Faculty of Engineering and Built Environment

School of Engineering

Mechancial Engineering Project B

Semester 1 - 2019

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

FINAL YEAR PROJECT

PROJECT TITLE

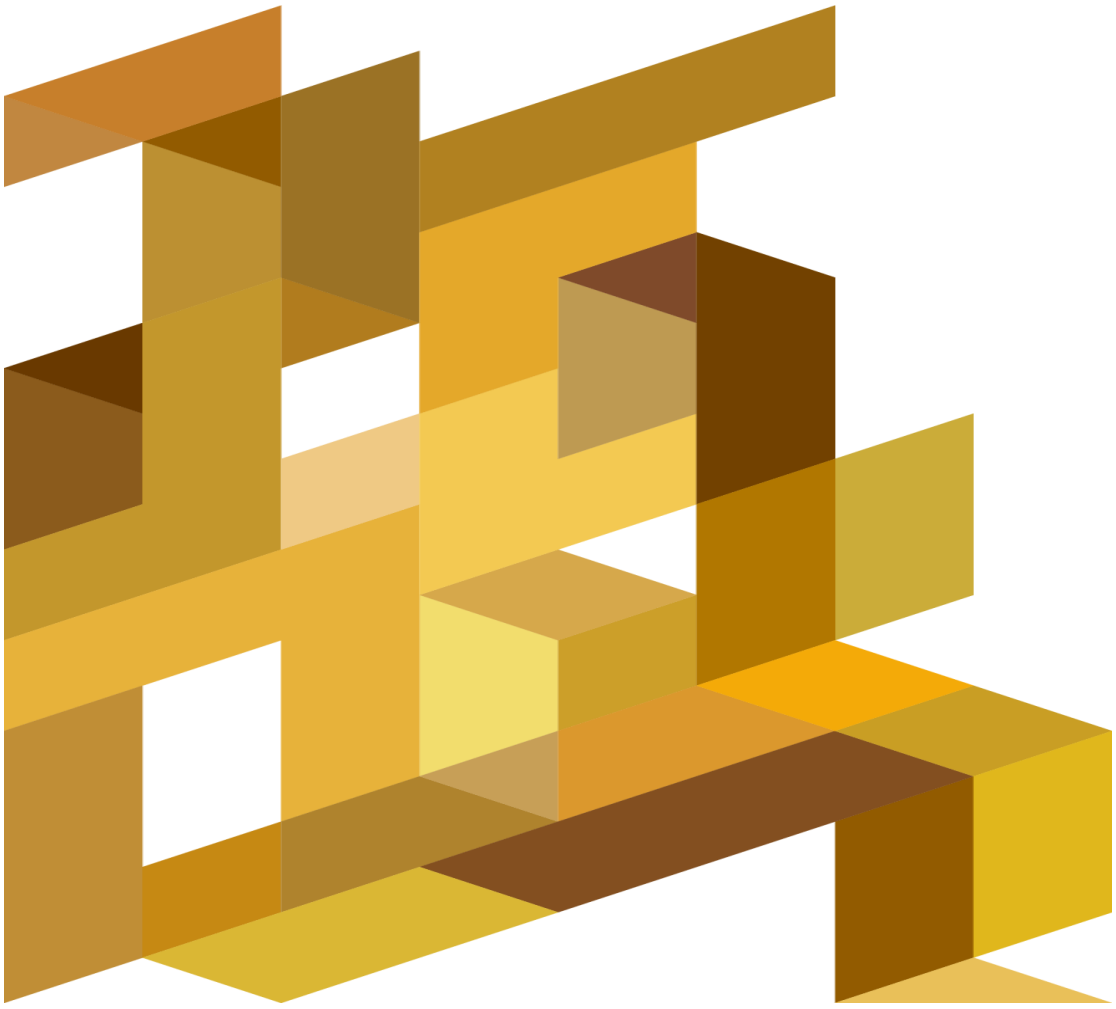
Optical Flow Based Obstacle Avoidance for a Fixed Wing UAV

NAME

Patrick Prell

SUPERVISOR

Dr Christopher Renton



School of Engineering, Discipline of Mechanical Engineering and Mechatronics, pp. 1–29

Optical Flow Based Obstacle Avoidance for a Fixed Wing UAV

Final Year Project Report - MECH4841 Part B

June 2019

Patrick Prell [1]

[1] *Student of Mechatronics Engineering,*
The University of Newcastle, Callaghan, NSW 2308, AUSTRALIA
Student Number: 3204734
E-mail: Patrick.Prell@uon.edu.au

Abstract

Remember that executive summary may include the following information:

- Defines the intention of the report.
- Places the report in context so the reader knows why it is important to read it.
- Why is it important?
- What problem is addressed?
- Briefly states the results
- Briefly presents the implications and recommendations

Acknowledgements

You may like to say thank you to someone that helped you with your project.

Contents

1. Introduction

Exploring and navagating unknown environments is an essential instinct in biology and avoiding obstacles is an major component. For robotic vehicles to perform high level tasks, they must also have these instincts.

- **Position**: Show there is a problem and that it is important to solve it.
- **Problem**: Describe the specifics of the problem you are trying to address
- **Proposal**: Discuss how you are going to address this problem. Use the literature to back-up your approach to the problem, or to highlight that what you are doing has not been done before

The rest of the report is organised as follows. Section 2 describes items related to the core content. Section 8 concludes the report. Appendix shows an example of how to make a Table.

2. Background

To the average human or most mammals for that matter, visual tasks such as identifying objects and interpreting visual cues might seem like a trivial task. However, implementing algorithms that mimic a mammalian visual cortex is infuriatingly difficult and largely remains an unsolved problem (Hartley and Zisserman, 2003). One fact often overlooked is that for all of its amazing abilities, approximately 1/3 of the human brain is dedicated to analysing information collected from our eyes. This statistic helps to put into perspective how challenging the field of computer vision can be.

The concept of computer vision has been explored since the inception of artificial intelligence in the late 1960s. Computer vision was made possible by leaps in computer power, allowing manipulation of large (for the time) data sets such as images and videos. The following decade saw research attempting to extract information from a sequence of images using visual cues embedded in the motion of an image. This brought about research on visual navigation as a viable method for a mobile robot to navigate an environment. Using Visual Navigation on a robot to move through an unstructured environment with no prior knowledge of the map is often based on optical flow. A motion field is estimated from fusing the motion of the robot with the motion of the pixels of a video feed. Furthermore, visual navigation techniques have been inspired by biology. Observing the way that animals and insects utilise their visual systems to move about and map their environments, the processes for which this is achieved can be replicated and implemented on a robotic system (Altshuler and Srinivasan, 2018).

The following section builds the background knowledge required to utilise visual cues as input to an obstacle avoidance system. In particular, a history of visual navigation is outlined; the geometry of a camera and the tools for calibrating it are introduced; and the kinematics of vision optical flow and optical flow on the view sphere. Finally, this section discusses the approach taken to validate the obstacle avoidance algorithm.

2.1. Biomimicry

Biology so often holds a wealth of inspiration for engineers. From the material science of the microstructures in a butterfly wing to the computational fluid dynamics of a Kingfisher diving into water (Benyus, 1997). visual sensors are found so frequently in nature, making it very productive to look at nature and tap into the eons of knowledge distilled by evolution.

2.1.1. Inverse Optimal Control

The traditional way of designing an autonomous system is to create a cost function for the system and use this cost function to produce an optimal policy. This method often involves tweaking tuning prametrs and weight values that are too complicated to understnd intuitively. This is known as optimal control. as the name suggests, inverse optimal control flips these techniques around. Inverse optimal control can be a far more intuitive method of obtaining the desired behavior from a robot.

Rather than defining a cost function that generates the policy for the robot, inverse optimal control uncovers the cost function that best explains demonstrated optimal behavior (Ratliff, ????). This behavior can be obtained through the intuition of the mechatronics engineer designing the robot. Or this behavior can be derived from biology, such as the work by

2.1.2. Visual Navigation

2.2. Optical Flow

In computer vison we are constanly running into the problem of dimentional compression. A lot of information is lost in the projection from 3 dimentions to a 2D plane, and when that projection is inverted, the lost data needs to be inferred. In the case of optical flow, the lost data comes mainly from different assumptions that are made to make the optic flow algorithm solvable. Each Optical flow algorithm has its own set of assumptions and constraints.

Beginnning with a basic description of optical flow. Optic flow is the apparent movement of objects, textures and edges between a sequence of frames. Optic flow can be thought of as the time derivitive of two images. It is dificult to imagine daking the derivitive of an image, and it is this which makes evaluating optic flow so dificult. One major probem with calculating optic flow is the *aperture problem*. This probelm manifests in many optical illutions that trick the human brain aswell. An example of this effect can be seen in 1. To an unassuming camera, it is impossible to tell the direction of the striped pattern.

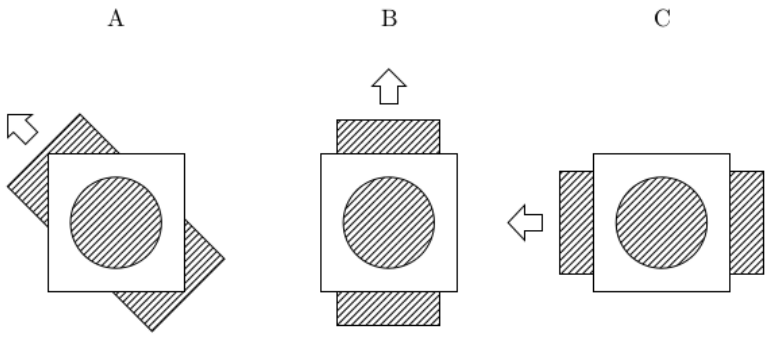


Figure 1: The direction of the stripes through the aperture is unclear without more context

There have been many solutions proposed to calculate the motion of pixels in an image. Each method being designed for different tasks and having their own set of advantages and disadvantags for their particular situation. The following section outlines two early methods for calculating optical flow, these being the *Horn and Schunk* and *Lucas - Kanade* method. Although both of these methods were devised over 20 years ago, they have layed a foundation for many optic flow algorithms following (Sun et al., 2010). The farnback algorithm, a much more recent algoithm will then be discussed as this is the algorithm used for the project.

2.2.1. Horn & Schunk

Horn and Schunck optic flow is based around an assumption known as the *Brightness constancy*. This says that the intencity of a pixel remains constant over a small step in time.

$$\frac{dI}{dt} = 0 \tag{2.1}$$

Defining a vector field $h = (u, v) = \left(\frac{du}{dt}, \frac{dv}{dt}\right)$ that hold the brightness constancy true, the chain rule is used to derive the following *optic flow constraint* equation.

$$\nabla I\mathbf{h} + I_t = 0 \tag{2.2}$$

This equation exposes the aperture problem in 1 mathmatically. There are more variables to solve for than there are linearly independent equations meaning there is no unique solutioin for these variables. in Horn and Schunck (1981), This problem addressed by formulating the estimation of optical flow as a minimisation problem. where the vector field (u, v) is presented as the minimiser for an energy function $J(u, v)$. this energy function is made up of the optic flow constraint based on brightness constancy, and another term based on the gradient of the flow. The following energy function is as follows:

$$J(u, v) = \int_{\Omega} (I_x u + I_y v + I_t)^2 + \alpha^2(|\nabla u|^2 + |\nabla v|^2) \tag{2.3}$$

where α is the weight of the smoothness of the resulting flow field.

Horn and Schunck's optical flow algorithm was a huge step forward in computer vision research, although this algorithm is limited in that it typically can only estimate small motions, the method fails in the presence of large motion when the gradient of the image is not smooth enough (Meinhardt-Llopis et al., 2013).

2.2.2. Lucas - Kanade

The Lucas-Kanade method is a widely used method for calculating optical flow. This method operates under the assumption that the flow is effectively constant in a small cluster of pixels, the method can then solve the optical flow equation for all the pixels in that neighbourhood using least squares regression. The Lucas-Kanade method improves on previous methods by using the spatial intensity gradient to find the disparity vector h, reducing the algorithms calculation time to $O(M^2 log(N))$ for the average case.

2.2.3. What is Observable in Optical Flow

Pose rate estimation

- Angular velocity,
- reflect on the work presented,
- make recommendations,
- suggest future work or improvements.

2.3. Reference Frames and Coordinate Systems

In general, the motion of an object needs to be described relative to some coordinate system and the reference frame. Defined by Perez (2014), a reference frame is a perspective from which the motion is described by an observer. A reference frame can be defined by a set of at least 3 non-colinear points that are rigidly connected. The reference frames used in this project are $\mathcal{C}$, $\mathcal{B}$ and $\mathcal{N}$ for the camera, the vehicle (body) and the world frames respectively.

Furthermore, a coordinate system is a way to describe the position and motion of objects relative to its reference frame. A coordinate system in three-dimensional Euclidean geometry is made up of at least three orthogonal basis vectors of unit length. This report uses the same notation as used in Perez (2014) to describe a vector from point O to point P as $\vec{r}_{P/O}$. Figure 2 shows an example reference frame $\mathcal{C}$ attached to a camera with an associated coordinate system $\{c\}$.

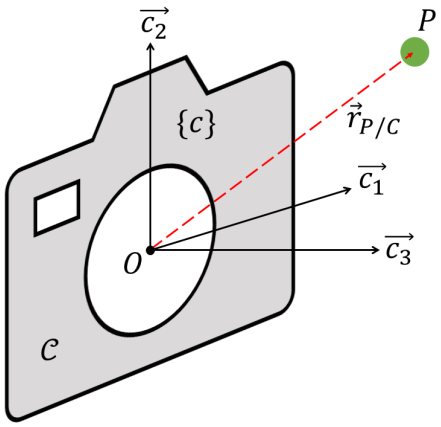


Figure 2: A reference frame $\mathcal{C}$ with its associated coordinate system $\{c\}$

When designing computational models, it is often convenient to represent vectors as column matrices in $\mathbb{R}^3$. The basis of the original vector is denoted as a superscript of the matrix. That is, a vector $\vec{v}$ in basis $\{a\}$ is written as $\mathbf{v}^a$, or more generally:

$$\mathbf{v}^a = \begin{bmatrix} v_1^a \\ v_2^a \\ v_3^a \end{bmatrix} \tag{2.4}$$

In this form, the dot product of two vectors $\vec{u} \cdot \vec{v}$ (inner product) in the same basis becomes a matrix operation:

$$(\mathbf{u}^a)^T \mathbf{v}^a = \begin{bmatrix} u_1^a & u_2^a & u_3^a \end{bmatrix} \begin{bmatrix} v_1^a \\ v_2^a \\ v_3^a \end{bmatrix} \tag{2.5}$$

Additionally, the cross product of two vectors $\vec{w} = \vec{u} \times \vec{v}$ can be represented as the *skew-symmetric* matrix of $\mathbf{u}^a$ multiplied by $\mathbf{v}^a$. That is:

$$\mathbf{w}^a = \mathbf{S}(\mathbf{u}^a)\mathbf{v}^a \tag{2.6}$$

The skew-symmetric operator $\mathbf{S}(\mathbf{u})$ arranges the elements in u such that the resulting matrix is skew-symmetric.

$$\mathbf{S}(\mathbf{u}) = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \tag{2.7}$$

2.3.1. Coordinate Transforms and Rotation Matrices

This project uses multiple coordinate systems to deal with the complex geometry of the camera and vehicle. Therefore, in order to have context for measurements and actions of different sensors and actuators, their orientation needs to be transformed into the relevant coordinate system. The following report will use rotation matrices, as derived from a coordinate transform.

A vector can be described from multiple bases, for instance, $\vec{r}$ can be represented in both basis $\{n\}$ and basis $\{c\}$ as:

$$\begin{aligned} \vec{r} &= r_1^n \vec{n}_1 + r_2^n \vec{n}_2 + r_3^n \vec{n}_3 \\ \vec{r} &= r_1^c \vec{c}_1 + r_2^c \vec{c}_2 + r_3^c \vec{c}_3 \end{aligned} \tag{2.8}$$

Expressing $\vec{r}$ as an inner product of the basis $\{c\}$ and r^n:

$$\vec{r} = \begin{bmatrix} \vec{c}_1 & \vec{c}_2 & \vec{c}_3 \end{bmatrix} \begin{bmatrix} r_1^c \\ r_2^c \\ r_3^c \end{bmatrix} \tag{2.9}$$

Finally, express both sides in $\{n\}$:

$$\mathbf{r}^n = \begin{bmatrix} \mathbf{b}_1^c & \mathbf{b}_2^c & \mathbf{b}_3^c \end{bmatrix} \mathbf{u}^c \tag{2.10}$$

This can be denoted more generally in matrix form:

$$\mathbf{r}^c = \mathbf{R}_n^c \mathbf{r}^n. \tag{2.11}$$

In robotics, it is often desirable to rotate a vector to a particular angle. The method used in this project is known as *euler rotation*. Euler angles discretise a complex three-dimensional rotation into three separate planar rotations about each basis vector. The rotation matrices about each basis vector are given by:

$$\begin{aligned} \mathbf{R}_x(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \\ \mathbf{R}_y(\theta) &= \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \\ \mathbf{R}_z(\psi) &= \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2.12}$$

The order of these rotations effects the final position, for this reason in this project the common robotics convention of *Roll, Pitch, Yaw* (RPY) will be used. The notation for a rotation matrix from $\{n\}$ to $\{c\}$ is given by $\mathbf{\Theta}_c^n \triangleq \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^\top$. using the consecutive planar rotations from equation (2.12) we get $r_a^b = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) r_b^a = \mathbf{R}(\mathbf{\Theta}_c^n) r_b^a$. From this, the structure of a full rotation matrix:

$$\mathbf{R}(\mathbf{\Theta}_c^n) = \begin{bmatrix} c_\psi c_\theta & s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi c_\phi s_\theta \\ s_\psi c_\theta & c_\psi c_\phi + s_\phi s_\theta s_\psi & c_\psi s_\phi + s_\psi c_\phi s_\theta \\ s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \tag{2.13}$$

where $s \triangleq sin(\cdot)$ and $c \triangleq cos(\cdot)$.

Due to the orthogonality of rotation matrices, they have the property that the inverse of the rotation matrix is equal to its transpose:

$$\mathbf{R}^{-1} = \mathbf{R} \tag{2.14}$$

This is useful, as rotating a vector back to its original orientation is as simple as applying the transposed rotation matrix. That is $\mathbf{R}_a^b = \mathbf{R}_b^{a\top}$.

2.4. Kinematics of Vision

While discussing the motion of objects in a space, it is helpful to properly define a geometric definition for these points and the relative motion between them. Kinematics defines these such relationships. More specifically, kinematics defines the reference frame from which the motion of a point can be described and the transformation that associates this motion in a separate reference frame. These concepts will be used to describe the relative motion between a camera mounted on a vehicle in an environment.

2.4.1. Time Derivative of Vectors and Transport Theorem

The subtle difference between the time-derivative of a scalar magnitude and a vector magnitude is that the vector derivative depends on the reference frame from which they are being observed Perez (2014). Figure 3 shows a vector $\vec{r}$ observed from two separate reference frames, $\mathcal{A}$ and $\mathcal{B}$.

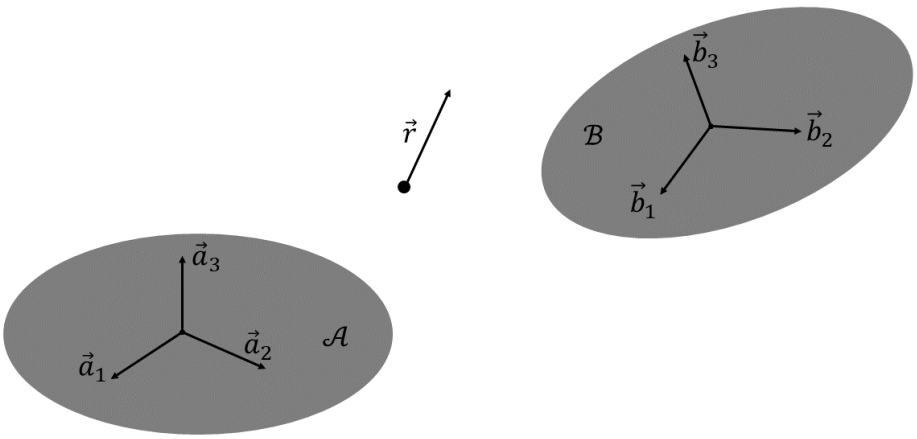


Figure 3: A vector $\vec{r}$ seen from two reference frames.

Taking the time derivative of $\vec{r}$ in each vector frame:

$$\begin{aligned} \frac{^{\mathcal{A}}d\vec{r}}{dt} &= \frac{dr_1^a}{dt}\vec{a}_1 + \frac{dr_2^a}{dt}\vec{a}_2 + \frac{dr_3^a}{dt}\vec{a}_3 \\ \frac{^{\mathcal{B}}d\vec{r}}{dt} &= \frac{dr_1^b}{dt}\vec{b}_1 + \frac{dr_2^b}{dt}\vec{b}_2 + \frac{dr_3^b}{dt}\vec{b}_3 \end{aligned} \tag{2.15}$$

Consider the case of a frisbee flying through the air carrying a spider S as a passenger. Reference frame $\mathcal{F}$ is attached rigidly to the frisbee and an observer O is watching it from the field in reference frame $\mathcal{O}$. Clearly, the movement of the spider is different from $\mathcal{A}$ as it is from $\mathcal{F}$ or more generally:

$$\frac{^{\mathcal{O}}d\vec{r}}{dt} \neq \frac{^{\mathcal{F}}d\vec{r}}{dt} \tag{2.16}$$

To find the time derivative of the spider relative to $\mathcal{O}$, two methods could be used: the time derivative of the vector $\vec{r}_{S/O}$ can be taken, which will prove to be a tedious exercise. Or the *transport theorem* can be used.

In accordance with Perez (2014), the transport theorem states that there exists a unique vector $\vec{\omega}_{\mathcal{B}/\mathcal{A}}$ called the **angular velocity** of $\mathcal{B}$ with respect to $\mathcal{A}$ such that:

$$\frac{^{\mathcal{A}}d\vec{r}}{dt} = \frac{^{\mathcal{B}}d\vec{r}}{dt} + \vec{\omega}_{\mathcal{B}/\mathcal{A}} \times \vec{r} \tag{2.17}$$

Using the transport theorem, the velocity of the spider relative to $\mathcal{O}$ is simply the velocity of the frisbee's centre P plus the rotational velocity of the frisbee, resulting in:

$$\frac{^{\mathcal{O}}d}{dt}\vec{r}_{S/O} = \frac{^{\mathcal{F}}d}{dt}\vec{r}_{P/O} + \vec{\omega}_{\mathcal{F}/\mathcal{O}} \times \vec{r}_{P/O} \tag{2.18}$$

2.4.2. Spherical Flow Model

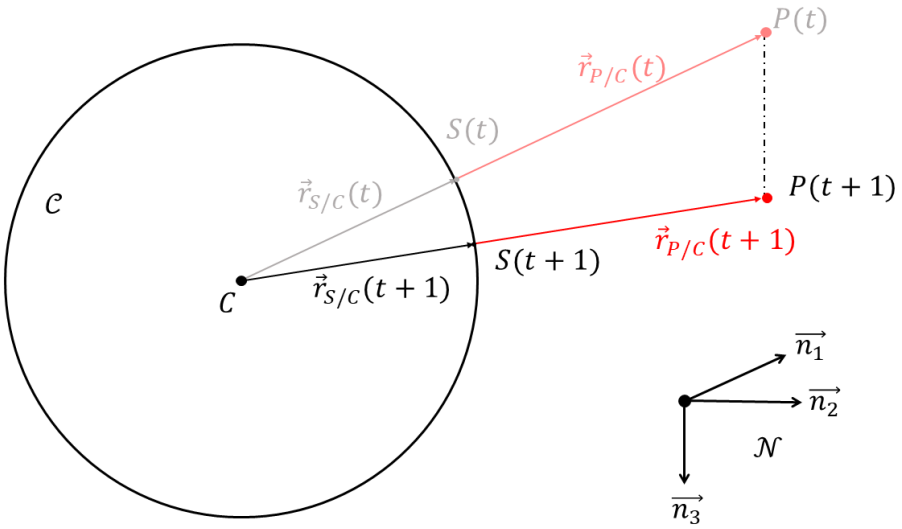


Figure 4: Write something here

Assuming the viewsphere has a unit radius, the point P can be projected onto the view sphere to point S by:

$$\vec{r}_{S/C} = \frac{\vec{r}_{P/C}}{\left\|\vec{r}_{P/C}\right\|} \tag{2.19}$$

Applying the concepts from 2.4.1, the time derivitive of $\vec{r}_{S/C}$ with respect to reference frame $\mathcal{N}$ can be taken:

$$\begin{aligned}
\frac{{}^{\mathcal{N}}d\vec{r}_{S/C}}{dt} &= \frac{{}^{\mathcal{N}}d}{dt}\left(\frac{\vec{r}_{P/C}}{\left\|\vec{r}_{P/C}\right\|}\right) \\
&= \frac{{}^{\mathcal{N}}d}{dt}\vec{r}_{P/C}(\vec{r}_{P/C}\cdot\vec{r}_{P/C})^{-\frac{1}{2}} \\
&= \frac{{}^{\mathcal{N}}d}{dt}\left(\vec{r}_{P/C}\right)(\vec{r}_{P/C}\cdot\vec{r}_{P/C})^{-\frac{1}{2}} + \vec{r}_{P/C}\frac{{}^{\mathcal{N}}d}{dt}(\vec{r}_{P/C}\cdot\vec{r}_{P/C})^{-\frac{1}{2}} \\
&= \frac{\vec{v}_{P/C}}{\left\|\vec{r}_{P/C}\right\|} - \frac{1}{2}\vec{r}_{P/C}(\vec{r}_{P/C}\cdot\vec{r}_{P/C})^{-\frac{3}{2}}\frac{{}^{\mathcal{N}}d}{dt}(\vec{r}_{P/C}\cdot\vec{r}_{P/C}) \\
&= \frac{\vec{v}_{P/C}}{\left\|\vec{r}_{P/C}\right\|} - \frac{1}{2}\vec{r}_{P/C}(\vec{r}_{P/C}\cdot\vec{r}_{P/C})^{-\frac{3}{2}}\left(2\vec{r}_{P/C}\frac{{}^{\mathcal{N}}\vec{r}_{P/C}}{dt}\right) \\
&= \frac{\vec{v}_{P/C}}{\left\|\vec{r}_{P/C}\right\|} - \frac{\vec{r}_{P/C}(\vec{r}_{P/C}\vec{v}_{P/C})}{\left\|\vec{r}_{P/C}\right\|^3} \\
&= \frac{\vec{v}_{P/C}}{\left\|\vec{r}_{P/C}\right\|} - \frac{\vec{r}_{S/C}\left\|\vec{r}_{P/C}\right\|(\vec{r}_{S/C}\left\|\vec{r}_{P/C}\right\|\vec{v}_{P/C})}{\left\|\vec{r}_{P/C}\right\|^3} \\
\vec{r}_{S/C} &= \frac{\vec{v}_{P/C}}{\left\|\vec{r}_{P/C}\right\|} - \frac{\vec{r}_{S/C}(\vec{r}_{S/C}\vec{v}_{P/C})}{\left\|\vec{r}_{P/C}\right\|}
\end{aligned} \tag{2.20}$$

As previously mentioned these calculations were with respect to the world reference frame $\mathcal{N}$. However, spherical flow is measured in the camera reference frame $\mathcal{C}$. Therefore we need to apply the transport theorem (2.17) obtaining the velocity of a point S as seen from $\mathcal{C}$

$$\frac{{}^{\mathcal{C}}d\vec{r}_{S/C}}{dt} = \frac{\vec{v}_{P/C}}{\left\|\vec{r}_{P/C}\right\|} - \frac{\vec{r}_{S/C}(\vec{r}_{S/C}\vec{v}_{P/C})}{\left\|\vec{r}_{P/C}\right\|} + \vec{r}_{S/C}\times\vec{\omega}_{\mathcal{C}/\mathcal{N}} \tag{2.21}$$

2.5. Camera Calibration

Computer vision begins with the camera detecting light from a scene. This light has travelled from its source, reflecting from an object, through the camera lens, and is recorded on the image sensor. For these rays of light that make up an image to be useful for representing the real-world, the geometry that transforms these rays from the three-dimensional real world to a two-dimensional image plane must be understood; i.e. the transformation from world coordinates to image coordinates.

The camera calibration process describes the camera with two sets of parameters, namely the intrinsic parameters, and the extrinsic parameters. The intrinsic parameters describe the geometric relationship between the camera model and its projection. And the extrinsic parameters describe the location and orientation or pose, of the camera with respect to the world frame.

This project uses a GoPro Hero 7 to test and verify the calibration routine. This section will cover camera models and outline why a planar model is used for the GoPro. It covers the mathematical

tools used to capture the different effects that make our camera imperfect for representing the real world, and how these deviations can be correrected.

2.5.1. Camera Models

The most elementary way of representing a camera is with the pinhole model. Light enters the aperture (pinhole) after reflecting from objects in a scene. For an ideal pinhole camera, only one ray enters the aperture from any point in the scene. The light is then projected onto a surface behind the aperture resulting in an image of the scene that is scaled proportionally to the focal length f.

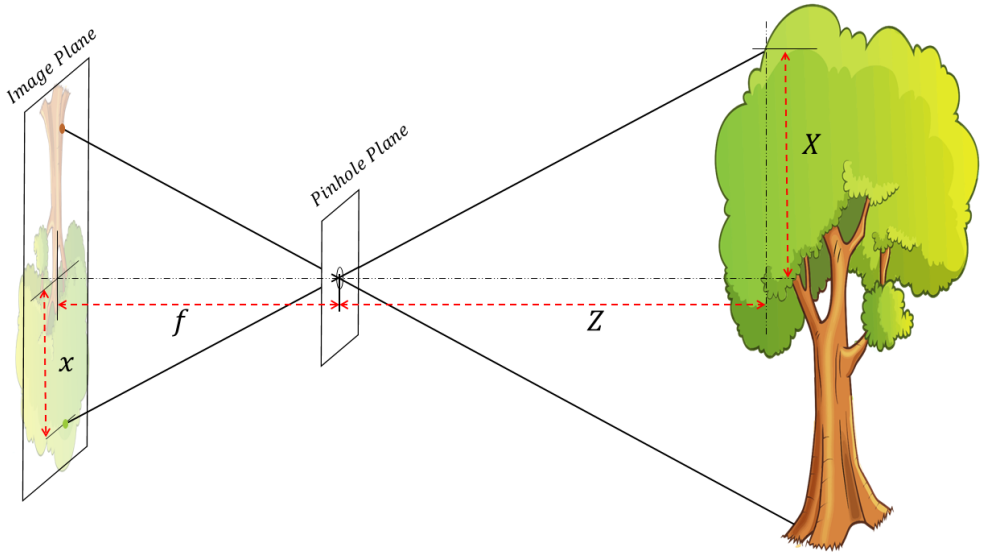


Figure 5: Elementary pinhole model of a camera: In practice, the appeture must be big enough to let light in, allowing a narrow ray of light from each point in the scene blurring the image. Therefore there is a tradeoff between brightness and sharpness.

Figure 5 shows an ideal pinhole camera model and its parameters, where f is the focal length, Z is the distance along the optical axis to an object, x is the size of the projected object and X is the size of the object. For this model, (2.22) is the relationship between the projected image and the object

$$\frac{-x}{f} = \frac{X}{Z} \tag{2.22}$$

Due to how light enters the camera, the image projection from a pinhole model is projected upside down. This issue is fixed in a planar camera model. The planar model is similar to a pinhole model, however rather than passing through an aperture, light travels from each point in the scene, intersects the image plane and converges to a single point known as the *centre of projection* Figure 6. The Plainar Model better represents the geometry of a camera with a lens, albeit a camera with an impossibly perfect lens.

The idea behind modelling and calibrating a camera is to identify and cancel out imperfections caused by slight defects in the manufacturing process. Firstly, the image sensor is often mounted out of line from the optical centre of the lens. Therefore, the parameters u_0 and v_0 are introduced to model this displacement from the optical centre. Furthermore, each pixel is not necessarily square, causing a

C : Centre of projection
$\{c\}$: Camera co-ordinates
P : Point in world space
P' : Point on image plane
$\vec{u}, \vec{v}$: Image co-ordinates
f : Focal length
I : Image plane

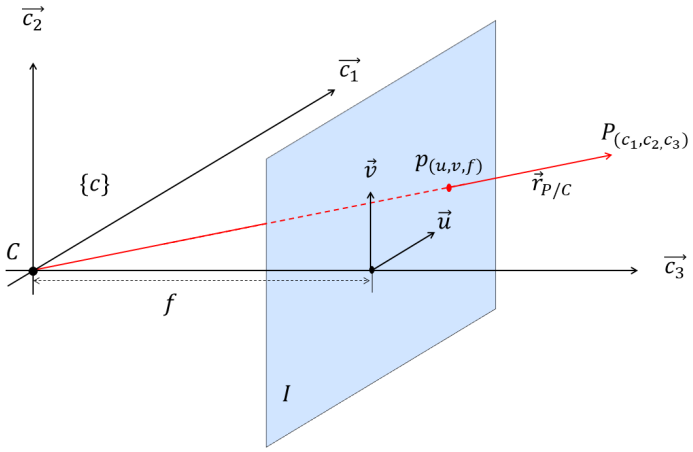


Figure 6: Ideal camera model. A point P in camera coordinates (c_1, c_2, c_3) is projected and displayed as a point p in image co-ordinates (u, v)

small discrepancy between the focal length in the u and v directions. These discrepancy are captured in the f_u and f_v parameters. The following Equation (2.23) project a point P_{c_1,c_2,c_3} to image sensor co-ordinates (u, v):

$$\begin{aligned} u &= f_u \times \frac{c_1}{c_3} + u_0 \\ v &= f_v \times \frac{c_2}{c_3} + v_0 \end{aligned} \tag{2.23}$$

2.5.2. Projective Geometry

Projective geometry is a tool describing the relationship that transforms a point from world coordinates (n_1, n_2, n_3) into image coordinates (u, v). When applying such transformations, it is convenient to add an extra dimension to each vector in the transform. for instance, a point that lies in a n dimensional space is represented as a $n + 1$ dimensional vector. this coordinate system is known as *homogeneous coordinates*. for example, a point lying on a plane at (x, y) is represented by the homogeneous coordinates (x, y, z). It is important to note that in homogeneous coordinates, any two vectors that share a common factor represents the same point in Euclidean geometry. This has the effect that a point in Euclidean geometry can be represented by an infinite number of homogeneous vectors. for instance $\alpha * (q_1, q_2, q_3) \equiv (q_1, q_2, q_3)$. The original primitive point is recovered by dividing through by q_3.

the utility of homogenious coordinates becomes clear when common transformations such as translation, rotation and scailing are attempted in . Homogenious coordinates allow these transformations to be implemented with matrix opperations. This is how projective geometry allows the camera intrinsic parameters f_u, f_v, u_0, v_0 to be arranges into a single 3×3 *camera intrinsic matrix* matrix. Bringing all of thees concepts together, the Equation (2.23) can be implemented as a single martix opperation of the form:

$$p = \mathbf{M}\mathbf{r}^C_{P/C} \tag{2.24}$$

where:

$$P = \begin{bmatrix} u \\ v \\ w \end{bmatrix}, M = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{r}^C_{P/C} = \begin{bmatrix} c^c_1 \\ c^c_2 \\ c^c_3 \end{bmatrix} \tag{2.25}$$

Some older camera manufacturing processes produce nonrectangular image sensors, i.e. they produced parallelogram shaped sensors. To account for this, a final *skew* parameter was included. Most modern cameras are produced with close to no skew therefor the error that this effect causes is below the noise floor and assumed to be 0.

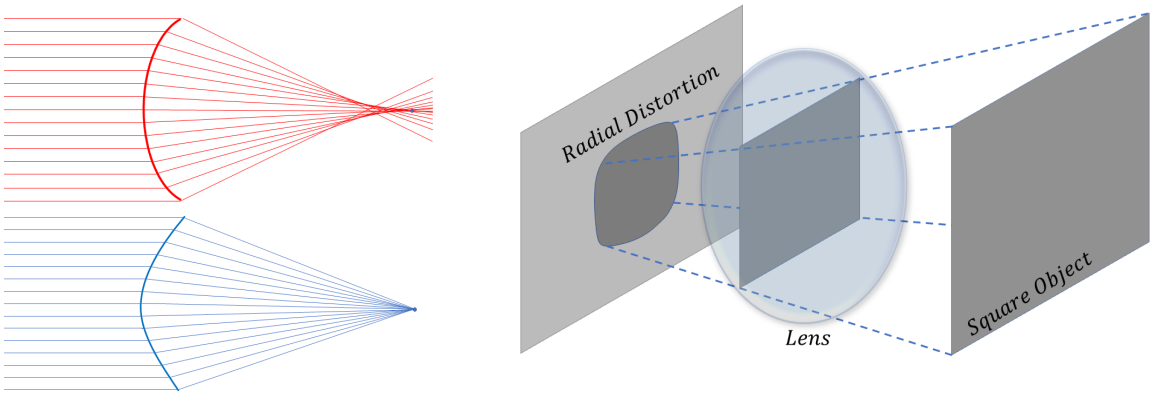
2.5.3. Lens Distortion

The optics inside a camera are yet another source of error in an image. Similar to correcting for the cameras geometry (captured in 2.25), lens distortion must be accounted for. The leading method for correcting Lens distortion uses a *Brown-Conrady model* from Fryer and Brown (1986). This model characterises the two most prominent sources of distortion, naimley *radianl distortion* and *tangental distortion* and undistorts the image pixel locations.

In an ideal camera, the lense would take the shape of a perfect paraboloid. However it is much easier and cheeper to produce a sperical lens. The spherical shape of the lens tends to "over bend" rays entering farther from the optical center producing the effect known as *Radial Distortion* figure 8. Typically a taylor series is used to characterise the distortion caused by the lens Kaehler and Bradski (2016). This polynomial takes the form:

$$f(r) = a_0 + a_1 r + a_2 r + a_3 r \ldots \tag{2.26}$$

where $r = \sqrt{u^2 + v^2}$ is the radial distance from the optic centre.



(a) Red - "spherical" lens
Blue - paraboloidal lens

(b) Radial distortion is most clear on square objects

Figure 7: Radial distortion also known as *Barrel Distortion*. The farther from the centre a ray enters, the larger the barrel distortion bends it.

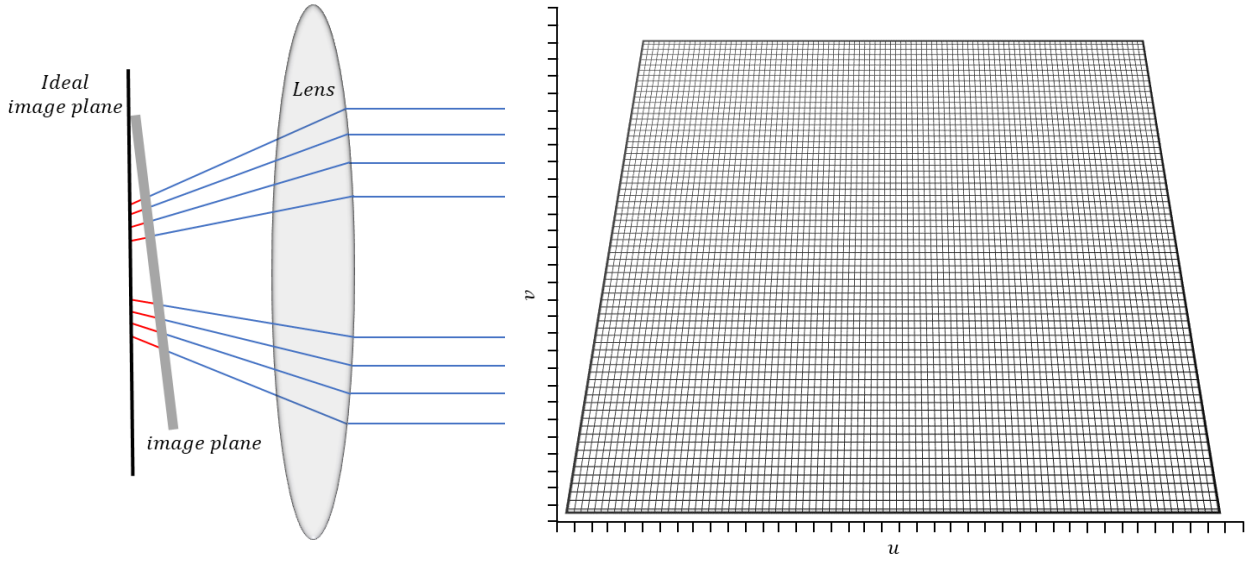
However, for radial distortions, at the optical centre of the lense, deviation cause to the light rays is minimal i.e. $f(r) = 0$ at $r = 0$. This further implies that $a_0 = 0$. Additionally, becaues the distortion must be symetrical in r, the coefficents of the odd powers of r equal 0. This means that only the

parameters needed for radial distortion correction are the coefficents of even powers of r Kaehler and Bradski (2016). Typically, no more than three radial distortion parameters (k_1, k_2, k_3) are needed. However, for high *field of view* cameras, upto K_6 may be necessary. Once these parameters are found, each pixel will be rescaled according to the following equations:

$$\begin{aligned} u' &= u \times (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ v' &= v \times (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{2.27}$$

where u' and v' are the undistorted pixel coordinates.

Not only is it expensive and dificult to produce a lens that perfectly trains rays of light onto the *centre of projection* of a camera, it is also dificult to mechanically mount the image sensor to the camera body such that it is parallel to the lense.



(a) Ray diagram of tangental distortion

(b) Tangental distortion of a flat grid

Figure 8: Extream example of tangental distortion.

Two additional parameters p_1 and p_2 are introduced to minimally characterise this tangental distortion such that:

$$\begin{aligned} u' &= u + [2p_1 uv + p_2(r^2 + 2u^2)] \\ v' &= v + [p1(r^2 + 2v^2) + 2p_2 uv] \end{aligned} \tag{2.28}$$

There are many other aspects within an average camera that cause image distortion, howerver these effects contribute a minor amount to the accuracy of measurements from an image so for this project will be assumed 0. Both Matlab and Open CV based camera calibration rotines are implemented and evaluated in this project. The camera calibration routine for Matlab *estimateCameraParameters();* returns an object containing the camera matrix 2.25, radial distortion parameters 2.27 and tangental distortion parameters 2.28, along with information about the calibration photos. When using *Open*

CV, these parametetrs are packaged into a *distortion vector*. For this project, six radial distortion parameters were used to capture the distortion for a Go Pro Hero 7. Therefor the distortion vector from Open CV is organised as $(k_1, k_2, p_2, p_1, k_3, k_4, k_5, k_6)$. From equation (2.27) and equation (2.28), the *Brown-Conrady model* is as follows:

$$\begin{aligned} u' &= u\left(\frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{k_4 r^2 + k_5 r^4 + k_6 r^6}\right) + 2p_1 uv + p_2(r^2 + 2u^2) \\ v' &= v\left(\frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{k_4 r^2 + k_5 r^4 + k_6 r^6}\right) + p_1(r^2 + 2v^2) + 2p_2 uv \end{aligned} \tag{2.29}$$

Now that the camera can be discribed mathmatically through the intrinsic parameters and distorion parameters, pixel location in the image can be *undistorted*. Figure 9 shows the movement of a subset of pixes from a GoPro Hero 7 image. Notice the movement of pixels in the centre of the image move far less than thoes radialy further away.

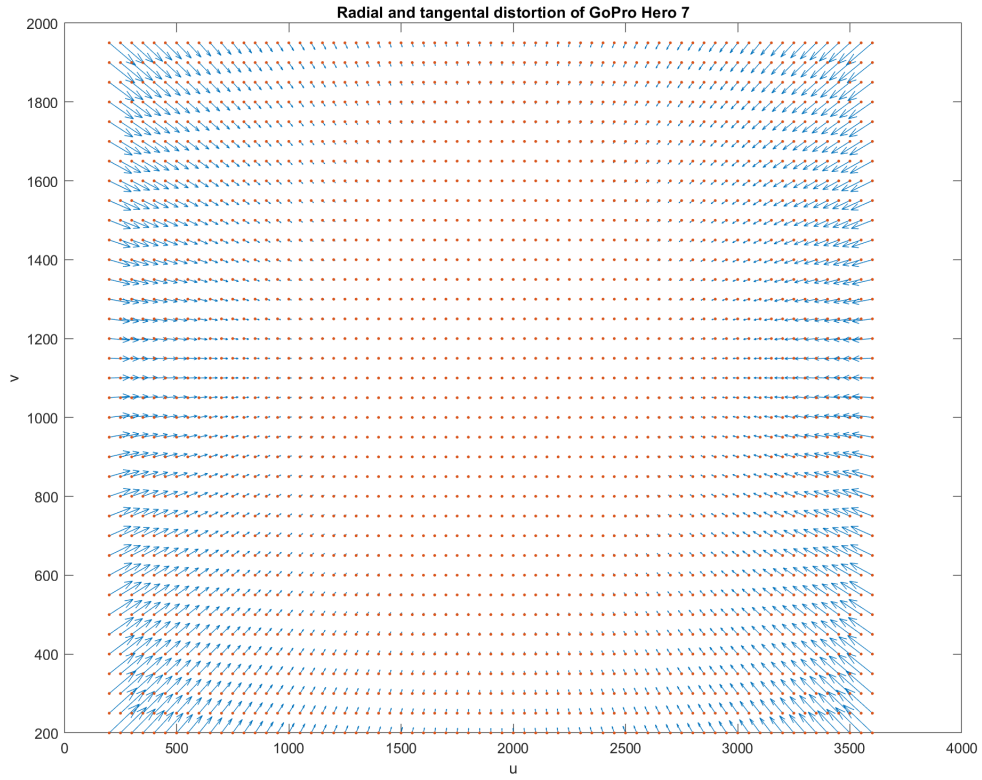


Figure 9: Shows the movement of a sample of pixels based on the parameters of a GoPro

2.6. Trajectorie Planning

3. Data simulation and Collection

Simulating the model discussed in The project is primaritly simmulation based. However, the camera calibration routine (explored in following sections was tested and evaluated on real world images collected from a GoPro Hero 7. Were this project to be implemented on a real world fixed wing UAV, the kinematic data would need to be obtained through sensors on the UAV.

4. Optical Flow - Simulation

As discussed in sing the spherical flow model discribed in 2.4.2

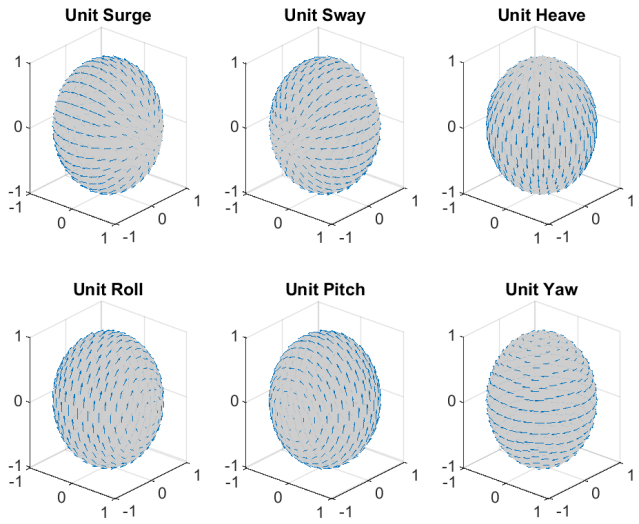


Figure 10: Write something here

4.1. Measurement Model

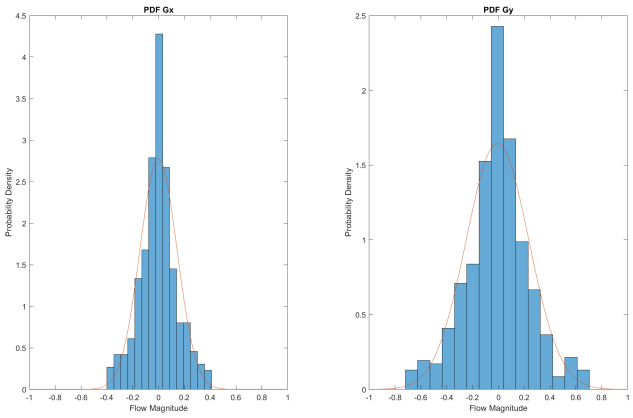


Figure 11: Write something here

4.2. Farnback Algorithm - OpenCV

5. Camera Calibration

As discussed in Section 2.5 it is imperitive to understand and account for the construction and imperfections in a camera, if the measurments taken need to be relevent outside image coordinates. This section outlines the procedure used in this project to correct a GoPro hero 7 using a planar camera model and a 6th order equation to correct for distortion parameters.

5.1. Calibration Boards

Theoretically, to calibrate a camera any object could be used as a calibration object, so log as it is appropriately characterised. However it is much more practical to use a well defined and easily defined object. For this reason, a flat rigid checker pattern was chosen. The geometry of the calibration pattern can be seen in Figure 12

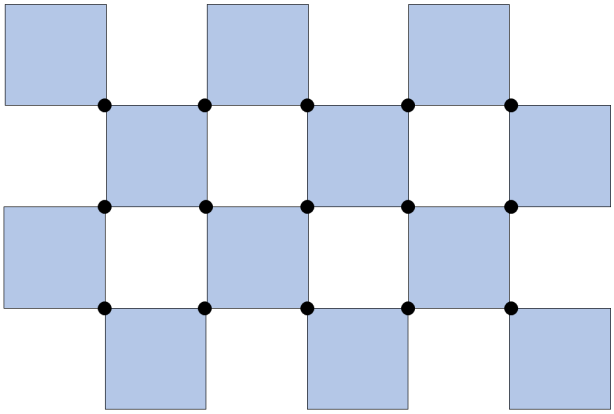


Figure 12: Write something here

5.2. Calibration Tool

To calibrate the GoPro Hero 7 for in this project, A camera calibration tool was written in C# using WPF as a graphical user interface(see Appendix B)).

5.3. Calibration Results

Figure 13: Caption

6. Vector Field Divergence for Object Avoidance

Figure 14: Caption

7. Discussion

Environment navigation is an extremely broard topic, and this project contributes a small amout towards the obstacl avoidance

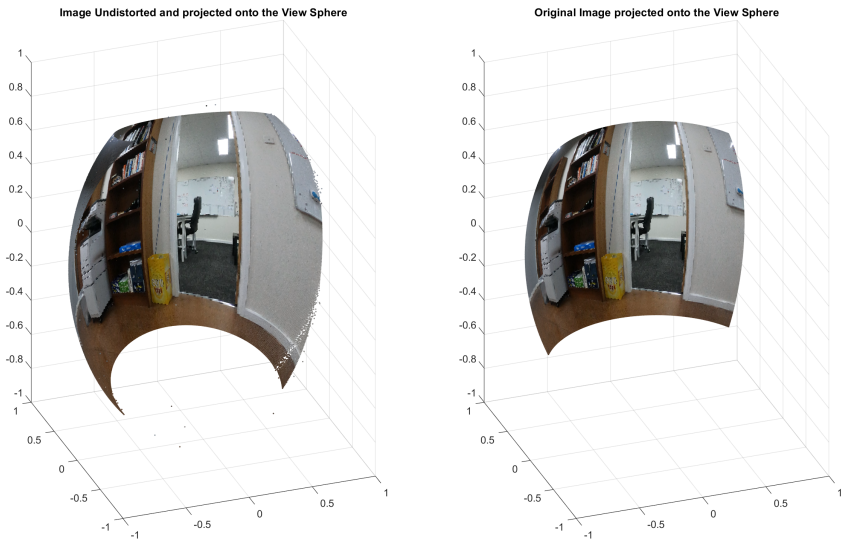


Figure 15: Caption

8. Conclusion

The goal of this project was to investigate the plausability of using optical flow cues as a method for avoiding obstacles in a scene. Results of the simulations show that the a cost function based on divergence of an optical flow field generates a suitable control policy, stearing the UAV in a direction away from obstacles such as walls and buildings.

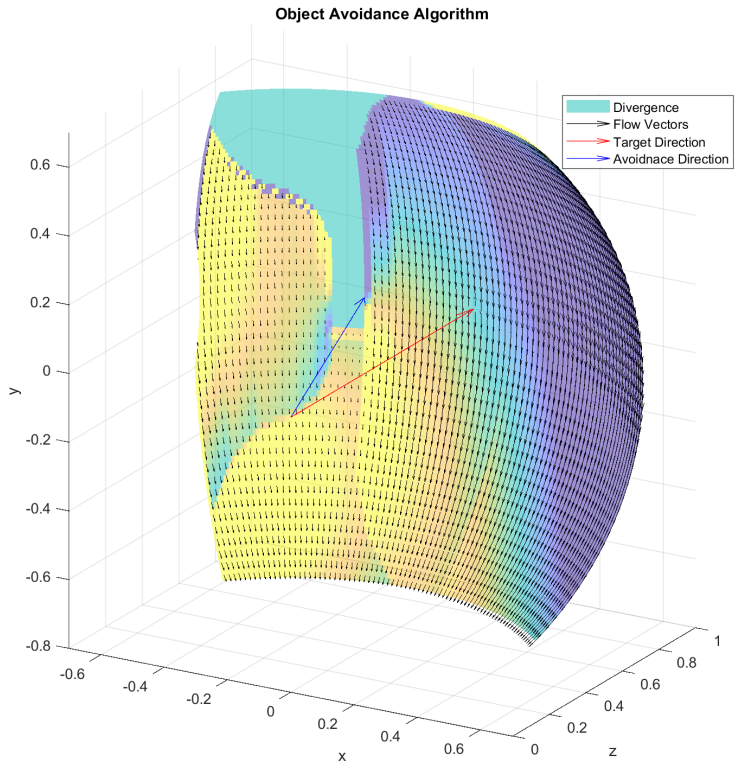


Figure 16: Caption

9. Recomendations

For the results of this project to be implemented on a robotic platform, an optical flow algoithm would need to be run in real time. for this, an image flow algorithm that can be implemented on a descrete graphic card should be implemented such as that ceated by Adarve and Mahony (2016).

References

Adarve, J. D., Mahony, R., 2016. A filter formulation for computing real time optical flow. IEEE Robotics and Automation Letters 1 (2), 1192–1199.

Altshuler, D. L., Srinivasan, M. V., 2018. Comparison of visually guided flight in insects and birds. Frontiers in neuroscience 12, 157.

Benyus, J. M., 1997. Biomimicry: Innovation inspired by nature.

Fryer, J. G., Brown, D. C., 1986. Lens distortion for close-range photogrammetry. Photogrammetric engineering and remote sensing 52 (1), 51–58.

Hartley, R., Zisserman, A., 2003. Multiple view geometry in computer vision. Cambridge university press.

Horn, B. K., Schunck, B. G., 1981. Determining optical flow. Artificial intelligence 17 (1-3), 185–203.

Kaehler, A., Bradski, G., 2016. Learning OpenCV 3: computer vision in C++ with the OpenCV library. " O'Reilly Media, Inc.".

Meinhardt-Llopis, E., Pérez, J. S., Kondermann, D., 2013. Horn-schunck optical flow with a multi-scale strategy. Image Processing on line 2013, 151–172.

Perez, T., 2014. Fundamentals of mechanical systems (with application to robotics). lecture notes mcha3900-mechatronic design ii, 11–13.

Ratliff, N., ???? Inverse optimal control: What do we optimize?

Sun, D., Roth, S., Black, M. J., 2010. Secrets of optical flow estimation and their principles. In: 2010 IEEE computer society conference on computer vision and pattern recognition. IEEE, pp. 2432–2439.

A. Journal

B. Calibration Tool

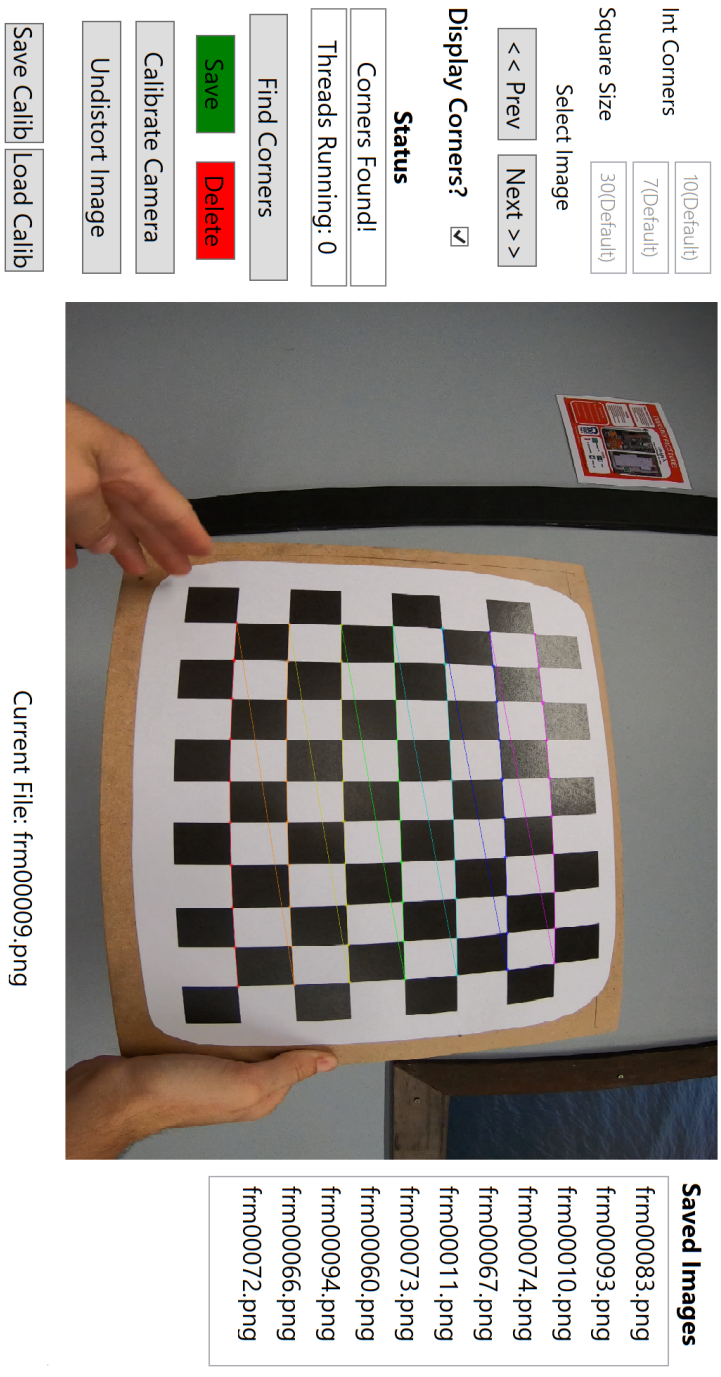


Figure 17: Write something here