This can still be used to detect a fall. If the robot can detect this swirl pattern on the perch (perhaps if perched on some kind of cable), then it should consider the optical flow in the v direction as well.

The last type of perch is the tree branch. This perch behaved much more in the manner that would be expected. A tree branch is rich with features that can be tracked, but they are not arranged in a pattern that warps the perception of the movement like the spiral. As a result, the optical flow gave much more accurate data about the motion of the perch relative to the robot. The relative motion is seen in the slip and swing trials, where the motion of the perch remains small until a slip occurs. While there is more motion than in the painted PVC perch, this is because there is still some small amount of relative motion due to various effects. The behavior seen in the static test is also expected. The motion of the background and the perch is similar, however, the magnitude of the motion of the perch is slightly less. This is expected due to the fact that the perch is not moving relative to the background. This difference is a result of the perch being closer to the camera than the background.

Figure 6 illustrates how the maximum number of votes allowed in the filter affects the results. As can be seen in the plot, increasing the number of votes is somewhat like lowering the cutoff frequency of a low-pass filter. However, this analogy is not entirely true. Since this is only changing the areas of the image considered for calculating the optical flow of the perch, sudden changes in the motion are still detected no matter what the maximum number of votes is, as can be seen in the spike near frame 183. Where this does play a role is in how noise from the perch changing position in the frame, as described earlier, effects the overall signal. In this case, a slower responding filter can add to this, however, this can be a good thing indicating a fall in a situation where the motion of the perch itself is difficult to discern. A fast-changing filter can limit this, but it will still exist. The noise will appear, then the filter will correct itself, causing the signal to be more jumpy.

Figure 7 illustrates how the threshold affects the output of the amplified algorithm. Sharp peaks tend to be smoothed by low thresholds and by high thresholds, however, this effect is not significant enough to be useful in classifying a fall, and this only affects some parts of the signal. It does not affect the overall result of the algorithm in a meaningful way. It is also worth noting that the effect is in the magnitude of the response, but does not create or eliminate any lag in the signal. The final conclusion is that the threshold does not affect the results in the experiments we performed. We used a threshold of 50% for the rest of this thesis, but any number between 10% and 90% should work with minimal differences in the results. 50% was chosen because it is the middle of this range and indicates a majority vote.

Overall, the method is able to detect falls from the perch and distinguish this from the UAV just moving with the perch. It is able to do this on multiple types of perches, even perches where the motion of the perch relative to the UAV is difficult to directly detect. This can be done by exploiting the lag in the filter used.

Advantages and Limitations

The main advantages of the proposed vision method include the fact that it requires limited additional hardware—only a camera and a computer to process the vision data. Webcams are fairly inexpensive and easily acquired, and they generally do not require a lot of setup time and effort.

The use of OpenCV as a tool for performing various computer vision tasks is also a huge advantage. OpenCV's algorithms have been written and optimized independently and are available for free online as an open source package.

Another advantage of this method is based in the fact that it uses vision. Vision methods give position data as opposed to accelerometers or gyroscopes, which give acceleration and velocity data, respectively. Because of this, accelerometers and gyroscopes must be integrated to get position data or velocity data in the accelerometer's case. Offsets in the sensor readings can lead to drift in the final estimates. However, with vision systems, since the data are position data from the start, this drift does not occur. The downside to this, however, is that to get velocity data as proposed, a derivative is required, provided in this case by the optical flow algorithm, and this can result in noisy signals.

While there are many advantages to this method, it is not without its limitations. One major limitation of this algorithm is processing power and time. Because there is a good amount of image processing, a computer more powerful than the average microcontroller is needed. For tests done in this chapter, the data were post processed on an external computer. However, for future research, plans are in place to utilize an Odroid single board computer from Hardkernel. This ARM-based computer running an Xubuntu operating system will have the processing power to run these algorithms, while being small enough to be mounted on the airframe of a quadrotor.

While this algorithm can be run on an onboard computer, it still takes a significant amount of time to execute. Current tests put it at around 1 Hz on the Odroid single board computer used on the test quadrotor. Another issue from the timing is the time between frames. With up to a whole second between frames, it is possible for there to be significant changes between the two frames. This could result in some instability in the optical flow algorithm. To counter this, two frames are collected, one right after another, and the optical flow is run on these two frames. While this will lead to better estimates of the optical flow, this still leaves the system updating once a second, which is not fast enough to catch a fall in time to correct it. Faster processing and a more streamlined algorithm may be necessary to accomplish this.

Another limitation of this algorithm is the kinds of perches that it will work with. The tests were performed on a tree branch which has a random texture with many distinguishable and unique features conducive to the optical flow. Other perches tested, such as a painted PVC pipe, do not have enough detail for the optical flow to detect motion. However, exploiting the lag in the image filter and the noise that the background can inject as a result can be used to find falls in this case. Another interesting case is where the texture of the perch is in a pattern that appears to move differently than the actual movement of the perch. An example of this would be spiral pattern (common on cables due to the winding used in their manufacturing).

This pattern appears to move perpendicular to the actual motion of the surface when the perch is rotated. If this phenomenon is known a priori, then it can be exploited in the fall detector.

The method proposed here works in principle, but it has some issues when it comes to real-time implementation. The method requires more processing power than currently available on the platform used in the experiments, however, it is not infeasible to have the required processing power onboard.

CHAPTER 3

INSTANTANEOUS CENTER OF ROTATION APPROACH

The second proposed approach for determining if a UAV is falling from its perch involves estimating the instantaneous center of rotation (ICR) of the UAV's rigid body frame and comparing it with known factors about the motion of the perch it is sitting on. The idea is that these comparisons will be indicative of whether or not the UAV is, in fact, falling.

In this chapter, the ICR will be discussed in more detail, then how it can be used to detect falling will also be discussed. Later, the matter of calculating the ICR using onboard inertial sensors will be discussed. Finally, the advantages and flaws of this approach will be examined.

The material presented in this chapter is from a paper presented at the IEEE International Conference on Robotics and Automation (ICRA) in Seattle, Washington, United States, on May 26-30, 2015 [17].

Instantaneous Center of Rotation

In this section, the nature of the ICR of a rigid body and its physical meaning will be discussed. Then how this data can be used to determine if a rigid body is falling or not will be explored.

What is an instantaneous center of rotation?

Any motion of a rigid body can be defined as a rotation about a point. A pure rotation is simply a rotation about the constant point, while a pure translation is a rotation around a point at an infinite distance from the object.

This point can be determined a number of ways. The most common method involves knowledge of the velocity at two points on a rigid body. Two lines are drawn perpendicular to these velocities, and the point where these lines intersect is the ICR. However, in our implementation, velocity cannot be directly measured. A second method, which will be utilized later in this chapter for one of the proposed ICR location algorithms, requires knowledge of the velocity at a point and the angular rate of the rigid body. In this case, a line is again drawn perpendicular to the velocity vector. A circle is then drawn centered on the point with a radius of $\frac{1}{\omega}$ where ω is the angular rate of the rigid body. Where the circle intersects the line provides two possibilities for the ICR. The sign of the angular rate determines which point is the actual ICR. If the sign of the angular rate is positive, the ICR is the point pointed to by the velocity vector when rotated 90 degrees about the axis of rotation the angular velocity is defined about If the angular rate is negative, the ICR is the point pointed to by the velocity vector when rotated -90 degrees. A third technique, which is utilized by the second method of finding the ICR, requires sensor estimates of the distance from three points on the rigid body to the ICR. This results in three circles around the three points on the rigid body. The ICR will be the intersection of these three circles.

Using ICR to Detect Falling

The motion of the UAV is the superposition of two different motions. The first is the motion of the perch. Because the UAV is attached to the perch, the motion of the two bodies is coupled. The second component of the motion is the motion due to any slip between the UAV and the perch. If there is not motion from the second component, then the ICR of the UAV and the perch will be the same. If the ICRs are different, then there is significant relative motion between the UAV and perch.

Two major assumptions are made for the methods proposed in this chapter. The first is that there is a priori knowledge of the motion of the perch. Given that the perch being considered is a swinging perch like a power line, the ICR is a constant point. With a perch like a tree branch, the majority of the motion is translational, as the perch itself does not rotate a significant amount. This method would also work with a static perch because the motion (or rather lack thereof) is known. In both of these cases, if the ICR is near the UAV, slip can be assumed, as that indicates the UAV is undergoing motion that is different from the motion of the perch.

The second major assumption made for these methods is that the motion is constrained to two dimensions. While the methods could be extrapolated to three dimensions, we are constraining the motion to a plane perpendicular to an axis along the perch. The perching mechanism we are considering largely constrains the actual motion of the UAV to a plane through its natural mechanics, so this constraint is assumed to simplify the equations for analysis.

Another point to consider is the reference frame the ICRs are measured in. They can either be reported in local coordinates of the UAV or global coordinates. The ICR of the perch would generally be known in global coordinates, while the ICR detected by onboard sensors, as proposed later in this chapter, would more likely be in local coordinates. As long as these two ICRs can be transformed to be in the same coordinate frame, they can still be compared.

Finding Instantaneous Center of Rotation Using Onboard Sensors

The next logical question is how to detect the instantaneous center of rotation of a UAV. This is relatively simple using external sensors, such as a Vicon motion capture system. Markers could be placed on the rigid frame of the UAV and the sensors could measure their linear velocity, thus the first method described in the previous section on calculating the ICR can be used. Unfortunately, this approach's usefulness is limited by the fact that it is only valid when these external sensors can see the UAV. A more useful method would make use of onboard sensors such as accelerometers and gyroscopes mounted onboard the vehicle. We propose two methods for doing this and compare them to the external sensor method. The first method, known as the Integration Method, involves integrating the accelerometer to get a linear velocity estimate and using this estimate along with the angular velocity of the rigid body to estimate its ICR. The second, known as the Radius Method, uses accelerometer data to estimate the distance between the said point and the ICR at three different locations. This results in the ICR being estimated as the intersection of three circles centered on the accelerometers. Both methods use an EKF to estimate the angular position and velocity of the UAV on the perch. This section will go over the design of the EKF and then discuss both methods in detail.

Extended Kalman Filter

Both the integration method and the radius method will use estimations of the angular velocity and position to locate the ICR. To come up with these estimates, an Extended Kalman Filter (EKF) is used. This filter uses the difference between the accelerations at two points on a rigid body to estimate the angular acceleration and the angular velocity squared. These, along with the readings from the gyroscope, are fed into the EKF, which estimates the angular velocity and position based on those data.

Figure 8 shows how the sensor positions relate to each other and the world reference frame, O_O. The two accelerometers are located at points O_a and O_b, while the IMU is located at point O_c. The vectors $\boldsymbol{r}_{ca}$ and $\boldsymbol{r}_{cb}$ are constant within reference frame c, as this is the rigid frame of the UAV. The world reference, O_O, is an arbitrary frame with no specific definition. These definitions will be used throughout this chapter.

The following equations show what the accelerometers and gyroscopes would read.

$$A = \begin{bmatrix} -r_{cax} & -r_{cay} \\ -r_{cay} & r_{cax} \end{bmatrix} \quad (2)$$

$$B = \begin{bmatrix} -r_{cbx} & -r_{cby} \\ -r_{cby} & r_{cbx} \end{bmatrix} \quad (3)$$

The matrices in equations (2) and (3) are made of the components of the vectors $\boldsymbol{r}_{ca}$ and $\boldsymbol{r}_{cb}$ in coordinate frame c. These matrices are defined to simplify the following equations.

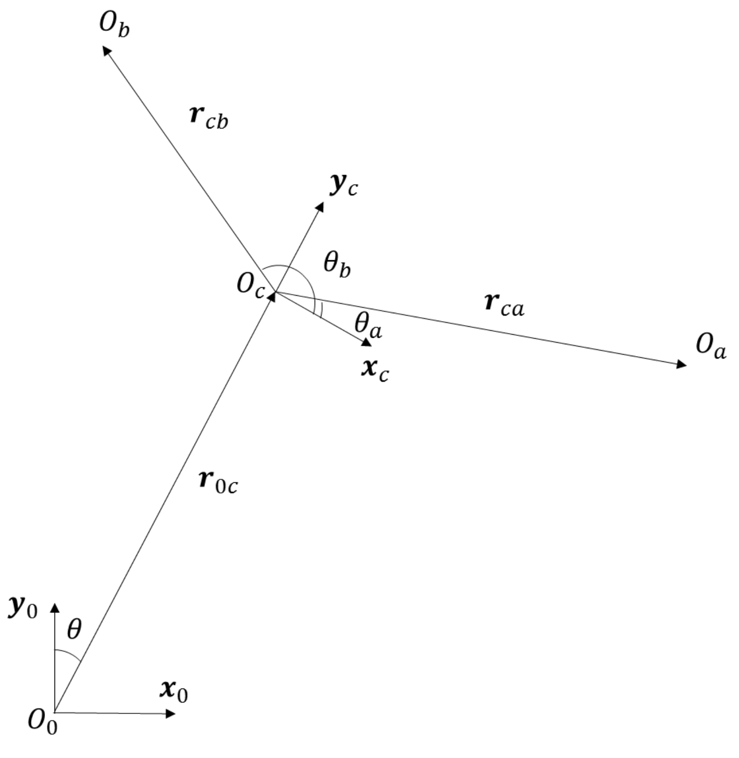

Figure 8: Sensor position and vector definitions.

$$\boldsymbol{a}_a = g\begin{bmatrix} sin(\theta) \\ cos(\theta) \end{bmatrix} + \ddot{\boldsymbol{r}}_{0c} + A\begin{bmatrix} \omega^2 \\ -\alpha \end{bmatrix} \quad (4)$$

$$\boldsymbol{a}_b = g\begin{bmatrix} sin(\theta) \\ cos(\theta) \end{bmatrix} + \ddot{\boldsymbol{r}}_{0c} + B\begin{bmatrix} \omega^2 \\ -\alpha \end{bmatrix} \quad (5)$$

$$\boldsymbol{a}_c = g\begin{bmatrix} sin(\theta) \\ cos(\theta) \end{bmatrix} + \ddot{\boldsymbol{r}}_{0c} \quad (6)$$

$$g_z = \omega \quad (7)$$

$\boldsymbol{a}_a$, $\boldsymbol{a}_b$, and $\boldsymbol{a}_c$ are readings of the accelerometers at points O_a, O_b, and O_c, respectively, measured in the local coordinate frame. $\boldsymbol{r}_{ca}$ and $\boldsymbol{r}_{cb}$ are the vectors shown in Figure 8 pointing from O_c to O_a and O_b, respectively, within the local frame. g_z is the measurement from the gyroscope at point O_c in the z direction. Lastly, g is the magnitude of the acceleration of gravity, for the purposes of this paper, it is assumed to be 9.81m/s^2. Note that there is not a negative sign missing in (4-6). Due to the way the accelerometer works, gravity is read as acceleration in the opposite direction to what it actually is. All four of these values are in the c reference frame. The angular position is denoted by θ.

$$\begin{bmatrix} {\omega_1}^2 \\ \alpha_1 \end{bmatrix} = (A - B)^{-1}(\boldsymbol{a}_a - \boldsymbol{a}_b) \quad (8)$$

$$\begin{bmatrix} {\omega_2}^2 \\ \alpha_2 \end{bmatrix} = A^{-1}(\boldsymbol{a}_a - \boldsymbol{a}_c) \quad (9)$$

$$\begin{bmatrix} {\omega_3}^2 \\ \alpha_3 \end{bmatrix} = B^{-1}(\boldsymbol{a}_b - \boldsymbol{a}_c) \quad (10)$$

Equations (8-10) illustrate the kinematic model for using the accelerometer and gyroscope readings to estimate the angular acceleration and velocity. In these equations, the angular acceleration is denoted by α_a and the angular rate is denoted by ω_a. Since comparing three accelerometer readings provides three separate estimates for these values, the subscript a=1,2,3 is used to differentiate between these estimates.

$$\boldsymbol{x} = \begin{bmatrix} \omega \\ \theta \end{bmatrix} \quad (11)$$

$$\boldsymbol{u} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \tag{12}$$

$$\boldsymbol{z} = \begin{bmatrix} g_z \\ {\omega_1}^2 \\ {\omega_2}^2 \\ {\omega_3}^2 \end{bmatrix} \tag{13}$$

$$\boldsymbol{x}_k = \begin{bmatrix} 1 & 0 \\ \Delta t & 1 \end{bmatrix} \boldsymbol{x}_{k-1} + \begin{bmatrix} \frac{\Delta t}{3} & \frac{\Delta t}{3} & \frac{\Delta t}{3} \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{u}_{k-1} \tag{14}$$

$$\boldsymbol{z}_k = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} 0 \\ \omega^2 \\ \omega^2 \\ \omega^2 \end{bmatrix} \tag{15}$$

Equations (11-15) describe the kinematic model derived from (7-10). The vector $\boldsymbol{x}$ is the state vector for the model, consisting of the angular rate and the angular position. The input to the model is the vector $\boldsymbol{u}$, which consists of the estimates of the angular acceleration from (8-10). The vector $\boldsymbol{z}$ is the sensor output vector, consisting of the estimates of the angular rate from the gyroscope as well as the angular rate squared, also from (8-10). An EKF is then generated using this model and is subsequently used to estimate the angular rate and acceleration. The outputs of this EKF are used by both the Integration and Radius methods for finding the ICR.

The noise estimates for the EKF are based on estimates for the noise in the sensors. The sensor noise is applied to the model and the resulting covariance matrices are calculated for the process and sensor noise.

Integral Method

The Integration Method of finding the ICR of the perched UAV relies on integrating the signals from the accelerometers shown in (4-6). The major problem posed in doing this is the fact that these sensors have a gravity component in their readings. To compensate for this, these signals are run through a digital high-pass filter.

After the accelerometer is filtered, the new, filtered signal is integrated to obtain estimates of the linear velocity of the rigid body of the UAV at the three points to which the accelerometers are mounted. This, coupled with the estimates of the angular velocity from the EKF, gives us three estimates of the ICR based on (16).

$$\boldsymbol{r}_{icr} = \frac{1}{\omega}\begin{bmatrix} -v_y \\ v_x \end{bmatrix} \tag{16}$$

These three estimates are then averaged to get a final estimate of the location of the ICR. This estimate is then compared to the expected value of the ICR based on the mode of the perch motion. If the perch is static, then there should not be an ICR, and any movement would be indicative of a slip. In the case of a swinging perch, it is considered whether the ICR is located above the UAV, indicating that the UAV is just moving with its perch, or if it is located underneath the UAV, indicating that the UAV is rotating about the perch that is below it, thus slipping. Another condition that could be tested is in the case of a translating perch. In this case, the ICR should be at very large distances from the UAV if there is no slip, because the UAV would be translating with the perch, and not rotating.

Radius Method

The second method for finding the location of the ICR on the UAV is the Radius method. This method does not rely on integrating the accelerometer signal, thus there is no risk of integrator windup from bias due to the gravity term. The gravity term, however, does still need to be eliminated.

This method is based on the idea that the acceleration at a point on a rigid body can be described by (17), where r_{icr} is the distance to the ICR, α is the angular acceleration, ω is the angular rate, $\boldsymbol{e}_n$ is the unit vector pointing toward the ICR, and $\boldsymbol{e}_t$ is the vector tangent to the motion at a given time.

$$\boldsymbol{a} = r_{icr}\omega^2\boldsymbol{e}_n + r_{icr}\alpha\boldsymbol{e}_t \qquad (17)$$

From this equation, it can be shown that the distance from a point to the ICR is proportional to the magnitude of the linear acceleration by way of the angular rate and acceleration. The proportionality term can be found with the results from the EKF discussed above, and is denoted as p in (18).

$$p = \alpha^2 + \omega^4 \qquad (18)$$

This proportionality term is used with the value reported by the accelerometer minus the acceleration of gravity, based on the estimates of the angular position by the EKF, to calculate the distance from points O_a, O_b, and O_c to the ICR. Once these three radii are obtained, it becomes a problem of finding the common point of three nonconcentric circles. See Figure 9 for an illustration of these circles.

Because of noise and other factors, it is unlikely that the circles will intersect at exactly one point. To estimate the ICR, first we generate a set of the six points where the three circles intersect. We then locate the set of three points within this set that are closest together. These points are averaged to find the ICR estimate. To find these points, first, the set of six points is found by locating the intersection points of the three circles with each other. Figure 9 shows these circles and illustrates the points we are looking for. To find these points, we compare the set of points from the intersections of circles about O_a and O_b and the intersections of the circles about O_a and O_c. We find the closest set of 2 points consisting of one point from each set of 2 points. Lastly, we find the point closest to these two points from the set of intersecting points of the circles centered about O_b and O_c. Once these three points are found, they are averaged to estimate of the location of the ICR. It is important to note that this method creates one of the few constraints on sensor placement in that the sensors positions cannot be

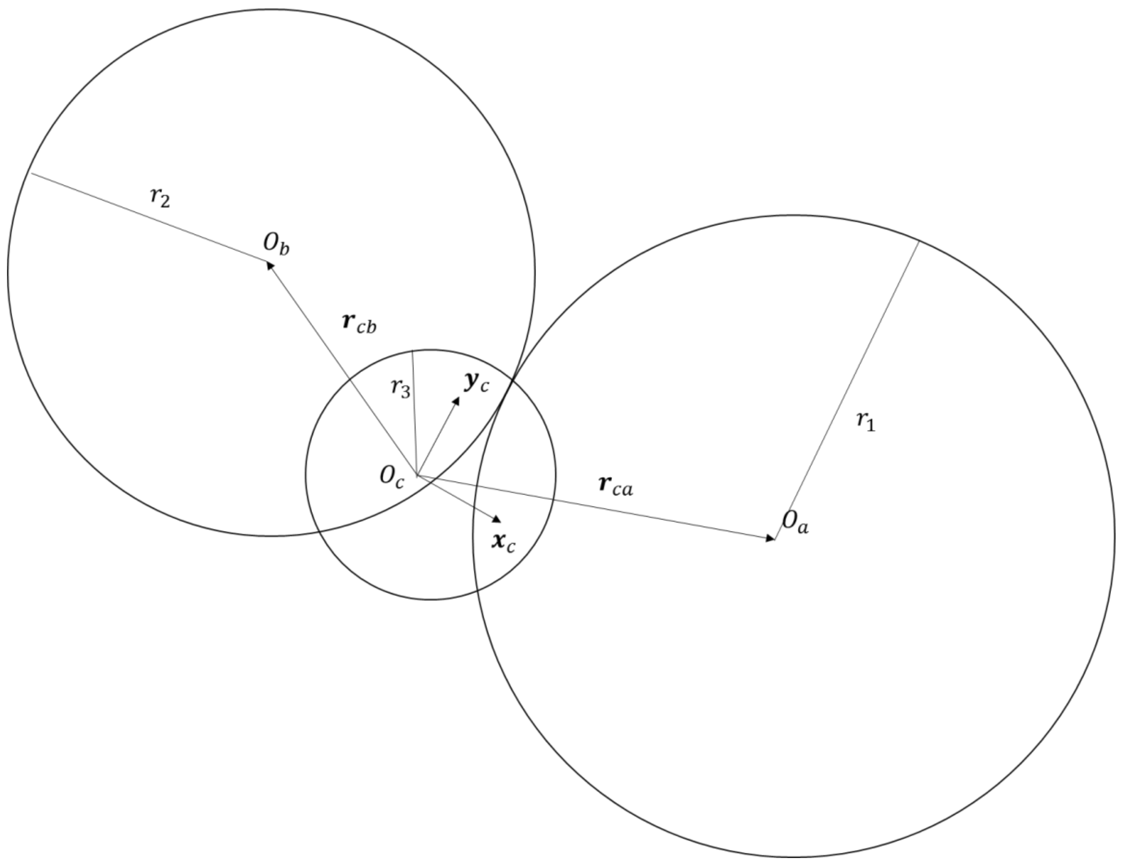

Figure 9: Radii from radius method.

collinear. If this is the case, then there will always be two possible solutions to the circle intersections. When noise is taken into account, this will cause the estimate of the ICR to jump across the line that the sensors are mounted on.

Once the ICR has been located, it is compared to the expected ICR from knowledge of the motion of the perch in the same way the ICR calculated from the Integration Method is.

External Sensor Method

The final method of finding the ICR that will be addressed in this paper is using external sensors. In this case, a Vicon motion capture system is used to find the ICR. The Vicon system gives a position vector and rotation quaternion for a rigid body based on a series of at least three markers placed on the quadrotor. Because we are considering this as a two-dimensional problem, this position and rotation are projected onto the plane where the motion takes place. By taking the derivative of these values, we then get a velocity and angular rate. Using these values and equation (16), the ICR can be found. The ICR is then translated and rotated from a global frame to the local frame of the UAV so it can be compared to the results of the other methods. This method is proposed as a baseline to compare the other methods to. It will provide more accurate results, but is cumbersome and ill-suited for general outdoor application.

Experimentation and Validation

To show that one can indeed be used to detect a fall from a moving perch, a simulation was run before any experiments to validate this theory. This simulation looked at how the ICR changes when the UAV slips from its perch. The perched UAV was modeled as a double pendulum. The first link is a model of the swinging perch, the second link a model of the UAV perched on the swing. The simulation set the angles of each joint as a sinusoidal function with different magnitudes and frequencies.

After this simulation, several different experiments were used for verifying the methods described in the previous section. Both inertial Sensor methods utilize an 18-inch-long balsa wood rod with two 3-axis accelerometers and an IMU mounted to it. This rod has been dubbed the “Divining Stick.” The accelerometers used are Freescale Semiconductor MMA8452Q 3-axis accelerometers and the IMU used is an InvenSense MPU-6050 6-degree-of-freedom IMU. This IMU contains a 3-axis accelerometer and a 3-axis gyroscope. The MMA8452Q chips are set up with a sensitivity of 2g with 12-bit resolution, the MPU-6050 accelerometer is set up with a sensitivity of 4g and the gyroscope is set up with a resolution of 250°/s. Both the gyroscope and the accelerometer on the MPU-6050 have 16-bit resolution. These values were chosen based on the magnitude of the acceleration that each sensor is expected to undergo and the available values for each sensor. These values maximize the resolution of each sensor.

Before any of the discussed experiments take place, a two-step calibration is applied to the sensors. The first step of the calibration is to calculate the intrinsic bias and gains for each accelerometer. The method used is the non-iterative method presented by Grip and Sabourova [18]. The second step in the calibration is a linear fit to rotate the accelerometer frames at O_a and O_b into the c reference frame. This fit uses the following governing equation,

$$R\boldsymbol{g}_a = \boldsymbol{g}_c, \tag{19}$$

where $\boldsymbol{g}_a$ is the measurement of gravity from accelerometer a with the divining stick at an arbitrary orientation. $\boldsymbol{g}_c$ is the same thing but for accelerometer c with the divining stick at the same orientation. The rotation matrix R is the rotation between these two coordinate frames; it is broken into its rows and the equation is rewritten as follows:

$$R = \begin{bmatrix} \boldsymbol{r}_1^T \\ \boldsymbol{r}_2^T \\ \boldsymbol{r}_3^T \end{bmatrix} \tag{20}$$

$$\begin{bmatrix} \boldsymbol{g}_a^T & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{g}_a^T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{g}_a^T \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \boldsymbol{r}_3 \end{bmatrix} = \boldsymbol{g}_c \tag{21}$$

This equation can be expanded to take into account multiple data samples at different orientations.

$$\begin{bmatrix} \boldsymbol{g}_a^T & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{g}_a^T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{g}_a^T \end{bmatrix} = G \tag{22}$$

$$\begin{bmatrix} G_1 \\ \vdots \\ G_n \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \boldsymbol{r}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}_{c1} \\ \vdots \\ \boldsymbol{g}_{cn} \end{bmatrix} \tag{23}$$

In this case, the matrix G_i is the matrix G generated from the readings from accelerometer a in the i'th orientation. The vector $\boldsymbol{g}_{ci}$ is the reading from accelerometer c at the i'th orientation. At this point, a pseudo-inverse analysis is used to solve for the rotation matrix. This is done with both accelerometers a and b. The data used to perform the calibration consists of six data samples taken with the Divining Stick held with the gravity vector pointing along the positive and negative directions of each of the primary axes. It is important to note that these calibration methods do not require precise orientations for calibration, only that they result in nonsingular matrices. For each of these samples, the Diving Stick was stationary, and all the outputs of the inertial sensors are averaged over the length of each sample to get a mean value, as well as an estimate of the noise induced by each sensor for use with the Kalman Filter. These values are then used in both calibration steps.

The noise in the accelerometers and the gyroscope are calculated by finding the covariance matrices of the measurements taken at the beginning of the experiment while the UAV is stationary. These estimates are then used to calculate the process and sensor noise for the EKF.

Experimental Procedure

The experiments were performed with two controlled variables: the type of motion and the type of perch. Three different motions were used: a stationary perch with slipping, a swinging perch with no slipping, and a swinging perch with slipping. The static perch consisted of rotating the UAV about the perch without moving the perch. The expected results of this motion would be an ICR located at the feet of the perch. The swinging perch with no slip experiment involved placing the UAV on a swing and allowing it to swing for a minute with no slip between the UAV and the perch. For this test, the expected result is an ICR located above the UAV. The final test is the swing with slip. For this test, the UAV was placed on a perch and the perch was rotated until the UAV fell off. The expected result for the final test is for the ICR to be located above the UAV until the slip occurs, in which case it should move.

The second variable considered is type of perch. Three types of perches were used: a plain, painted PVC perch, a PVC perch with a spiral pattern painted onto it, and a tree branch. These three perches were more for the benefit of the vision method proposed in the previous chapter, however, they also demonstrate that the results of these methods are independent of the perch itself.

Figure 10 is two images of the experimental setup. The first image is from above and shows the quadrotor equipped with the divining stick and an Arduino for gathering data. The second image is a side view of the setup showing the quadrotor on its perch. The perch shown here is the swing used in the swinging tests with and without slip. The perch used in the images is the plain painted PVC pipe.

There was a total of 9 independent experiments, where each experiment was executed 5 times. The data from the inertia sensors, the onboard camera, and the Vicon system were all recorded using the Robot Operating System (ROS). This means that all the data are timestamped and can be synced later. The data from the inertia sensors and Vicon system were then run through the process proposed in this chapter and the results compared. The video

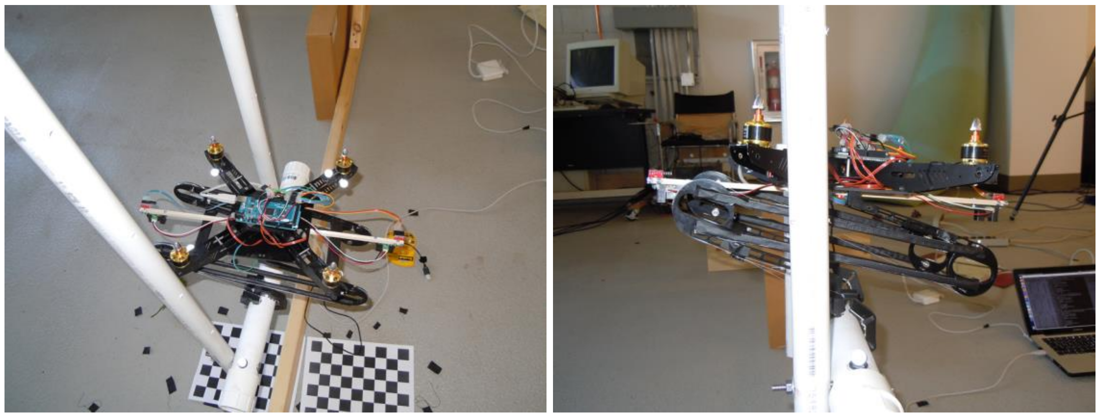
Figure 10: Photos of the experimental setup.

from the onboard camera was processed by the vision system proposed in chapter 2.

Experimental Results

The simulation results are fairly straightforward. When the perch link was held steady and the quadrotor was allowed to rotate about a stationary perch, the ICR holds steady at that location. When the quadrotor link is held steady and the perch is made to swing, the ICR is steadily at the point the swing is moving about. When the two joints are rotated together however, it was found the ICR moved along the line between the two points of rotation, as shown in Figure 11. These results show that the ICR can in fact be used to estimate a fall in a UAV. When the UAV is only moving with a perch, its ICR is the same as the perch, however,

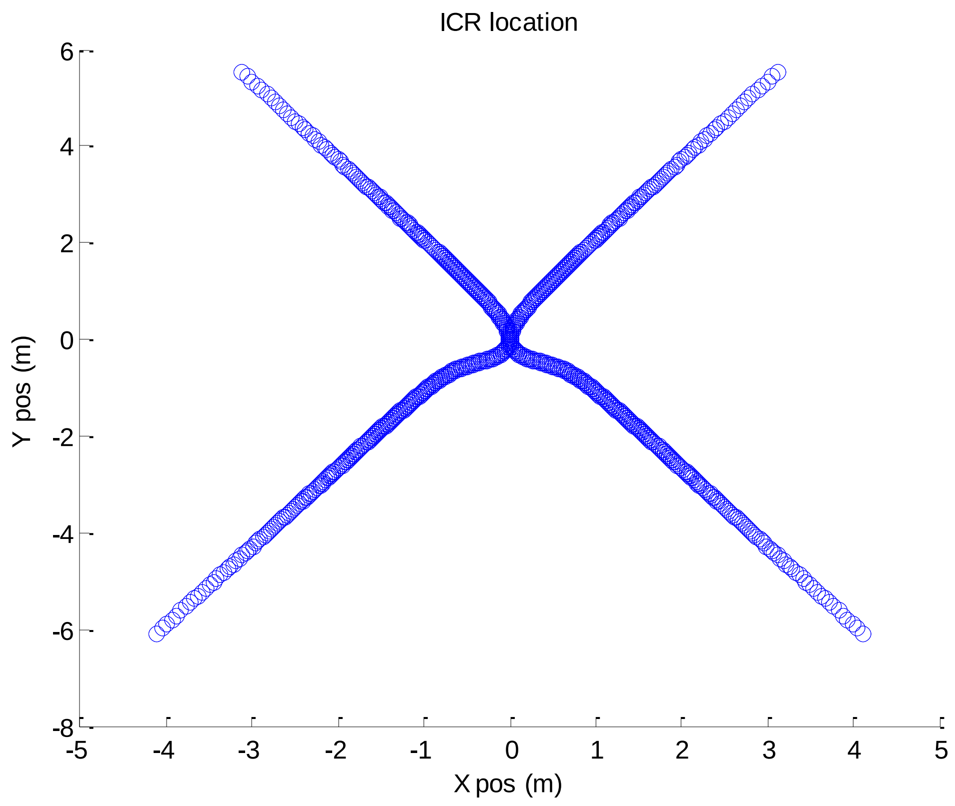

Figure 11: ICR location from the simulation of a perched UAV

when it starts to slip, the ICR deviates from that of the perch. It does not necessarily move toward the UAV. Depending on the motion of the perch, when the UAV slips, its ICR can move either toward it or away.

Figure 12 shows the results of the Integration and Radius Methods during the swing test with the white painted PVC perch, for calculating the ICR, as well as the results from the Vicon system. These figures show the x (solid) and y (dashed) position estimates as functions of time. These plots are generated from the same test, however the plots are comparable across the five

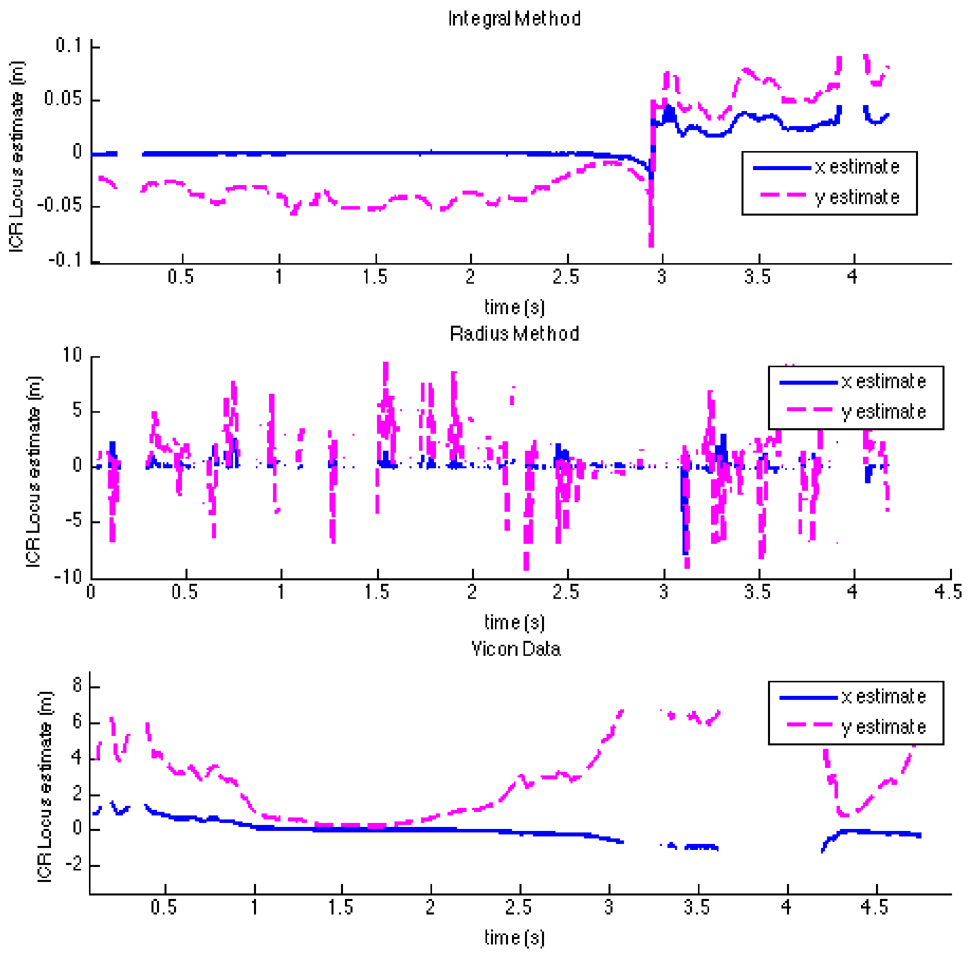

Figure 12: Plots of the ICR location vs. time from the integration, radius, and external sensing methods.

tests.

As can be seen in the plots in Figure 12, the Radius Method clearly failed. That being said, the integration method does give useful information. It is clear when the quadrotor fell, as the y signal jumps from negative to positive. This point corresponds with the point where the quadrotor falls within a few tenths of a second. This indicates that the ICR has suddenly switched from being above the quadrotor to below it. When these plots are compared to the plot from the Vicon data, it is clear that the two ICR methods do not estimate the ICR very well. However, they do provide a key flag for when the quadrotor slips off its perch that is not as apparent in the Vicon data.

Of special note with these results are the results from the radius test. While the results here look very sporadic, we believe this is due to sensor placement. Looking back to Figure 12, it can be seen that if the three points where the measurements are taken are collinear, then there would be two possible solutions to the location of the ICR. The sensor placement on the divining stick only had the middle accelerometer approximately one centimeter off of the line formed by other two accelerometers. With this small amount of spacing, when noise in the accelerometer signals is taken into account, we believe that the estimation is jumping between these two solutions. When analyzing the actual data, this does seem to be the case.

We were able to demonstrate this phenomenon in simulation. A rigid body attached at the end of a pendulum oscillating at a specified amplitude and frequency was simulated such that the ICR should always be at -1m in the x direction in the local rigid frame. Accelerations were measured for virtual accelerometers mounted in two different configurations shown in Figure 13 and Figure 14. Configuration 1 has a sensor configuration close to collinear, as in the experiments performed. As can be seen in Figure 13, there are two clusters of circle intersection points on either side of the y axis. In general the correct cluster on the negative x

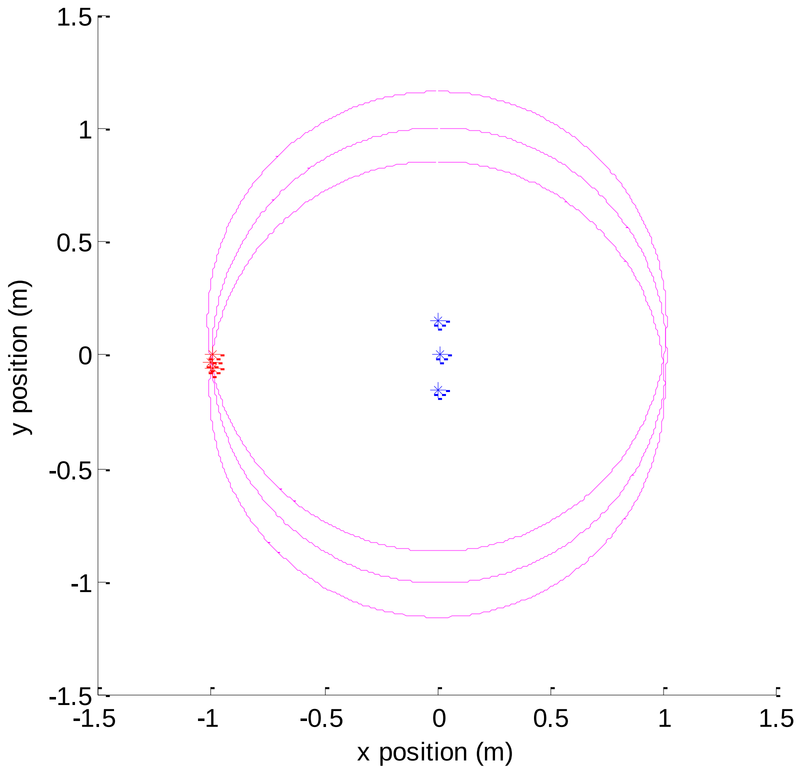

Figure 13: Sensor configuration 1 with circles of estimated radii and intersection points labeled. The blue stars represent the sensor locations. The circles are centered at the sensor locations, and have a radii based on the proportionality constant described in (17) and the magnitude of the acceleration. The red stars show the closest grouping of three points within the set of six points that are the intersection points of the three circles.

side is closest, but when noise is introduced, the cluster on the positive x side is sometimes closer. This causes the estimated location of the ICR to bounce across the y axis. The second sensor configuration has more eccentricity from the collinear state, as suggested. Figure 14 shows that cluster of intersection points is clearly in the negative x side of the plane and the points are closer together, yielding a more accurate result.

Figure 15 shows the results of the simulated ICR estimator for both sensor configurations when sensor noise is injected into the system. The results for sensor configuration 1 occasionally jump across the y axis. This is the phenomenon we observed in the

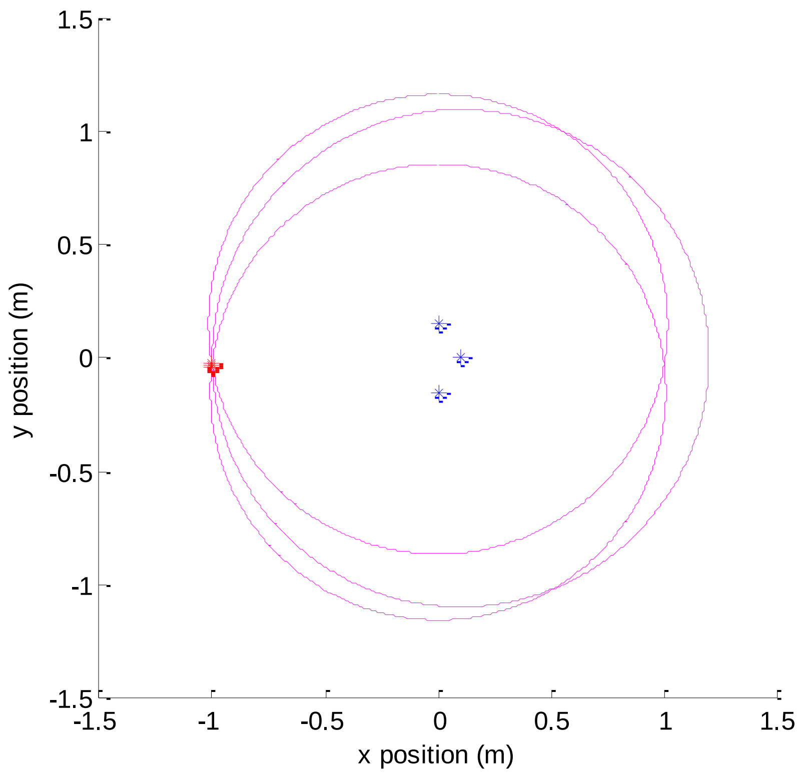

Figure 14: Sensor configuration 2 with circles of estimated radii and intersection points drawn. The points and circles are the same as in figure 13.

actual test results from the experiments. The second sensor configuration proves to be robust to this noise while still accurately estimating location of the ICR. We plan to demonstrate this experimentally in future work.

Another important point to note is that the data shown in Figure 12 are filtered. Even the results from the external sensor method, which were based on readings from a Vicon motion capture system, a system with submillimeter accuracy in its position measurements, were noisy. This indicates that the ICR location method based on finding velocities and traveling a distance perpendicular to them is highly sensitive to noise. This makes sense, especially as the angular velocity goes to zero. As can be seen in equation (16), this noise in a small ω results in

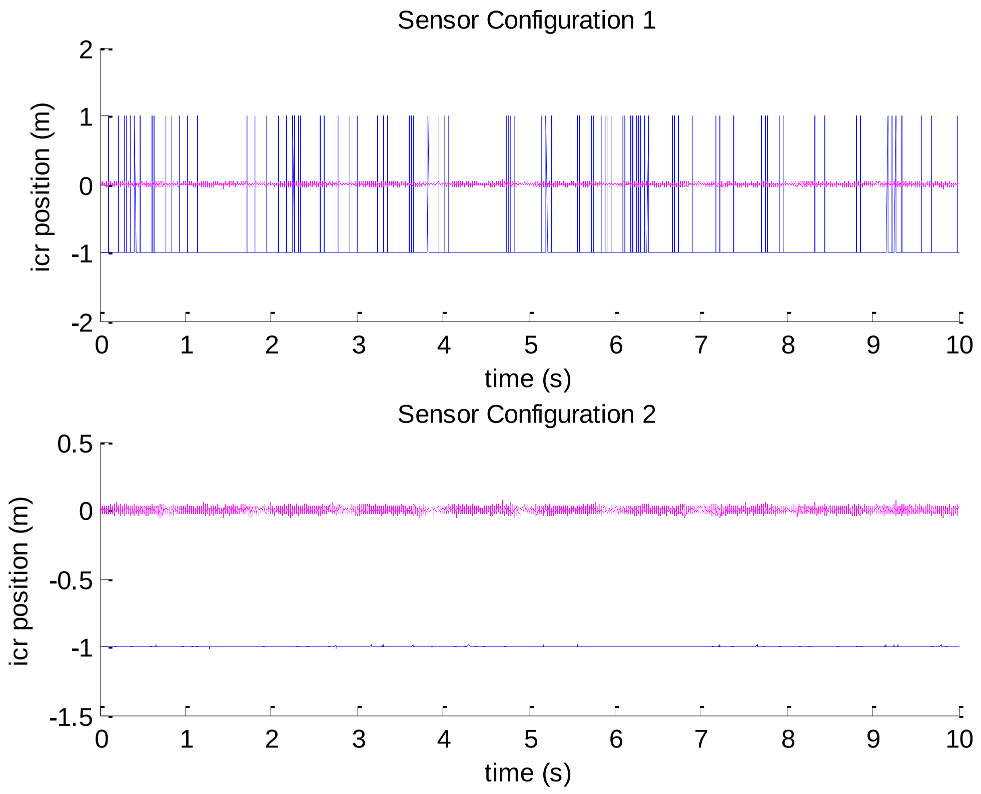

Figure 15: ICR location estimates from both configurations. Blue lines is the estimated local x coordinate, magenta is the estimated local y coordinate.

large changes to the radius estimation. With the radius method, noise in the estimates of the radii would not be inversely proportional to the magnitude of the angular velocity. Since the radius is proportional to the square of the angular acceleration estimate, and the fourth power of the angular velocity, the effect of the noise is further minimized. For these reasons, we believe that the radius method, despite its apparent failure in the experiments, is the most promising of the proposed methods. It is less sensitive to noise, and its failure may be due to poor sensor placement rather than a flaw in the underlying theory.

Advantages and Limitations

The main advantage of these ICR methods over the vision methods from the previous chapter is in the amount of computation power required. Even though these methods do implement EKFs, they use a relatively small number of states and are relatively simple, especially when compared to the computation required to perform the image processing and computer vision algorithms required for the computer vision approach. The EKF is even less complicated when it is taken into account that all the matrices for the EKF are precalculated, which reduces the amount of computation need for the EKF significantly.

Another advantage of the ICR method is that it works independent of the type of perch (e.g., a tree branch, power line, pvc pipe, etc.). As long as the motion of the perch is known, the ICRs of the perch and UAV can be compared to detect slip.

One of the main limitations of the ICR methods lies in the fact that ICR method requires either noisy, onboard sensors, or external sensors. While the theory of comparing the ICRs of the UAV and the perch does work in simulation, there are issues with the practical implementation of the methods for calculating the ICR.

The other main limitation of the ICR methods overall is that they require some sort of a priori knowledge of the motion of the perch the UAV would be landed on. The exact amount of hindrance this limitation offers is dependent on the application.

The integration method does not calculate the ICR very accurately, with a few meters of error, however, the values returned by this method may still be useful for detecting falls. The main advantages of this method are that it involves inexpensive onboard sensors and has limited processing requirements. The main disadvantage is that this method does not do an adequate job detecting a fall. Noise in the sensors as well as the lack of a good way to measure velocity are the primary sources of error in the velocity method.

The advantages of the radius method are the same as those of the velocity method, but this method has even worse results. However, these results are likely due to poor sensor placement. Simulation results were able to recreate the noise seen in the radius method and also demonstrate an alternative sensor configuration that is robust to this noise. While some additional experimentation is planned for future work, the radius method shows promise to be less sensitive to noise and be able to estimate the location of the ICR.

The external sensor method does provide relatively accurate estimates for the ICR, however, this method does rely on expensive, external sensors. This is a major disadvantage to this method, as it severely limits the applications of this method and thus its overall usefulness.

These onboard ICR methods seem like a good idea in theory—they use readily available sensors, require very little in the way of additional hardware, and the computation is simple, compared to vision methods. While there is more work to be done on these methods, they do show promise and are worth investigating. That said, the other main disadvantage of these methods is that if the ICR is calculated, some a priori knowledge of the motion of the perch is still required to predict falling accurately.

CHAPTER 4

CONCLUSION

This thesis proposes several methods for detecting a fall in a UAV perched on a moving perch. The first is an approach based in computer vision and relies on a camera being aimed at the surface the UAV is perched on. A few methods are proposed to use the Instantaneous Centers of Rotation (ICR) and inertial sensors to detect the fall.

The vision methods are based on a feed from a camera mounted under the UAV pointed at the perch. Analysis of this video feed using OpenCV gives an estimate of the relative motion of the two rigid bodies using optical flow, and a specialized filter to locate the perch in the image.

The ICR approaches focus on calculating the ICR using various onboard inertial sensors such as gyroscopes and accelerometers. Another approach uses external motion capture sensors to determine the motion of the UAV. All these methods involve comparing the ICR of the UAV to the expected ICR of the perch. This expectation is based on the type of perch and the type of motion the UAV is expected to undergo. This can be used to detect a fall.

Summary of Advantages and Limitations

The vision method is fairly accurate, however, it also has high computational requirements. The other major disadvantage of the vision system is that it relies on the perch having some detectable texture or unique features that can be tracked by the computer vision algorithms to accurately characterize the relative motion between the perch and the UAV. However, lag in the filter can be exploited to detect a fall in a featureless perch. This method does rely on a dedicated camera pointed at the perch, at least while the UAV is expected to remain perched. It also requires significant computational resources in order to perform the analysis in a timely fashion.

The ICR of the UAV can also be used to detect a fall from a moving perch. The most promising of the proposed methods involve using magnitude of the acceleration vector at three different points on the frame of the UAV and calculating the intersection of three circles. While we were not able to get this method to work experimentally due to poor sensor placement, we believe that it is still a viable method if the sensors could be place in a more ideal fashion. We were able to show in simulation that the proposed sensor placement will eliminate a lot of the noise we were seeing. We believe that with this configuration, we will be able to accurately estimate the ICR of the UAV. Simulation results as well as the other approaches, particularly the external sensor approach, did show that the ICR can be used to detect a fall. However, using external sensors severely limits the usefulness of this solution. The most promising aspect of using the ICR is that it makes use of sensors that every UAV would already have on board, possibly with the addition of only a couple of inexpensive, light, and easily mounted sensors elsewhere on the frame of the UAV.

For best results, both vision and ICR methods should be implemented in parallel. This would provide a more robust system overall, and the two systems could work together under a wide range of conditions to provide an accurate to the question, “am I falling?”

Future Work

There are several directions that this research could take at this point. The first step would involve polishing some of the methods discussed in this thesis. The vision method could potentially be improved if the camera was adjusted such that it had one of the feet of the landing gear in its view. Using a similar optical flow algorithm and filter as that provided in Chapter 2, the computer could look for differences in the optical flow of the feet and the perch underneath them. Another possible area to look at with the computer vision method is using the sparse optical flow algorithm supplied by OpenCV. Since the dense optical flow requires good texture with good features anyway, the sparse method could improve timing by only considering a few points rather than the whole image. The down side to this is that the segmentation filter would need to be rewritten to use only the pixel values and possible position for the segmentation.

The ICR methods could be improved, as mentioned earlier, with better sensor placement. The main reason for the failure of the radius method for ICR detection in our experiments, we believe, is poor sensor placement. By placing the sensors in a less collinear configuration, the results of this method should greatly improve. Simulations show that this new sensor configuration will be more robust to noise. There is not, however, much that can be done to improve the integration method. Some filtering could be used to clean up the signals a bit maybe, but in the end, the radius method is probably the better of the two methods. If the ICR could accurately be measured with onboard sensors, or if the ICR was implemented utilizing an external motion capture system, there is still the major limitation that the ICR of the perch is assumed to be known a priori, or at least there is a general idea of the expected motion. While this is a reasonable assumption for some types of perches, such as a tree branch or a power cable as considered in this thesis, a more general approach would be ideal. Some more general approaches to consider would be comparing the ICR of the UAV to some intrinsic property of itself, such as its center of mass or the location of the feet. Comparing the ICR to the center of gravity could give some insight into the stability of the motion the UAV is undergoing, whether it is pendulum-like, or inverted-pendulum-like. Comparing the ICR of the UAV to the location of the UAV's feet could tell how much of the motion is from rotation about the feet as opposed to perch motion. The motion constraints of the perch itself could also be used. In the case of the perch used in our experiments, the perching mechanism restricts the motion of the UAV during a perch to be a rotation about the feet. This could possibly be exploited to detect a fall. If the ICR of the UAV is near the feet, then the robot is likely falling. In short, there are several areas that can be explored relating to using the ICR to detect a fall.

It may also be beneficial to look at some additional analysis tools, specifically bidirectional filters such as a recursive least-squares filter. We think that such tools may provide better estimates than the Kalman filter used in the ICR methods. We did not consider these filters in this thesis since they are not useful in real-time applications, and the Kalman filter worked for our needs. We do feel that it is worth looking at these tools in the context of a future paper. Such filters could provide smoother, more accurate data that could provide more insight into how these methods perform. They could also be used to verify the EKF, or even to replace it in some cases if data can be extrapolated from filter results.

While some of the methods proposed are not fully flushed out, they did all show some promise for the most part. After polishing some of these methods a little more, the next step is to combine the best methods presented, namely the vision method and the radius ICR method, with some tool such as a support vector machine or other classification tool to combine the advantages of each method. Another area where the two methods could be combined would be looking at using vision data to aid in the state estimation of the Kalman filters used in the ICR methods.

The other major next step in this line of research is consideration of additional sensors. Tactile sensors in the feet of the perching mechanism have the potential to detect slip between the feet and the perch. Other sensors include potential use of range finders to estimate distance from the ground. Using an additional camera to get stereo vision could also work to augment the ICR method. The focus of this thesis was to avoid implementing lots of additional sensors, aiming more to create a process that can be implemented easily on almost any UAV. However, these additional sensors could assist in the task, and may be worth considering, depending on the application.

Final Thoughts

In conclusion, this thesis set out to propose a valid method of allowing a UAV perched on a moving perch to detect slip between itself and the perch in a simple way, making the most of sensors commonly found on most UAVs. The key to doing this is distinguishing between the motion from being attached to a moving perch and the motion, if any, from a slip between the perch and the UAV. There are two main methods proposed for doing this, a vision-based method looking for relative movement between the UAV and the perch, and an instantaneous-center method that compares the ICR of the UAV to the ICR of the perch. Both approaches work in their own right and can be used to detect a fall. Both have their advantages and disadvantages, but they also complement each other well, and the best approach will likely be a combination of the two.

BIBLIOGRAPHY

[1] M. Kovač, J. Germann, C. Hürzeler, R. Y. Siegwart, and D. Floreano, "A perching mechanism for micro aerial vehicles," *Journal of Micro-Nano Mechatronics,* vol. 5, pp. 77-91, 2009.

[2] A. L. Desbiens, A. T. Asbeck, and M. R. Cutkosky, "Landing, perching and taking off from vertical surfaces," *The International Journal of Robotics Research,* p. 355-370, 2011.

[3] L. Daler, A. Klaptocz, A. Briod, M. Sitti, and D. Floreano, "A perching mechanism for flying robots using a fibre-based adhesive," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013, pp. 4433-4438.

[4] K. Mohta, V. Kumar, and K. Daniilidis, "Vision-based Control of a Quadrotor for Perching on Lines," presented at the IEEE International Conference on Robotics & Automation, Hong Kong Convention and Exhibition Center, 2014.

[5] C. E. Doyle, J. J. Bird, T. A. Isom, J. C. Kallman, D. F. Bareiss, D. J. Dunlop*, et al.*, "An avian-inspired passive mechanism for quadrotor perching," *Mechatronics, IEEE/ASME Transactions on,* vol. 18, pp. 506-517, 2013.

[6] C. Melchiorri, "Slip detection and control using tactile and force sensors," *Mechatronics, IEEE/ASME Transactions on,* vol. 5, pp. 235-243, 2000.

[7] A.-V. Ho and S. Hirai, "Slip Perception Using a Tactile Array Sensor," in *Mechanics of Localized Slippage in Tactile Sensing*, ed: Springer, 2014, pp. 155-178.

[8] B.-c. Kim, H. Cho, D. Kim, U. Kim, H. R. Choi, H. Moon*, et al.*, "Design of slip detection sensor for artificial skin," in *Ubiquitous Robots and Ambient Intelligence (URAI), 2012 9th International Conference on*, 2012, pp. 510-511.

[9] J. Chung, B.-c. Kim, H. Choi, H. Moon, Y. Lee, J. Nam*, et al.*, "A Flexible Tactile Capacitive Sensor for Slip Detection," in *ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2011, pp. 507-510.

[10] G. Reina, G. Ishigami, K. Nagatani, and K. Yoshida, "Vision-based estimation of slip angle for mobile robots and planetary rovers," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008, pp. 486-491.

[11] R. Gonzalez, F. Rodriguez, J. L. Guzman, C. Pradalier, and R. Siegwart, "Control of off-road mobile robots using visual odometry and slip compensation," *Advanced Robotics,*

vol. 27, pp. 893-906, 2013.

[12] A. Ikeda, Y. Kurita, J. Ueda, Y. Matsumoto, and T. Ogasawara, "Grip force control for an elastic finger using vision-based incipient slip feedback," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 2004, pp. 810-815.

[13] B. K. Horn and B. G. Schunck, "Determining optical flow," in *1981 Technical symposium east*, 1981, pp. 319-331.

[14] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis*, ed: Springer, 2003, pp. 363-370.

[15] (2014). *OpenCV Documentation for CalcOpticalFlowFarneback*. Available: http://docs.opencv.org/modules/video/doc/motion_analysis_and_object_tracking.html#calcopticalflowfarneback

[16] G. D. Finlayson and G. Schaefer, "Hue that is invariant to brightness and gamma," in *BMVC*, 2001, pp. 1-10.

[17] K. L. Crandall and M. A. Minor, "UAV fall detection from a dynamic perch using Instantaneous Centers of Rotation and inertial sensing," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, 2015, pp. 4675-4679.

[18] N. Grip and N. Sabourova, "Simple non-iterative calibration for triaxial accelerometers," *Measurement Science and Technology,* vol. 22, p. 125103, 2011.