Visual sensing and control of hover for Micro Air Vehicles

Samuel A. Wassef

University of New South Wales at the Australian Defence Force Academy

Insects have the ability to control their flight through the use of visual sensors. The use of visual sensors enables insects to control their forward flight, landing, response to changes in the environment and also hover. This behaviour in insects has inspired attempts to recreate the same behaviour in robots. Visual sensors have already been used in conjunction with other technologies such as Global Positioning Systems (GPS) or laser rangefinders to sense and control the hover of a Micro Air Vehicle (MAV). However, MAVs are not suitable for use with current GPS or laser rangefinders because the sensors are impractical due to their size and weight. Since insects are able to control flight using visual and other neural sensors, it is desired to have a system which uses small visual and inertial sensors to control hover of an MAV. This paper presents the theory and simulation results for a visual sensing and control system for hover of an MAV. The simulation was done using proposed optic flow algorithms to show that visual and inertial sensing suitable for an MAV can be used to simulate hover. The algorithms use optic flow and loom to calculate velocity and position in x, y and z directions, as well as estimating the height of the MAV by integrating the loom. This therefore allowed for simulating two-dimensional and then three-dimensional snapshot hover for the MAV. Tests were also done to determine the ability of the simulation to launch the MAV to a certain height and then simulate hover. The results of the computer simulations show that hover for an MAV can be achieved using visual and inertial sensors. Furthermore, the method is somewhat robust to various effects of noise that would be experienced in real applications.

Contents

I. Introduction 3
II. Aim 4
III. Thesis Scope and Methodology 4
   A. Phase One 5
   B. Phase Two 5
   C. Phase Three 5
   D. Phase Four 5
IV. Biological Inspiration 5
   A. Visual sensors, image motion pattern and landmark learning 5
   B. Visual control of flight 6
   C. Landing strategies 7
   D. Insect hover 7
V. Development of insect vision to optic flow computation 7
   A. Optic flow 7
   B. How does it work? 7
   C. The I2A and I3A algorithms 8
VI. Optic flow computation in robotics 8
   A. Forward flight/obstacle avoidance 8
   B. Landing strategies and hover 9
VII. Problems with using optic flow for hover 9
   A. Intensity constancy assumption 9
   B. The aperture problem 10
   C. Other problems 10

1 LT, School of Engineering and Information Technology. Mechanical Engineering Project, Thesis and Practical Experience ZEIT4501.
VIII. Methodology to simulate snapshot hover of the MAV
   A. Overcoming reliance on other technology 10
   B. Approach 11
   C. Simulation 11
   D. Controller 12
IX. Calculation of X and Y velocity and position 12
   A. X and Y velocity 12
   B. X and Y position 13
X. Loom calculation to estimate Z velocity and position 14
   A. Loom 14
   B. Z velocity 15
   C. Estimating the height 15
   D. Controller used for the simulation 15
   E. Launch to a certain height 16
XI. Results of the simulation 16
   A. Optic flow vectors 17
   B. Results for 2D snapshot hover 17
      i. X and Y velocity and position 17
      ii. Z velocity and position 18
      iii. Summary 19
   C. Estimating the height and 3D snapshot hover 19
      i. Comparison of the height estimation to actual height 19
      ii. X and Y velocity and position 20
      iii. Z velocity and position 21
      iv. Summary 22
   D. Effect of noise 22
      i. Noise added to optic flow for position 22
      ii. Noise added to optic flow for velocity 23
      iii. Noise added to optic flow for both position and velocity 23
      iv. Noise added to loom 24
   E. Launch to a certain height 24
      i. X and Y velocity and position 25
      ii. Z velocity and position 26
      iii. Summary 27
   F. Summary of results 27
XII. Recommendations and future work 27
XIII. Conclusions 28
Acknowledgements 29
References 29

APPENDICES (In supplementary document)
Appendix A. Client Brief A1
Appendix B. Task Breakdown Structure A2
Appendix C. Gantt Chart A3
Appendix D. Milestone Chart A4
Appendix E. I^2A Algorithm Derivation A5
Appendix F. MATLAB Code for Optic Flow Calculation A6
Appendix G. Simulink Diagrams Used in the Controller A7
Appendix H. MATLAB Code for X and Y Velocity Calculation A8
Appendix I. MATLAB Code for X and Y Position Calculation A9
Appendix J. MATLAB Code for Loom Calculation A10
Appendix K. MATLAB Code for ∆Z/Z Calculation A11
Appendix L. MATLAB Code of Inputs for Simulation A12
Nomenclature

2D = two-dimensional
3D = three-dimensional
I2A = Image Interpolation Algorithm
I3A = Iterative Image Interpolation Algorithm
v = Velocity [ms⁻¹]
H = height [m]
Q = optic flow [s⁻¹]
Δx = translation along the x axis
Δy = translation along the y axis
Δz = rotation around an axis perpendicular to the plane with the origin at the centre of the picture
Ψ(x,y) = window function
P = half-width of the Gaussian [pixels]
v_f = forward speed [ms⁻¹]
ω_set = the angular velocity in the previous snapshot [rads⁻¹]
ω_measured = the measured angular velocity in the current frame [rads⁻¹]
Q_x = horizontal component of optic flow
Q_y = vertical component of optic flow
F_x = horizontal focal length of the camera
F_y = vertical focal length of the camera
X = forward/rearward shift in velocity or position
• Y = sideways shift in velocity or position
• Z = vertical change in velocity or position
V_x = X component of velocity
V_y = Y component of velocity
V_z = Z component of velocity
X = X position
Y = Y position
Z = Z position
p = pitch rate
q = roll rate
r = yaw rate
θ = pitch angle
φ = roll angle
ψ = yaw angle
U = horizontal pixel locations of features in the field of view
V = vertical pixel locations of features in the field of view
L = loom
\( \overline{Q}_{set} \) = the next estimate of the shift reference
\( \overline{Q}_s \) = the previous estimate of the shift reference
\( Q_{set} \) = the actual measurement of the shift
\( \nabla I \) = the magnitude of the intensity gradient vector, \( \nabla I = (I_x, I_y) \)
\( I_t \) = the temporal derivative of the intensity function

I. Introduction

Studies show that insects use image motion from visual sensors to control several aspects of their flight. Bees are able to acquire 3D perception of their world from the pattern of image motion that their visual system experiences as they fly through the environment [1]. This pattern of image motion is called ‘optic flow’, which is a visual phenomenon experienced everyday. Optic flow can be defined as the apparent visual motion that you experience as you move through the world – or in simpler terms, as you move forwards or backwards, objects that are closer appear to move faster than objects that are further away [2]. Theories suggest that several insects use optic flow to control their flight, such as flies, bees and grasshoppers. Experiments also suggest that bees have applied optic flow to visually control several aspects of their flight including flight stabilisation, landing,
estimation of object range, safe negotiation of narrow gaps and also hover [1]. These studies in biology have inspired developments in Unmanned Aerial Vehicles (UAVs) and MAVs due to the fact that such tiny insects are capable of guiding their flight through visual sensors [3].

The University of New South Wales at the Australian Defence Force Academy (UNSW@ADFA) is interested in contemporary work with MAVs and developing a project to contribute to the use of optic flow in MAVs. The overarching project is sensing and controlling hover of an MAV and the interest is in designing a system suitable for both rotary wing and flapping wing implementations. The first year of the project is divided into a number of activities. This project will focus on one of the tasks in the project which is achieving visual snapshot hover in simulation using hardware in the loop image processing and rapid synthetic image generation.

Current research has been applied to MAVs in forward flight, obstacle avoidance and hover using an external object such as a GPS, laser rangefinder or radar. These objects are used to determine either the speed of the object (from GPS) or the range to other objects (laser rangefinder). From this, optic flow can be calculated [4]. The problems with GPS’s and laser rangefinders are that they are simply impractical for use on MAVs due to the physical barriers to miniaturisation that exist. The fact that bees are such tiny creatures and theories suggest that they can control their flight using visual sensors has inspired mechanisms for control of flight using small visual and inertial sensors. GPS and other technologies also have added problems, such as not operating well in some areas due to weak coverage and ability to be interfered with due to electromagnetic emissions. This makes it more desirable to develop a passive system to sense and control hover [3,4,5,6].

Snapshot hover is a passive system, using small visual and inertial sensing technology and eliminating the use of other technologies, by using optic flow calculation to control hover of an MAV. By employing snapshot hover techniques in MAVs, the reliance on other technologies will be reduced. Furthermore, a passive, small, lightweight and low cost system will be developed for MAVs, which allows for very advantageous military applications.

There are, however, some problems with optic flow computation and the employing snapshot hover for MAVs. Traditional methods for computing optic flow are based on the intensity constancy assumption and the assumption that the flow field will be smooth [7]. The intensity constancy assumption leads to problems in real world images because the velocity fields are neither locally constant nor globally smooth, rather, they are piecewise continuous [7]. Another fundamental problem in the estimation of optic flow is the aperture problem. When viewing a moving object through an aperture, where there is a spatial gradient in only one direction, it is possible to determine a precise estimate of the optical flow perpendicular to the gradient [8]. However, this is insufficient in specifying the true direction of the moving optic flow vector, and more information is required to determine the motion completely [9,10].

This project will be done using both MATLAB and Simulink. Simulink will allow for a model representing the dynamics of the MAV and for the attitude control system to be developed. The attitude control system will be developed using a controller in Simulink to control the dynamics of the MAV and simulate hover. MATLAB will be used to code the calculation of optic flow and loom, as well as estimation of velocity and position from the optic flow and loom calculations. These calculations will be used as inputs in the Simulink model.

II. Aim

The aim of this thesis is to develop a control system to sense and control hover of an MAV using snapshot hover techniques with the \( \hat{T}A \) algorithm. This aim can be divided into four sub-aims:

1. To understand how insects use visual sensors for motion,
2. To relate how insects use vision to the computation of optic flow,
3. To understand how optic flow and the \( \hat{T}A \) algorithm can be used in snapshot hover, and
4. To design a control system for an MAV to hover.

III. Thesis Scope and Methodology

The scope of this thesis involves understanding and modelling snapshot hover in Simulink. It was first necessary to understand biological vision and how it is used for flight control. It was then necessary to understand optic flow computation and how it is currently used in robots. The approximations for optic flow computation were then examined and used in the simulation in Simulink. Currently, research has already been done using a GPS or laser rangefinder for control of flight of helicopters.

The aim of this thesis is to design a control system to simulate hover of an MAV using snapshot hover. Snapshot hover seeks to eliminate reliance on other technologies because they pose some problems in use with MAVs. GPS’s are not suitable for use in cluttered environments because they have a tendency to break down due to their reliance on a signal. Laser rangefinders are too bulky for use on MAVs and it is impractical to use them on MAVs due to size. Various simulations were therefore conducted in Simulink to determine whether optic flow and other inertial sensors can be used to control hover of an MAV. The simulations were firstly run to simulate 2D snapshot hover.
A secondary aim is to simulate 3D snapshot hover and to be able to launch to a height and then simulate hover. With 2D snapshot hover, the height is assumed to be known by some other means, but in the 3D case, another method is needed to determine the height of the MAV. The proposed method that was used, was integrating the loom to obtain an initial height estimate to start the snapshot working. Launching to a height would simply be simulated using a step input to a certain height and then transitioning to hover.

To achieve these objectives the project is broken down into four phases. These phases are as follows: (1) understanding biological vision and how it is used for flight control, (2) applying biological vision to mathematical algorithms, (3) design of the sensing and control system to simulate 2D snapshot hover of the MAV, and (4) extending to 3D snapshot hover. These phases are chronological in order, to firstly understand how insects use vision to control flight, then investigate the current approaches and finally use this to extend research to snapshot hover of the MAV. Appendices A through D detail the project management aspects of this thesis, an explanation of each phase is detailed below.

A. Phase One
This phase involves understanding biological vision and how it is used for flight control. Insect vision is a widely studied area and several experiments suggest theories on how insects use vision to control their flight. Srinivasan et al [11,12] have completed many experiments explaining how insect vision works, how they use optic flow and how this is used in forward flight and landing strategies.

B. Phase Two
Equations and algorithms have been derived to mathematically determine how insects use visual sensors to control their flight. This has been developed from the more simple equation for optic flow to Srinivasan’s I2A algorithm and the Garratt’s proposed I3A algorithm [2,13,14]. The I2A algorithm is currently being used in robotics to either achieve or simulate control of flight for computer-based systems. Research has been developed to several aspects of flight, which will be investigated in this phase to not only understand the algorithms, but also how they are applied to the different aspects of flight.

C. Phase Three
This is the most intensive phase, which involves the simulation to sense and control hover of the MAV. This will involve defining the approach to how hover will be simulated for the MAV, which will be using a snapshot hover algorithm, in order to overcome the reliance on other technologies. This algorithm will then be used in Simulink to simulate hover of the MAV. The simulation will be evaluated for how well it simulates snapshot hover of the MAV, as well as examining environmental effects such as wind gusts. This is considered essential prior to phase four because Deng et al [15] argue that although rotary-wing MAVs have the potential ability to hover, they are more susceptible to wind gusts, and are slower in response when compared to the ability of insects to hover.

D. Phase Four
If the 2D snapshot algorithm is successful, and there is time permitting, this will be extended to 3D snapshot hover, controlling horizontal and vertical position. This will therefore have to include a system to control the height of the object, which will involve designing an algorithm to bootstrapping the MAV from take-off.

IV. Biological inspiration
Insects have quite remarkable vision capabilities, which has inspired much research. Their ability to use visual sensors to determine image motion patterns allows them to control several aspects of their flight. Experiments have been conducted on insect flight through a tunnel and the effects of stationary and moving walls, as well as the effect of contrast patterns to change the insect’s image perception. Studies have also shown how insects apply their ability to visually control their flight in free flight, landing and hover.

A. Visual sensors, image motion pattern and landmark learning
Current theories of insect motion suggest that several insects, particularly bees, deliberately impart a lateral sinusoidal displacement of known amplitude upon their flight path in order to generate optic flow in a defined way. As the velocities corresponding to the lateral wiggle are a function of the amplitude, the bees are able to deduce range from the lateral optic flow signal and thereby control their height. Apparently, bees can also regulate their flight speed based on the longitudinal optic flow signal. With a range known from lateral optic flow and forward speed known from longitudinal optic flow, bees are able to regulate both speed and height using the one sensory system [16]. This sensory system is a combination of inertial, visual and other sensing modes, with visual sensors as the primary sensor [17]. Grasshoppers also derive their motion through visual cues. This ‘peering behaviour’ of grasshoppers is a reaction to visual cues and their motion is derived from...
changes in the environment caused by this peering behaviour’. This inspired research into bees, and how they navigate through corridors, which was conducted by Srinivasan et al [18].

B. Visual control of flight

Using these visual cues, insects are therefore able to control their flight. Srinivasan et al [12] have demonstrated through experiments that free flying honeybees tend to fly along the axis of a tunnel with a so-called ‘centering response’. By equalising the apparent angular speeds on the two eyes, bees are able to balance the distances to the two walls, thus achieving the centering response [12]. Srinivasan et al [12] explains that this centering response occurs when the two walls are stationary; however, bees were also tested for a combination of stationary and moving walls. Figure 1 [12] shows the results of these tests. When there was one moving wall and the bees were moving in the same direction, results show that they tended to fly closer to the moving wall. This is because the wall that moves in the same direction as their flight in figure 1 b generates a lower optic flow, which is accounted for by the bees flying closer to that wall, thus keeping optic flow constant. Conversely, when the moving wall was moving in the opposite direction to the bees, they tended to fly closer to the stationary wall. This is because the wall moving in the opposite direction appears to move at a higher velocity thus generating a higher optic flow, which is accounted for by the bees flying further away from the wall, again to maintain a constant optic flow [12]. This behaviour shows that bees use their visual cues to generate optic flow from the apparent motion of the walls and adjust their flight based on the image motion pattern.

Biomechanical studies of flies suggests that their hind wings called halteres are capable of measuring body rotations through gyroscopic forces [19]. This causes a reaction of the halteres during flight to beat up and down out of phase to the wingbeat frequency [20]. This suggests that the haltere is able to detect rotations, sensing angular velocity and therefore reacting during movement in flight [15,21]. In particular, it is argued that fruit flies use mechanosensory feedback from halteres to rapidly change direction in response to visual cues, according to the schematic diagram in figure 2 [20]. Approaching objects or sideways body translation generates image expansion of the centre of the image. These visual cues are relayed from the brain along specialised descending premotor networks to trigger rapid all-or-none saccades. Once initiated, mechanosensory feedback from the halteres
probably serves to terminate the saccade, limiting the body rotation to 90 degrees. This then initiates the resultant motor output, which changes the flight of the fly accordingly [20].

C. Landing strategies

Image motion cues are used in a fairly similar manner in the way of holding image velocity constant to perform smooth landings. One strategy which bees use to perform grazing landings on a flat surface, is described by Srinivasan et al [11] as a surprisingly simple and elegant strategy. The bees hold image velocity constant as they approach the surface, therefore automatically ensuring that flight speed is close to zero on touchdown. This requires no explicit knowledge of flight speed or height above ground and simply uses optic flow [11]. Studies of this landing behaviour has shown that as the surface is approached, the expansion of the image of the surface provides strong cues that are used to control deceleration and trigger extension of the legs in preparation for contact. Studies also suggest that the rate of expansion of the image is used to infer the time to contact the surface, despite that speed of flight or height above the surface is not required [11]. To achieve this, bees hold the angular velocity of the image constant, which allows the velocity of the bee to decrease at the same rate as the height above the surface decreases. This achievement of a smooth landing implies that the bees control their forward flight speed to maintain constant image velocity of the ground. The bees also control the descent speed to be proportional to their forward speed so that as both components of the speed decrease, it becomes zero at landing [11]. This therefore shows another aspect of an insect’s flight that is mainly controlled by visual sensors.

D. Insect hover

Insects are also able to use visual sensors to allow them to hover. Kelber and Zeil [22] argue that guard bees are able to extract the flow features from the retinal image, which will tell them the direction and the magnitude of the displacement that must be compensated for in order to keep a constant position relative to the nest. Kelber and Zeil’s [22] observations demonstrate that guard bees are able to analyse complex image flow patterns and use them for orientation.

V. Development of insect vision to optic flow computation

The aspects of insect flight including free flight, landing and hover has been approached in depth to develop strategies to determine optic flow algorithms so that it can be used computationally. This involves determining how bees use visual sensors to control flight and attempting to explain how this works mathematically.

A. Optic flow

Optic flow is the term given to what theories suggest that insects use to control their flight. Garratt [17] defines optic flow as the motion of visual features across the field of the observer caused by translation or rotation of the observer. A more simplified definition is as you move either forward, backwards, sideways or rotationally, the objects closer to you appear to move faster [2]. When applied practically, optic flow can therefore be used to perceive the relative range of objects in the environment. This is because closer objects, in more mathematical terms, exhibit a higher angular motion (since they appear to move faster) than more distant objects when the observer is in motion [17]. For example, if you double your velocity, the optic flow will also double and if you double the distance to an object, the optic flow will decrease to half its original value. This can be easily understood by comparing motion of walking compared to motion in a car. In a car moving at, say 60 km/h, stationary objects appear to move much faster than if one were walking, i.e. the objects generate a much higher optic flow due to a larger increase in velocity. Conversely, the optic flow will decrease with objects that are further away because they appear to move slower. In very simple mathematical terms, optic flow can be defined as

$$Q = \frac{v}{H}$$  \hspace{1cm} (1)

showing the relationship between the velocity of an object to its distance to the object. In this relationship, the simple example of an object in forward flight at a certain height above a surface is used to calculate optic flow [17].

B. How does it work?

When optic flow is applied, using speed from a known source, and using range detection from visual sensors, optic flow can be calculated. Experiments have shown that bees tend to fly down the centre of a tunnel exhibiting this response [12]. Conversely, maintaining a constant image motion pattern (constant optic flow), an object can control other aspects of its flight. Experiments have shown that bees use this technique to control
their free flight with moving and stationary walls in tunnels by maintaining a constant optic flow and velocity and adjusting their distance away from the moving wall [12]. Also in landing, while maintaining a constant optic flow, bees decrease their velocity at the same rate as their height decreases so that when their height is approaching zero, as does their velocity, thus performing a smooth or grazing landing [11]. Furthermore, it has been shown that guard bees achieve hover simply by maintaining a constant optic flow that is very close to zero and therefore their velocity is practically zero (except for environmental effects), allowing them to remain stationary at a constant height [22].

C. The $\mathcal{I}A$ and $\mathcal{I}4A$ algorithms

Srinivasan’s [23] $\mathcal{I}A$ algorithm is a single-stage, non iterative procedure, which interpolates the position of a moving image in relation to a set of reference images [2]. The algorithm sets the shift references as the maximum shift between two snapshots (or images). If the real shift exceeds the shift reference, estimates given by the algorithm will be wrong [2]. A derivation of this algorithm is in appendix E [2,13,23]. To overcome this problem, $\mathcal{I}4A$ algorithm was developed.

The new $\mathcal{I}4A$ algorithm predicts the image for the next time step to allow for the image shift to be small, since the prediction should be close to the actual image in the next time step. This overcomes the problem of the real shift exceeding the reference shift. This prediction of the next shift reference can be estimated using the formula [2]

$$\overline{Q}_{x,t+1} = \overline{Q}_{x,t} + \lambda (\overline{Q}_{x,t} - \overline{Q}_{x,t})$$

(2)

Snapshot hover uses a stored image of the ground as the snapshot, taken of the ground directly under the MAV. This snapshot forms a visual anchor point for the MAV and by comparing subsequent frames with this snapshot, absolute translation from the datum can be calculated using optic flow equations which will be discussed later.

VI. Optic flow computation in robotics

Higgins [24] argues that engineers have a lot to gain from studying biological neural systems. Due to biological inspiration, optic flow research was extended with attempts to use optic flow calculations and algorithms to control flight of robots, aircraft and other computer based objects. This was done in a number of aspects of flight, using knowledge from previous research and experiments done with insects (mainly bees) to explore what could be simulated using man-made objects.

A. Forward flight/obstacle avoidance

Garratt has done much work in forward flight of helicopters using remote controlled helicopters with onboard image processing to determine range of objects and GPS to calculate velocity. His experiments show that the helicopter is able to successfully fly above a certain surface of undulating terrain and maintain a constant height above the terrain [17]. Figure 3 shows a graph depicting the helicopter’s ability to maintain a constant height above the terrain when in forward flight using optic flow [17].

Further work by Garratt was also done using optic flow in forward flight to simulate obstacle avoidance of helicopters. Results showed that when a helicopter was set on a certain path, it would travel roughly along the path but correcting its route to avoid obstacles using optic flow. Figure 4 shows a simulation with the helicopter set on a heading of 22.5° [3]. As can be seen from figure 4, the simulation generally follows the original heading, but as it identifies obstacles it avoids them along the defined path.

Studies of halteres in flies has also inspired experiments on flapping wing implementations. Deng [15] conducted simulations and experiments to mathematically model halteres and optic flow sensors on flapping wing models based on a blow fly Calliphora. Reiser and Dickinson [25] also applied the sophisticated sensory-motor control of flies on robotics by designing a five-degree-of-freedom system that serves as a novel

![Figure 3. Helicopter using optic flow in forward flight to follow terrain. In the figure, the solid line shows the path of the helicopter, while the dashed line shows the terrain. As can be seen in the figure, the paths are almost parallel, showing that optic flow can successfully be implemented on helicopters to control flight. [17]](image_url)
It allowed the implementation of a fly-inspired control system that uses visual and mechanosensory feedback. The results from both studies show that by using a simple control system, behaviour of robots can be simulated or achieved to be very similar to fly-like behaviour [15,25].

Srinivasan et al [18] have explored several other insects, which have inspired research in robotics. The observations of peering grasshoppers, bees landing on artificial flowers as well as of bees flying through tunnels and their centering response has led to the development of novel algorithms and devices, such as the I2A and I3A algorithms, for rangefinding. Results again showed that the I2A algorithm performed well in simulating the flight of insects [18].

B. Landing strategies and Hover

Studies of landing strategies of bees was extended to conduct tests on robotic gantry. A computer-controlled robotic gantry carrying a visual system (provided by a mounted video camera), placed under a closed-loop control system, was used to test its ability to land. Velocity of the image motion was measured using the I2A algorithm. Landing was controlled by maintaining a constant descent angle and constant image angular velocity, which is very similar to the landing strategy of bees. A time step was used for the changes in velocity and angular velocity according to equation:

\[ v_f(i+1) = v_f(i) \frac{\omega_{set}}{\omega_{measured}}. \]  

(3)

This shows the relation between the forward speed in the next time step, \( v_f(i+1) \), with the forward speed at the current time step, \( v_f(i) \), where:

\[ \frac{\omega_{set}}{\omega_{measured}} \]

provides the correction factor to set the desired descent angle. This correction factor uses the image velocity to maintain a constant angular velocity, and therefore the forward and descent speeds will decrease continuously as the camera descends [11].

VII. Problems with using optic flow for hover

When computing optic flow, some problems arise due to estimations that are made in derivations. The most common problems associated with optic flow are the intensity constancy assumption and the aperture problem.

A. Intensity constancy assumption

The intensity constancy assumption is the assumption that the intensity of light reflected by a point on an environmental surface and recorded in the image remains constant during a short time interval, although the location of the image at that point may change due to motion [28]. This, of course becomes a problem when computing optic flow for real world images and for extended periods of time. This therefore has an effect on hover. The intensity constancy assumption is violated at points in the image region which contain transparency, specular reflections, shadows, or fragmented occlusion (for example looking through the branches of a tree or spacing of a fence) [7]. These are all problems that would be experienced with real world images; however, since the simulation does not use real graphics or imagery, these effects will not be visible in the results. Instead, however, use will be made of Simulink’s noise block inputs because all these effects on images effectively translate into a noisy signal. Therefore to test the robustness of the algorithm against effects of real world images, noise will be added as an input to the optic flow signal used to calculate image motion.

The intensity constancy assumption leads to the intensity constraint, which is defined by the equation

\[ \nabla I = -I_t \]  

(4)
The normal-flow at the edge, $\vec{u}_\perp$, is the component of the image velocity $\vec{u}$ parallel to $\nabla I$. The other component $\vec{u}_T$, along the direction perpendicular to $\nabla I$, is unspecified by the constraint. Since the orientation of the intensity-gradient vector is normal to the direction of the edge at a point, $\vec{u}_\perp$ and $\vec{u}_T$ are called the normal-flow and tangential-flow components of the edge respectively. The lack of information regarding the tangential-flow component is known as the aperture problem [28].

B. The aperture problem

The aperture problem is essentially equivalent to viewing a moving object through an aperture [9]. Since the problem leads on from the intensity constancy assumption, the problem is that the only information that can be extracted is of the motion component perpendicular to the local orientation of the element, as motion along the element would be invisible [10]. This can be explained by supposing that an extended line moves through an aperture, as in the diagram in figure 5 [9]. The velocity in the aperture can be described by $V_I$, the local velocity orthogonal to the orientation of the line, as shown in figure 5(A). This local velocity is insufficient to specify the true direction of the moving line because $V_I$ could be generated by an infinite set of true velocity vectors $V$, shown in figure 5(B). Thus, an analysis of local motion cannot specify the true velocity to anything better than 180 degrees. Because of this 180 degree ambiguity, a single local reading is not very informative, and multiple optic flow vectors are required to overcome this problem [8,9].

C. Other problems

Another important problem in using optic flow for stabilisation in hover is that near perfect hover will result in a very weak optic flow signal because motions are small. Further, optic flow does not provide for identification of landmark features that can be used to maintain station. Research by Bianco et al [26] suggests that some insects use landmark-guided navigation, storing images of landmarks as a snapshot. To overcome this problem, snapshot hover is proposed. This uses a stored image of the ground, forming a landmark as a visual anchor point for the MAV. Also, a measure of scale is still required for the algorithm to simulate 3D snapshot hover. For hover after a vertical take off, the proposed method is to integrate the loom to obtain an initial height estimate, which will be used to start the snapshot working. The vertical velocity may be estimated from the inertial sensors and optic loom.

Figure 5. Pictorial description of the aperture problem.
In (A), circular region designates the local window or aperture, and the local velocity $V_I$ is orthogonal to the contour and moving up and to the right at a 45 degree angle. In (B) the possible real motion vectors $V$ are shown which could have given rise to $V_I$ and the line where the vectors end is the constraint line. [9]

VIII. Methodology to simulate snapshot hover of the MAV

Current research of hover is still reliant on devices to be used in conjunction with cameras to calculate optic flow. Garratt [17] points out that the use of optic flow is not limited to forward flight, nor should it be restricted to use with other technologies. Work by Baird and Srinivasan [1,12] suggests that bees, simply from visual cues and other sensors, use optic flow to control their height. Garratt [17] argues that similar systems could be implemented on a helicopter to achieve hover through the use of an open-loop system where a controller would act to maintain zero longitudinal optic flow.

A. Overcoming reliance on other technology

Since bees do not use any forms of technology and experiments suggest that they use mainly visual sensors to control their flight, one would think that a helicopter should be able to eliminate the use of GPS and laser rangefinders [5,6]. GPS and laser rangefinders tend not to operate well in cluttered environments such as urban areas. They will not work indoors, and are dependent on an area where there is good coverage to operate efficiently. The application of MAVs to military operations is becoming more varied including surveillance inside buildings, search and rescue hostage situation monitoring, chemical agent detection and bomb search. In the current theatres to which the Australian Defence Force (ADF) is deployed, it is likely that GPS coverage may not exist, or coverage will be weak in cluttered urban environments. Therefore it is desirable to have a passive system that does not rely on other unreliable and bulky technologies. This has many added advantages,
such as: not producing any electromagnetic emissions, enabling use in operational environments where stealth is important, not breaking down in various environments, ability to create small MAVs with visual sensing technology, as well as being low cost [3,4]. It is therefore desirable to develop an algorithm capable of simulating hover without the use of an external source to determine velocity.

B. Approach

The approach to the problem will be using the optic flow equations shown later to simulate snapshot hover. Theories suggest that landmark-guided navigation in insects is based on storing the image of the landmark as a snapshot and as the insect continues to move, it strives to achieve a match between the currently viewed image of the landmark and the previously stored snapshot [26]. A similar principle being applied to MAVs where the I3A algorithm will be used to estimate the pixel shift between the stored snapshot and the current camera frame at various parts in the image. In achieving hover, the snapshot will be used to anchor the MAV to a particular area, by using optic flow to calculate the amount of drift from the stored snapshot and return the MAV to its original position. Many of the experiments and studies done on bees show their ability to control their forward flight and landing, and current robots are able achieve forward flight, obstacle avoidance and landing because there is a strong optic flow signal when the object is in motion. However, the problem with snapshot hover is that ideally perfect hover will result in a weak optic flow signal because motion between images is very small. Furthermore, Deng et al [15] argue the susceptibility of MAVs to wind gusts and their slower response when compared with insects that can hover. Therefore, the simulation will be run first and then checked against environmental effects such as wind gusts, to determine its success in a variety of environments.

C. Simulation

The current model being used in Simulink is in figure 6 [3]. This shows the general control system for how the entire system of the MAV will work. As is seen in the model, the helicopter dynamics are given as an‘actual state’ which are input to the controller and the state display. The state display simply extracts the information of the attitude of the MAV and plots each aspect of attitude, velocity and position. Once the state of the MAV is input to the controller, the controller then stabilises the MAV to hover.

For 2D snapshot hover, the mathematical approach is calculated using the equation [3,27]:

\[
\begin{bmatrix}
Q_x \\
Q_y
\end{bmatrix} =
\begin{bmatrix}
\frac{F_x}{Z} & 0 & -\frac{U}{Z} & -\frac{UV}{F_x} & -\frac{F_x^2 + U^2}{F_x} & -\frac{V}{U} \\
0 & \frac{F_y}{Z} & -\frac{V}{Z} & -\frac{F_y^2 + U^2}{F_y} & -\frac{UV}{F_y} & -\frac{U}{V}
\end{bmatrix}
\begin{bmatrix}
\dot{X} \\
\dot{Y} \\
\dot{Z} \\
\dot{p} \\
\dot{q} \\
\dot{r}
\end{bmatrix}
\] (5)

The simulation will use inertial sensors for the attitude of the MAV. The position and velocity of the MAV includes \(X, Y, Z, V_x, V_y\) and \(V_z\), while the attitude includes \(p, q, r, \theta, \phi\) and \(\psi\). Using the input of these values from the inertial sensors, this calculation can therefore be performed by calculating the equations in MATLAB, using the code in appendix F.
D. Controller

The calculation for optic flow is performed in the controller in figure 7. From the optic flow, position and velocity are calculated. The position and velocity are then used in the outer loop and inner loop controllers to simulate hover. The outer loop controller, shown in figure 8, has inputs of X and Y velocity and position, as well as the desired state. PID controllers are used in the outer loop controller to dampen the velocity and position of the MAV using the actual velocity and position, as well as the desired velocity and position. This allows for the calculation of the pitch and roll commands required to achieve the desired state (hover). These pitch and roll commands then become inputs to the inner loop controller. The inner loop controller then uses the pitch and roll commands to adjust the MAV’s dynamics and allow it to hover. This is done again using PID controllers to adjust the pitch, roll and yaw angles of the MAV. The collective pitch controller takes the height estimate to stabilise the height of the MAV. The outputs from the inner loop controller and the collective pitch controller form the servo commands which become inputs back to the MAV. These servo commands are then used so that the MAV can simulate hover.

IX. Calculation of X and Y velocity and position

X and Y velocity and position was calculated by inputting values from inertial sensors into equation 5. Firstly the optic flow was calculated, then using the height, velocity and position was calculated using the relationship in equation 1. The Simulink diagram showing this relation is in appendix G. This shows the velocity and position of the MAV from the inertial sensors, then the block to calculate optic flow, which is input into the block to calculate velocity.

A. X and Y velocity

To simplify equation 5, the pitch, roll and yaw rates of the MAV were set to zero. This allowed for simple calculation of optic flow, resulting in calculation of velocity from the simple relationship from equation 1, shown below:
\[ V_x = Q_x Z \]  
\[ V_y = Q_y Z \]  

Once this was compared the values from the inertial sensors, this showed that the calculations of the horizontal and focal lengths of the camera, as well as the horizontal and vertical pixel locations were calculated correctly, hence optic flow was calculated correctly. So now pitch, roll and yaw could be set to their original values according to the inertial sensors. This changed the calculation of velocity because optic flow is affected by changes in pitch, roll and yaw. Mathematically, there is no effect of yaw on X and Y velocities, so there is no need to compensate for yaw in the calculation. Figure 9 [2] shows the relation between the motion of pitch, roll and yaw on the MAV and how this is related to the different axes. As is shown, pitch affects \( V_y \), while roll affects \( V_x \), therefore these must be subtracted off to calculate \( V_x \) and \( V_y \) correctly.  

Equation 5 was then used to calculate X and Y velocity and to compensate for pitch and roll, the following equations are used, which are derived from equation 5:  

\[ V_x = \left( Q_x - \frac{F_x^2 + U^2}{F_x} q \right) Z \]  
\[ V_y = \left( Q_y + \frac{F_y^2 + U^2}{F_y} p \right) Z \]  

The Simulink diagram allowing the calculation of the velocity from the optic flow is in appendix G. Equations 8 and 9 were coded in MATLAB to allow the diagram to output the velocities. The MATLAB Code to perform these calculations is attached in appendix H. The outputs of the velocities are then used in the controller.  

B. X and Y position  
Since equation 5 calculates optic flow based on a shift in the image, the equation can also be adapted to calculate X and Y position. This is done by substituting in the position from the dynamics of the MAV according to the equation 5. Similar to the calculation for velocity, compensating for pitch and roll, position was calculated from the equations:  

\[ \Delta X = \left( Q_x - \frac{F_x^2 + U^2}{F_x} q \right) Z \]  
\[ \Delta Y = \left( Q_y + \frac{F_y^2 + U^2}{F_y} p \right) Z \]  

Since this was calculated directly, without setting pitch, roll and yaw to zero, this was checked against the inertial sensor as was done with velocity, before the values of position were used in the controller. Similar to the velocity calculations, this was coded in MATLAB, with the code attached in appendix I. Also, the Simulink
A diagram in appendix G shows the relationship of the calculations performed from optic flow to position. The outputs of X and Y position shown in appendix G are then used in the controller.

X. Loom calculation to estimate Z velocity and position

Calculation of optic flow accounts for the calculation of velocity and position in X and Y. However, since this is a downwards looking camera, another means is required to determine Z velocity and position for 3D snapshot hover. This phenomenon is loom.

A. Loom

Loom corresponds to expansion or contraction of an image due to motion perpendicular to the image plane of the camera [3]. Loom represents the time to contact and depends on both the change in motion (height or velocity) and the distance away from the object, according to the equation:

\[ L = \frac{\dot{X}}{Z} \]  

Consider looking at an object directly ahead. If this object starts moving towards you with a certain velocity, the image of the object appears to expand. Conversely, if it moves away, it appears to contract. If the object moves at twice the velocity towards you from the same distance away, the expansion of the image occurs twice as fast. Mathematically this means that the velocity, \( \dot{X} \), doubles, while \( Z \) (distance away) remains constant, therefore the loom (expansion of the image) also doubles. On the other hand, if the object moves at the same velocity at twice the distance away, the image will contract by a factor of two. Mathematically this means that \( Z \) (the distance away) doubles, while \( \dot{X} \) remains constant, and therefore the loom is halved. The loom value is extracted by subtracting the sum of the optic flow vectors on the left-hand side of the image from the sum of the optic flow vectors on the right-hand side of the image [3]. Optic flow is calculated using equation 5. From this equation, the horizontal component of the optic flow vectors is defined by \( Q_x \) at a location \((U, V)\) in the image plane. Due to the symmetrical and even distribution of patches summed over the left and right sides of the image, subtraction of their sum results in the cancellation of many terms from equation 5. The resulting relationship is:

\[
\frac{\dot{X}}{Z} \left[ \sum_{\text{left}} U - \sum_{\text{right}} U \right] = \sum_{\text{left}} Q_x - \sum_{\text{right}} Q_x
\]  

Finally, the loom term can be calculated using the equation below:

\[
L = \frac{\dot{X}}{Z} = \frac{\sum_{\text{left}} Q_x - \sum_{\text{right}} Q_x}{\sum_{\text{left}} U - \sum_{\text{right}} U}
\]  

This is done simply using the same calculations of the pixel locations and optic flow used to calculate velocity and position for X and Y. Using the symmetry of the calculation of the pixel location on the left and right sides of the image, the denominator can be simplified because:

\[
\sum_{\text{left}} U - \sum_{\text{right}} U = 2 \sum_{\text{right}} U
\]  

This therefore simplifies the loom equation to:

\[
L = \frac{\dot{X}}{Z} = \frac{\sum_{\text{left}} Q_x - \sum_{\text{right}} Q_x}{2 \sum_{\text{right}} U}
\]
The MATLAB code to calculate the pixel locations on the left and right sides of the image, the optic flow on the left and right sides of the image, and finally the loom, is in appendices J and K. This was done twice to calculate the loom for a height estimate and $\Delta Z/Z$ for the vertical velocity, which will become more apparent with the controller.

B. Z velocity
Using the relationship in equation 12 for loom, the Z velocity can be calculated. To be able to calculate the vertical velocity in the simulation, however, an estimation of the height was required first.

C. Estimating the height
Since the loom allows for the calculation of $V_z$, a relationship between Z and $V_z$ is required. This can simply be determined by the fact that $V_z$ is a measure of the change in Z over the time step $dt$. Therefore by integrating the loom, and multiplying it by Z, a value for the change in Z can be obtained, as shown below by the relationship:

$$\Delta Z = \int V_z dt = Z \int L dt \quad (17)$$

This is done in Simulink simply using the calculation of $V_z$, and using an integral block which directly obtains the value for the change in Z. This calculation also used the MATLAB Code in appendix K. This change in Z can be used as a direct input into the collective pitch controller, as shown in figure 10. This allowed obtaining an initial height estimate. Then, once the snapshot was working, the height could be calculated more accurately. The estimate of the height could then be used to estimate the vertical velocity. This was simply done by multiplying the loom value by the estimated value of the height, according to equation 12.

D. Controller used for the simulation
The controller in figure 7 had to be modified slightly to account for shifts in position of the MAV. As shown in figures 7 and 8, the inputs to the outer loop controller are only the X and Y velocities. This was modified to include inputs of X and Y position as well, which would simulate hover in position, and was therefore more suitable than the original controller. Figure 11 shows a diagram of the controller used for the simulation. The
other major change that occurred was the creation of the optic flow blocks used to calculate velocity and position. Figure 10 and the Simulink diagrams in appendix G were created in the controller using the inputs from the state, as shown in figure 11, and were then used as inputs into the outer loop controller. Figure 10 and the Simulink diagrams in appendix G were created based on the algorithms and derived equations, using the MATLAB Code in appendices F, H, I, J and K to supplement the Simulink blocks. The outer loop controller in figure 12 shows that the inputs of X and Y velocity and position are taken, and the PID controllers are used to determine the pitch and roll commands to input into the inner loop controller.

![Diagram of the outer loop controller used in the simulation.](image12)

**E. Launch to a certain height**

To simulate a launch to a certain height, the collective was set to a constant value for a certain amount of time to launch the MAV to a certain height. Then a switch was used once the MAV had reached the certain height to change the input back to simulate hover. Figure 13 shows how the switch was implemented in Simulink to simulate launching the MAV to a certain height. The first input is a constant of zero collective, and the step increases the collective to a certain value after the desired amount of time. Because the collective from the collective pitch controller is greater than the increase from the step, the switch then takes the input from the collective pitch controller after the desired time for the step.

![Diagram of the set up of the switch to simulate launching to a certain height.](image13)

**XI. Results of the simulation**

Simulink produced several plots of the attitude of the MAV, which are used to show hover. This is done by extracting plots from the actual state of the MAV in the ‘State Display’ block in the Simulink model, shown in figure 14. A scope is used to plot each aspect of the attitude of the MAV, and also the velocity and position, which was used to determine whether hover was simulated.

![State Display of the actual state of the MAV.](image14)
The MATLAB code containing inputs for the attitude and conditions of the MAV is in appendix L. This sets the initial values for the MAV, and varying these will show a range of outputs. Simulations were run the settings as shown in appendix L. The mask in figure 15 was created for Simulink blocks containing optic flow calculations to easily vary the number of patches in the image (in x and y directions) and the number of pixels in the image. This was set to 8 x 8 patches and 400 x 400 pixels. This was a compromise between not having too many patches in the image, as this would slow computation time, and still having multiple optic flow vectors to avoid problems that would be encountered with a single optic flow vector.

A. Optic flow vectors
   A MATLAB function called 'quiver' was used to display what is called a 'quiver plot' of the optic flow vectors. This shows the movement of the pixels in the field of view of the camera, therefore showing the relative movement of the ground to the downwards-looking camera. It allowed for more understanding of what was occurring in the simulation. This was initially used to check that optic flow was being calculated correctly, and then subsequently turned off due to excess computational time required to produce the plot. Figure 16 shows an example of a quiver plot taken of the simulation, showing all the optic flow vectors pointing out from the centre. As explained previously, loom is the expansion or contraction of an image. Since all the vectors point outward, meaning that the pixels in the image are moving outward, this represents the expansion of the image. This means that the simulation is of a decrease in height, or descent.

B. Results for 2D snapshot hover
   Ideally, the plots of velocity would tend to zero quickly to simulate hover of the MAV. Firstly, X and Y velocities were checked, as only optic flow was needed to calculate these velocities. Since loom was calculated later, this is when Z velocity and position was calculated. Several simulations were run with changes in the input values to check that the algorithm worked.

   Initially, tests were run and were only stabilising the velocities, using a different outer loop controller which did not control position. This allowed for the velocities to reach very close to zero so that the MAV was quite effectively hovering. However, there was still some slight drift of the MAV due to the small value of velocity. This resulted in hover not being completely effective, as when the X, Y and Z position plots were examined, there was a change in the position of the MAV.

   To overcome this problem, the outer loop controller in figure 12 was used to control both velocity and position. It was therefore necessary to examine the plots of position in addition to velocity to ensure how effectively the MAV was simulating hover. As is shown in figure 17, the velocities appear to reach exactly zero, which should show the MAV maintaining a constant position with no drift. Figure 17 also contains the corresponding plots of position for the same simulation that was run for 80 seconds.

   1. X and Y velocity and position
      Figure 17 shows the plots of x and y velocity and position against time for a simulation of 80 seconds. Figure 17(A) is a plot of the x-velocity against time and figure 17(B) is a plot of the x-position against time. As is shown in the figure 17(A), there is some movement rearward and then forward until the velocity reaches zero at approximately 30 seconds. Since the x-velocity is maintained at zero by the controller for the remainder of the simulation, the MAV is simulating hover in the x-direction. This shows that the controller was effective in stabilising the MAV to simulate hover and optic flow can be used to calculate the velocity in the x-direction.

      As can be seen in appendix L, the initial position of the MAV is at -130m. Therefore the plot of position starts at -130m, as shown in figure 17(B) and since the x-velocity changes (shown in figure 17(A)), this
corresponds to movement in the x-position until it maintains a constant position at approximately -130.26m. This occurs after approximately 30 seconds, which corresponds to the time taken for the x-velocity to reach zero. This shows that the MAV has maintained hover in the x-position.

Figure 17(C) is a plot of the y-velocity against time for and figure 17(D) is a plot of the y-position against time. As is shown in the figure, there is some translation to the left and then back to the right until the velocity reaches zero at approximately 25 seconds. Since the y-velocity is maintained at zero by the controller for the remainder of the simulation, the MAV is simulating hover in the y-direction. This shows that the controller was effective in stabilising the MAV to simulate hover and optic flow can be used to calculate the velocity in the y-direction.

As can be seen in appendix L, the initial position of the MAV is at 80m. Therefore the plot of position starts at 80m, as shown in figure 17(D), and since the y-velocity changes (shown in figure 17(C)), this corresponds to movement in the y-position until it maintains a constant position at approximately 79.37m. This occurs after approximately 25 seconds, which corresponds to the time taken for the y-velocity to reach zero. This shows that the MAV has maintained hover in the y-position.

2. Z velocity and position

Figure 18 shows a plot of z velocity and position against time for a simulation of 80 seconds. This was simulated with an external sensor to measure z velocity and position, as loom was not yet used to estimate the z velocity and position. This was done to firstly test the estimate of optic flow in calculating x and y velocity and position, as well as the ability of the controller to stabilise the MAV to hover. Figure 18(A) is a plot of the z-velocity against time. As is shown in the figure, the velocity increases and then decreases before reaching zero after approximately 22 seconds. The z-velocity is then maintained at zero, showing that hover is being simulated. Figure 18(B) is a plot of the z-position against time for 80 seconds. As can be seen in appendix L, the initial position of the MAV is at -2m, which corresponds to a height of 2m above the ground. Therefore the plot of position starts at -2m, and since the z-velocity changes (shown in figure 18(A)), this corresponds to...
movement in the z-position until it maintains a constant position at approximately -2m. This occurs after
approximately 22 seconds, which corresponds to the time taken for the z-velocity to reach zero. The controller is
therefore effective in stabilising hover in the z-direction. Optic flow is therefore suitable to use for hover in 2D,
using an external sensor to simulate hover in the z-direction.

3. Summary
Since optic flow was used to simulate hover in the x and y-directions, it is therefore suitable for use in
simulating 2D snapshot hover. This was done using an external sensor to simulate hover in the z-direction, and
therefore does not effectively simulate 3D snapshot hover. The proposed method for estimating the height is
therefore used to test simulations of 3D snapshot hover.

C. Estimating the height and 3D snapshot hover

Figure 18. Plot of Z velocity and position against time for t=80s. Figure 18(A) shows the plot of Vz
against time for t=80s and figure 18(B) shows the corresponding plot of Z against time for t=80s.

1. Comparison of the height estimation to actual height
Estimating the height was done by integrating the loom, as explained. To check how accurate the estimation
of the height was, it was first calculated and compared to the actual height of the MAV. This was done with a
various estimates of the height close to the actual value of the height, which was -2m. This enabled checking the
effectiveness of the approximation of integrating the loom to obtain the height. The initial estimate was set to -2m,
as this was the actual value of the height and would isolate other factors and purely show how effective the
approximation is. Figure 19 shows the results of the comparison.
This shows that there is some discrepancy between the actual and the estimated values. As is shown in figure 19(A), the estimated height drops to almost -0.8, which is a height of 0.8m above the ground, whereas the actual height drops further to only -1.6m, which is a height of 1.6m above the ground. The estimated height then returns to a value lower than that of the actual height, again showing some discrepancy between the values. The estimated vertical velocity is also different from the actual vertical velocity. The plot shows that the estimated curve increases to 0.5m/s, then decreases to -0.2m/s, whereas the actual curve only increases to -0.2m/s and decreases to 0.1m/s, before both curves stabilise on zero. Although there is some error in the values as shown by the curves, the general trend of the curves is much the same. The error is expected due to the fact that the method of integrating the loom to obtain a height estimate is only an approximation, and should not be the same value as the actual height. Therefore, the results are acceptable to use in the simulation.

This was then done changing the initial estimate by increments of 0.1m, which would show how close the initial guess of the height had to be to the actual height for the simulation to work. After changing the value of the initial guess and running a simulation each time the guess was changed, it was found that the estimate of the height had an upper limit of -3.2m and lower limit of -1.7m. Figure 20 shows a plot of the comparison of the actual height to the estimated height for an initial guess of -1.6m and -3.3m, which were lower and upper limits for which the approximation strayed too far from the actual value and the trend in the plots were not the same. What can also be deduced from the plots in figure 20, is that if the initial guess is too far from the actual height, then the algorithm will not be able to simulate hover. Particularly evident in figure 20(B) and to an extent in 20(A), the yellow (lighter) line of the height does not stay constant after 40 seconds of the simulation. When the plot of x, y and z velocity and position were examined, they showed that the MAV did not hover after the 40 seconds, with the velocities not tending to zero, and position was not constant. Therefore, for a height of 2m, it is better to overestimate the initial guess of the height slightly, and hover can be simulated if the initial guess is less than 3.2m. This estimate is quite achievable considering the room for error allowable and the order of magnitude of the height.

The estimated height and vertical velocity were then used as the inputs to the collective pitch control of hover, to see if using this estimation of height could be used to simulate hover. The initial guess was set back to -2m, to again check how effective the approximation is. The results are shown for x, y and z-velocity and position, which are comparable to the results for 2D snapshot hover.

2. X and Y velocity and position

Figure 21 shows the plots of x and y velocity and position against time for a simulation of 40 seconds. Figure 21(A) is a plot of the x-velocity against time for and figure 21(B) is a plot of the x-position against time. As is shown in the figure 21(A), there is some movement rearward and forward until the velocity reaches zero at approximately 30 seconds. Since the x-velocity is maintained at zero by the controller for the remainder of the simulation, the MAV is simulating hover in the x-direction. This shows that the controller was effective in stabilising the MAV to simulate hover and optic flow can be used to calculate the velocity in the x-direction.

![Figure 20. Comparison of the estimation to actual values for the limits. Figure 20(A) is a plot of the height for 40 seconds of the simulation for an initial guess of -1.6m for the height. The yellow (lighter) line indicates the actual height and the purple (darker) line indicates the estimated height. Figure 20(B) is a plot of the height for 40 seconds of the simulation for an initial guess of -3.3m for the height. The yellow (lighter) line indicates the actual height and the purple (darker) line indicates the estimated height.](image)
As can be seen in appendix L, the initial position of the MAV is at -130m. Therefore, the plot of position starts at -130m, as shown in figure 21(B) and since the x-velocity changes (shown in figure 21(A)), this corresponds to movement in the x-position until it maintains a constant position at approximately -130.26m. This occurs after approximately 30 seconds, which corresponds to the time taken for the x-velocity to reach zero. These results can be compared to results for the 2D snapshot hover, as there is slightly more movement in the x-velocity when comparing figure 21(A) to figure 17(A). This then corresponds to the less smooth change in x-position shown in figure 21(B) compared to the smooth transition to a constant x-position in figure 17(B). Nevertheless, the results show that the MAV has maintained hover in the x-position.

Figure 21(C) is a plot of the y-velocity against time for and figure 21(D) is a plot of the y-position against time. As is shown in the figure, there is some translation to the left and then back to the right until the velocity reaches zero at approximately 25 seconds. Since the y-velocity is maintained at zero by the controller for the remainder of the simulation, the MAV is simulating hover in the y-direction. This shows that the controller was effective in stabilising the MAV to simulate hover and optic flow can be used to calculate the velocity in the y-direction.

As can be seen in appendix L, the initial position of the MAV is at 80m. Therefore the plot of position starts at 80m, as shown in figure 21(D), and since the y-velocity changes (shown in figure 21(C)), this corresponds to movement in the y-position until it maintains a constant position at approximately 79.37m. This occurs after approximately 25 seconds, which corresponds to the time taken for the y-velocity to reach zero. These results are comparable to the 2D snapshot hover simulation in figure 17. Again, as for x-velocity and position, there is a less smooth transition to zero velocity and constant position when compared to the 2D snapshot hover case. Nevertheless, the results show that the MAV has maintained hover in the y-position.

3. Z velocity and position

Figure 22 shows a plot of z-velocity and position against time for a simulation of 40 seconds. Since loom was used to approximate the velocity and position in the z-direction, and the height was approximated with an
initial guess of the height and integrating the loom, this was key to testing the robustness of the algorithm in 3D. As is shown in the figure, the algorithm to calculate z-velocity and the controller are quite accurate in stabilising hover. Figure 22(A) is a plot of the z-velocity against time. As is shown in the figure, the velocity increases and then decreases before reaching zero after approximately 20 seconds. The z-velocity is then maintained at zero, showing that the MAV is simulating hover. Figure 22(B) is a plot of the z-position against time for 40 seconds. As can be seen in appendix L, the initial position of the MAV is at -2m, which corresponds to a height of 2m above the ground. Therefore the plot of position starts at -2m, and since the z-velocity changes (shown in figure 22(A)), this corresponds to movement in the z-position until it maintains a constant position at approximately -2.05m. This occurs after approximately 20 seconds, which corresponds to the time taken for the z-velocity to reach zero. The controller is therefore effective in stabilising hover in the z-direction. Comparing these results to the 2D snapshot hover case, where an external sensor was used to obtain the height, these results are quite accurate. Figure 18 shows that there is some movement in the z-velocity and that the z-position is maintained at -2m, whereas figure 22 shows more movement in the z-velocity and the z-position is at -2.05m, slightly different to the 2D case. Nevertheless, snapshot hover is achieved using an estimation of the height, thus simulating 3D snapshot hover.

4. Summary
Since optic flow, loom and the proposed method for estimating the height were used in this simulation, this shows that 3D snapshot hover was quite well achieved. As shown, there is some discrepancy in the estimated z-velocity and position when compared with the actual z-velocity and position. Also, the initial guess of the height has to be quite close to the actual height of the MAV, which was set to -2m. However, the simulation was nevertheless quite effective in simulating 3D snapshot hover, despite some slight differences when compared with the results produced using an external sensor to measure z-velocity and position.

D. Effect of noise
To evaluate the simulation against noise, a “Bandwidth-Limited White Noise” block from Simulink was used to simulate a noisy signal input to whichever variable was chosen. This noisy signal could be added to any variable as desired, such as optic flow, loom, velocity, position, etc. This would test the robustness of the calculation of optic flow as well as the effectiveness of the controller. The variables chosen to test the simulation were optic flow and loom. This is because, these are the variables most likely to encounter effects of noise if the MAV were to practically achieve hover. The problems associated with using optic flow for hover stem from the approximations used when calculating optic flow and loom, and noise is a factor which can simulate these problems. Noise was added in the same way to optic flow used to calculate velocity and loom. As will be shown in all cases, adding noise resulted in hover not being perfectly simulated.

1. Noise added to optic flow for position
When noise was added to optic flow, the MAV was able to simulate hover for the z-velocity and position quite well, however, there was a slight drift in the x and y position. The velocities were around zero, despite that they were oscillating about zero. As is shown in figure 23, despite that the plots of position appear to drift back and forth from the datum, the order of magnitude of the shift is in centimetres. When there was no noise, results showed that hover was reached after 25-30 seconds. From 20 seconds to 40 seconds, the x position shifts approximately 5cm one way and then 6cm the other, while the y position moves 4cm one way and 5cm the
other. Over this amount of time, these drifts are not significant. Therefore the calculation of position using optic flow is quite robust to noise, as the plots indicate that the hover is still simulated with an insignificant drift.

2. Noise added to optic flow for velocity

Figure 24 shows the plots of x and y velocities against time for a simulation of 40 seconds, as this is what was directly affected by adding noise to the optic flow calculation for velocity. As is shown in the figure, both velocities bounce around zero. Although it does not appear to be quite significant, this corresponds to a larger increase in the position shift. Because the velocities can be up to 0.6m/s this can correspond to a shift of 6m in 10 seconds, which is quite significant. However, because the velocities hover around zero, the maximum shift in 10 seconds for the position was 2m. This shows that the noise has quite a significant impact when added to the calculation of optic flow for velocity.

3. Noise added to optic flow for both position and velocity

This had an even larger impact, as is expected, than only adding noise to either position or velocity optic flow calculations. Figure 25 shows the plot of x and y-position against time for 40 seconds of the simulation, as these were affected the most. As is shown, changes of approximately 3m over 10 seconds is quite significant and shows that the simulation is quite sensitive to noise added to the calculation of optic flow for both position and velocity. Although the shifts in position are back and forth about the datum, these shifts are quite significant over the time period taken.
4. Noise added to loom

Adding noise to the loom did not affect the x and y velocity and position of the MAV. It did, however, have a significant effect on the z-velocity and position, as is expected. Figure 26 shows plots of the z-velocity and position against time for a simulation of 40 seconds. This shows that the z-velocity changed over the time period, corresponding to a considerable change in z-position, thus showing that the simulation did not effectively maintain a z-position during the simulation. This shows that 3D snapshot hover was not simulated with a noisy signal from the loom. This suggests the requirement for loom damping, which is discussed later in the launch to a height and also in the recommendations and future thesis work.

![Figure 25. Plot of X and Y position when adding noise.](image)

Figure 25(A) is a plot of the x-position for 40 seconds of the simulation. Figure 25(B) is a plot of the y-position for 40 seconds of the simulation.

![Figure 26. Plot of X and Y position when adding noise.](image)

Figure 26(A) is a plot of the z-position for 40 seconds of the simulation. Figure 26(B) is a plot of the z-velocity for 40 seconds of the simulation.

E. Launch to a certain height

Launching to a certain height was done as explained and shown in figure 13. Launching to a height was expected to yield similar results for x and y velocity and position as obtained in previous results. What would be different, however, was the z position. It was expected that the z position would not reach a constant value as soon as the other results achieved this. This is because of the time taken for the controller to adjust for movement of the launch and adjust its position accordingly. Furthermore, if the launch occurred to a height that strayed too far from the predicted height and a snapshot was not obtainable, the collective would be too high and the simulation would continue to increase in height. Some form of loom damping would therefore be required to control a significant overshoot from the datum height.
Since the collective had a step input, controlling it too would also take time to maintain a constant value. A plot of the collective from the launch is compared with the collective from the 3D snapshot hover case in figure 27.

![Figure 27](image)

**Figure 27. Comparison of Collective for launch against simple case.** Figure 27(A) is a plot of the collective against time for 40 seconds for the launch simulation. Figure 27(B) is a plot of the collective against time for 40 seconds for the simple 3D snapshot hover case corresponding to the plots in figures 21 and 22.

In the figures, the negative corresponds to an increase in height, and a step decrease was used to simulate this. The collective in figure 27(A), from the launch, shows the step decrease of the collective, whereas the collective in figure 27(B) shows a smooth decrease to the minimum value on the plot. Figure 27(B) maintains a smooth curve throughout until it maintains a constant value after 15 seconds. However, the collective in figure 27(A) shows a much steeper decrease and to a much lower value or approximately 50 less due to the step increase. Furthermore, it does not have as smooth a transition until it reaches a constant value after 20 seconds.

1. **X and Y velocity and position**

Since the launch did not have a significant effect on the x and y velocity and position, only the plots of x and y position are shown in figure 28. When compared with figure 21 from the simple 3D snapshot hover case, there was not a noticeable difference in the velocity plots, and only a slight difference in the position plots. Both position plots appear to be slightly less smooth, but take effectively the same amount of time to simulate hover as in the simple 3D case. This shows that launch does not have a significant effect on x and y position and velocity.

![Figure 28](image)

**Figure 28. Plot of X and Y position for the launch.** Figure 28(A) is a plot of the x-position for 40 seconds of the simulation. Figure 28(B) is a plot of the y-position for 40 seconds of the simulation.
2. **Z velocity and position**

As shown in figure 29, simulating a launch has quite an effect on the z position and velocity, which is expected. When comparing figure 29 to figure 22, the simple 3D snapshot hover case, there is quite a significant difference in the plots of position and velocity. Due to the launch it is expected that there will be a much greater shift from the datum, due to the initial overshoot of the launch from the datum. The simulation shows a much sharper increase in downward velocity to a much greater magnitude, when compared with figure 22. This corresponds to a much greater decrease in height, as the simulation decreases to a height of 0.2m above the ground. Hover is then simulated after approximately the same time as in figure 22, but the height is maintained at almost 1m above the ground, rather than 2m.

Furthermore, it was noticed that too large a step decrease (simulated by increase the time of the step), resulted in a continual decrease of the height. Figure 30 shows a plot of the z and x position that resulted from too large a step increase. The z position is shown to continually decrease through the ground, and the x position (which is similar to the results for y position) reaches a constant height, but has a very noise signal about the constant height. This occurred due to overshooting the datum, which may be a factor to consider for future projects. This may be overcome by use of some loom damping prior to the snapshot simulating hover. This would remove the stray of the launch height away from the datum to hover.

![Figure 29. Plot of Z velocity and position for the launch.](image1)

**Figure 29.** Plot of Z velocity and position for the launch. *Figure 29(A) is a plot of the z-position for 40 seconds of the simulation. Figure 29(B) is a plot of the z-velocity for 40 seconds of the simulation.*

![Figure 30. Plot of Z and X position for the launch with a large step increase.](image2)

**Figure 30.** Plot of Z and X position for the launch with a large step increase. *Figure 30(A) is a plot of the z-position for 20 seconds of the simulation. Figure 30(B) is a plot of the x-position for 20 seconds of the simulation.*
3. Summary
Launching to a certain height did not have a significant effect on the x and y velocity and position, but effects on z velocity and position were noticeable. Although there was significant difference to the height in the simple 3D snapshot hover case, hover was still simulated. However, increasing the step size of the collective did impact the z velocity and position significantly and hover was not simulated.

F. Summary of results
The simulation was able to effectively simulate both 2D and 3D snapshot hover. Although there was some discrepancy in the height estimate from the actual height, it was quite an accurate method to obtain the height and effectively simulate 3D snapshot hover. Inputting noise had quite an effect on the results. It showed that the MAV was not able to simulate exact hover, and that it shifted slightly in position and was not able to maintain zero velocity. In some cases noise had a significant impact on hover, despite that changes in position were about the datum. Although there were slight changes compared to the simple 3D snapshot hover case, hover was still simulated, with launching to a certain height, provided the step increase for the launch was not too significant to cause a large stray from the datum.

XII. Recommendations and Future Work
Optic flow is currently a very widely researched area. Optic flow has already been used in robotics to achieve various controls of flight. However, this research has relied on the use of either a GPS or laser rangefinder. Due to the size of laser rangefinders and their impractical use on MAVs, as well as the problems with GPSs in cluttered environments, research has been extended to the use of snapshot hover to overcome the reliance on these technologies. Snapshot hover makes use of other smaller inertial sensors and allows for use on MAVs. This was simulated in this thesis.

Since this was only a simulation, however, there remains much more work prior to its actual implementation. Firstly, overcoming the problems shown in simulating a launch to height and noise is necessary. Secondly, the use of graphics and imagery with the algorithm will better test for robustness against noise. Thirdly, using the same imagery with different brightness to appropriately test the intensity constancy assumption. Finally, after testing the simulation in several different conditions with these more realistic situations, the algorithm can then be tested for practical use on an MAV.

The launch to a height was still effective in simulating hover, provided the step size was not too large. When the step size of the collective was too large, the datum point was lost, and the simulation had no reference to control the height. This may be achieved through the use of loom damping prior to controlling the height. It is recommended that this be examined prior to any additional progressions of the simulation. Loom damping may also aid in reducing the effects of noise added to the loom, which caused hover to not be simulated.

Graphics and imagery can be used in Simulink with the simulation in addition to using inertial sensors for the state of the MAV. Using graphics and imagery will enable the calculation of the pixel shift in the image from the optic flow calculations, simulating what the camera on an MAV would see and thus making the simulation more realistic. Using actual graphics and imagery would test the robustness of the algorithm, as there will be other factors affecting the calculation of optic flow such as noise. Although the effects of noise have been tested for in this simulation, the graphics and imagery will provide a more realistic noisy signal to affect the optic flow calculation.

In addition to using different graphics and imagery to test for robustness of the algorithm, the same images of different brightness should also be used in the simulation to test the intensity constancy assumption. As explained previously, calculations of optic flow are approximated based on the intensity constancy assumption. But this is violated at points in the image region that contain transparency, specular reflections, shadows, or fragmented occlusion [7]. This also leads to the aperture problem because at edge points in an image, or when the image is viewed through an aperture, there is a lack of information regarding the tangential component of the velocity at that point [9]. This can be somewhat overcome by using multiple optic flow vectors, rather than just a single optic flow vector [8,9], hence in this simulation the image was set to 8 patches in both X and Y directions across 400 pixels in both directions. However, using images of different brightness will have the appropriate noisy signal to determine the robustness of the algorithm. This is especially important for hover, because for extended periods of time, images may change brightness for various reasons. For example, consider an MAV hovering in a shallow valley for an entire day. If the sun is out, the image will appear quite bright, however as the day progresses, clouds may obscure the sun slightly changing the brightness of the image, or later in the day the sun will set behind the mountains, which will gradually change the brightness of the image due to the shadows that will be viewed in the camera. This is an important aspect to test prior to use practically with an MAV.

Once all these tests and simulations are complete, it will be ready for use practically with an MAV. This is essentially the final work that is required to be achieved for hover of an MAV. Once tests are complete for
practical use, this is likely to be amalgamated with other research to achieve a fully autonomous MAV. Other research includes aspects of the MAV’s flight such as forward flight and obstacle avoidance. Once research is complete and a fully autonomous MAV has been developed, this is likely to be used in various military applications. Due to overcoming the need for GPS and laser rangefinders, MAVs will be used in various environments that they could not be used in previously, such as indoors, cluttered environments and with changes to the environment. This will provide the military with a very strong technological advantage due to the capability that a fully autonomous MAV can provide.

XIII. Conclusions

Insect vision is quite remarkable in the sense that it is used to control various aspects of their flight. Flies, bees, and also grasshoppers exhibit the ability to use visual sensors in a mechanosensory feedback system to control their forward flight, landing, response to changes in the environment and also ability to hover. These aspects of biological vision have inspired research in the field of robotics to create and simulate systems that are able to replicate these responses to visual sensors. Much research into this area has shown that it is possible to replicate insect flight control behaviour using flapping wing, fixed wing and rotary wing implementations.

Currently, hover can be achieved using a GPS, laser rangefinder or radar in conjunction with a control system to calculate optic flow. However, research has suggested that insects such as bees hover using motion cues derived from visual sensors. Due to the associated problems with these technologies such as bulkiness, cost and operability in cluttered, indoor or urban environments, it is desirable to achieve snapshot hover which eliminates the use of such technologies.

This paper has proposed a method to achieve snapshot hover for an MAV. MATLAB and Simulink were used to simulate hover for an MAV using the proposed method. The proposed method was using the equations derived for optic flow and for the loom. Using these equations, velocity and position could be calculated in 3D. The height estimation was done using an initial guess of the height and using the loom calculation. The controller used PID controllers to simulate controlling the velocity and position of the MAV. MATLAB was used to program the equations and Simulink was used to run the calculations and the entire simulation.

The results of the simulation firstly showed that the algorithm was quite successful in simulating 2D snapshot hover. This was done assuming the height was known from another source, and the results showed that the velocity could be maintained at zero and position could be constant in x, y and z. This showed that optic flow could be used to simulate snapshot hover in 2D.

Loom was then used to estimate the height, which required in initial guess of the height. Comparing the estimated height and velocity and the actual height with an initial guess of the height of 2m (which was the actual height), showed some discrepancy between the plots. This was expected, however, due to the calculation of the height only being an estimate. After running several simulations in 0.1m increments of the height away from the actual height (i.e. 2.1m, 2.2m, 2.3m etc), it was found that overestimating the initial guess of the height was better. Results showed that up to 3.2m and as low as 1.7m were still allowable guesses for the MAV to still have results quite close to the actual results, and still be able to simulate hover. A guess outside of these bounds, however, showed a significant stray from the actual results and the MAV was not able to simulate hover. Despite the discrepancy in the estimated height from the actual height, 3D snapshot hover was still simulated. The results again showed that the MAV reached zero velocity and constant position in x, y and z, this time with the z position calculated using loom and guessing the initial height.

Inputting noise generally had a small impact and the MAV still simulated hover, despite slight drifts in position. This was the case when adding noise to either the calculation of optic flow for position or velocity. When noise was added to the calculation of optic flow for position and velocity, or the loom calculation, this resulted in a more significant impact on the simulation of hover. In the optic flow case, the MAV shifted about the datum, whereas in the loom case, there was quite a significant impact and the MAV did not simulate hover in the z-position. As discussed, loom damping should be considered in future work to overcome this problem.

Launching to a height was done by setting the collective to an initial value of zero and then using a step to increase the collective to the desired height of the MAV. This showed results of x and y position and velocity much the same as in the simple 3D snapshot hover case. There was some difference in the z position and velocity when compared with the simple 3D snapshot hover case, which was expected. However, it was noticed that increase the step of the collective too much resulted in overshooting the datum by too much, and that a continuous decrease in height was simulated. Again, this may be overcome with loom damping, which is suggested future work for this project.

Overall, using the proposed method effectively simulated 2D and 3D snapshot hover. There was an expected slight discrepancy in the estimated height from the actual height, but 3D snapshot hover was still simulated. The simulation was also able to launch the MAV to a height and then hover, but was slightly affected by noisy inputs of optic flow and loom.
Acknowledgements

The successful outcome of this thesis could not have been accomplished without the contributions provided by various staff members at UNSW@ADFA who generously offered me their time, support and assistance. Without their specialist knowledge, dedication and willingness to help, the completion of this thesis would have been far too difficult and time consuming. It is therefore important to acknowledge their contribution.

Firstly, I would like to thank my thesis supervisor Dr. Matthew Garratt, for his time, mentoring and assistance throughout this project. Thankyou for setting aside time out of your busy schedule to explain concepts; and for giving me assistance when I found things too difficult. Your guidance, patience and understanding through countless difficulties encountered when using MATLAB and Simulink were imperative to the outcome of this project. Without you this project would not have been accomplished.

Secondly, I would like to thank Dr. Andrew Neely, for introducing me to how to conduct and do major project research. Your guidance was well needed in ensuring the production of good research, project work and a final report.

Lastly, I would like to thank for family and friends for their support, understanding and sacrifices made in order for me to conduct this research project. There were times that I was very busy, and I appreciate your understanding during these busy times.

References


