The Pennsylvania State University
The Graduate School

ESTIMATION OF BIO-INSPIRED VISUAL CUES USING ONBOARD CAMERA IMAGES FOR LANDING A NANO-QUADCOPTER

A Thesis in
Mechanical Engineering
by
Wesley Huff

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2022

The thesis of Wesley Huff was reviewed and approved by the following:

Bo Cheng
Associate Professor of Mechanical Engineering
Thesis Advisor

Katie Fitzsimons
Assistant Professor of Mechanical Engineering

Daniel Haworth
Professor of Mechanical Engineering
Associate Head of Graduate Program of Mechanical Engineering

Abstract

Unmanned aerial vehicles have become increasingly prevalent in a multitude of industries across the globe. However, many challenges prevent their wide adoption, specifically their limited flight time. In order to alleviate this problem, previous work done by Habas *et al* implemented an inverted landing strategy using bio-inspired visual cues that were calculated with external positioning. This thesis extends the previous work by estimating these visual cues on-board the micro unmanned aerial vehicle. Additionally, the limits of the developed on-board visual cue estimation algorithm are explored in regards to ceiling pattern, frame rate, and computational load. Once the visual cue algorithm was explored, it was run through the same reinforcement learning algorithm from Habas *et al* to determine the landing robustness.

In nature inverted landing maneuvers are observed in several organisms such as flies, bees, and bats. In this thesis, the maneuvers observed in blue bottle flies are used as inspiration to achieve the inverted landings for a small quad-copter. In order to implement the inverted landing strategy, two biological cues are used, time to contact and optical flow. Specifically, for the blue bottle fly, time to contact determines when the flip maneuver should initiate, while optical flow helps with the final landing phase. To emulate these biological cues the Horn and Schunck method is employed to segment image data from an onboard camera. The major contribution of this work is to explore the main variables that introduce error due to the onboard noisy sensors and how they affect landing performance. In order to explore these variables the algorithm testing and the landing policy learning were conducted in a physics-based simulation. Future work will then be done in order to experimentally determine the landing robustness of the implemented algorithm.

Table of Contents

List of Figures

List of Tables

List of Symbols

τ	Time to contact, p. 9
$d_{ceiling}$	Distance of quadrotor to ceiling surface, p. 9
V_z	Quadrotor velocity in the z direction, p. 9
V_x	Quadrotor velocity in the x direction, p. 9
I	Pixel Intensity, p. 15
u_p	Pixel coordinate in the u direction, with units of pixels, p. 15
v_p	Pixel coordinate in the v direction, with units of pixels, p. 15
t	Time, p. 15
ϑ_u	Pixel velocity in the u direction, p. 15
ϑ_v	Pixel velocity in the v direction, p. 15
f	Camera focal length, p. 16
ω	Rotational velocity of quadrotor body, p. 16
G	Radial Gradient, p. 17
K_u	Normalized Sobel Kernel in the u direction, p. 17
K_v	Normalized Sobel Kernel in the v direction, p. 17
O_x	Pixel offset, x component, p. 18
O_y	Pixel offset, y component, p. 18
U	Pixel coordinate in the u direction, with units of meters, p. 18
x_p	Pixel index x component, p. 18
W	Pixel width in meters, p. 18

U Pixel coordinate in the v direction, with units of meters, p. 18

y_p Pixel index y component, p. 19

Acknowledgments

Bryan Habas for all of his help in integrating me into this project. All my friends who I made read my paper in the worries of it not being clear enough. And finally flies for their inspiration on how to make a drone land on inverted surfaces.

Chapter 1
Background and Motivation

1.1 Practical Aerial Vehicle Applications

Unmanned Aerial Vehicles (UAV), have started to make a substantial impact on both the consumer and industrial markets. In 2021, the global consumer UAV market was valued at $6.51 billion, with a projected growth of up to $47 billion by 2029 [6]. The consumer market typically uses UAV systems for purposes such as drone racing or photography. While businesses use UAV systems for shipping goods [7], inspecting crops [8], as well as navigating and mapping mines [9]. UAVs are even being used for search and rescue [10], as well as other humanitarian tasks like delivering vaccines or medical aid [11]. Hence, UAVs, such as quadrotor robots, have become multifaceted devices used in a many different situations due to a multitude of factors, such as their high maneuverability as well as ease of both autonomous and user capabilities.

Autonomous UAV systems are useful in removing workers from potentially hazardous conditions. From the years, 2003-2008 delivery workers have accounted for 17% of all workplace fatalities [12]. The reason a significant amount of the fatalities come from the delivery industry is because workers are generally worked long hours with monotonous work, notably delivery and truck drivers. One way UAVs have been seen as a solution to this problem is by aiding delivery workers with the final part of a package's journey, also known as last mile deliveries. Although their wide adoption has not been implemented, companies like Amazon, Google, UPS, and DHS have been researching UAVs for last mile deliveries since 2005 [13]. Their aim is to reduce the most expensive part of a package's journey, increase efficiency, but also decrease the environmental impact, with preliminary results showing great promise [14] [15] [16].

However, UAV technology can be potentially used more extensively, notably by integrating multiple solutions within one platform. One such example is Precision

Agriculture, which is a new method of managing crops by increasing the amount of information available to those responsible so the farmers can act intelligently rather than broadly. This increases not only the farmer's productivity but also output, while decreasing user input [17]. Not only does this help reduce waste and the use of harmful chemicals, it also removes workers from contact with these potentially hazardous chemicals. Furthermore, if less pesticides are used the resulting environmental impact on the surrounding water, soil, and other organisms in the ecosystem will be reduced [18].

Although these technologies have great potential for large scale applications, the high level of automation and upfront costs can be daunting due to the steep learning curve associated with advanced hardware and software implementations. Thus, leading to a niche group adopting the extremely specific hardware that is developed. However, UAV systems do not need to solve the entirety of the problem, their aid can come from decreasing the amount of effort required for certain tasks, most notably, surveying, wildlife management, and relief and rescue operations. UAVs excel at aiding users with collecting vast amounts information thanks to the help of powerful consumer technology and sensors being readily available to users. Coupled with the ease of user control for consumer products the barrier to entry can be significantly reduced.

In all of these various applications UAVs share one thing in common: they have onboard sensors, most commonly a camera. Cameras allow users to rapidly log large quantities of data and quickly relay it to the user. Wherein, the users can utilize it for their specific needs. Whether it is for simply recording video, or using specialized camera equipment to plan, sense, or gather data, cameras are essential tools for UAVs. Because of this, the following work will utilize a camera sensor, which is commonly on many UAV platforms.

1.2 Micro Aerial Vehicles: Limitations and Challenges

Even though UAV technology has vast applications and shows great promise, it still suffers several limitations. Most notably, the trade-offs required to make an engineered system capable of flight ultimately lead to UAVs that are under-actuated with limited computational capabilities and short flight times. An under-actuated system is one in which the movement in various Degrees of Freedom (DoF) are coupled together such that the system cannot move in all 6 DoF independently. Thus, these coupled actions result in systems that utilize more energy to change states compared to fully actuated ones. In comparison fully actuated flying systems outperform engineered under-actuated flight

by a significant margin. For example, bats, which are fully actuated, can complete 180° turns all while maintaining their original forward velocity, while not even moving as far as their own wingspan [19]. While newer UAV systems are actively trying to combat this by adding more actuators this lends to even more pressure on the computational abilities of the hardware, which as previously mentioned is limited due to the miniaturization of the onboard components. Although computers have been advancing their computing power and reducing their size at extraordinary rates the complex problems flight controllers and other hardware have to synthesize and solve, like trajectory generation, remain extremely difficult [20]. Therefore, adding even more actuators, and further complicating the dynamics of the system, means more expensive hardware is required. In an effort to solve these problems some systems have to rely on external sensors and ground stations in order to collect and compute desired information because onboard systems are not sufficiently powerful. Even if it is possible to get a powerful computer airborne powering the system for long periods of time remains one of the biggest challenges. The problem of powering UAVs is complicated further the smaller the system becomes due to the smaller Reynolds number, which reduces the flight time, due to increased viscous effects. Further, due to the aerodynamics and miniaturization of the motors, the flight velocity also decreases the smaller the system becomes, lowering the Reynolds number even more [21]. Ultimately, the combination of these drawbacks leads to most smaller systems with a flight time of around thirty minutes and under. As shown in Figure 1.1 the energy required to move a specific velocity for smaller systems is significantly lower than larger systems. Additionally, batteries, which are commonly used power sources for UAVs, also suffer when scaled down in size. Even though battery technology has been advancing at an ever increasing rate their energy density still pails in comparison to typical fuels other engineered flying systems use. For example, typical aircraft fuels are nearly 80 times more energy dense than the average battery [22].

1.3 Solutions for Alleviating Limited Battery Life

Resolving the limited battery life is one of the major issues many researchers are attempting to solve. Extensive studies done to increase the operating time of UAVs for various shipping tasks do not consider how much power onboard sensors use [1]. Therefore, stationary collection of information while the system is perched will greatly increase the operating time of the system. The reason onboard sensors are not included in the models used to predict flight time is due to the fact that motors utilize orders of

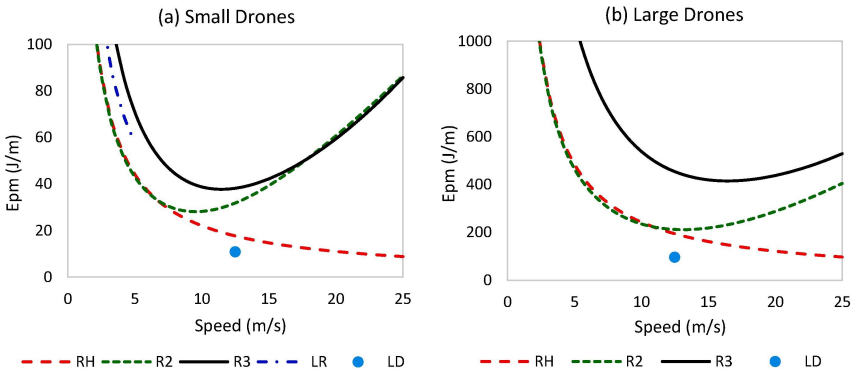

Figure 1.1. An example comparing the amount of energy UAVs have to use in order to move at a certain velocity determined by different flight models [1]

magnitude more power than onboard sensors, regardless of what UAV platform is used. Thus, in order to increase the operating time of UAVs, its imperative to decrease the amount of time the motors are in use. However, this is where the challenge lies, how can a drone not use the only component that keeps it in the air?

To begin to solve this problem understanding what exactly a UAV encompasses is needed. UAVs do not have a standard definition due to the multitude of factors that they can be classified by; from operational weight, to number of motors, and body design, UAV is a catch-all for a multitude of UAV designs. Therefore, some of these systems rely on more intelligence put into the frame design that aids with extending battery life, for example, fixed wing hybrid UAVs. However, fixed wing hybrid UAVs come at the cost of maneuverability which is the advantage other UAV platforms rely on, including this study. For this thesis the dynamics of the system is modeled after the Bitcraze Crazyflie 2.1 Micro UAV (MAV) system, which is a quad-copter design as seen in Figure 1.2. The reason this platform was chosen was for its maneuverability as well as its ease of control and low level programming of the hardware due to its open source nature. The current battery life of the Crazyflie drone is about 10 minutes which is not nearly long enough if, for example, this drone was being used for surveying purposes.

The way many have tried to alleviate the constrained battery life of UAVs is by taking inspiration from biology. In engineering there are countless examples of every day items being inspired from something observed from nature, from VELCRO™ all the way to wing design. After all, why wouldn't designers try to copy from nature; biology has had millions of years of survival pressure to adapt these creatures to be able to solve highly dynamic and niche problems in extremely robust and efficient manners. This thesis aims to investigate landing strategies that will allow the drone to extend its lifetime

Figure 1.2. The Crazyflie 2.1 with AI Deck attached.

by decreasing the amount of time spent powering the motors. More specifically, our inspiration for this task comes from nature's robust ability to perch. Biological flyers like flies have been observed rapidly determining not only where but how to land in a multitude of complex environments [2] [23]. Which in comparison to the vasts amounts of highly advanced sensors and controllers engineers have designed they still struggle to solve this feat as efficiently and elegantly as nature. Achieving these perching strategies with also only around 100,000 neurons which pails in comparison to billions of transistors on a typical programmable logic controller. Ultimately leading researchers are attempting to copy them.

1.4 Biological Inspirations

Specifically in this thesis the inverted landing strategy observed in blue bottle flies is of interest for bio-inspiration. When it comes to biological flyers the same exact motivation they have for landing is mirrored by several tasks drones routinely do. Whether it is simply to rest or to gather more information from the surrounding area, the parallels are apparent. If nature does not fly constantly why should our attempts at copying them be any different. Doing so not only increases the amount of time the UAV can take in data but it also allows for stationary collection of data, opening the door for closer and more advanced sensors to be used. Furthermore, it allows the ability for drones to potentially

dock and charge on specific charging locations, which will expand the drone's operational limits tremendously. However, simply trying to copy from nature is merely a starting point. In order to do so one must first understand how nature completes these amazing feats.

1.4.1 Tau Theory

Reacting to thrown objects, whether to dodge or catch them, is a trait most are familiar with, and research has discovered that the ability to quickly respond and determine the appropriate reaction may come from unconscious processing [24]. Research has observed that in order to react to an object organisms of all kinds visually or audibly observe objects of interest and determine a specific value of Time to Contact (TTC or τ), also known as Tau, in order to couple an appropriate response to the stimulus [25]. This area of research is known as Tau theory and its role serves as a powerful shortcut organisms use to solve problems that arise in complex dynamic environments. The findings from Tau theory enable organisms to solve problems by measuring one value, referred to as Tau, as opposed to several spatial and temporal derivatives that classic kinematics uses to solve these problems. Thankfully, research from the collaboration of biologists and engineers has become a popular interest due to nature's robust ability to solve difficult problems with limited sensing and computational capabilities. Thus, allowing engineers to apply and advance systems using the elegant solutions nature has evolved to perch.

When biological flyers approach a perching target they utilize two main visual cues, Time to Contact and Optical Flow (OF). TTC is simply the time remaining before anticipated contact with the target, assuming constant velocity. While optical flow relates translational movement with respect to another object. The way these triggers are used depends on the specific animal and behaviour. However in general, depending on the type of landing that is initiated, these values are used by the animal to trigger the appropriate physical response in order to ensure a successful landing. For example, pigeons maintain a constant rate of change in TTC in order to ensure that they do not over or undershoot their perching target [26]. This strategy ensures robust landing performance by being able approximate the dynamics of landing as opposed to somehow determining not only the distance but also their velocity with respect to the perching target.

Another example can be observed in blue bottle flies, which are smaller in size than bats, yet are still able to complete impressive aerial maneuvers all in a fraction of a second. Their ability arises from utilizing a certain threshold of Relative Retinal Expansion Velocity (RREV), which is simply the inverse of TTC, in order to determine

when to flip. As well as relying on mechanical intelligence that is observed in their landing gear. An example of which can be seen in Figure 1.3. Thus, leading to where we are getting our inspiration from biology, the inverted landing techniques observed in blue bottle flies [2].

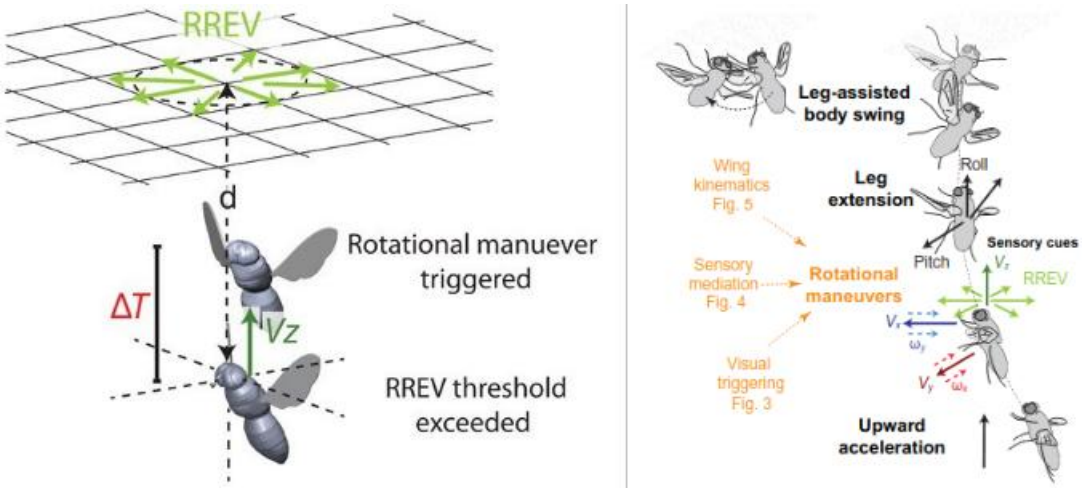

Figure 1.3. Inverted landing strategy of Blue Bottle flies as observed in [2]

1.4.2 Bio-inspired Landing Strategies

Several research teams have taken the angle of bio-mimicry in attempts to increase the performance of UAVs. Many of their efforts are focused on the mechanical intelligence that arises from perching animals. Mellinger *et al* have created a powerful gripping mechanism for robust perching of drones [27]. While MIT has developed an algorithm that detects power lines in order to perch and harvest energy [28]. The University of Utah has designed a passive gripping mechanism that can grab onto curved surfaces [29]. However, few have applied the sensing intelligence that nature has evolved. Souhila *et al* achieved obstacle avoidance utilizing optical flow [30]. While McCarthy *et al* developed a strategy that utilizes TTC estimates coupled with tracking the Focus of Expansion (FOE) of an image sequence to dock mobile robots [31]. Lastly, Mellinger *et al* have developed a robust strategy that has been shown to land on various surfaces [32]. However, these systems either do not respond fast enough for practical drone applications or rely on external positioning. Which is where this work intends to both increase the efficiency of the algorithms that enable robust landing strategies, while removing the reliance of external systems.

1.5 Goals and Objectives

This thesis is an extension of Habas *et al*, where they developed an EM-based Policy Hyper Parameter Exploration (EPHE) Reinforcement Learning (RL) algorithm which learned numerous landing control policies in both simulation and experimentally [3]. However, the successful inverted landing strategy was implemented with external positioning. Although, the RL algorithm was fed optical data in a fashion that specifically allowed for future implementation of on-board sensor data to be used in place of the external positioning information.

The same exact landing strategy will be employed in the following work however the objective of this work is to remove all external positioning. For this purpose, specific image processing techniques will utilize an onboard camera in order to use successive image sequences to calculate the visual cues that are fed into the RL algorithm. The specific desired outcomes are to determine the acceptable limits of error that arise from the hardware limitations being modeled. As with many electromechanical systems many of the challenges lie with the difficulties that come from utilizing digital sensors. Therefore, the first three experiments will focus on exploring the limitations of the image processing technique utilized. Specifically, it will be determined what ceiling pattern can be used to garner accurate estimations, the effects of variance in frame rate, and the effects of varying percent of image data used in the calculations. The last experiment will analyze how these three variables affect the RL algorithm previously developed by Habas *et al.*

Chapter 2
Methods

2.1 Optical Flow Estimation Techniques

While understanding how nature solves the complex problem of perching, the engineering challenge is determining how to mimic or translate these strategies considering the difference in design and hardware. First and foremost discerning the egomotion, which is defined as the three dimensional movement of a camera within an environment, has been of interest to researchers since digital cameras have become widely available. Determining egomotion is useful for numerous tasks like image stabilization and visual odometry, the latter of which is an essential tool for UAVs. Visual odometry is essential for standalone UAV systems because traditional methods for determining position like wheel encoders do not apply to flying systems. The goal of visual odometry is to analyze image data to determine the position and or attitude of the camera. Specifically, we are interested in the broad form of egomotion, as shown in previous work from Habas *et al*, by leveraging minimal visual observables it is possible to formulate a robust landing strategy [3]. Of these visual cues two are of interest, TTC and OF, with their mathematical equivalents defined below.

$$TTC,\ \tau = \frac{d_{ceiling}}{V_z} \tag{2.1}$$

$$OF_y = \frac{-V_x}{d_{ceiling}} \tag{2.2}$$

Originally, we explored $RREV$ in lieu of time to contact, this is because, from a biological perspective animals are not extracting position and velocity information from visual data *per se*. Instead, animals are determining how fast an object expands in their field of view. Therefore, while $RREV$ is mathematically equivalent to the inverse of

TTC, it is a more visually intuitive idea in reference to visual information. While $RREV$ and TTC are mathematically equivalent to the inverse of the other, $RREV$ arises from the analysis of visual information as opposed to TTC, which encodes specific state data that is of interest; specifically the upward velocity and distance to the ceiling as seen in Figure 2.1. Alternatively, $RREV$ can be thought of as a biological estimation, while TTC is a kinematic description of the system. Further, TTC or τ, is more extensively used throughout literature, thus more studies have been done with respect to Tau Theory. In total, while these variables are different yet related, RREV may be referenced in place of TTC.

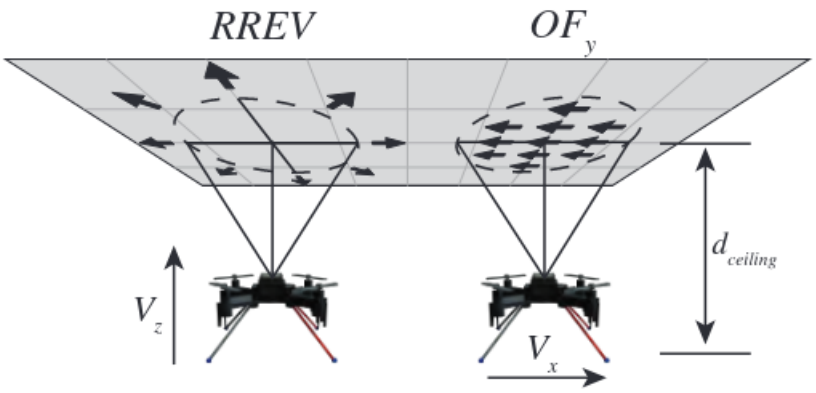

Figure 2.1. RREV and Optical Flow with respect to the quad-rotor's coordinates frame [3].

In literature there exist many methods to determine both TTC and OF, such as derivative methods, energy methods, and phase based methods [33], while only a few are used for visual odometry. The reason only a few of these methods are used for visual odemetry arises from the difficulty of utilizing visual information in order to determine specific visual cues. Particularly, monocular cameras remove the ability to directly extract depth information without utilizing other assumptions and or information from other sensors. Extracting all three dimensions and the corresponding motion in each is known as the aperture problem and is at the forefront of computer vision research, an example of which can be seen in Figure 2.2.

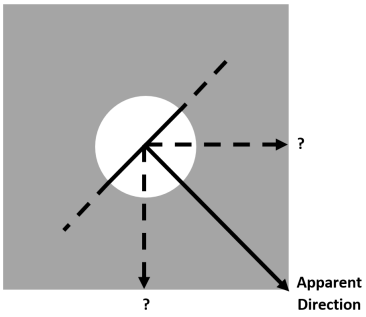

Figure 2.2. An example of the aperture problem. No matter which way the solid line moves in frame it appears to move diagonally because the edges of the object are required discern the object's motion. Another example of which can be seen in barbershop poles, while only rotating the lines appear to translate down.

However, both nature and engineering have a solution for this in the form of stereo vision an example of which is seen in 2.3. The ability to extract depth information is something humans are very familiar with and generally take for granted how accurately they can predict the distance of objects. While engineering has borrowed this template in order to mimic stereopsis and calculate the visual cues we are after, they are generally too large to fit on MAVs, such as our system. Due to the drawbacks associated with depth sensing cameras, like the need for more computing power, and periodic calibration, in order to extract depth information accurately the following study will not be implementing this technology; ultimately leaving us with monocular camera estimation.

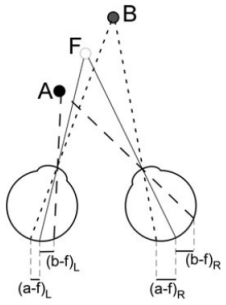

Figure 2.3. How binocular disparity is used in humans to determine depth. The difference in where the different points (A,B,F) land on the retina can be used to infer how far the object is [4].

Within the scope of monocular visual odometry there lies three popular options, feature based estimation, direct (featureless) methods, or phase based methods [34] [35] [36]. Phase based methods rely on frequency domain representation of the image data which

is generally computed using Fast Fourier Transforms (FFT). While these methods are robust with respect to noise, they fall short on accuracy when simpler models are applied and generally rely on nonlinear optimization techniques in order to gather accurate data [37]. Further, because these methods rely on FFT algorithms as well as nonlinear optimizations, this study will not consider their applications due to the direction of future work. We know that the hardware we plan on using does not perform floating point operations quickly, meaning this will drastically reduce the performance of these methods.

Feature based methods rely on simplifying the problem of determining TTC by tracking the change in size of specific objects in frame. If S is the size of an object in frame then TTC can be determined by tracking the change of it's apparent size: $\tau = S/\dot{S}$. However, the accuracy of these methods now relies on the ability to accurately track specific points in the image sequence, introducing a new problem; determining where feature points are and how these feature points move with respect to the camera. In order for this method to be effective it relies on precise feature point tracking algorithms which work well on powerful computers but struggle with lightweight implementations, especially with real image data [38]. While these segmenting and tracking algorithms are constantly improving they generally introduce specific constraints on the tracking points which makes highly dynamic systems, like UAVs, a bad candidate for feature based tracking.

Lastly, featureless estimation relies on the change in pixel intensity over time in order to determine TTC and OF. This is done by assuming a global smoothness constraint which allows it to solve the original ill posed aperture problem. These methods are more prone to sensitivity in noise due to the discrete nature in which the derivatives are calculated. They also struggle with scale factor ambiguity, meaning that if there are two objects in frame and one is twice the size and twice as far as the other they will appear to change the exact same amount as the camera moves. However, their strength comes from the ability to determine accurate measurements under specific conditions with lightweight implementations.

2.1.1 Selection of Optical Flow Method

In total the problem we are trying to solve boils down to balancing two conflicting trade offs. On one hand we observe creatures with extremely limited sensing capabilities completing acrobatic feats that still trump what we see in engineered flight methods. On the other, we have computational methods that exist to calculate the motion primitives

that can be used to replicate nature's robust ability to perch. However, they are generally computationally complex thus requiring simplifications that generally lead to unsatisfactory precision. Deciding what method to use complicates the problem further because research is either focused on accuracy of the methods or the efficiency of the methods, generally not considering both simultaneously. While there are several metrics that these methods are measured by, it is difficult to translate these findings to different hardware setups. Therefore, in absence of many benchmarks comparing both lightweight implementations as well as real time calculation on mobile robots we narrowed the wide selection of choices down by comparing two popular hardware implementations of optical flow estimation algorithms, the Lucas-Kanade (L-K) and Horn-Schunck (H-S) methods.

The Lucas-Kanade method is a feature based tracking method, while Horn-Schunck is a featureless method. The L-K method relies on the assumption that within a specific window in the image, generally the center but it can be over the whole image, the displacement of these feature points remains small. This assumption allows for a small amount of optimization in the algorithm, if the tracked points do not move much between frames then the computationally expensive feature tracking algorithm does not need to run as frequently, an example of which can be seen in Figure 2.4. However, as previously stated because of the highly dynamic nature of UAVs these assumptions are generally not met. Meaning that the edge detection algorithm must run more frequently, which not only leads to more computations but it also increases the variation in time to calculate. Thus, exacerbating the downsides this method has of not performing well in dynamic settings.

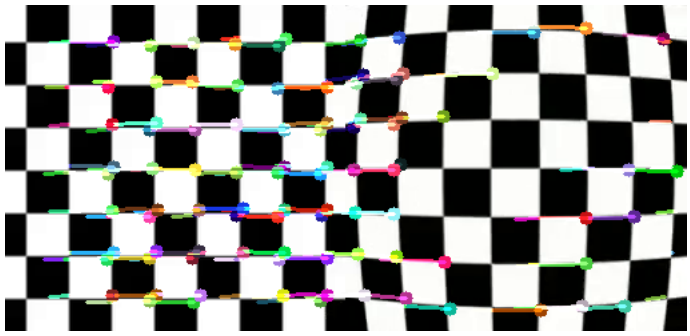

Figure 2.4. An example of the LK algorithm running on a video sequence of a checkerboard translating to the right.

In comparison, the Horn-Schunck method relies on the assumption that the image sequence varies smoothly. In this sense, it seems that the H-S method would suffer the same drawbacks of the L-K method in not being able to handle highly dynamic scenarios,

but this is not the case. The reason being, UAV's typically follow smooth trajectories, which in turn smoothly translates the camera through the environment. Because quad-copters mostly accelerate smoothly this leads to image sequences that generally satisfy the requirements of the HS method. Finally, when comparing both methods on the same hardware setup then the conclusion becomes apparent which method is more applicable to MAV systems. In an experiment done by Zhang *et al*, a mobile robot equipped with a Raspberry Pi and camera is translated towards a wall in order to capture the image sequence of a typical docking scenario [5]. The image sequence is then processed using both the L-K and H-S methods and the comparison of each can be seen in Figure 2.5. Furthermore, they also show that when running these image processing methods on the image sequence the H-S method is also almost four times faster, while having less error. Thus, following these results we then implemented the Horn-Schunck method to calculate TTC and OF on board the MAV with a monocular camera.

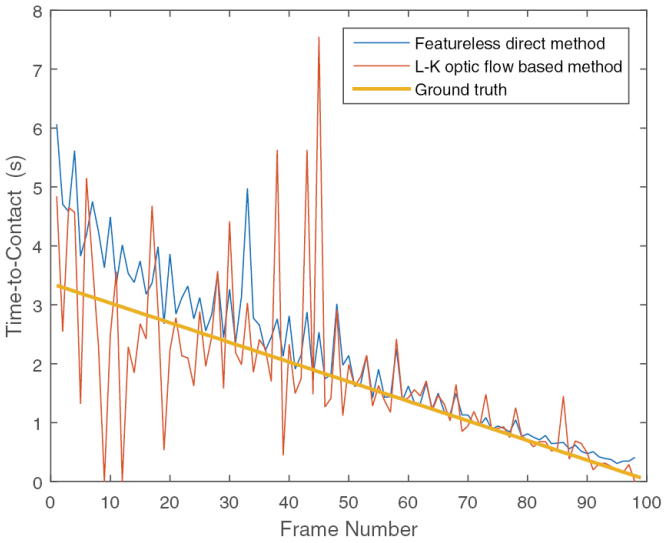

Figure 2.5. Comparison between LK (Orange) and HS (Blue) for estimation of TTC [5].

2.1.2 Algorithm Derivation

In order to fully understand the derivation of the Horn and Schunck method we first mathematically describe the aperture problem, then in order to explain how the visual cues will be calculated the derivation will follow the work of Horn *et al* [35]. In order to extract information from image sequences the coordinates of the pixels must be explored. For example, take an image with respect to a stationary surface with point P on it at

time t. Then if the camera translates, point P is expected to have moved by Δu,Δv, and Δt, where u_p and v_p denote the pixel coordinates in units of pixels with the origin being in the Center of the Projection (COP) and I denotes the intensity of the pixel at a certain location. The equation describing which can be seen in 2.3, is known as the brightness constancy constraint, which assumes that between the two images the brightness of the objects, and therefore pixels, remains the same. Which when generalized for all pixels within the image coordinate frame can be expressed with a first order Taylor series expansion as Equation 2.4. Then by dividing by Δt and simplifying the final result yields equation 2.5 which is the previously mentioned aperture problem, with ϑ_{u_p} and ϑ_{v_p} denoting the optical flow vectors in pixels per second. The aperture problem is ill posed because it cannot be solved without another equation, which in the case of all optical flow methods comes from some additional constraints.

$$I_P(u, v, t) = I(u_p + \Delta u_p, v_p + \Delta v_p, t + \Delta t) \tag{2.3}$$

$$I(u + \Delta u, v + \Delta v, t + \Delta t) = I(u_p, v_p, t) + \frac{\partial I}{\partial u_p}\Delta u_p + \frac{\partial I}{\partial v_p}\Delta v_p + \frac{\partial I}{\partial t}\Delta t \tag{2.4}$$

$$\frac{\partial I}{\partial u_p}\vartheta_{u_p} + \frac{\partial I}{\partial v_p}\vartheta_{v_p} + \frac{\partial I}{\partial t} = 0 \tag{2.5}$$

As mentioned above the H-S method solves the aperture problem by including a global smoothness constraint. In order to show the implications of this constraint a better picture of the previous example combined with the initial phase of the MAV's landing sequence can be seen in Figure 2.6. The previously described example of a stationary point projected onto the image plane can be seen, where f is the focal length of the camera lens in meters. Now consider a camera with both translational and angular velocities (ω). If it is then assumed that the camera frame and corresponding COP remain perpendicular to the ceiling plane equations 2.6 and 2.7 can be obtained. In which ϑ_{u_p} and ϑ_{v_p} denote the optical flow vectors in pixels per second in their respective directions, and are identical to the OF_x and OF_y values used in the previous work [3]. While x_c and V_c denotes the position and velocity of the camera and lastly, u_p and v_p are the pixel index of the $m \times n$ image. The limitations of this assumption will be discussed later, but this allows the following derivation to be simplified greatly (for the full derivation see [39] and explanation of why this assumption is made see [35]).

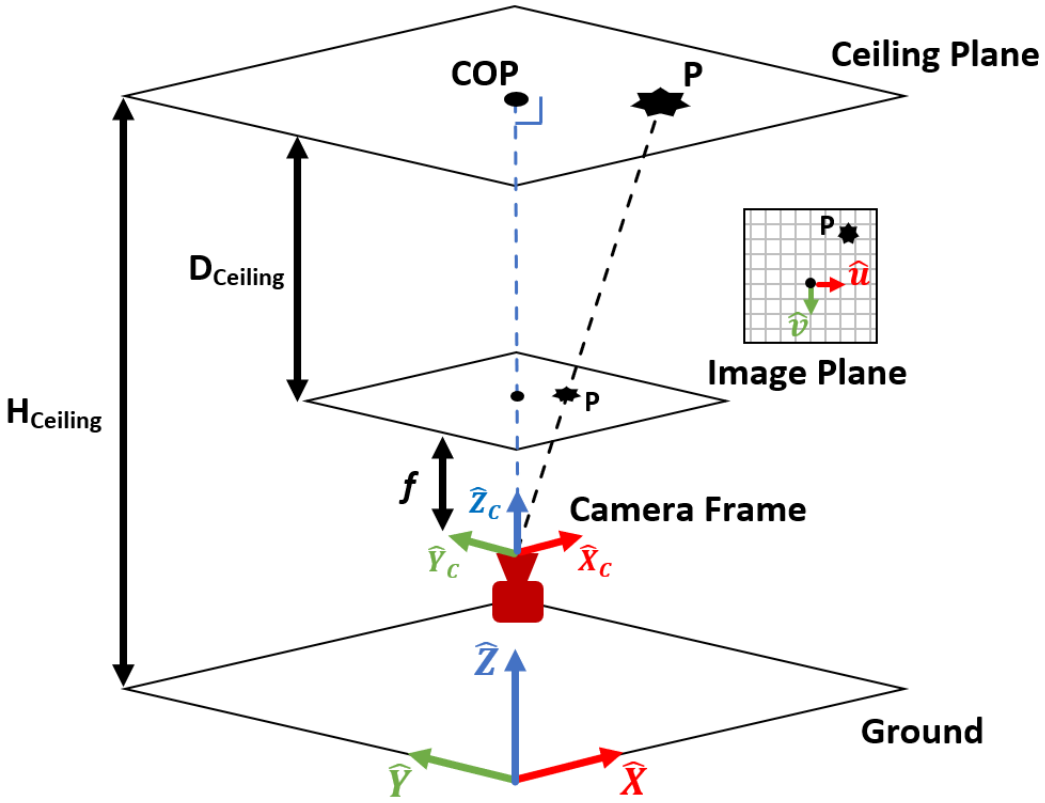

Figure 2.6. Perspective projection model of a pinhole camera without distortions. With Corresponding image visualized on the right.

$$\vartheta_{u_p} = -f\frac{V_{cy}}{x_c} + u_p\frac{V_{cx}}{x_c} + \omega_{c_y}\frac{u_p v_p}{f} - \omega_{c_z}\frac{f^2+u_p^2}{f} + v_p\omega_{c_x} \tag{2.6}$$

$$\vartheta_{v_p} = -f\frac{V_{cy}}{x_c} + u_p\frac{V_{cx}}{x_c} + \omega_{c_y}\frac{u_p v_p}{f} - \omega_{c_z}\frac{f^2+u_p^2}{f} + v_p\omega_{c_x} \tag{2.7}$$

Depending on how the system moves towards the ceiling plane determines what further simplifications can be applied to the above equations. In continuation of the previous work by Habas *et al*, the typical landing sequence of the system approaches the ceiling plane with different components of translational velocity (V_{c_x}, V_{c_y}, V_{c_z}), while minimizing rotational velocities (ω_{c_x}, ω_{c_y}, ω_{c_z}), before the flip maneuver occurs. Thus, we will assume those same parameters to simplify and solve equations 2.6 and 2.7. Assuming negligible rotational velocity and combining equations 2.6 and 2.7 with the brightness constancy equation 2.3, while defining the radial gradient G in equation 2.8, we obtain a equation 2.9 shown below. To simplify the equations $\sum$ is used in place of $\sum_{u_p=1}^{m}\sum_{v_p=1}^{n}$.

$$G = (\frac{\partial I}{\partial u_p} u_p + \frac{\partial I}{\partial v_p} v_p) \tag{2.8}$$

$$-\frac{\partial I}{\partial t} = f\frac{\partial I}{\partial u_p}(\vartheta_{u_p}) + f\frac{\partial I}{\partial v_p}(\vartheta_{v_p}) + (\frac{\partial I}{\partial u_p} u_p + \frac{\partial I}{\partial v_p} v_p)(\frac{1}{\tau}) \tag{2.9}$$

$$\sum (\frac{\partial I}{\partial t} + f\frac{\partial I}{\partial u_p} v_u + f\frac{\partial I}{\partial v} v_u + \frac{G}{\tau})^2 \tag{2.10}$$

Finally we can then use the least squares optimization method to minimize equation 2.10. Ultimately, giving the following system of equations seen in 2.11. This system of equations contains brightness gradients in the spatial dimension on the left-hand side, with temporal gradients on the right-hand side, all of which are then accumulated over the range of the entire image.

$$\begin{bmatrix} f\sum \frac{\partial I}{\partial u_p}^2 & f\sum \frac{\partial I}{\partial u_p}\frac{\partial I}{\partial v_p} & \sum G\frac{\partial I}{\partial u_p} \\ f\sum \frac{\partial I}{\partial u_p}\frac{\partial I}{\partial v_p} & f\sum \frac{\partial I}{\partial v_p}^2 & \sum G\frac{\partial I}{\partial v_p} \\ f\sum \frac{\partial I}{\partial u_p} G & f\sum \frac{\partial I}{\partial v_p} G & \sum G^2 \end{bmatrix} \begin{bmatrix} \vartheta_{u_p} \\ \vartheta_{v_p} \\ 1/\tau \end{bmatrix} = \begin{bmatrix} -\sum \frac{\partial I}{\partial t}\frac{\partial I}{\partial u_p} \\ -\sum \frac{\partial I}{\partial t}\frac{\partial I}{\partial v_p} \\ -\sum \frac{\partial I}{\partial t} G \end{bmatrix} \tag{2.11}$$

In order to approximate the image gradients a 3×3 normalized Sobel kernel is utilized as shown in Figure 2.7 and Equation 2.12. The discrete convolution process can approximate the value of the partial derivatives seen in Equation 2.11 with the addition of the 1/8 weighting factor. Specifically for our implementation, we cropped the outer edge of the image data in order to increase the efficiency of the algorithm since it no longer needed to use edge detection methods to determine if the kernel is over the bounds of the $m \times n$ image.

$$K_u = 1/8 \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, K_v = 1/8 \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.12}$$

The system of equations can then be solved with numerous linear systems solvers but we went with QR decomposition. While compared to LU decomposition it is objectively slower, however because the final solution is a square 3×3 matrix the extra time is worth the numerical stability due to the reduced condition number that QR decomposition holds.

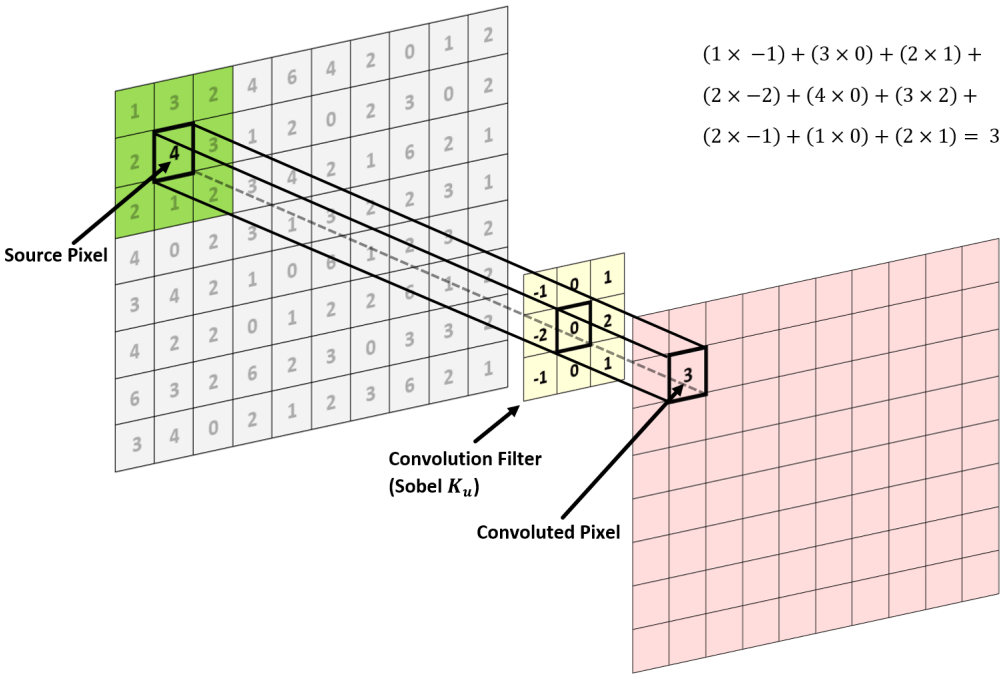

Figure 2.7. An example of the convolution process for one pixel.

Through all of these derivations it is paramount that consistent and proper units are used. Because this method utilizes the COP as the origin of the coordinate frame Equations 2.14 and 2.15 are used to scale and shift the origin from the top left corner (which is the standard origin for images) with units of pixels, to the center of the image with units of meters. An example of which can be seen in Figure 2.8 where x_p and y_p denote the pixel index in their respective directions for standard image coordinate frames. While W represents the width of a pixel in meters. O represents the offset from the image plane origin which is equal to half of each respective component's value as shown in 2.13. To not confuse the notation the lowercase values will be in units of pixels, while uppercase denotes meters. The following equations will then be applied using the camera parameters listed in the following section to ensure that the proper units are used when determining τ and OF.

$$O_x = m/2 \ , \ O_y = n/2 \tag{2.13}$$

$$U = (x_p - O_x)W + W/2 \tag{2.14}$$

$$V = (y_p - O_y)W + W/2 \tag{2.15}$$

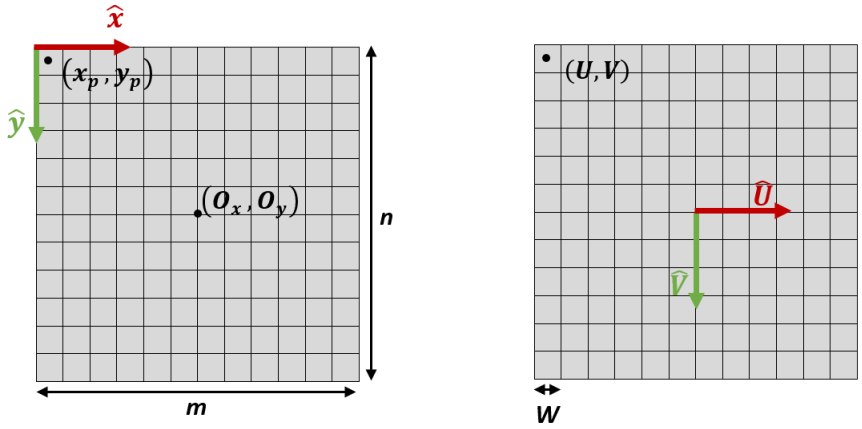

Figure 2.8. Image coordinate transform from pixels to meters.

2.2 Simulation Setup

The experiments of this thesis will be run solely in a simulation environment. While experimental setups are certain to introduce unforeseen drawbacks, the previous work done by Habas *et al* has shown that the simulation results translate to experiments with reasonable accuracy, as shown in Figure 2.9. Further, our recent focus has been on estimating the most accurate model parameters possible. This includes more accurate moment of inertia values for all axes, the proper mass of the system with new motors, all for each respective set of landing gears. Also more accurate sensor models, from proper battery compensation to measuring sensor noise in order to make sure each sensor matches what we observe in the lab.

A simulation environment allows for rapid testing to be done over a larger parameter space than would not be feasible with physical experiments. The simulations were run in Gazebo 11 with the Robot Operating System (ROS) Noetic package. Within the simulation we have modeled the Crazyflie 2.1 with some slight modifications as previously mentioned. The specific system setup used for the following tests utilized the Narrow-Long landing gear due to it having the most robust landing success over the tested parameter space [3]. Additionally, the systems sensors are modeled after the onboard sensors of the Bitcraze AI-deck, specifically the Himax HM01B0 gray-scale

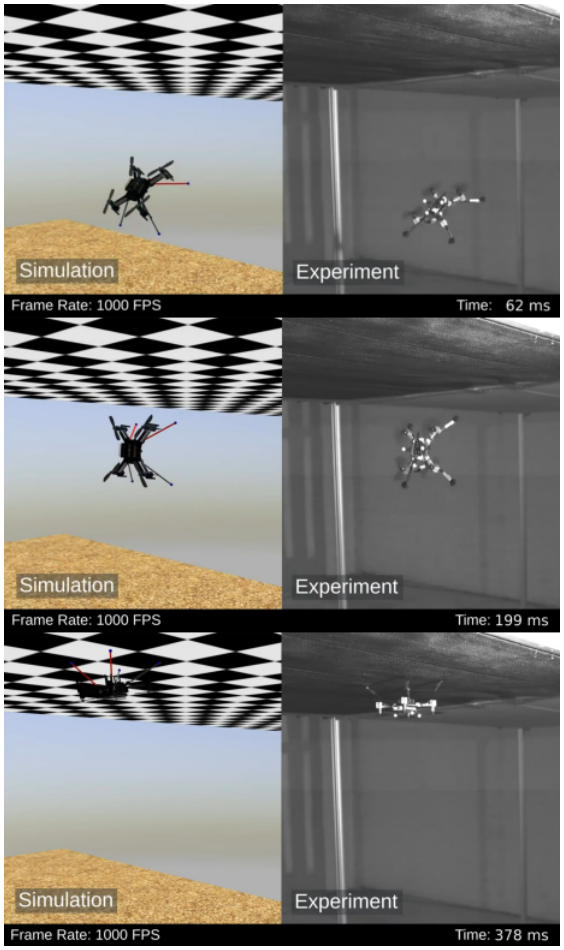

Figure 2.9. Side by side comparison of the MAV in simulation and laboratory setup [3].

camera, the important values of which can be found in Table 2.1. Since we do not have a good measure of the noise the camera generates we modeled the noise as Gaussian with a standard deviation of 0.005.

Table 2.1. Himax HM01B0 Parameters

Focal Length (f)	3.3×10^{-4} m
Pixel Array ($m \times n$)	160×160
Pixel Size (W)	3.6×10^{-6} m

Chapter 3
Simulation Experiments

3.1 Experimental Setup

Before discussing the experiments run it must be noted that due to the vast parameter space to test over the following experiments were conducted in a specific order in order to systematically determine certain variables.

Due to the high maneuverability of the MAV, and therefore high variability of potential experiments, the experiments in the following section were conducted in a specific order to understand how each variable tested impacted the performance of the estimated visual cues. The variables that were explored were modifying the ceiling pattern and filtering image data, modifying the frame rate of the camera, and lastly sub-sampling the image data. The default frame rate of the camera was set to be 50 frames per second. The effective area the partial derivatives were calculated in was 158×158 due to the aforementioned cropping to remove edge detection as well as modeling the Himax camera's ability to utilize a 2×2 binning mode that reduces the image data by half. An Exponential Moving Average (EMA) filter was applied to the TTC estimate; the equation for which can be seen in 3.1. Where, S_t represents the output of the filter at time t, and alpha represents a weighting factor, the higher the value the quicker it rejects previous data.

$$S_t = \begin{cases} \tau_0 & , t = 0 \\ \alpha\tau_t + (1-\alpha)\tau_{t-1} & , t > 0 \ ; \ \alpha \in [0,1] \end{cases} \tag{3.1}$$

Lastly, the first two experiments follow identical flight parameters with $V_z = 3m/s$ while keeping all other velocities zero. This upward velocity was chosen for two reasons. First, the landing success rate for the entirety of the past experiments utilizing external

positioning was almost 100% for all landing gear designs and flight angles [3]. Second, because the H-S method prefers solutions that show more smoothness, the high vertical velocity tests the limits of this method by having more movement between frames. Thus, allowing for room to experiment and determine appropriate flight parameters for the complications that arise from moving away from the motion capture system previously used.

3.2 Effects of the Ceiling Pattern

The first problem to be tested comes from determining what ceiling pattern should be used. Because digital cameras have to quantize optical information the collected data will have quantization error. Additionally, the fixed focal length camera will inevitably suffer from blurring due to the objects being out of focus. In order to solve this problem, high contrast images, like checkerboards, are utilized to give lots of information in regards to brightness contrast. In a controlled laboratory setting our system operates approximately two meters from the ceiling until contact with it. Due to the wide operating range of the modeled system it introduces the problem of feature scaling. For example, large features would increase the accuracy when the MAV is further away. However, it then removes information when the MAV approaches the ceiling. Especially if the features are too large and it only has one color within the field of view during it's final touchdown phase as shown in Figure 3.1. To alleviate this drawback filtering incoming image data can not only reduce noise, but also smooth out the high contrast markers in the landing area.However, we have found that with camera noise and highly varied flight parameters this process can be side stepped by not using high contrast images, instead utilizing ones that smoothly transition between different intensities. All while maintaining acceptable levels of error. An example of which can be shown in Figures 3.2 and 3.3. In order to better compare the following experiments we set a baseline goal of keeping the error below 0.05 when $\tau < 1s$, which can be seen in the following examples between the dashed black lines. For each ceiling pattern the system was flown directly towards the ceiling with $V_z = 3m/s$. The software then saved the images for each flight in order to compare the effects of raw versus Gaussian smoothed data with respect to each ceiling pattern shown below. For the checkerboard ceiling pattern it is apparent that not only is the feature size not scaled small enough, hence the error rising as the MAV gets closer to the ceiling, but in order to obtain accurate results the images must be filtered as shown in green.

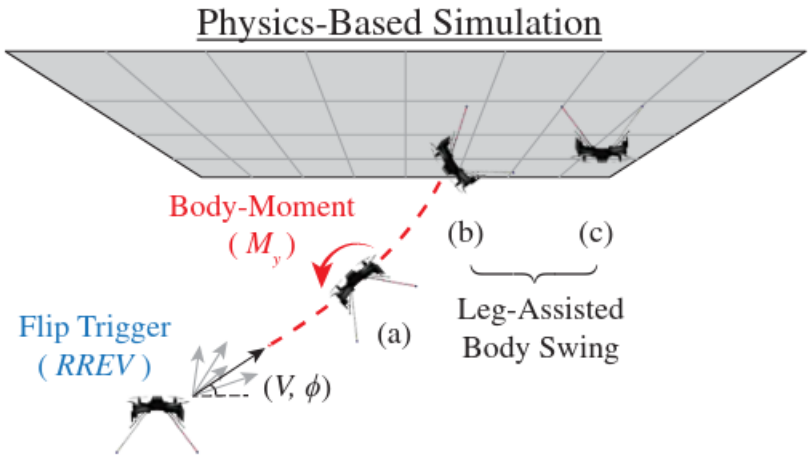

Figure 3.1. Final phase of the landing sequence the system follows [3].

Compared to the smooth ceiling pattern as shown in Figure 3.3, the error in the estimates only changes marginally when the image data is filtered. Another reason the smooth gradient ceiling pattern is preferred is because eventually this system will be put to the test in a laboratory setting. In order to make a VELCRO™ checkerboard ceiling pattern the squares are meticulously cut out making it difficult to create. Further, if the size of the features are not right, then the whole process must be repeated again until satisfactory results are obtained. Instead, it would be easiest to have one color put on the ceiling then the other spray painted in order to emulate the smooth gradient pattern depicted in Figure 3.3.

The motivation behind not smoothing the images is also done to save computation time. In order to filter image data, it is typically smoothed with a Gaussian kernel. However, because the brightness gradients are approximated with the same convolution method filtering image data would nearly double computation time. Doubling the time it takes to compute is a drawback that cannot be afforded since the system generally executes it's flip maneuver around 250 milliseconds before contact. Further, is has been shown that excessive image smoothing, while helping with reducing noise, introduces other errors, notably in the velocity estimates [40], which it also cannot be afforded as it would further reduce landing robustness.

Now that it is apparent that a smoothed ceiling pattern, as opposed to a high contrast one, should be used we then modeled the ceiling pattern with a sine function. Because the ceiling pattern can now be iterated over by simply changing the periodicity of the peaks a more uniform study can now be conducted. The equations for how the ceiling pattern was generated is shown below. Equation 3.2 was adapted from the work of

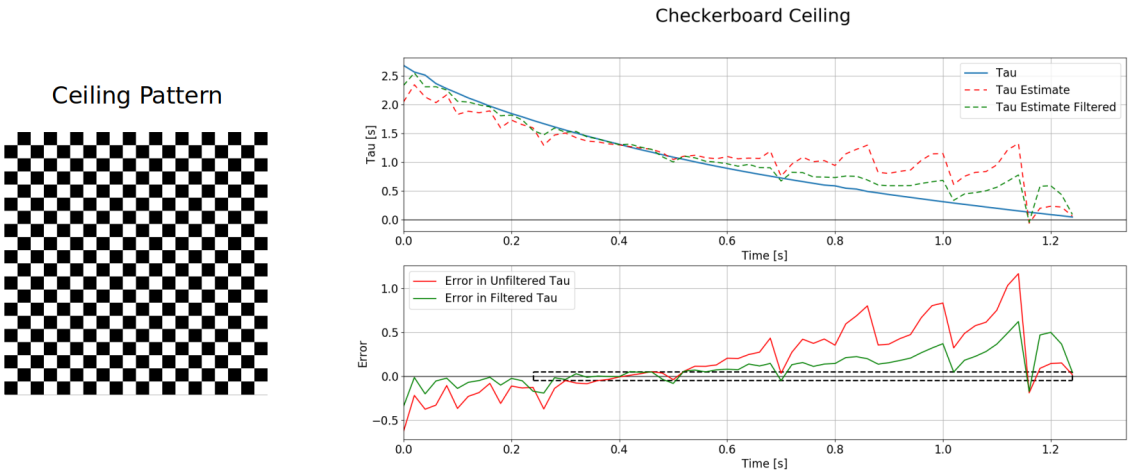

Figure 3.2. Comparison of Gaussian filtered image data and unfiltered images with the high contrast ceiling pattern. Blue is the ground truth, red represents the unfiltered estimates, while green represents the Gaussian smoothed data.

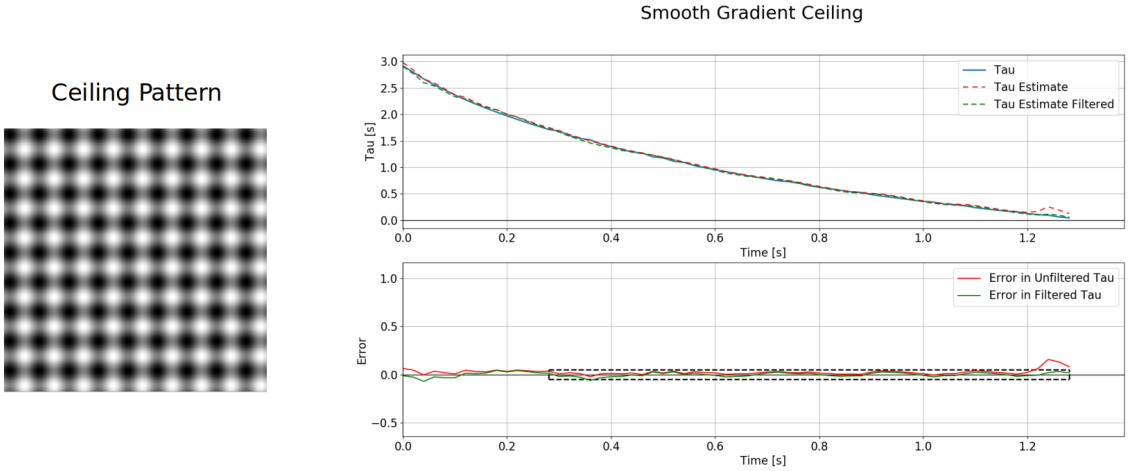

Figure 3.3. Comparison of Gaussian filtered images and unfiltered images with the smooth gradient ceiling pattern. Blue is the ground truth, red represents the unfiltered estimates, while green represents the Gaussian smoothed data

Chirarattananon and further extensions of the equation were made as shown in Equations 3.3 and 3.4 [41]. I_0 denotes the maximum brightness of the image, which in the case of the Himax camera is 8-bit or 255. X and Y is simply a mesh grid of points over which the image is generated. Thus, making L the period of the sine waves and is what will be changed in the following experiments to determine what feature size works best for the system. Ultimately, I is the final image and examples of various periods L can be seen in Figure 3.4.

$$I_x = \frac{I_0}{2}(\sin(2\pi X/L) + 1) \tag{3.2}$$

$$I_y = \frac{I_0}{2}(\sin(2\pi Y/L) + 1) \tag{3.3}$$

$$I = \frac{I_x + I_y}{2} \tag{3.4}$$

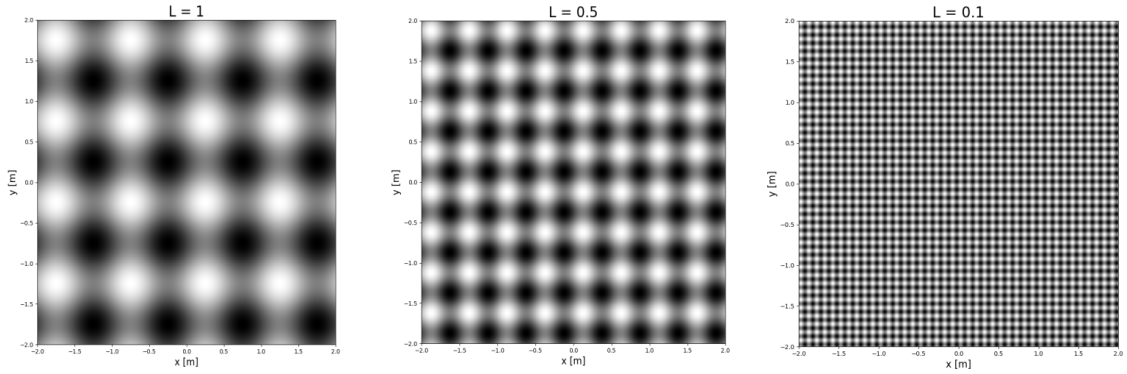

Figure 3.4. Comparison of different periods L used to generate the ceiling patterns

The previous Figure 3.4 shows examples of the range that was tested over, first the exact values depicted above ($L = 1$, $L = 0.5$, and $L = 0.1$) were tested and the results of which can be seen in Figures 3.5-3.7.

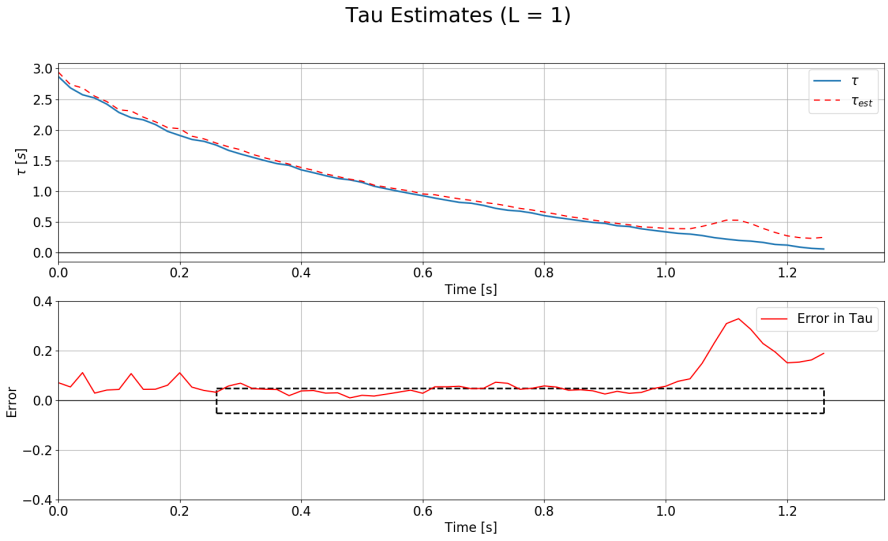

Figure 3.5. Flight test showing the estimated values compared to ground truth with $L = 1$.

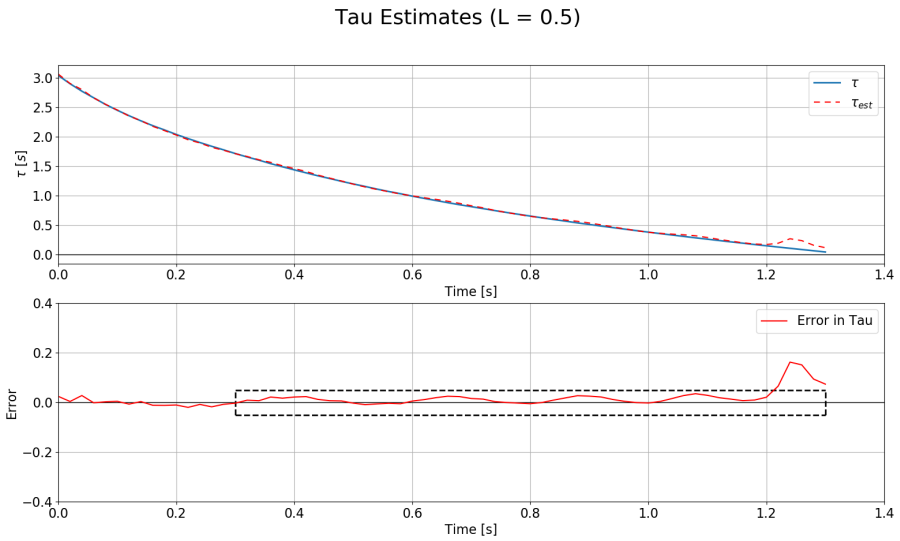

Figure 3.6. Flight test showing the estimated values compared to ground truth with $L = 0.5$.

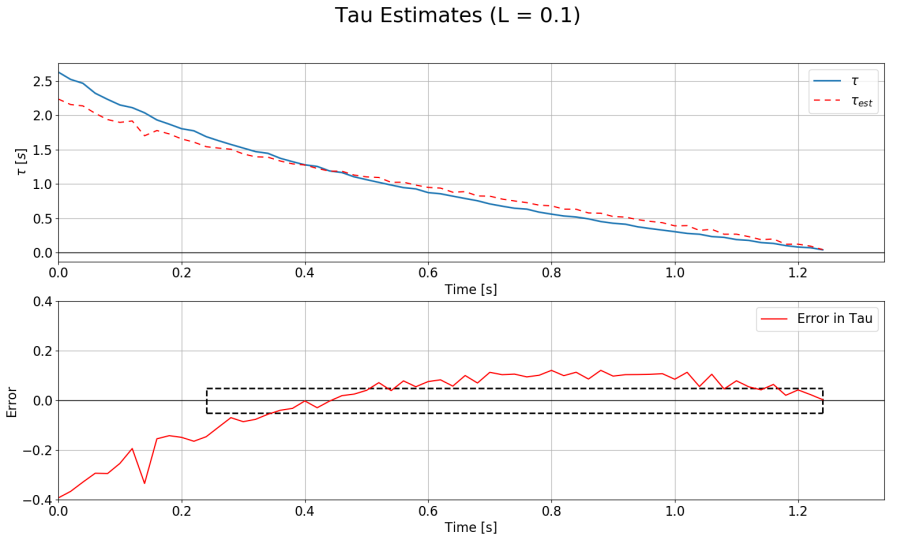

Figure 3.7. Flight test showing the estimated values compared to ground truth with $L = 0.1$.

These results were then used to narrow down the selection of L, the upper end of the range was removed due to it's constant offset from the ground truth as shown in blue, as well as the error steadily rising when the system approaches contact with the ceiling. The lower range of L was slowly iterated through in increments of 0.05 eventually landing on $L = 0.25$ with a total of 15 ceiling patterns tested. When L was increased or decreased the error in the estimate increased, the results of which can be seen in Figure 3.8. Therefore, in the following experiments the ceiling pattern will be kept at $L = 0.25$

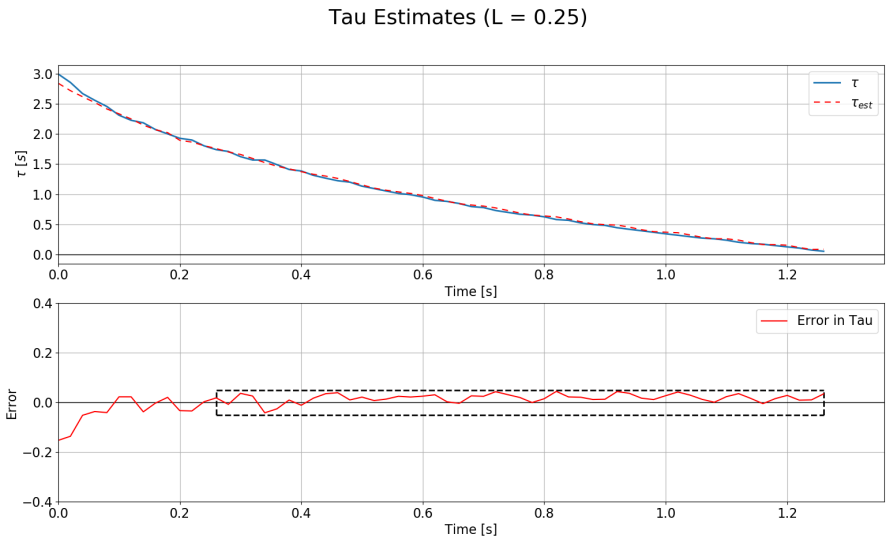

Figure 3.8. Final flight test showcasing the estimated values compared to ground truth for $L = 0.25$.

3.3 Effects of Varying Frame Rate

The next variable to explore is the Frames Per Second (FPS) of the camera. Because the HS method prefers solutions that show more smoothness, a higher frame rate allows the camera to capture less movement between frames, which in turn will increase the accuracy of the predicted values. In order to conduct this experiment and directly compare the effects of varying the frame rate the MAV was flown at the ceiling with the same upward velocity of $V_z = 3m/s$, and the ceiling pattern was constant with $L = 0.25$, with the only variable changing between tests being the FPS. As show in Figure 3.9 when the frame rate is lowered to 25 the accuracy decreases as expected. The last step was to assess how much the accuracy improves when the frame rate is set to 100 FPS in comparison to the baseline 50 FPS as seen in Figure 3.10.

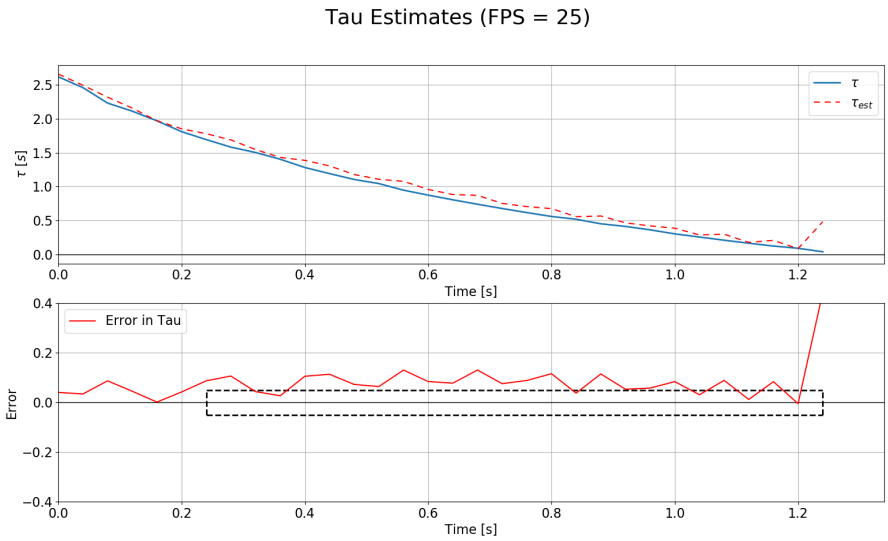

Figure 3.9. Flight test with FPS set to 25, $V_z = 3m/s$ and $L = 0.25$.

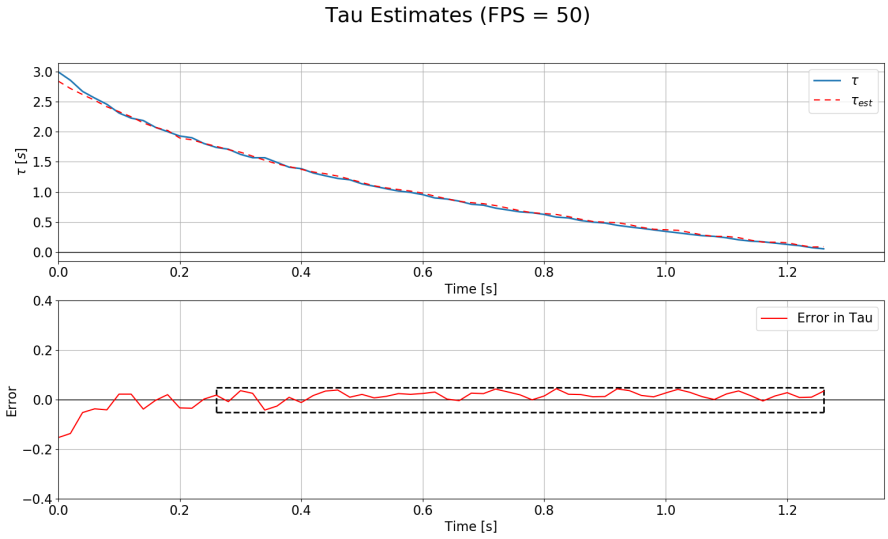

Figure 3.10. Flight test with FPS = 50, $V_z = 3m/s$, and $L = 0.25$

When the FPS is set to 100 the error rarely crosses the baseline error threshold when the quad-rotor is less than two seconds from contact as shown in Figure 3.11. When compared to the previous work the camera refresh rate matches the 100 Hz refresh rate of the motion capture system [3]. While these results are promising if the most accurate estimates are desired, then following the trend of increasing the FPS is the obvious easy choice. However, one must not forget that even if the camera in use can capture images at

higher frame rates the algorithm behind the scenes must be able to keep up and calculate the motion cues before the next frame is received. One option to achieve higher refresh rates comes from the optimization of the implemented featureless algorithm, however because it is already a lightweight algorithm not many improvements can be made to increase efficiency. Therefore, the following experiment will explore whether it is possible to decrease the computation time by only accumulating over a certain portion of the image data, while maintaining acceptable limits of accuracy.

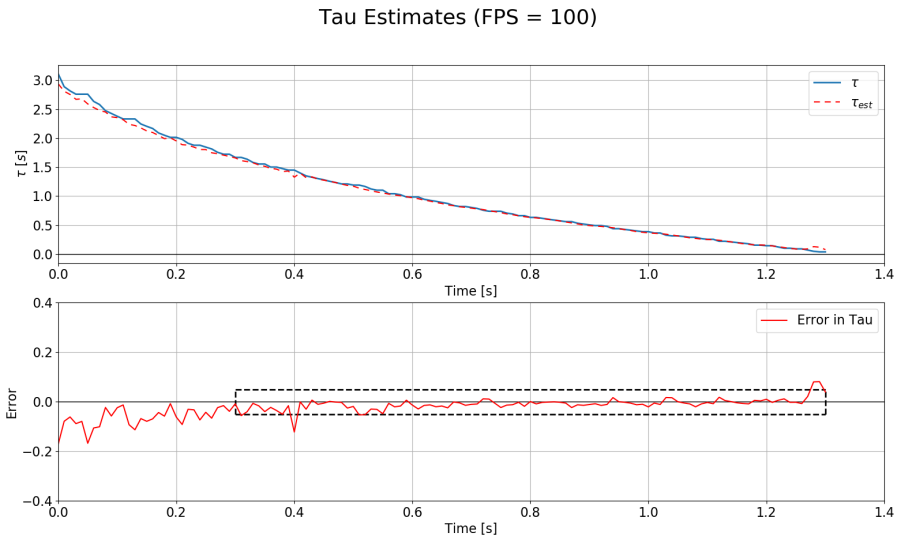

Figure 3.11. Flight test with FPS = 100, $V_z = 3m/s$, and $L = 0.25$.

3.4 Effects of Sub-Sampling Image Data

Although there are many benefits that come from running these experiments in simulation, like being able to test over 15 ceiling patterns in a reasonable amount of time, there comes drawbacks as well. In this case the following experiments will focus on the accuracy of the algorithm while varying the total amount of the image used, instead of the computation time. Because the simulation is run on a powerful desktop PC measuring the time it takes to process images will not directly translate between different hardware setups. Instead, by focusing on the accuracy of the model then the following experiments will more easily be transferred between different hardware. First, by determining how fast the featureless algorithm runs when utilizing all of the image data, then by determining the acceptable balance between the time to calculate and deviation from the ground truth for the specific hardware utilized. The following experiments were done on the same

image sequence that was captured as the MAV traveled to the ceiling with $V_z = 3m/s$ and $L = 0.25$. The image data was kept identical to directly compare the differences of using varying amounts of image data.

In order to remove data from the image the convolution process that approximates the image derivatives will be moved by larger increments. Standard convolution which utilizes all of the image data the convolution kernel will be shifted by one pixel in the $\hat{U}$ direction until it reaches the edge of the image. When the edge of the kernel reaches the edge of the image the kernel will then be shifted back to the opposite end where it is then moved by one pixel in the $\hat{V}$ direction. The following process repeats until the entire image is shifted over. Therefore, the following experiments will vary how much each direction translates, also known as the stride N, in order to reduce the amount of data used. While there are many ways to utilize different amounts of image data, varying the stride in $\hat{U}$ and $\hat{V}$ was chosen over cropping the image. Because, by removing data at the edges of the image the sensitivity decreases due to the importance that the radial gradient G places on changes at the edge of the image. In total, reducing the amount of image data used should have a linear relationship with respect to processing time; meaning that the lower the amount of data used the faster the time to calculate. The linear relationship arises from the deterministic nature of the implemented algorithm and it's reliance on basic operations that generally have fixed computation times. By changing the stride the reduction in calculations compared to the standard convolution over a square $M \times M$ image can be calculated using equation 3.5. To ensure a uniform distribution of data points across the image is sampled the stride is kept equivalent in both $\hat{U}$ and $\hat{V}$ directions. Examples of the strides used in the following experiment and the corresponding computational cost reduction compared to the full image convolution can be seen in table 3.1.

$$\left[\frac{(M-2)^2}{\lceil (M-2)/N \rceil^2}\right] \times 100 \tag{3.5}$$

Table 3.1. Computational cost for Varying strides (N) for a 158×158 image with 3×3 kernel.

Stride (N)	Computational cost (%)
1	100 %
2	50 %
4	6.409 %
8	1.602 %
16	0.401 %
32	0.100 %

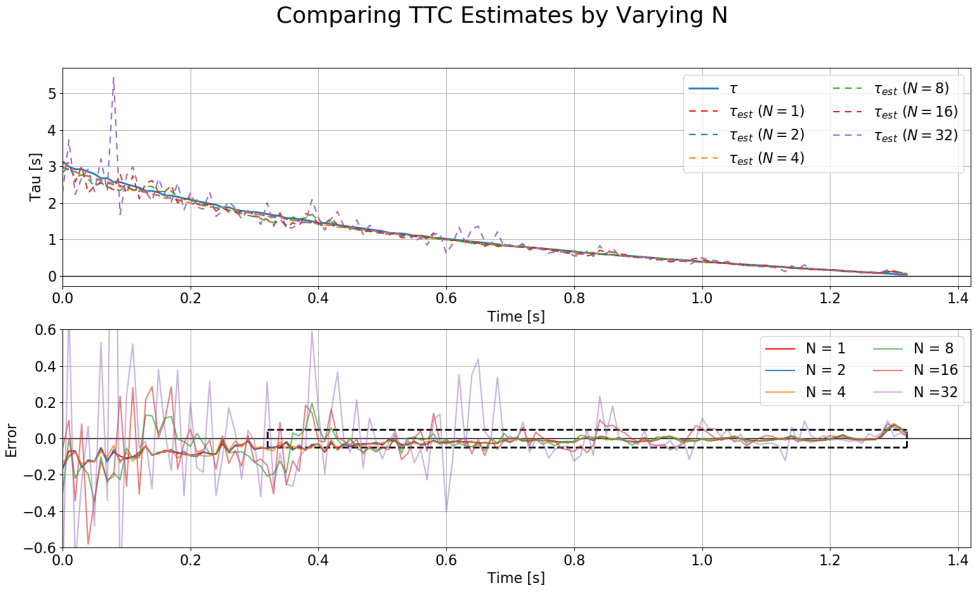

Figure 3.12. Error observed by varying the stride (N) in order to reduce the amount of the image data used.

As shown from the above results the error negligibly changes when removing significant amounts of image data. These results at first seem counter intuitive, however, while reducing the number of convolution processes, and therefore spacing the kernels further apart, the kernel still extends into the areas where the image information is shifted over. Meaning, the average space skipped by removing percentages of the image data does not exceed the dimensions of the 3×3 kernel utilized until approximately 16% of the image data is used. Therefore, removing data does not have a large impact on the estimation of TTC for the case of strictly vertical flight. In comparison similar work done using the same featureless algorithm shows that utilizing 6.25% of the image data yields results

that allow for accurate state estimation of a quad-copter [41]. However, further studies must be done in order to confirm that the results shown above translate to experimental settings.

3.5 Learning Inverted Landing Motor Control Using Reinforcement Learning

After determining how the ceiling pattern, frame rate, and amount of image data affect the performance of the featureless algorithm the results were then synthesized and the best from each variable that was explored was implemented for a final test. The ceiling pattern of $L = 0.25$, frame rate of 100, and the stride $N = 1$ were used for the last experiment. The ultimate goal of this work was to remove the dependence of external positioning and in order to do so the featureless algorithm was tested using the same RL algorithm that was used previously [3]. The RL algorithm used is a parameter optimization algorithm which samples values of τ and M_y to find an optimal set of parameters that yields a successful inverted landing. The RL algorithm then receives a reward value based off the landing performance which is then used to converge onto an optimal set of parameters. For example, if the MAV missed the ceiling the τ threshold (τ_{cr}) was decreased, and if the moment did not yield a four legged landing it was also varied. The following process is then repeated multiple times over a certain number episodes until the values of τ_{cr} and M_y converged for the specific flight conditions. The same flight conditions that were tested prior to this were also implemented in the following experiment the results from which can be seen in Figure 3.13.

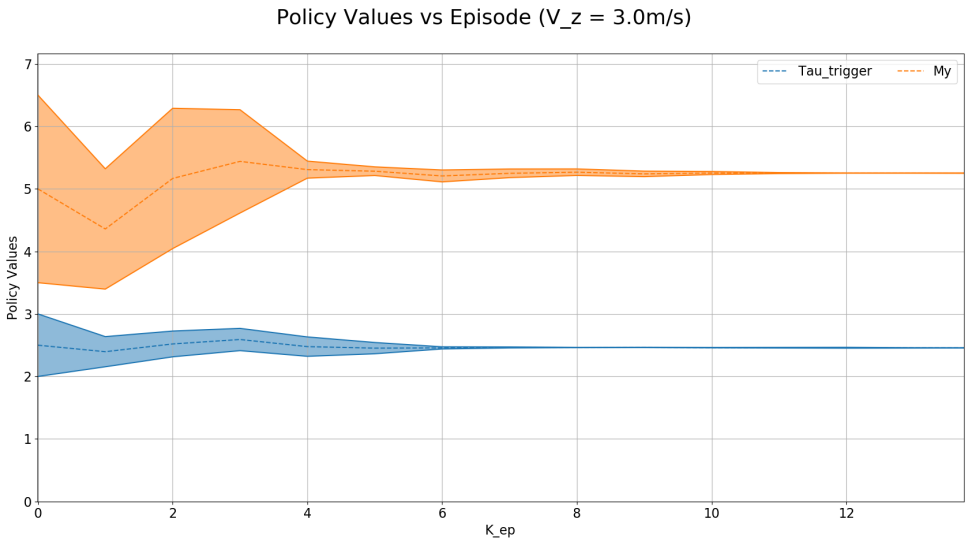

Figure 3.13. Policy convergence plot showcasing a successful learned policy. The dashed blue line indicates the specific value of τ that was used to trigger a flip maneuver, while the dashed orange line depicts the corresponding moment (M_y) used to flip the MAV. The shaded region represents the confidence in the current estimate of the respective parameter with $\pm 2\sigma$.

Because the camera algorithm introduces noise no matter how confident the RL algorithm is with the policy values the landing robustness can still suffer from early triggering of the flip maneuver. Hence the reasoning for the baseline goal of maintaining an error below 0.05 when $\tau < 1s$. The results from the learned policy above can be seen in Figure 3.14. The highest reward is given to a successful four legged landing, with the second highest being a two legged landing. Lastly the reward was lowered if during the flip maneuver the body or rotors of the MAV contacted the ceiling plane.

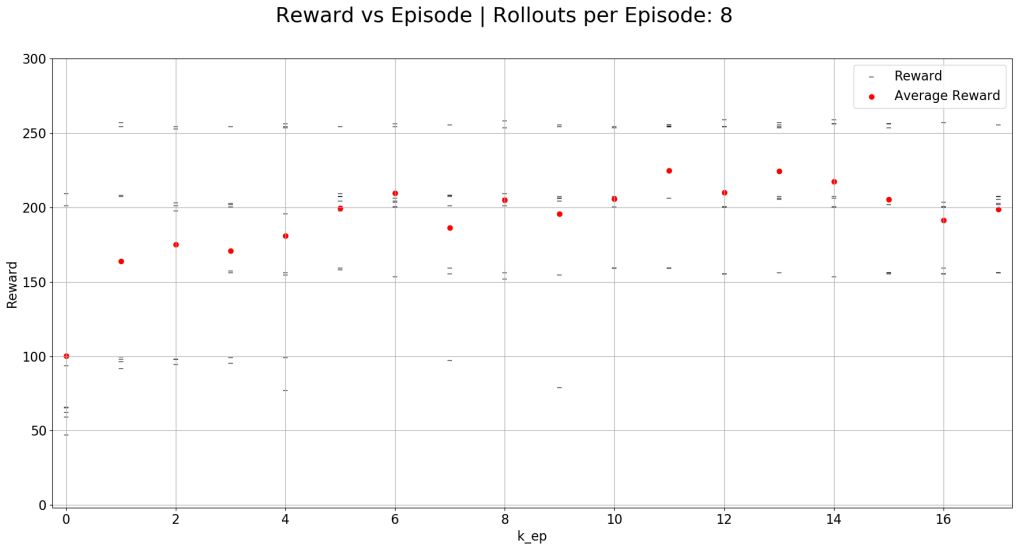

Figure 3.14. The reward data for the first 17 episodes. The reward shown in black indicates the performance of the flip maneuver, the highest value indicates a four legged landing. The red indicates the average reward for the corresponding episode.

Chapter 4
Conclusions

4.1 Summary

This thesis applies the H-S method to estimate bio-inspired visual cues for an inverted landing strategy. It first examines the effects of image processing parameters on the estimation accuracy and then applies the H-S method in learning the landing policy with the selected parameters. Additionally, the goal of the experiments were to determine how these image processing parameters affect the accuracy of the implemented algorithm. In these studies three main variables were explored, the ceiling pattern, the frame rate, and lastly varying the stride of the kernel. First, the most abstract problem was explored, determining what ceiling pattern yielded the best results; the findings from which showed that $L = 0.25$ was the ideal pattern. The following experiment determined that increasing the frame rate also increases accuracy. However, regardless of how promising these results may be there are hardware limitations that prevent exceptionally large frame rates, hence the frame rate was kept at 100 FPS. While higher frame rates could be achieved with advanced algorithm implementations, the hardware limitations of the MAV make it infeasible. Therefore, the following experiment focused on reducing the amount of image data that was used in order to decrease processing time. The findings from which determined that the implemented algorithm did not suffer from reducing the amount of image data used until significant percentages of image data was excluded. Further investigation and verification is required in order to determine why there is not a significant degradation of accuracy as the image data decreases. Ultimately, the main findings from each study were then applied to the RL algorithm in order to determine the landing robustness in comparison to previous experiments done by Habas *et al.*

The policy the RL algorithm converged on has a landing success rate of 23.7% for four legged landings. Even though the landing robustness is significantly lower compared

to external positioning, the entire parameter space still remains unexplored with the featureless algorithm implemented. Initial observations of the performance conclude that the poor performance of the landing rate is due to the offset in the estimated visual cues which trigger an early flip. However, the landing rate for two legged landings was 21.8%. Although these are considered failed landings, because they cannot be corrected, the increased rate of early or late triggering due to the noisy output from the featureless algorithm may be compensated with mechanical intelligence. The mechanical intelligence for the MAV is also inspired from the blue bottle flies. Although the static landing gear cannot be actuated like the flies' legs the thin plastic allows them to deflect to absorb impacts from the high velocity approach. Therefore, further investigations using multiple sets of landing gear may aid in reducing the number of two legged landings by leveraging more on the mechanical intelligence of the varying designs. Further, because the MAV is highly mobile flight angles can vary anywhere from 20° to 90°. Meaning that while the system successfully converged with a vertical velocity of 3 m/s and a flight angle of 90°, extending the current algorithm beyond this proves difficult. The reason the algorithm struggles with extending the tested parameter space is due to the assumption that the COP remains perpendicular to the ceiling plane. Due to the underactuated nature of the MAV the body must rotate in order to translate with varying horizontal velocities. If the system's body angle changes then so does the apparent angle of the ceiling. Therefore, in order to extend past strictly near vertical landings further extensions of the implemented algorithm must be made. Horn and Schunck have made several updates to the implemented algorithm as seen in [35]; in which they have derived a method for determining τ for arbitrary ceiling orientations. However, when removing the assumption that the landing area is perpendicular to the COP the derivation no longer provides closed form solutions and instead relies on an iterative solution method. Because the derivation no longer yields closed form solutions, this leads to a variable calculation time along with increased computational load, as well as irregular accuracy; which our system cannot afford. Therefore, other methods in order to solve this problem will have to be explored in future work.

4.2 Future Work

All the previous experiments were conducted in a linear fashion in order to isolate specific variables that were identified to contribute to the accuracy of the featureless algorithm. However, there may be potential relationships between the variables that

were tested and fully understanding these connections will aid future work in creating an accurate understanding of the underlying methods and how they vary with respect to the goal of inverted landing. Additional studies into the relationship between removing image data and how the kernel translates, as well as varying the kernel size and the effects these variables have on performance will aid future hardware implementations. Further experimentation with the hardware modeled in this study will continue to aid it's accuracy, notably determining the approximate value for the standard deviation of the camera noise, as well as other unknown factors that were not modeled that might effect the performance. Potentially utilizing other onboard sensors, like the Inertial Measurement Unit (IMU), in order to determine the angle of the quad-copter may be used to estimate the corresponding angle of the ceiling assuming it is horizontal. Possible solutions to the issues that arise due to the under-actuated MAV could take inspiration from flying animals robust head stabilization techniques in order to keep the camera in a desired orientation. However, due to the small scale nature of the MAV, as well as the strict carrying capacity restrictions, including gimbals to stabilize the camera will prove to be a mechanically challenging feat.

Additional improvements of the software implementation of the featureless algorithm could include the use of spatially separable convolution. Which is a relatively recent variation of standard convolution in which the number of total operations to calculate the output is reduced [42]. For a standard 3×3 kernel the number of operations is reduced by approximately 33%, as long as the image dimensions are much greater than the kernel size. The reduction in calculations is achieved by splitting the convolution process into two smaller and simpler convolutions. In the case of a 3×3 kernel the kernel is split into a 1×3 and 3×1, an example of which can be seen in Figure 4.1. Separable convolution has the main drawback of the kernel used must be spatially separable to begin with, hence spatially separable convolution has not seen wide application. Thankfully the kernels used in this study are spatially separable leading to potential algorithm enhancements.

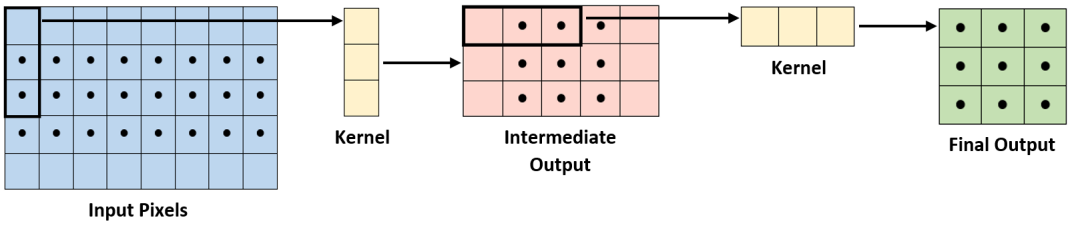

Figure 4.1. An example of the spatially separable convolution algorithm.

Bibliography

[1] Zhang, J., J. F. Campbell, D. C. Sweeney II, and A. C. Hupman (2021) "Energy consumption models for delivery drones: A comparison and assessment," *Transportation Research Part D: Transport and Environment*, **90**, p. 102668.

[2] Liu, P., S. P. Sane, J.-M. Mongeau, J. Zhao, and B. Cheng (2019) "Flies land upside down on a ceiling using rapid visually mediated rotational maneuvers," *Science Advances*, **5**(10), p. eaax1877, https://www.science.org/doi/pdf/10.1126/sciadv.aax1877.
URL https://www.science.org/doi/abs/10.1126/sciadv.aax1877

[3] Habas, B., B. AlAttar, B. Davis, J. W. Langelaan, and B. Cheng (2021) "Optimal Inverted Landing in a Small Aerial Robot with Varied Approach Velocities and Landing Gear Designs," *arXiv preprint arXiv:2111.03539*.

[4] Lazareva, O. (2017) *Binocular Disparity*, Springer International Publishing, Cham, pp. 1–4.

[5] Zhang, H. and J. Zhao (2017) "Bio-inspired vision based robot control using featureless estimations of time-to-contact," *Bioinspiration & biomimetics*, **12**(2), p. 025001.

[6] (2022) "Commercial drone market size," *Fortune Business Insights*, https://www.fortunebusinessinsights.com/commercial-drone-market-102171.

[7] Sudbury, A. W. and E. B. Hutchinson (2016) "A cost analysis of amazon prime air (drone delivery)," *Journal for Economic Educators*, **16**(1), pp. 1–12.

[8] Mogili, U. R. and B. Deepak (2018) "Review on application of drone systems in precision agriculture," *Procedia computer science*, **133**, pp. 502–509.

[9] Shahmoradi, J., E. Talebi, P. Roghanchi, and M. Hassanalian (2020) "A comprehensive review of applications of drone technology in the mining industry," *Drones*, **4**(3), p. 34.

[10] Karaca, Y., M. Cicek, O. Tatli, A. Sahin, S. Pasli, M. F. Beser, and S. Turedi (2018) "The potential use of unmanned aircraft systems (drones) in mountain search and rescue operations," *The American journal of emergency medicine*, **36**(4), pp. 583–588.

[11] Scott, J. E. and C. H. Scott (2020) "Drone delivery models for medical emergencies," *Delivering Superior Health and Wellness Management with IoT and Analytics*, pp. 69–85.

[12] Chen, G. X., H. E. Amandus, and N. Wu (2014) "Occupational fatalities among driver/sales workers and truck drivers in the United States, 2003–2008," *American journal of industrial medicine*, **57**(7), pp. 800–809.

[13] Lee, H. L., Y. Chen, B. Gillai, and S. Rammohan (2016) "Technological disruption and innovation in last-mile delivery," *Value Chain Innovation Initiative*.

[14] Aurambout, J.-P., K. Gkoumas, and B. Ciuffo (2019) "Last mile delivery by drones: An estimation of viable market potential and access to citizens across European cities," *European Transport Research Review*, **11**(1), pp. 1–21.

[15] Goodchild, A. and J. Toy (2018) "Delivery by drone: An evaluation of unmanned aerial vehicle technology in reducing CO2 emissions in the delivery service industry," *Transportation Research Part D: Transport and Environment*, **61**, pp. 58–67, innovative Approaches to Improve the Environmental Performance of Supply Chains and Freight Transportation Systems.

[16] Park, J., S. Kim, and K. Suh (2018) "A Comparative Analysis of the Environmental Benefits of Drone-Based Delivery Services in Urban and Rural Areas," *Sustainability*, **10**(3).
URL `https://www.mdpi.com/2071-1050/10/3/888`

[17] Maes, W. H. and K. Steppe (2019) "Perspectives for Remote Sensing with Unmanned Aerial Vehicles in Precision Agriculture," *Trends in Plant Science*, **24**(2), pp. 152–164.

[18] Damalas, C. A. and I. G. Eleftherohorinos (2011) "Pesticide exposure, safety issues, and risk assessment indicators," *International journal of environmental research and public health*, **8**(5), pp. 1402–1419.

[19] Tian, X., J. Iriarte-Diaz, K. Middleton, R. Galvao, E. Israeli, A. Roemer, A. Sullivan, A. Song, S. Swartz, and K. Breuer (2006) "Direct measurements of the kinematics and dynamics of bat flight," *Bioinspiration & biomimetics*, **1**(4), p. S10.

[20] Goerzen, C., Z. Kong, and B. Mettler (2010) "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *Journal of Intelligent and Robotic Systems*, **57**(1), pp. 65–100.

[21] Tennekes, H. (2009) *The Simple Science of Flight, Revised and Expanded Edition: From Insects to Jumbo Jets*, MIT press.

[22] Hileman, J. I., R. W. Stratton, and P. E. Donohoo (2010) "Energy content and alternative jet fuel viability," *Journal of propulsion and Power*, **26**(6), pp. 1184–1196.

[23] Balebail, S., S. K. Raja, and S. P. Sane (2019) "Landing maneuvers of houseflies on vertical and inverted surfaces," *PloS one*, **14**(8), p. e0219861.

[24] Bahadori, M., P. Cesari, C. Craig, and M. E. Andani (2021) "Spinal reflexive movement follows general tau theory," *BMC neuroscience*, **22**(1), pp. 1–7.

[25] Lee, D. N. (1998) "Guiding movement by coupling taus," *Ecological psychology*, **10**(3-4), pp. 221–250.

[26] Lee, D. N., M. N. Davies, P. R. Green, and F. . Van Der Weel (1993) "Visual control of velocity of approach by pigeons when landing," *Journal of experimental biology*, **180**(1), pp. 85–104.

[27] Mellinger, D., M. Shomin, and V. Kumar (2010) "Control of quadrotors for robust perching and landing," in *Proceedings of the International Powered Lift Conference*, pp. 205–225.

[28] Moore, J. and R. Tedrake (2009) "Powerline perching with a fixed-wing UAV," in *AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference*, p. 1959.

[29] Doyle, C. E., J. J. Bird, T. A. Isom, C. J. Johnson, J. C. Kallman, J. A. Simpson, R. J. King, J. J. Abbott, and M. A. Minor (2011) "Avian-inspired passive perching mechanism for robotic rotorcraft," in *2011 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, pp. 4975–4980.

[30] Souhila, K. and A. Karim (2007) "Optical Flow Based Robot Obstacle Avoidance," *International Journal of Advanced Robotic Systems*, **4**(1), p. 2, `https://doi.org/10.5772/5715`.
URL `https://doi.org/10.5772/5715`

[31] McCarthy, C., N. Barnes, and R. Mahony (2008) "A robust docking strategy for a mobile robot using flow field divergence," *IEEE Transactions on Robotics*, **24**(4), pp. 832–842.

[32] Mellinger, D., N. Michael, and V. Kumar (2012) "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, **31**(5), pp. 664–674.

[33] Barron, J. L., D. J. Fleet, S. S. Beauchemin, and T. Burkitt (1992) "Performance of optical flow techniques," in *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, pp. 236–237.

[34] Lucas, B. D., T. Kanade, et al. (1981) "An iterative image registration technique with an application to stereo vision," Vancouver.

[35] Horn, B. K., Y. Fang, and I. Masaki (2009) "Hierarchical framework for direct gradient-based time-to-contact estimation," in *2009 IEEE Intelligent Vehicles Symposium*, IEEE, pp. 1394–1400.

[36] Fleet, D. J. (2012) *Measurement of image velocity*, vol. 169, Springer Science & Business Media.

[37] Sjödahl, M. and L. Benckert (1993) "Electronic speckle photography: analysis of an algorithm giving the displacement with subpixel accuracy," *Applied Optics*, **32**(13), pp. 2278–2284.

[38] De Croon, G., D. Izzo, and G. Schiavone (2012) "Time-to-contact estimation in landing scenarios using feature scales," *Acta Futura*, **5**, pp. 73–82.

[39] Longuet-Higgins, H. C. and K. Prazdny (1980) "The interpretation of a moving retinal image," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, **208**(1173), pp. 385–397.

[40] Bainbridge-Smith, A. and R. Lane (1997) "Determining optical flow using a differential method," *Image and Vision Computing*, **15**(1), pp. 11–22.

[41] Chirarattananon, P. (2018) "A direct optic flow-based strategy for inverse flight altitude estimation with monocular vision and IMU measurements," *Bioinspiration & biomimetics*, **13**(3), p. 036004.

[42] Mamalet, F. and C. Garcia (2012) "Simplifying convnets for fast learning," in *International Conference on Artificial Neural Networks*, Springer, pp. 58–65.