PRIFYSGOL ABERYSTWYTH UNIVERSITY

Estimation of Terrain Gradient Conditions and Obstacle Location Using a Monocular Vision-based System

FINAL REPORT

CS39440 - MAJOR PROJECT

G601 (MENG SOFTWARE ENGINEERING)

Author:
Connor Luke Goddard (clg11)

Supervisor:
Dr. Frédéric Labrosse (ffl)

This report was submitted as partial fulfilment of an MEng degree in Software Engineering (G601)

May 8, 2015

Revision: 1.0

Status: Release

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature .....

Date .....

Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature .....

Date .....

Ethics Form Application Number

The Ethics Form Application Number for this project is: 841.

Student Number

Acknowledgements

I wish to thank my project supervisor, Dr. Frédéric Labrosse for all of the guidance and vast amount of patience he provided throughout the duration of the project. Many of the ideas discussed within this report were originally suggested by Fred, before being developed further through a combined effort.

I also wish to thank all of my family for their continued support during my academic studies, and in particular thank you to my partner Sioned who without all the support she has given, I would never have been able to achieve all that I have till now.

Abstract

The ability for an autonomous robot to navigate from one location to another in a manner that is both safe, yet objective is vital to the survivability of the machine, and the success of the mission it is undertaking.

While the problem of vision-based obstacle detection has enjoyed a high level of research focus over recent years, it would appear that much less work has concentrated on providing a means of reliably detecting changes in terrain slope through the exploitation of observed changes in motion.

Making use of recent advances in camera technology with appropriate computer vision techniques, this project focussed on conducting an investigation into the use of appearance-based template matching technique in conjunction with optical flow analysis to provide an estimation into the current gradient conditions of the environment terrain, while also supporting the detection of nearby obstacles, and providing an indication as to the egomotion properties of a mobile robot.

While not all of the original research aims were met, progress was made in terms of establishing a positive relationship between the vertical position of a feature and the level of vertical displacement that it demonstrates between two subsequent images. The investigation saw three key experiments conducted, all focussing on improving the approach to using appearance-based template matching with the aim of accurately determining the level of vertical displacement shown, and thus subsequently allowing the detection of potential changes in slope, or appearance of obstacles within the current field of view.

Although the final results were generally mixed, there were occasions where key conclusions could be drawn as to the performance characteristics of each evaluated experiment method. However due to the overall lack of consistency in the results, this project has laid the groundwork to contribute towards further development within this research area.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background & Objectives

1.1 Background

1.1.1 Introduction

In considering any kind of autonomous system that is expected to make unattended decisions, a crucial, yet not always obvious aspect is a reliance upon the idea of *trust*. As is observed in humans, such a system will typically depend upon underlying information in order to assess which, out of a selection of alternative choices, is the most appropriate for a given situation.

This information may be obtained from a variety of sources, ranging from raw sensor data up to the output of a sophisticated algorithm. While the type and source of information may vary wildly between systems, the underlying notion that this information can, in one form or another, be *trusted* to provide a truthful representation of the current situation remains the same.

Given the importance of the relationship between reliable input data and correct decision making, a crucial aspect associated with autonomous robotics is the attempt to maximise both accuracy of input data, and robustness of systems used to verify this accuracy.

The ability to navigate plays a vital role in enabling robots that require autonomous motion capabilities to make safe, yet objective decisions on how best to traverse from one position to another. This becomes evermore important when considering that a key use case for robotic systems typically involves the undertaking of tasks within environments deemed unsafe for humans to enter. From observing examples of tasks where autonomous robots are expected to operate within harsh environments including bomb disposal and planetary exploration [30], it becomes clear that the ability to perform safe and robust navigation is vital to the survival of both the mission and device.

A vital contributor to enabling successful navigation, is the ability to identify and subsequently avoid obstacles. As such, the development of accurate and robust obstacle detection systems is a keen and well-explored area of robotics research, with a variety of approaches now available adopting many types of sensor including sonar [51] and laser scanning [26].

An alternative approach that has enjoyed increasingly greater research focus over the past decade is that of vision-based obstacle avoidance. This involves the analysis of images captured from one or more digital cameras in order to detect and classify possible dangers situated within the robot's field of view.

Cameras are becoming an increasingly favourable sensor for use on robotic systems, due primarily to the impressive variety and quantity of potential data that can be captured *simultaneously* from a single device. They are also one of few sensors that have experienced a consistent reduction in price over the past decade [16], making them viable for many types of project application and budget.

1.1.2 Related Work

The challenge of providing accurate and robust obstacle detection remains to be a major topic of focus within the field of computer vision. While beginning to investigate this area of research, it quickly becomes clear that there are an enormous variety of solutions already available. In the majority of cases, these solutions propose new or improved approaches to combining "standard" computer-vision techniques with a variety of hardware configurations, with the aim of either improving on previously published results, or to focus on the avoidance of specific types of obstacle (e.g. precipice or animal detection).

The first approach to be noted is that from the work of Campbell *et al* [16], in which the authors propose a system for estimating the change in position and rotation of a robot using information obtained solely using a single consumer-grade colour camera. The approach describes the use of feature-based tracking to estimate the sparse optical flow-field between subsequent video frames, before taking these optical flow vectors currently corresponding to image coordinates, and back-projecting them onto a "robot-centred" coordinate system in order to establish the incremental 'real world' translation and rotation of the robot over a given time period. The detection and subsequent tracking of matching features between captured frames is provided via the *OpenCV* library implementation of the Lucas-Kanade algorithm [45].

Of particular interest from the work of Campbell *et al* [16] is the approach described for detecting environment hazards solely via the exploitation of the optical flow field. The proposed method bases itself around the observed discontinuities caused to the optical flow field by scene objects that appear at a different observed depth to camera than the ground. In the paper this behaviour is described as a violation of the "planar world hypothesis" [16], in which objects appearing closer to the camera relative to the ground cause a positive violation, and in the opposite case causing a negative violation.

As a consequence, the authors discuss how owing to the effects of motion parallax, it is possible to observe clear differences in the motion displacement of scene objects demonstrating either a positive, negative or no violation. This subsequently maps onto discontinuities observed in the optical flow field, with objects that move significantly closer to the camera relative to the ground demonstrating longer optical flow vectors, and objects further away from the camera demonstrating the opposite. This approach appears to demonstrate notable results, with the authors describing how in practice their system was able to "detect precipices as near as 3cm" [16].

An alternative approach proposed by Low and Wyeth [26] involved combining sparse

feature detection with appearance-based matching in order to track corresponding features between video frames. Using the template matching function provided by the *OpenCV* library, the authors describe how a variety of appearance-based matching metrics could easy be evaluated via simple changes to function parameters. They report on how they eventually chose Normalised Cross-Correlation as the final similarity metric, owing primarily to its improved robustness to lighting changes and simple score range falling between 0 and 1 (1 indicating a perfect match) that allowed for easy pixel thresholding.

A major challenge relating to the use of optical flow methods for obstacle detection, and one discussed in the work of both Low and Wyeth [26] and Campbell *et al* [16], is the effect that changes in rotation (deliberate or otherwise) can have on the observed optical flow vectors of features. While both papers propose alternative methods for removing the effects of rotation (the use of an Inertial Measurement Unit gyroscope in the case of Low and Wyeth, and a calibrated cylindrical coordinate model in the case of Campbell *et al*), both appear to agree that rotational movement should be accounted separately to translational movement in order to ensure accurate motion estimates can be inferred from the optical flow field.

Ulrich and Nourbakhsh [49] present an alternative approach to vision-based obstacle detection from the two previously discussed. As opposed to scrutinising differences in motion behaviour of scene objects, the authors instead focus on comparing differences in colour between the ground and other objects in the robot's field of view. Ulrich and Nourbakhsh argue that using colour as a detection metric for obstacles can prove to be less computationally expensive than "range-based" obstacle detection methods like optical flow or stereo vision, and in certain cases, have also shown to be more accurate and reliable - particularly in cases involving the detection of small or flat objects close the ground or where a system needs to be able to differentiate between terrain surfaces.

A key aspect from the Ulrich and Nourbakhsh paper discusses details of how their system is able to effectively handle image noise caused by changes in illumination. In their solution, the authors discuss the use of an alternative colour space to RGB for representing input images that allows for the undesired effects caused by changes in illumination between subsequent frames to be negated. By converting images to use the 'Hue-Saturation-Intensity' colour space, the actual colour of objects within the image (represented by the Hue and Saturation channels) become less sensitive to changes in brightness (represented by the Intensity channel). As well as providing greater resistance to illumination changes, using a colour space that separates colour from brightness also provided an easier platform from which the authors could further remove noise via channel thresholding [49].

As part of a survey conducted into various methodologies of appearance-based template matching methods, Perveen *et al* [36] discuss an enhancement to traditional template matching that has been shown to support the successful detection of objects whose observed appearance has been altered following a change in orientation.

The development of vision-based systems that are invariant to object rotation has been an area of keen research interest over recent years. While a number of solutions now exist for providing resilience to such changes [**?**], the majority of these solutions are only applicable to detection methods that are based upon the use of feature descriptors for

representing scene objects, rather than the use of an object's appearance.

This is particularly important given the assumption that for most objects, even a slight rotation will cause its *appearance*, in terms of pixel values, to be modified considerably. As such, a key weakness of appearance-based detectors describes a failure to find correct matches of an object that has undergone a rotation. This is generally caused as a result of such detectors relying on an explicit pattern of object appearance for use as the primary metric when comparing image patches for similarity equality.

In their paper, Perveen *et al* describe how "Grayscale-based template matching" utilises multiple pyramid structures (used traditionally in supporting degrees of scale invariance [27]) to provide support for rotational-invariance within appearance-based features detectors. By generating a new pyramid structure to represent every possible orientation of the target pattern, these structures can, once combined, be used to significantly improve the chances of a pattern being relocated in the current image, even after potentially experiencing a significant rotational transformation [36]. Additional work published in [23] discusses further improvements to the efficiency of this approach by removing pixels deemed unlikely to ever match a target pattern from further consideration, thereby reducing computational cost.

1.1.2.1 Hardware Configuration

While the majority of approaches mentioned thus far focus on the use of monocular-based systems (i.e a single camera) for their hardware configuration, many other examples of vision-based systems using alternative choices of hardware are also available. One of the most popular alternatives to monocular camera systems is a concept that can be easily described as 'natural progression' in terms of technological evolution. This description is of course talking about the use of two or more cameras, formally known as stereo vision.

Williamson [51] proposes a trinocular stereo vision system capable of detecting small obstacles over great distances - *"objects as small as 14cm high over ranges well in excess of 100m"*. He discusses how by using three cameras arranged in a triangular formation, his obstacle detection system is able to maintain high levels of reliability even in situations where an image demonstrates *"texture in one direction, but not in the other"*. This reliability is particularly important when you consider that typical examples of this type of behaviour have been found to originate around pixels representing the edges of objects that such a system is aiming to detect [51].

Williamson also later states that through the use of more than one camera, his system is able to enjoy an improved level of general accuracy, achieved as a result of the significant level of additional information that can be gathered. Taking this statement solely at face value, it could be feasible to assume that the addition of further cameras may improve on this performance further still. On a contrary to this, Williamson warns that using anymore than three cameras will not make any significant difference to performance results, and in fact may actually impede results as a consequence of over-complication [51].

As mentioned previously, Low and Wyeth [26] chose to combine the efforts of a monocular vision system, laser scanner and inertial measurement unit to provide highly accurate detection capabilities for a variety of obstacle types and sizes. A key advantage to

adopting a hybridised hardware configuration comes from benefit that should one system fail for any given reason, in most circumstances, alternative systems remain available to continue. As such, multi-instrument setups can help to ensure that potentially critical information regarding the location of possible obstacles is not lost.

However, an important consideration to note is that by increasing the variety and number of hardware components, other critical quantities relating to the design of a robot including aspects such as power usage and chassis space, may also need to revised accordingly.

1.1.3 Motivation

Motivation for the undertaking of this project comes following an increased interest into the field of computer vision and its associated topics, gained during completion of a university module surrounding the subject.

The project also presented a key opportunity for the author to engage in a substantial piece of work falling well outside of their prior experience, choosing to pursue a problem of a scientific or research-based nature, rather than one of 'traditional' software engineering.

Further opportunities were also presented for enhancing development skills using the C, C++ and Python programming languages, and in particular, the development of computer vision applications using popular image processing libraries.

1.2 Development of the Working Hypothesis

Focussing on conducting research as part of a scientific investigation, this project looks specifically into investigating the viability of the following proposed hypothesis:

"From images captured using a non-calibrated monocular camera system, analysis of optical flow vectors extracted using localised appearance-based dense optical flow techniques can provide certain estimates into the current condition of terrain gradient, the location and characteristics of obstacles, and robot egomotion."

In the context of this project, such a statement is technically characterised as a *working hypothesis*, which gives rise to the expectation for further investigative work to be conducted, even if this hypothesis is later discovered to fail [46].

The history of work that led to the proposal of the above hypothesis actually began focussing not on obstacle detection, nor the inference of terrain gradients, but instead on investigating potential solutions towards a lightweight visual-based odometry system for installation aboard a mobile robot.

It was not until the completion of initial background investigations into existing approaches (namely the work of Campbell *et al.* [16] - discussed further in Section 1.1.2) that key ideas surrounding the final hypothesis began to come into fruition.

1.2.1 Key Aspects & Assumptions

The hypothesis detailed in this report can be decomposed into four key areas of prediction. While all relate to the exploitation of observed optical flow behaviour, each focusses on providing an estimation into the status of a specific "condition metric".

While these will of course only be estimates, it was hoped that when brought together, these observations could provide a vital insight into the overall status and condition of the current terrain and robot egomotion.

The four "condition metrics" subject to observation under the proposed hypothesis were as follows:

- **Terrain gradient** - Indication of how "flat" the terrain surrounding the robot current is. Any significant changes to terrain slope (positive or negative) should be identified in order to allow any appropriate action (e.g. reduction in speed) to be taken.
- **Oncoming obstacles** - The detection of both positive and negative obstacles (e.g. rocks and pits respectively) encountered along the path of a mobile robot as it moves through its environment.
- **Robot speed** - Indication of a robot's speed based upon a prior knowledge of a fixed time interval and the distance moved by the robot within that time frame.
- **Robot orientation** - An indication of current orientation, based upon observations into changes in rotation made as a result of the robot turning.

Given the high level of potential scope for investigation presented under the working hypothesis, it was deemed important, especially in the early stages of investigation, to draw some appropriate assumptions that could help simplify the general ideas presented into manageable stages. Clearly, should the initial findings be found to be positive and enough available time remained, then it was expected that further work could be planned in order to try and tackle the more complex issues that would address these assumptions.

Taking guidance from the previous projects [16], [26], the chosen assumptions were deemed reasonable in relation to the main objective of providing visual-based terrain inference, obstacle detection and egomotion estimation of a wheeled robot:

1. In the first instance, it was assumed that the proposed system would only support motion following straight line trajectories. This meant that at least within the early stages of investigation, work would not focus on providing an estimation into robot orientation, given that theoretically this should not change while the robot is moving in a forward direction.
2. It is assumed that camera to be used as input will be mounted in fixed position facing in the same direction as the robot during forward travel.

The remainder of this section is dedicated to providing details regarding the core ideas proposed within the hypothesis, including a discussion into the general thought processes considered while being conceptualised.

1.2.2 Depth Estimation via Motion Parallax & Optical Flow

The concept of motion parallax, and its usefulness as means of inferring information about real-world environments, is an idea found to be commonplace in vision-based solutions wishing to extract details about 3D environments from 2D images [28].

The general theory behind motion parallax is simple, stating how as the visual field of an observer changes due to moving position, objects located a short distance away will appear to move across the scene faster than those objects located in the distance (Figure 1.1). This effect forms a type of visual cue, from which the depth of objects situated within a 3-dimensional scene can subsequently be perceived [41].

One of the key advantages to using motion parallax within the context of computer vision is that the perceived effects continue to be visible even if captured through only a single camera. Therefore strictly speaking, motion parallax is best defined as providing a *monocular* visual cue for depth, theoretically open to exploitation by any visual-based system making use of at least one camera for supplying input.

By measuring the level of displacement that an object demonstrates, it then becomes possible to infer the relative distance between that object and the current view position. Typical approaches within computer vision accomplish this task through the use of feature detection and tracking to subsequently calculate the displacement of static objects between consecutive images.

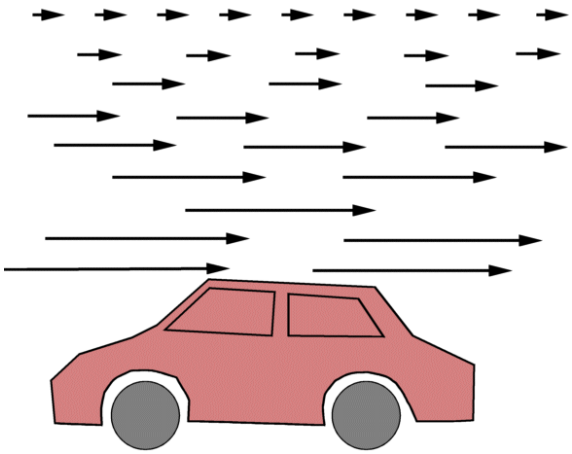

Figure 1.1: Visualisation of the changes in perceived displacement of objects within the visual scene as a result of the effects caused by motion parallax. Courtesy [42].

In the case of Campbell *et al.* [16], the authors make exclusive use of the observed differences in the displacement of objects as the underlying mechanism for detecting the presence potential obstacles.

Measuring the displacement by which a select number of "distinctive" image regions move over time, is a process commonly known as *sparse optical flow* [33].

By monitoring discontinuities in the length of calculated optical flow vectors (i.e. the distance and direction of displacement along the image plane), Campbell *et al.* were able to detect the presence of both positive and negative obstacles, characterised respectively by whether the features identified in the image demonstrated vectors that were proportionally *shorter*, or *longer* than the general consensus set [16].

A potentially problematic issue associated with sparse optical flow and feature detection in general, is how to ensure adequate coverage and frequency of features, when the scene represented in consecutive images fails to contain many regions "distinctive" enough for use by feature detectors. When faced with this issue, Campbell *et al.* [16] chose to

deliberately select a wider range of lower quality features within input images in order to ensure that they obtained reasonable coverage across the entire image space.

Naturally, choosing specifically to allow the detection of lower-quality features opens up a greater risk to outliers and false positives, which Campbell and his team later counteract through the combined filtering and 'peer-verification' of incremental optical flow vectors [16]. While this approach is noted to strike a good balance between the amount of information that can be extracted and resilience to potential outliers, it does of course also attribute itself to an increase in the overall computational cost of the system. This is a price that aboard some robotic systems could prove simply to be too high, particularly in situations where spare computing power is already considered a luxury (e.g. planetary rovers).

As alternative solution is to instead attempt to track the change in motion of *every pixel* in the image, known formally as *dense optical flow* [33]. While this approach guarantees coverage across the entire image, it can also prove to be noticeably slower to complete.

1.2.3 Supporting Inference of Terrain Gradient & Obstacle Detection

The proposed hypothesis aimed to take the same fundamental ideas proposed by Campbell *et al*. [16] for the detection of obstacles located within the visual field (discussed more in Section related-work), before attempting to apply them further to the inference of terrain gradient conditions.

It was considered that, just in the same way as positive obstacles appear closer to the camera than the ground on which they lie, an area of ground that demonstrates a sudden increase in its tendency upwards would also conceptually appear "closer" to the camera.

When the resulting optical flow field is later analysed, it would be expected that the same region of the ground plane (captured over consecutive images) would demonstrate optical flow vectors that became gradually longer than those previously recorded as a result of the parallax-induced motion being observed. Likewise, when presented with an area of ground sloping evenly downwards, a decrease in its displacement detected through the observation of proportionally shorter optical flow vectors would generally be expected.

Continuing to employ the concept of motion parallax as the primary means of estimating the 3D-geometric characteristics of a captured 2D scene, the hypothesis introduced two key differences in its general approach from those discussed within other projects investigated:

- The exclusive use of appearance-based template matching techniques to provide a *localised* variant of *dense optical flow analysis* over the use of sparse optical flow techniques relying upon a high number of available image features.
- The use of a formalised model to represent detected changes in vertical displacement as part of efforts to estimate changes in terrain gradient in addition to the location/characteristics of potential obstacles.

1.2.3.1 Appearance-Based Feature Detection & Tracking

The decision to use appearance-based matching for the detection and subsequent tracking of features was taken primarily out of an interest in conducting an evaluation into the performance of such methods with respect to optical flow analysis.

While examples of previous work describe the combination of feature-based detection with appearance-based tracking [26], this investigation wished to focus exclusively on the use of appearance-based methods for both the initial detection, and subsequent matching/tracking of image features necessary to establish the optical flow field.

One of the key motives behind this decision directed towards the common observation that appearance-based detectors on average tended to be less computationally expensive than their feature-based equivalents [26]. This was an aspect that tied in closely with the relationship that the project exhibited between the fields of computer vision and robotics, and subsequently was deemed an important benefit in relation to the ultimate goal of installation aboard a mobile robot that would most likely possess limited computing capability.

In the interests of time, the project decided to focus on the evaluation of four appearance-based similarity measures for template matching identified through the background research. These consisted of two *distance-based* measures, and two *histogram-based* measures.

1.2.3.2 Vertical Displacement Model

The use of a model for representing vertical displacement of image regions was an idea that lay at the centre of the main investigation. Its purpose was to provide a mechanism by which the core underlying prediction relating to the estimation of changes in terrain gradient and obstacle detection could be evaluated:

"Following the concepts of motion parallax, it is predicted that features captured towards the bottom of an image will show greater displacement between subsequent frames, than those features captured towards the top of the image."

The model was designed to take the form of a kind of "lookup-table", where a simple structure would be adopted to focus specifically on the relationship between the *height within an image*, and the *level of vertical displacement that the row located at that height demonstrated*.

When visualised as a 2D scatter plot (Figure 1.2), this structure definition would be represented as:

- **X axis:** The height of the image, plotted as 1px-high rows.
- **Y axis:** The vertical displacement in pixels recorded at a given height.

where the predicted trend would see as the row number (i.e. height relative to the top of the image) increased, so to would the level of vertical displacement.

The use of the model relies upon the completion of an initial calibration stage, in which a robot would navigate across an area of terrain that was generally considered to be both

flat and clear of obstacles. During this time, a "baseline" vertical displacement model is generated by taking an average of the displacement recorded for each image row across a number of separate measurements.

Within the scope of the proposed final system, subsequent comparison of displacement magnitude and direction recorded for a given row is performed against the general trend of this "baseline" model as a means of identifying cases of "unusual" displacement behaviour. This comparison is re-evaluated upon every system cycle, following the generation of a new model to represent the changes in the scene captured by the latest images from the camera.

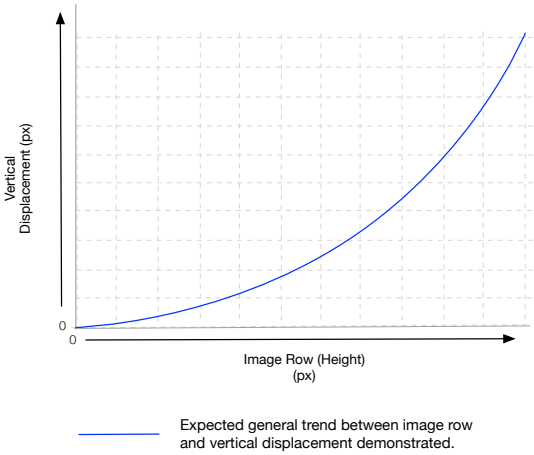

Figure 1.2: Example of a "perfect" vertical displacement model, showing the expected positive correlation between vertical position of a patch within an image, the the level of vertical displacement demonstrated.

Observing a significant extension or reduction in the length of displacement recorded between the "baseline" model and the latest version of the model would provide a strong indication to the presence of a potential obstacle (positive or negative) or change in the slope gradient of the ground (Figure 1.3).

Establishing the difference between a change in displacement caused by an obstacle, and one caused by a change in terrain gradient comes down to the evaluation of the pattern of displacement recorded across all of the rows in the image.

In the event of an oncoming obstacle, the rows representing that object in the image would be expected to demonstrate a significant increase (positive obstacle) or decrease (negative obstacle) in the level of displacement shown.

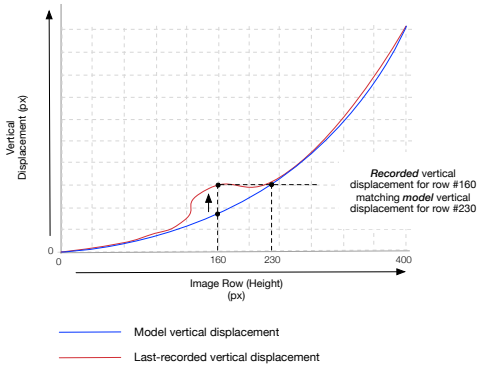

Figure 1.3: Example of vertical displacement model indicating the potential presence obstacle of a positive obstacle.

However importantly, this change would typically not be representative of the general trend in displacement that is observed across the entire height of image (indicated as a 'spike' in a 2D scatter plot (Figure 1.4).

Alternatively in the event of a change in terrain gradient, it would instead be expected that the vast majority of those pixels representing the ground would all demonstrate a similar change in displacement in terms of both direction and magnitude (Figure 1.5). This is because a change in the slope of the actual *ground* is likely to be observed across a much larger range of rows than what would be observed for an individual obstacle such as a rock or pit.

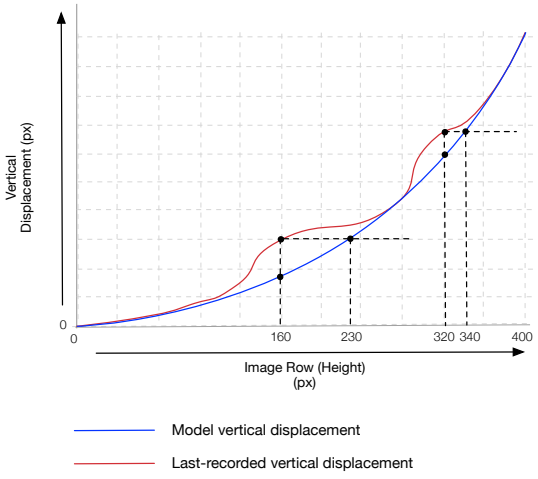

Figure 1.4: Simplified example of detection of multiple positive obstacles.

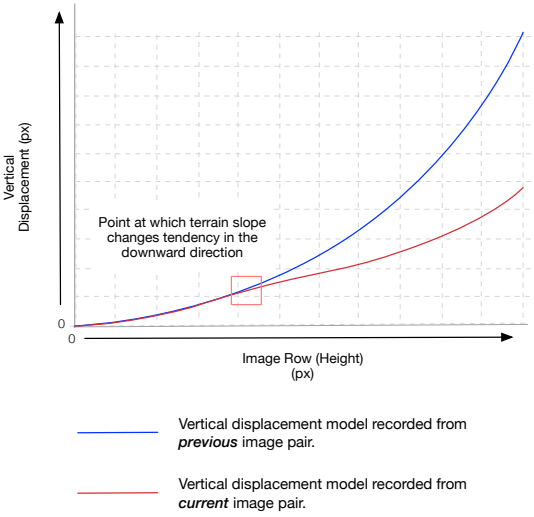

Figure 1.5: Simplified example of recorded change in terrain gradient indicating a downwards slope (general reduction in recorded displacement).

1.2.3.3 Calibration & Focus of Expansion

When considering the use of input supplied by an optical camera, a crucial aspect lies calibration of the camera lens. The primary requirement for performing calibration stems from the need to identify and correct the *distortion* observed in images that are captured through a *pinhole camera* lens [11].

Failure to correct such distortion can result in significant inaccuracies being observed between the perceived motion of an object captured in an image, and the actual movement that it demonstrates. The effects caused by *perspective distortion* on the perceived motion of objects positioned within the view of a forward-facing camera is a problem that is of particular importance to the success of the approach detailed in Section 1.2.3.

In the case of many robotic software projects engaging in tasks involving visual navigation or obstacle detection, the calibration of both the camera lens and physical setup allows for future conversion between native camera units (e.g. pixels) and their real world equivalents (e.g. centimetres) [16], [26]. While many computerised solutions to camera calibration do now exist [11], all continue to rely upon some form of manual involvement (normally surrounding calibration of the camera lens).

Through the understanding of a concept known as *Focus of Expansion* (FOE), it was identified that this additional level of complexity associated with camera calibration could instead be replaced with a much simpler approach, focussing exclusively on the analysis of optical flow of features that were contained within a confined "region-of-interest" located around the horizontal centre of the image.

When a camera is moved forward and an image is subsequently taken, all of the motion vectors observed in the resulting perspective image appear to begin from a single point centred along the horizon [40]. This 'single point' is known as the Focus of Expansion, and acts as the primary metric for an observer wishing to identify the direction in which they are currently headed [37] (Figure 1.6).

By restricting the tracking of features to only those that lay within a specific range of the FOE, it was hoped that this would allow for a reasonable reduction in the effects caused by perspective distortion without requiring the explicit calibration of camera parameters.

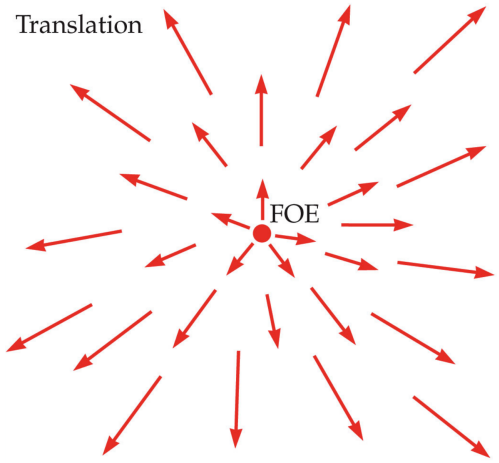

Figure 1.6: Visualisation of the perceived effects caused by Focus of Expansion. Courtesy [37]

1.2.4 Estimation of Robot Orientation

An important observation relating to the motion exhibited by a robot, is the notion of it being possible to separate such motion into two main constituent parts; *translation* and *rotation* [16].

In the context of this project, translation represents the displacement that a robot obtains while travelling along in a single direction. As already discussed in Section 1.2.3, it is currently possible using optical flow analysis, to record translational motion of a forward-moving robot by measuring the level of *vertical displacement* that is demonstrated throughout a series of images captured via a front-facing camera.

The other type of motion that can be demonstrated by a robot is of course, rotation. For a typical mobile robot, rotational movement actually constitutes a change in heading or orientation, which specifically, describes a rotation around the vertical axis [16]. In the opposite way to translational movement, changes in rotation can represented by the *horizontal* displacement of features tracked through a collection of images showing the scene in front a robot [25].

However, a crucial distinction to be made between these two modes of motion, is the difference observed towards the level of respective displacement that features within an image can demonstrate. In the case of translational motion, features detected towards the top of the image will show a much smaller level of vertical displacement than features located towards the bottom. This is a consequence of the effects caused by motion-parallax, and forms the foundation of the proposal made in Section 1.2.3.2 for supporting the inference of terrain gradient and obstacle detection.

In contrast, it is noted that a change in rotation will cause *all* features tracked within an image to move through the *same angle* [16] (Figure 1.7). This is very important, as it subsequently reveals how all of these features should theoretically demonstrate the same level of horizontal displacement regardless of how close or distant they are to the camera at the time of capture.

Taking inspiration from the work of Campbell *et al.* [16], it is predicted that taking the median of these horizontal displacement values will provide an estimation into the change of orientation that a robot has undertaken.

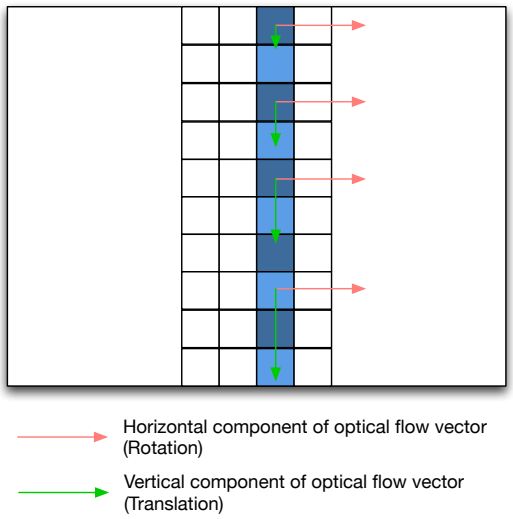

Figure 1.7: Diagram demonstrating the expected difference in behaviour between the separated vertical and horizontal components of observed optical flow vectors calculated from *forward-turning motion*. Notice that the horizontal-component vectors (i.e. rotation) all share the same magnitude, whereas the vertical-component vectors (i.e. translation) show an increase in displacement travelling towards the bottom of the image.

1.2.5 Estimation of Robot Speed

For supporting both the inference of terrain gradient conditions and detection of potential obstacles, it is the role of the vertical displacement model (Section 1.2.3.2) to provide a delineation of the distance by which tracked features within an image have moved, relative to the position at which they were last detected.

Using the relationship defined by the standard *speed-distance-time* equation:

$$\text{Speed} = \frac{\text{Distance}}{\text{Time}} \tag{1}$$

it becomes possible to estimate the speed at which the robot the travelling when it is assumed that the time interval between the capture of subsequent images remains constant.

Therefore, a proposed extension to the vertical displacement model would instead see a number of separate models produced during the calibration stage, where each would represent the distance of vertical displacement demonstrated by the terrain as the robot travels at differing speeds (Figure 1.8).

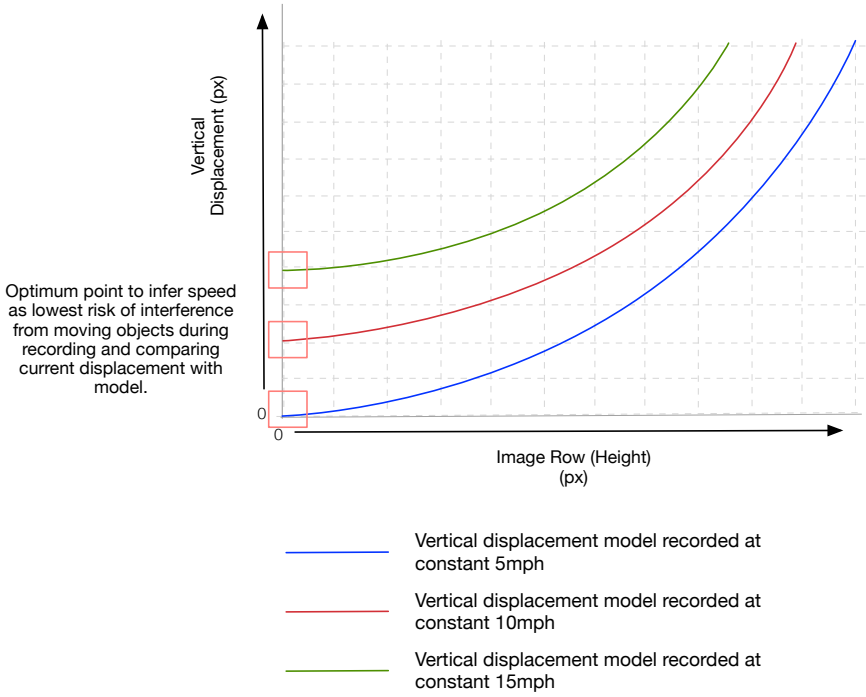

Figure 1.8: Diagram indicating examples of alternative vertical displacement models calculated while travelling at different constant speeds. To infer the current speed using the calibrated vertical displacement models, the current displacement would be compared with the recorded displacement across each of the alternative displacement models before selecting the model that best fits the current displacement as an indication of the current speed.

Once these multiple models have been established, it would not become an unfeasible task to further enhance the system in order for it to calculate which of the models best matches the current general trend of displacement demonstrated, which therefore would allow for it to present an estimation into current speed of travel.

1.3 Analysis

The main focus for this project falls upon conducting research as part of an investigation into the proposed hypothesis.

As such, the overall plan for the project intended to perform as much experimentation into the aspects discussed in the hypothesis (see Section 1.2.3) as was feasibly achievable within the three month deadline set for the major project.

1.3.1 Breakdown of Research Aims

Following the conceptualisation of the working hypothesis, it was necessary to devise a formal list of "high-level" research aims based upon the proposals that were previously set out.

Given that these were indeed research-focussed aims, it was expected that they would be completed within a specific sequential order, as this would allow for the work of each subsequent aim to build upon the work of its predecessor.

As there was a relatively high time constraint on the project (with respect to many of the previous projects investigated), it was decided that these aims should be decomposed further into primary and secondary categories.

1.3.1.1 Primary Aims

The following aims consist of those deemed most important in fulfilling the core motives behind the project investigation. Unsurprisingly, these 'primary' aims were also responsible for laying the 'groundwork' from which any 'secondary' aims could then continue.

1. Establish which out the following appearance-based template matching similarity measures:

   - Euclidean Distance
   - Correlation Coefficient
   - Histogram Correlation
   - Histogram Chi-Square

   best supports the predicted positive correlation between the vertical position of features within an image, and the level of vertical displacement demonstrated (see Section 1.2.3.2).

2. Implement a means of using the generated model of vertical displacement to identify potential obstacles and changes in terrain gradient based upon the predicted 'comparative trend' behaviour detailed in Section 1.2.3.2.

Note to the Reader:

It was deemed important at this point to inform the reader that due to significant issues discovered during experimentation, combined with a lack of author experience and general time constraints, it became progressively clear that work would only be able to focus exclusively on the first primary aim detailed above, before hitting the project deadline. Further discussion regarding this is available as part of the critical evaluation within Chapter 4.

1.3.1.2 Secondary Aims

Additional proposals from the hypothesis that provided further enhancements or abilities were classified as 'secondary' aims. While optimistically, it was hoped that such functionality could be implemented in the time available, it was deemed more than likely that such aims would fall outside of the scope of the major project.

1. Further extend the capabilities of the vertical displacement model as discussed in Section 1.2.5 to provide estimates into the current speed of travel demonstrated by the robot.
2. Add support for providing an estimation of the robot's change in orientation using the approach detailed in Section 1.2.4.

1.3.2 Breakdown of Experiment Work

As part of the initial planning stage, expected tasks relating to "experimental research" were further decomposed into one of four key stages. This was to ensure that there was a full understanding into all of the different kinds of work involved in performing research experiments, which in turn would allow for adequate attention to be given in planning the finer details for each experiment.

- **Stage One - Data Collection:** In order to evaluate the approaches proposed within the investigation, it would first be necessary to collect and verify image data that could provide an appropriate basis for testing.
- **Stage Two - Setup of Testing Environment:** Prior to beginning any experimentation, it was important to establish a stable and robust testing environment for the evaluation of the implemented methods. In the context of this project, this would include the implementation of an configurable 'testing rig' capable of running and evaluating multiple sequences of tests automatically.
- **Stage Three - Establishment of Ground Truth Data:** Compilation of verified results known to represent the "true" behaviour of the various aspects under investigation. Used as the means of evaluating the relative success of experiments conducted.
- **Stage Four - Implementation of Hypothesised Approaches:** Implementation of the actual approaches and predictions detailed within the proposed working hypothesis (see Section 1.2.3).
- **Stage Five - Evaluation of Experiment Results:** Analysis of the performance results obtained for each of the implemented approaches, focussing in particular on aspects regarding the accuracy of each method when tested against a variety of datasets.

Stages one and two would be performed initially, in order to ensure everything was prepared for the "bulk" of the investigative work to come within stages three and four. As

part of this investigative work, a number of potential appearance-based template matching methods were due to be implemented over a number of working 'iterations'.

Stages three and four would require repeating over every new iteration, each time focussing upon a different method to *implement* and subsequently *test*, before finally moving to stage five in order to conduct an evaluation into the performance of each of the implemented methods.

1.3.3 Deliverables

As is typically the case with research-based projects, it was expected that the majority of deliverables resulting from the completion of tasks would consist mostly of experiment results and evaluation conclusions:

- A collection of test images used to evaluate implemented experiments. This may either be images from an existing dataset [29] [8], [14], or images captured specifically for this project.
- A set of predicted experiment results used in evaluating the results obtained from each conducted experiment.
- A collection of recorded experiment results, represented as a series of plotted 2D-scatter graphs with the addition of moving averages to allow for the easy identification of expected behaviour trends.
- This report, containing information regarding the background to the experiment and hypothesis, details on the experiment methods and a discussion of the results obtained.

However as this project would also focus on producing software (an implicit requirement in the submission as department major project), inevitably there was the expectation of additional deliveries relating to both the development of software and undertaking of project-management:

- A series of individual 'test rig' applications provided for each primary experiment undertaken.
- Accompanying unit and regression tests suites for each separate software module.
- Accompanying documentation for produced software.
- Breakdown of the current status of remaining work, including details of both "high-level" and constituent tasks, in addition to details regarding previous project performance (e.g. 'burn-down' and 'throughput' charts (discussed further in Section 1.4.2).

1.4 Research Methodology

This was a project in which the work involved was potentially subject to a great deal of change and re-organisation, arising as a result of attempting to solve a problem that had no fully-defined solution.

Given this fact, any amount of significant upfront planning and design was deemed unfeasible, as in the most likely of cases, it was expected that a significant proportion of these "fixed" decisions would eventually undergo some kind of modification following the discovery of issues or better alternatives.

Experiments were instead conducted over a number of project 'iterations', in which all of the work associated with a particular experiment would be completed before moving onto the next experiment. This included the design of the experiment itself, the development and testing of associated software, and the gathering and evaluation of result data.

1.4.1 Qualitative & Quantitative Research

The project consisted of both qualitative and quantitative research aspects. Qualitative research focussed on providing the outline to the proposed hypothesis [7], aiming to investigate the reasons behind *why* it was possible to use visual stimuli in aid of detecting potential hazards and changes in gradient across the observed terrain. This kind of research was conducted early within the project timeline and took the form of reading existing papers and examples of previous visual-based projects relating to the topics identified for this project.

Quantitative research involved the gathering and subsequent analysis of data in order to try and identify *how* various approaches could be used to solve the question topics uncovered within the qualitative research (i.e. the hypothesis) [7]. In the context of this project, all of the experiments discussed in Chapter 2 fell under the category of quantitive, rather than qualitative research.

1.4.2 Management Process

Traditional, plan-driven approaches towards software development tend to expect all requirements are gathered up front prior to beginning any planning, design or implementation. For this project, such an approach would have been unsuitable, as it was likely that "requirements" would be created, modified and removed much later into the project as a result of recent discoveries in the successes and failures of previous experiments.

Instead, the decision was taken to adopt a SCRUM-based methodology for the planning, recording and organisation of tasks, work would be partitioned into a series of time-boxed releases and subsequent sprints. By following an agile methodology process, development work could begin more quickly, and in conjunction with ongoing research and experiment evaluation. Through choosing to begin building working software early, even with limited functionality, it would become possible to potentially contribute some of this work back into ongoing research (e.g. use of the same testing rig "shell" within a

new experiment) and in turn improve the overall efficiency.

As the SCRUM methodology is founded upon empiricism [44], this allows it to appreciate that unpredicted changes cannot effectively be addressed using a plan-driven approach. The work conducted for each iteration was prioritised based upon discussions with the project supervisor. New work items were added to a list of outstanding tasks (product backlog), before being organised into a specific collection of tasks (sprint backlog) scheduled for completion within a particular sprint.

Each sprint followed a "five-day on, two-day off" development cycle, which provided a buffer to help in situations where work took significantly longer than expected. Sprint planning meetings were arranged for the same time as the weekly group project meeting, allowing for planning of work for the coming week, along with the opportunity to discuss changes that either party felt were required in light of new experiment results.

In terms of tracking project progress, feedback for sprints and releases was provided via a combination of burn-down and throughput charts (see Appendix C - Figures C.1 and C.2), that were created using the *Burndown.io*[1] and *Waffle.io*[2] web services.

While both provided graphical representation of work completed, throughput charts focussed on comparing the frequency of completed tasks over a fixed range of weeks, whereas burn-down charts plotted the amount of outstanding work within an individual sprint or release, in relation to the time that was left before it closed.

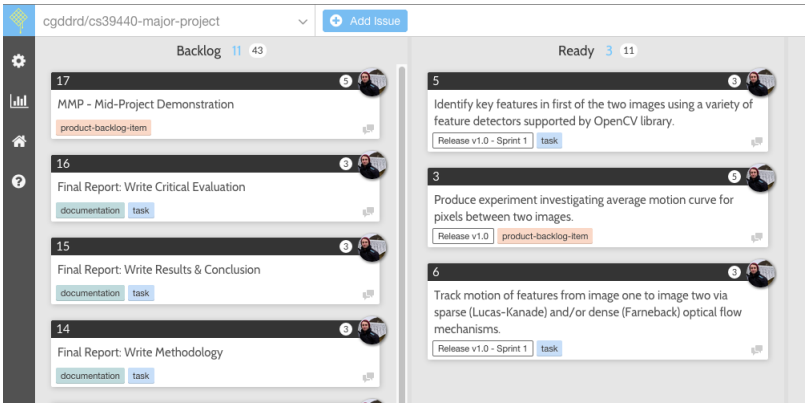

Figure 1.9: Day-to-day tasks were managed via the use of web-based KANBAN-style interface (Waffle.io[3]), that connected directly to the project Github account where all of the SCRUM backlogs were stored.

[1] `http://burndown.io/#cgddrd/cs39440-major-project`
[2] `https://waffle.io/cgddrd/cs39440-major-project`

Chapter 2

Experiment Methods

This chapter aims to provide a discussion into the experiments implemented with respect to the investigation detailed in Chapter 1, with emphasis on the adopted approaches adopted and the resulting actions of handling issues encountered.

The results subsequently collected from these experiments are presented in Chapter 3.

2.1 Collection of Appropriate Test Data

Prior to beginning the implementation of any experiments, it was first necessary to identify and collect sets of sample images which would accurately represent the types of input expected upon the deployment of a completed system into a live scenario.

2.1.1 Image Data Requirements

As discussed previously in Section 1.2.1, for the simplification of the *primary aims* presented in the working hypothesis, the assumption was drawn that at least in early stages of investigation, the system would make exclusive use of images captured from a single camera positioned to face in front of the robot that was also limited to moving in the same direction as the camera (i.e. forward-motion only).

When identifying appropriate test data for use in experiments, a the following characteristics were considered:

- The images must provide a reasonable field of view of the current scene above and below the horizon line (i.e it was important to capture objects within the scene located both far away and close to the camera)
- The images must show the act of forward translation through the current environment. This would be most obvious through the observation of a vertical displacement in the negative direction (i.e. downwards) visible in features located along the ground plane.

- The images must not show forward translation as horizontal movement across the image plane (i.e. no images captured from cameras looking out from the side of a robot moving in a forwards direction).
- The images must be of a size and quality reasonably expected of a standard "point-and-shoot" consumer-grade camera.
- The images must have been originally captured in colour, using the "default" colour space supported by the camera (typically this would be RGB for standard consumer-grade camera).
- The images should demonstrate minimal change in rotation or pitch (i.e. should be taken across a flat surface).

In addition to these "baseline" requirements, additional requirements were also defined with the intention for use within specific experiments focussing on the identification of particular aspects in motion behaviour. These secondary requirements were intended to be used on an "as needed" basis and came with the possibility of the opposite statement to the ones detailed below being desired in certain situations:

- The images should demonstrate minimal rotational motion (i.e. no examples of the robot turning to change direction)
- The images should capture terrain that is predominately flat and free of major obstacles both positive (e.g. rocks) and negative (e.g. pits).

Unfortunately due to time constraints, some the planned experiments could not be completed within the the major project, as a consequence of this, not all examples of images meeting every one these requirements were captured. However, it was still important to define these requirements before beginning any experimentation and given more time, would still prove to be valid.

2.1.2 Existing Datasets

At the beginning of the project, some time was initially devoted to searching for any existing datasets that could provide suitable imagery. One main advantage to using existing datasets, is that typically other previous projects have already had the opportunity to verify that the data is both accurate, and provides a sufficient level of variability that proves crucial in testing the robustness of the systems that make use them as part of their evaluation.

In the majority of cases, existing datasets also come pre-packaged with appropriately verified ground truth data, thereby preventing the need for projects that subsequently chose to use them having to produce their own ground truth results, consequently saving on both time and resources.

While a number of previously published datasets were found to be available [29], [8], [14], unfortunately none were found to be suitable in relation to the requirements detailed in the previous sub-section.

2.1.3 Manual Datasets

Following the lack of appropriate existing datasets with the field, it was necessary that a bespoke datasets would have to be created manually. While in the short term this meant an increased work load, it also presented the opportunity to capture datasets that would specifically meet the needs of the experiments.

2.1.3.1 Camera Rig Setup

In the pursuit of capturing image datasets, a wide range of approaches could have potentially been adopted. One initial idea considered requesting the use of one of several 'Pioneer' robots owned by the Computer Science department at Aberystwyth University. By rigidly mounting a camera to the front of one of these small wheeled robots, and remotely instructing it to move forward by a set distance before manually triggering the camera, it would be achievable to capture a collection of images in which each demonstrated an equal level of displacement between that and the next image.

As part of a particularly elaborate setup, it would have perhaps been feasible to provide an interface to the mounted camera (perhaps via USB or serial) before programming the Pioneer robot into automatically capturing images at set intervals, while following a pre-determined path through the environment (making use of the on-board sonar and odometry capabilities).

While these approaches would certainly provide an ample solution, following a discussion with the project supervisor, it was deemed that, at such an early stage in an investigation that was already limited in time remaining, efforts would be better spent focussing on conducting actual research, as opposed to the collection of test data.

Nevertheless, a means of manually capturing image datasets was still required. As such, the decision was taken to adopt a 'simplified' approach, that in exchange for greater manual involvement, could consequently be brought into service in a shorter timeframe.

Compared to the use of the robot, this approach was certainly less 'sophisticated' in terms of its setup (Figure 2.1), consisting only of:

- Two cardboard boxes (one rectangular and one angled);
- A consumer-grade "point-and-shoot" camera (Panasonic Lumix DMC-FS18 (2011));
- A standard 30cm ruler;
- A standard spirit level;

However, as well as being quick to initially build, this setup would go on to prove to be a lot more portable and a great deal cheaper to modify and fix than one of the Pioneer robots.

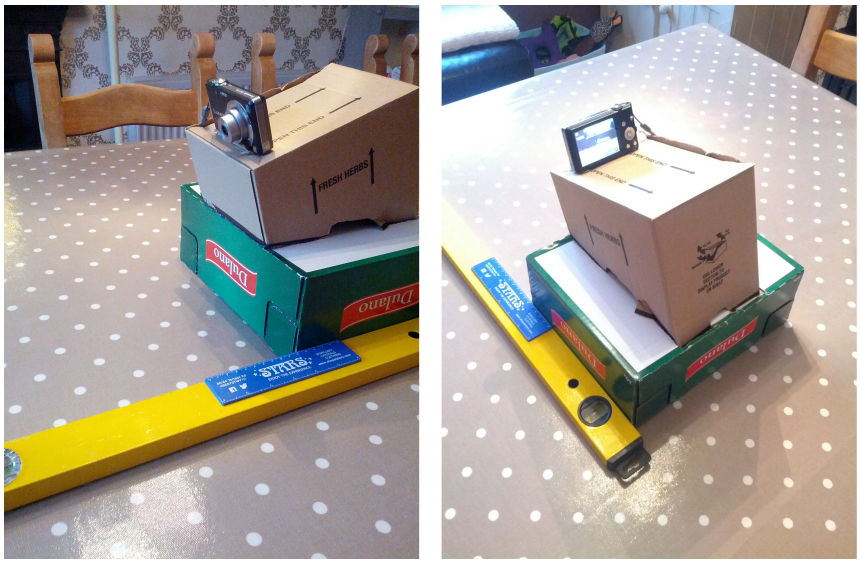

Figure 2.1: The camera-rig setup used to capture the experiment datasets from the front and back profiles.

2.1.3.2 Capture Locations

A selection of locations were chosen for the purposes of capturing datasets. To ensure an appropriate level of variability, each dataset focussed on a different type of terrain environment that it was predicted a robot may face during live deployment.

Datasets were captured at four separate locations (Figure 2.2), split between indoor and outdoor sites to provide variation in lighting conditions and terrain type:

1. **Site 1:** Living room rug with brightly-coloured simple-shapes (Indoors) (Natural light entering from the right)
2. **Site 2:** Brick-paved road (Outside) (Natural light)
3. **Site 3:** Heavily-patterned asian rug upon white-tiled floor (Inside) (Partial natural light from behind camera, partial artificial light)
4. **Site 4:** Slate-tiled footpath (Outside) (Natural light)

Figure 2.2: Examples of the first frame captured from each of the four location datasets.

2.1.3.3 Capturing Process

The method behind the use of the 'homemade' camera rig to capture the image datasets also proved simple in design. This was regarded as favourable, given that it relied on manual involvement where human error subsequently becomes a potentially big issue.

Figure 2.3 below details the outline of the method used to capture an individual image dataset.

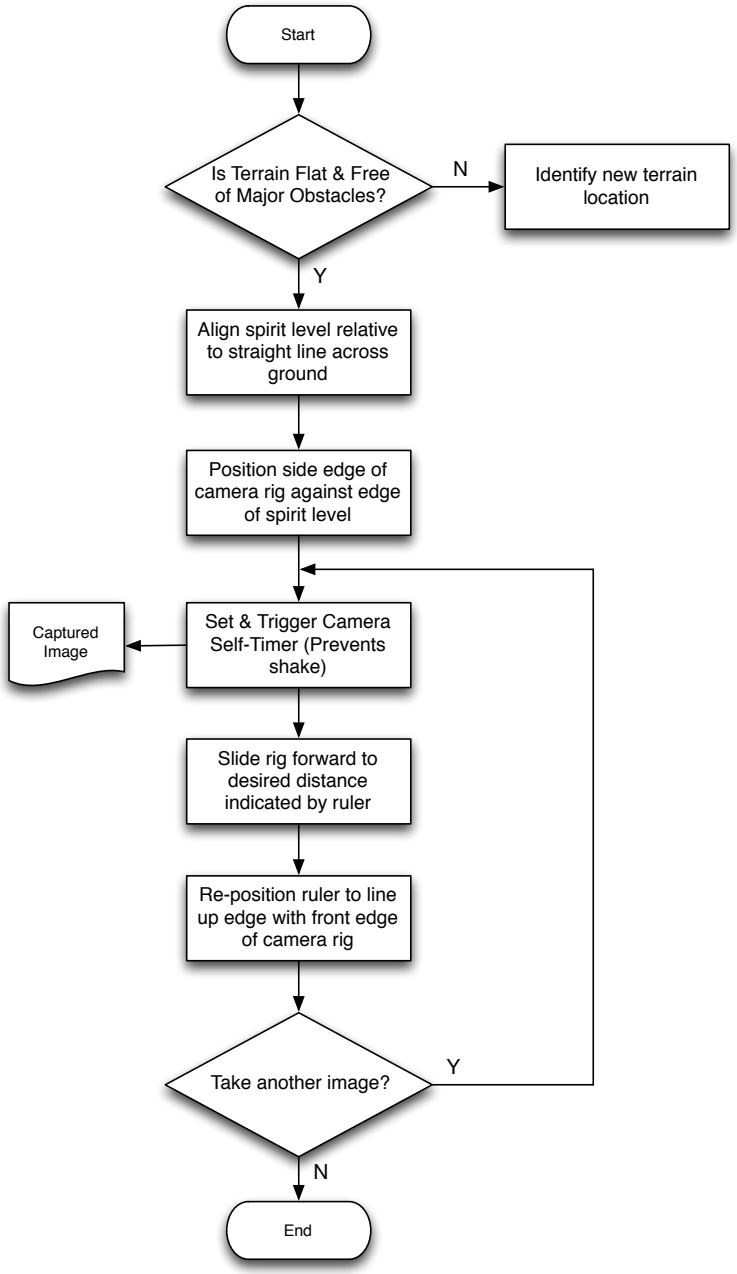

Figure 2.3: General method process employed for capturing dataset images.

While the method for capturing images followed a "fixed" procedure, it did allow for

changes in the type of terrain captured, and the level of set displacement by which the camera was moved between subsequent images.

2.1.3.4 Calibration of Physical Setup

Although one of the aims of the hypothesis was to avoid the need for calibration of the camera, it was decided that in the interests of good experiment practice, measurements regarding the physical setup of the camera rig should be taken.

Following the process outlined in the work of Campbell *et al.* [16], the calibration would provide a future opportunity, should the need arise, to calculate the mapping between coordinates in the image plane and the ground plane.

The physical setup of the camera rig matched the same general setup as described by Campbell *et al.* [16] (Figure 2.4), where h refers to the height of the camera from the ground, and d refers to the distance between the front edge of the rig base and the position at which the principle ray from the camera lens intersects with the ground plane [16].

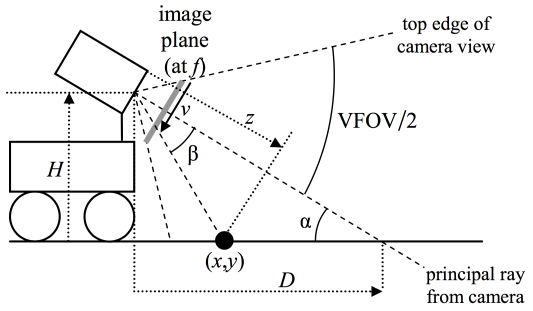

Figure 2.4: Diagram originally produced by Campbell *et al.* [16] depicting the physical setup of the camera rig used within their work. The camera rig built for this project followed a very similar setup.

To establish the distance d, a target T was positioned on the ground such that it lay within the centre of the camera's viewfinder (Figure 2.6). The distance between the position of the target T and the front of the rig was then manually measured in order to arrive at the final length.

After also manually measuring the height h, it became possible to calculate the tilt α of the camera using basic trigonometry:

$$\tan(\alpha) = \frac{h}{d} \tag{1}$$

2.1.4 Artificial Datasets

While unfortunately never pursued in this project due to time limitations, an alternative approach to gathering robust testing data was to capture motion sequences set within a 3D virtual environment.

A key advantage of creating artificial image datasets stems from the level of control that becomes available over properties including the choice of lighting, terrain textures and landscape geometry. This high degree of granularity that makes using computer-generated datasets a popular choice for many vision-based projects, perhaps especially within earlier stages of a project, where one is simply attempting to identify bugs in the

system, rather than evaluating hypotheses [29], [8], [14].

Other potentially very useful consequences of using computer-generated datasets include control over the amount of noise an image may contain (e.g. choice to include or remove specular reflections), and the high degree of speed at which new datasets adopting different condition configurations could be created almost "on the fly" in order to test new ideas and approaches within an experiment.

Figure 2.5: Image indication the visual target used as part of the manual calibration of the camera rig setup.

The creation of artificial datasets would have provided an opportunity to test the robustness of the experiment methods against a much broader range of conditions than what was feasibly possible using only real-world imagery. However the use of datasets captured from real-world environments was still arguably the most crucial in terms of evaluating the performance of experiment methods, given that these would ultimately provide the closest representation to the input expected within a live deployment.

2.2 Establishment of Ground Truth

With respect to scientific investigation, the term "*ground truth*" refers to measurement data or results that are known to represent the *absolute truth* of a particular condition or observation which subsequently can be used to determine the accuracy of reported results from a system or method currently undergoing testing and evaluation.

In the context of computer vision, ground truth data will represent observations within image datasets that are known to exist in the real-world, and be accurate.

Due to the very nature of what ground truth data represents, generating such data can still typically only be accomplished through means of manual verification, where human observation continues to be deemed as the 'state of art' in terms of providing the best possible accuracy available as to the *fundamental* truth of a condition or observation [24].

A fundamental issue with this approach to the collection and verification of ground truth data, is it tends to be costly in terms of both time and resources. Kondermann [24] states that this large requirement in effort to generate ground truth data in combination with its relatively limited applicability in terms of evaluation, consequently results in many vision-based projects failing to provide reference data that is either comprehensive or accurate.

For particular types of reference data, some semi-automated methods do exist for gathering high-accuracy measurement results (e.g. extracting depth information about a 3D environment using laser scanning techniques) [20]. However, these approaches still re-

quire at least some manual verification in ensuring that this accuracy is at a level sufficient enough for it to be trusted as ground truth.

In terms of generating ground truth, one of the most challenging types of visual behaviour to document is optical flow [24]. This is because no sensor currently exists that is capable of detecting the effects of optical flow directly, and in most situations (except perhaps within very simple artificial datasets) attempting to perform manual annotation also proves to be an extremely difficult task to achieve with the kind of accuracy that is expected for use as ground truth data [20].

Note to the reader:

Within the context of the experiments that were conducted it was agreed by both the author and the project supervisor that in the interests of time, the expected behaviour predicted in the hypothesis (namely "*greater vertical displacement is shown approaching the bottom of an image than towards the top*") would serve as sufficient 'un-official' ground truth from which the performance of the methods implemented could be evaluated.

2.3 Design of Experiment Setup

Prior to beginning the implementation of specific experiments, it was first necessary to design and implement an underlying software infrastructure with the aim of maximising the *efficiency*, *flexibility* and *repeatability* of the experiments conducted:

- **Efficiency:** The testing infrastructure should execute each experiment in a way that makes efficient use of data and system resources (including but not limited to: imagery datasets, experiment parameters and host computer CPU and memory allowance).
- **Flexibility:** The testing infrastructure should provide support for conducting an experiment under multiple combinations of method parameters entered by the user prior to initiation.
- **Repeatability:** The testing infrastructure should ensure that each run of an experiment is treated as new and conducted in complete isolation to any previous run.

2.3.1 Programming Languages

For this project, it was primarily the selection of specific external libraries that determined the programming languages that would be adopted.

The original intention saw the entire project developed in C++, as this matched the 'native' development language of the OpenCV library [10] that was expected to provide a significant contribution to the implementation of the appearance-based similarity measures under investigation. While the OpenCV library did provide a comprehensive range of bindings for other programming languages including Python, Java and C#, it was felt that choosing to develop in C++ would prove to be the most beneficial in terms of maximising available functionality (i.e. not all functions within OpenCV were available for

alternative languages) and maximising the level of support available from both official documentation and the wider community.

Following the completed implementation of the *first experiment* (detailed in Section 2.4.1), this original intention was altered to instead change from C++11 to Python 2.7 as the primary development language.

This decision was not taken lightly, with the author being very much aware that by choosing a language with which they were unfamiliar, there was a high probability of the project incurring delays resulting from both learning the actual language, while also having to most likely dedicate additional time to fixing "common" bugs typically associated with being new to the language. In spite of this, it was decided that while it may potentially cost time in the short term, switching to Python would provide a number of key benefits in the long term.

The first of these benefits was that in comparison to C++, the language provided significantly more functionality "out of the box" due to it being situated at a "higher" level programmatically. While reviewing the implementation of the first experiment, it was found that on a number of occasions, a function that had to written manually in C++ (e.g. standard deviation and variance of a numeric collection), was already available within either the Python language as standard, or one of its primary third-party libraries (e.g. *Numpy* [50]). While it was true that many of these functions did not take particularly long to implement in C++, there was a clear benefit to having such functions already implemented in libraries that were known to be both optimised and peer-tested.

Secondly, following research into the Python language, it soon became clear that it was a popular choice amongst scientific and engineering teams due mainly to its impressive level of support for scientific and numeric computing [35] and promotion of 'natural language' within its syntax that is known to aid in supporting rapid prototyping [39]. These were both characteristics that could be applied to the needs of this project, and hence were deemed beneficial to future development.

A further reason for switching to the Python language, which would in time become one of the most significant, was the prospect of being able to make use of the *iPython* interactive computing library [34]. Following the discovery of this library during research looking into the use of Python within scientific applications, it became apparent that it had the potential to provide significant capabilities in terms of supporting the automated execution and presentation of experiments and their results. This approach provided a more intuitive means of displaying and analysis than what was originally available using just a command-line based interface.

One of the most commonly noted disadvantages of using Python describes the fact that as a scripted language, its general performance is considered to be slower overall than that of compiled languages such as C++. While this was initially of some concern, it was later addressed through the use of a third-party library (Cython [9]) that enabled computationally expensive functions originally written in Python to be converted automatically into compiled C code.

2.3.2 Development Tools

Given the author's lack of prior knowledge in C++ or Python, it was decided that the use of an IDE for development would be most suitable for development, as they would be able to provide support facilities including code completion, automatic refactoring and powerful debugging facilities.

For C++ development, the Xcode IDE[1] was chosen. While this was primarily a decision based on ease of having the IDE already installed, Xcode did provide a high level of support for managing the deployment of C++ applications, including a facility to automatically generate the 'Makefiles' used to install the released version on another computer.

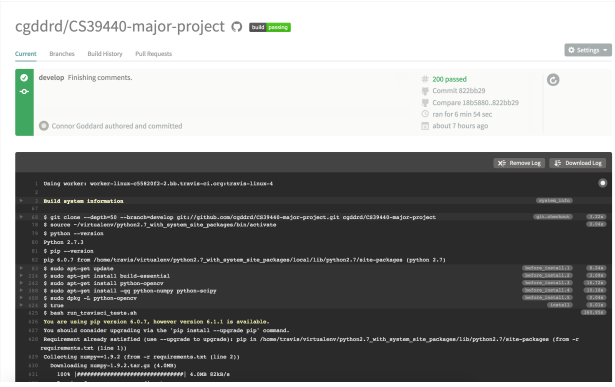

Figure 2.6: Travis CI provided automated builds and test suite execution upon each commit to the Github repository

Python development was conducted within the PyCharm IDE[2] making use of the educational license provided by JetBrains. Of particular benefit was the support provided for the automatic management of external dependencies and libraries, which included the ability to export a complete list of a project's dependencies and their precise version numbers.

Version control was managed by git, with repositories hosted on both Github[3] and a separate private git server. The project also made use of the Travis CI[4] continuous integration platform to provide automated building and testing of software. Integrating directly with the Github service, new builds would be triggered automatically upon pushing of a new commit to the remote repository. Should any part of the build process fail (including unit test failures), the service would send an email notification detailing the type and source of the error.

2.3.3 Core Libraries

The project made use of a selection of libraries to support in the development of the both the experiment testing framework, and the experiment methods themselves.

The *OpenCV* library [10] became a vital tool in the development of this project, providing a large number of "core" image processing and computer vision facilities including colour space manipulation, sub-region extraction and template matching (see Section 2.3.4). This project referenced the latest stable release at the time of writing (2.11.0), and made use of both the C++ and Python bindings as discussed in Section 2.3.2.

[1] `https://developer.apple.com/xcode/`
[2] `https://www.jetbrains.com/pycharm`
[3] `https://github.com/cgddrd/CS39440-major-project`
[4] `https://travis-ci.org/cgddrd/CS39440-major-project`

For aspects of the implementation developed in Python (approximately 75% of code written), the project made extensive use of the *Numpy* [50], *Cython* [9], *Matplotlib* [21] and *iPython* [34] libraries.

Please see Appendix A for a detailed description of all third-party libraries utilised within this project.

2.3.4 Implementation of Template Matching Similarity Measures

All of the experiments conducted within this project focussed on the analysis of four specific template matching similarity measures for identifying corresponding sub-regions between two subsequent images of a forward-motion sequence.

The general process for performing template matching, regardless of the choice of similarity metric, remains fundamentally the same [4]. Where a template image (i.e. the image we are searching for) is convolved through a (typically) larger search image (i.e. moved through the search image 1 pixel at a time). At each increment in position, a similarity "score" is calculated using a specific similarity metric calculation, if this new score is deemed "better" than the current high score (which can be lower or higher depending on the similarity metric chosen) then that position within the search image becomes the new location where the item in the template image is most likely to be.

An outline of the process described via pseudocode is available in Appendix B - Algorithm 1.

The similarity measures subsequently implemented in the experiments were chosen specifically in order to compare the performance between the two major categories of appearance-based matching technique.

2.3.4.1 Distance-Based Similarity Measures

The first two measures were implementations of *distance-based* matching. Under this approach (also known as *correlation-based* matching), the similarity between two images is calculated by comparing the pixel-wise properties of each image (i.e. each pixel is compared with the corresponding pixel in the other image) using a particular type of measure. [47].

For this investigation, the *Euclidean Distance* and *Normalised Cross-Correlation* measures were selected as they both demonstrated differences that were suitable for comparison with respect to their approach to calculating pixel-similarity and robustness to changes in lighting.

In the field of mathematics, the Euclidean Distance is a measure of the length of the vector that connects two distinct points located within Euclidean space [47]. In the context of appearance-based template matching, it is possible to utilise the Euclidean Distance to calculate a measure of similarity between pixel values of two images as defined by:

$$d(x,y) = \sqrt{\sum_i \sum_j [\mathcal{I}(x+i, y+j) - \mathcal{A}(i,j)]^2} \tag{2}$$

where $\mathcal{A}$ is the template image containing the pattern we are searching for within image $\mathcal{I}$.

A crucial characteristic of the Euclidean Distance (and indeed similar approaches including Sum of Squared Difference), is that it relies on comparing the *intensity* of pixels in order to provide a measure of image similarity. This is very important, as it subsequently enforces an implicit assumption (known as the *brightness constancy constraint*), that the values of corresponding pixels do not change between the two images [47]. This in effect, means that the measure can be substantially affected by changes in brightness between the two images under comparison.

Normalised cross-correlation takes its roots from the field of signal processing, and unlike Euclidean Distance does not rely on differences in pixel intensity to establish the similarity between two images [47]. Instead, this approach takes the product (known as the *cross-correlation*) of pixel values as defined by:

$$d(x,y) = \sum_i \sum_j [\mathcal{I}(x+i, y+j) \cdot \mathcal{A}(i,j)]^2 \tag{3}$$

However as it stands, this equation currently remains susceptible to the effects of differences in image brightness just in the same way as the Euclidean Distance. It is however possible to overcome this issue by first *normalising* the pixel values prior to scoring:

$$d(x,y) = \frac{\sum_i \sum_j [\mathcal{I}(x+i, y+j) \cdot \mathcal{A}(i,j)]^2}{\sqrt{\sum_i \sum_j [\mathcal{A}(i,j)]^2 \cdot \sum_i \sum_j [\mathcal{I}(x+i, y+j)]^2}} \tag{4}$$

As well as dramatically improving resilience to differences in image exposure, normalising also guarantees the returned score will fall between -1 and 1 [47]. This can prove helpful when used as part of larger application.

Given that both Euclidean Distance and Normalised Cross-Correlation were already available as optimised functions within the *OpenCV* library (`norm` (specifically the `L2_NORM` type)[5] and `CV_TM_CCORR_NORMED`[6] respectively), it was decided that in terms of performance, no real benefits would be gained from choosing to re-implement the functions manually. However during early stages of the investigation, some attempts were made to implement these functions regardless as part of a drive to enhance the author's own understand of these key topics.

By choosing to keep within the OpenCV eco-system, it also became possible to utilise the extensive template-matching facilities already supported within the library through the use of the `matchTemplate`[7] function.

[5] `http://docs.opencv.org/modules/core/doc/operations_on_arrays.html#norm`

[6] `http://docs.opencv.org/doc/tutorials/imgproc/histograms/template_matching/template_matching.html`

[7] `http://docs.opencv.org/modules/imgproc/doc/object_detection.html?highlight=matchtemplate#matchtemplate`

2.3.4.2 Histogram-Based Similarity Measures

An alternative approach to calculating the measure of similarity between two images using only their "appearance", is that of *histogram-based* matching.

While also focussing on the value of image pixels as a measure of similarity, this approach differs from distance-based approaches by choosing to represent and later compare this data as *histograms*, rather than comparing between the actual value of corresponding pixels directly.

By choosing to represent an image as a histogram, it becomes possible to organise the various data aspects that of an image's appearance into a collection of value-specific counts (known otherwise as *bins*). One of the advantages to using histograms is that it becomes easy to generate alternative representations of the same image, where each focusses upon a specific type of 'metric' for describing that image (e.g. A separate histogram for each channel in the 'RGB' colour space).

In order to use histogram representations for the comparison of two or more images, the histogram for each image must first be calculated. As such, when performing histogram-based matching a couple of additional stages are required:

1. Compute the histogram for the first image;
2. Compute the histogram for the second image;
3. Perform comparison between the first and second histograms;

This is in contrast to distance-based approaches, which as a result of working within the original representation of an image, can be completed within a single comparison operation.

The approach to comparing between two histograms follows closely with the method of comparison adopted within distance-based matching, whereby a specific *distance metric* is calculated in order to provide a measure of similarity [12].

The two metrics selected for evaluation within this investigation were *Correlation* and *Chi-Square*. Unlike the distance-based metrics chosen previously, these were selected entirely at random from four possible similarity measures provided by the *OpenCV* library (Correlation, Chi-Square, Intersection and Bhattacharyya distance)[8].

The Correlation metric is defined within the *OpenCV* library as:

$$d(H_1, H_2) = \frac{\sum_I [(H_1(I) - \bar{H_1})(H_2(I) - \bar{H_2})]}{\sqrt{\sum_I [(H_1(I) - \bar{H_1})^2 (H_2(I) - \bar{H_2})^2]}} \tag{5}$$

where

$$\bar{H_n} = \frac{1}{N} \sum_J [H_n(J)] \tag{6}$$

[8] http://docs.opencv.org/doc/tutorials/imgproc/histograms/histogram_comparison/histogram_comparison.html

and N represents the number of 'bins' used to organise histogram data.

When scoring the similarity between two histograms via correlation, a higher score represents a better match, with 1 representing a complete match, 0 representing that no correlation could be obtained (i.e. random) and -1 representing a complete mismatch [12].

Originating from the field of statistical analysis, the Chi-Square metric defines a class of tests that can be used to determine how closely a collection of 'observed' data matches the data that would be expected according to a particular hypothesis [32].

For the purposes of comparing image histograms, the *OpenCV* library provides an implementation of a specific type of chi-squared test known as Pearson's chi-square test [12]:

$$d(H_1, H_2) = \sum_I \frac{(H_1(I) - H_2(I))^2}{H_1(I) + H_2(I)} \tag{7}$$

For chi-square, scoring is conducted in the opposite fashion to correlation, where a lower score indicates a better match. In this case, 0 represents a perfect match and any value greater than 0 represents a worse match.

Just as in the case of the distance-based metrics, the *OpenCV* library provided optimised support for computing and subsequently comparing image histograms via the `calcHist`[9] and `compareHist`[10] functions respectively.

While histograms provide an alternative representation of an image, they still contain fundamentally the same data. This means that in the case where one image is significantly brighter than the other, it becomes unlikely that a histogram-based technique will be successful in identifying a correct match or non-match.

Using the same underlying theory as Normalised Cross-Correlation, it was possible to help minimise brightness-relating matching errors by ensuring that both histograms were normalised. Although none of the histogram similarity metrics included a built-in normalisation step (as is the case with Normalised Cross-Correlation), *OpenCV* provided a specific `normalize`[11] function that enabled the histograms to become normalised prior to any comparison.

2.3.4.3 Colour Space

While the normalisation of image data can help reduce matching errors caused by changes in brightness between two images, a means of further enhancing this resilience is to convert the colour space of both images to one that is capable of separating the *colour* of a pixel from its *brightness* [49].

As a result it then becomes possible to compare the two images using only the *colour information*, effectively ignoring the effects of changing brightness that can cause two pixels that are in fact visually the same, to be deemed as different.

Within this investigation, all testing images were converted from RGB (loaded as BGR in

[9] `http://docs.opencv.org/modules/imgproc/doc/histograms.html#calchist`
[10] `http://docs.opencv.org/modules/imgproc/doc/histograms.html#comparehist`
[11] `http://docs.opencv.org/modules/core/doc/operations_on_arrays.html#normalize`

OpenCV) to the *Hue-Saturation-Value* colour space prior to the beginning of any experiment tests. This was achieved using the `cvtColor`[12] *OpenCV* function with the `BGR2HSV` conversion type parameter. This later allowed the 'Value' channel to be ignored as part of all image comparisons made during the template matching procedures conducted within an experiment.

2.3.5 Automated Test Rig

One of the most significant design aspects of the project was the decision to implement an automated testing rig to facilitate in the conducting of experiments. The core motivation behind the test rig, was to find an efficient approach to executing experiments wishing to test a wide combination of potential method parameters, in a manner that was both comprehensive and repeatable.

While of course defining these parameters would remain the responsibility of the user, the primary use case of the system would see all of the possible test parameters specified *right at the start*, before initiating the test rig to repeatedly perform and document each experiment in isolation, until all possible combinations of the test parameters for that experiment had been attempted.

The most obvious advantage to using such a system was the significant reduction that could be made in terms of human resources. Thus, allowing the user to focus on tasks that were less suited or perhaps not possible for a computer to achieve, such identifying key aspects within the results gathered from the last run of the test rig.

However, another advantage perhaps not as immediately visible, was the almost complete removal of the risk for human error that otherwise (assuming a moderate number of potential parameters) would be set fairly high, given the situation where a user had to manually run each possible combination of experiment. Although, it should be noted that as the user would continue to interact in some form, there was always still a risk of experiment results being tainted as a consequence of human error being introduced.

2.3.5.1 Execution of Experiment Tests

In terms of interaction with the user, the test rig provided a command-line based interface that provided support for the input of all test parameters and configuration options using a collection of specific parameter arguments.

While certain arguments related only to specific experiments, an example selection of arguments common to all were defined as follows:

In addition to the command-line interface, it was also possible to interact with the test rig programatically. This was an approach utilised extensively while generating the final results as discussed in Section 2.3.7.

[12] `http://docs.opencv.org/modules/imgproc/doc/miscellaneous_transformations.html#cvtcolor`

Parameter Name	Argument Specifier	Type	Input Format	Description
Image Pairs	`--images`	`(String, String)[]`	`"<pair_1_image_1>, <pair_1_image_2>", "<pair_2_image_1>, <pair_2_image_2>"`	Specifies the file paths for one or more pairs of images.
Patch Sizes	`--patches`	`Int[]`	`[50, 100, 200]`	Specifies a collection of square-dimension sizes to use for extracting template patches.
Matching Methods	`--methods`	`String[]`	`["DistEuclidean", "HistCorrel"]`	Specifies which similarity measures to use in matching corresponding patches.
Draw Plot of Results	`--drawplot`	`Boolean`	`--drawplot`	If specified, instructs the test rig to render a graphical plot of the results for each experiment.

Table 2.1: Table of common input parameters used to run automated tests via the testing rig.

2.3.6 Library of 'Core' Functions

During a review of the completed C++ implementation for the first experiment, it came to light that it would be desirable in many cases to reuse the same functions that had previously been written across a number of future experiments.

Following the decision at that same point to switch from C++ to Python (discussed in Section 2.3.1), it was deemed as an opportune moment to re-factor all of the common functions into a single library that could be shared across all projects, but whose code remained de-coupled from any single experiment implementation.

2.3.6.1 Structure

The design of the *'Terrain Shape Estimation'* (TSE) library saw the creation of nine key component Python modules, each focussing on providing specific functionality for a given catagory of tasks:

- **TSEImageUtils:** Provides functionality relating to image processing and template matching.
- **TSECImageUtils:** Provides compiled (C code) versions of certain functions available in *TSEImageUtils* which are known to be computationally expensive (Cython `.pyx` source class)
- **TSEDataUtils:** Provides common data handling and statistical processing functionality for use with single and multi-dimensional data structures.

- **TSEFileIO:** Provides functions relating to data file input and output (not images - this is provided by the *OpenCV* library).
- **TSEGeometry:** Provides mathematical functions relating to the geometric transformation of image pixel coordinates (primarily scaling).
- **TSEEnum:** Represents the definition of an `enum` type for use in Python which does not come as standard in Python 2.7.
- **TSEMatchMethod:** `Enum` type used to represent the three possible matching metric categories available: distance-based (normal), distance-based (Euclidean) and histogram-based (The two types of distance-based category are as a result of the way *OpenCV* handles calculation of Euclidean Distance in a separate manner to other distance-based similarity measures).
- **TSEMatchType:** Provides a class representation for a single appearance-based similarity measure to hold specific properties regarding each metric.
- **TSEPoint:** Provides a class representation for a single 2D-coordinate point within an image used when performing geometric transformations of pixel coordinates.
- **TSEResult:** Provides a class representation for a single 'result unit' recorded within an experiment.

Due to the page restrictions in this report, the UML class diagram produced as part of the design effort for this library is available within Appendix C - Figure C.3.

2.3.6.2 Deployment

Falling in line with standard Python convention [38], the library also included an additional `setup.py` file that allowed it to be installed upon a computer via the use of common Python package managers including *Pip*[13] and *Setuptools*[14].

Each separate 'experiment-specific' Python application was able to subsequently add the installed *'TSE'* library as a package dependency in order to then access the functions provided.

2.3.7 Presentation & Analysis of Results

The ability to present results in a clear and organised fashion is naturally an aspect that has great importance within scientific experimentation.

One of the key issues facing this project was how to present the results of a potentially large number of individual experiment tests, in manner that a human could observe, manipulate and compare. While the solutions that were eventually chosen differed between the first experiment implemented in C++ and the rest implemented in Python, all had the

[13] `https://pypi.python.org/pypi/pip`
[14] `https://pypi.python.org/pypi/setuptools`

same aim of providing the user with a means of easily extracting and viewing the results that were of particular interest to their own needs.

For the first experiment it was decided that, in the interests of both time and the primary project objective, it would be best to provide support only for exporting results data to file, rather than attempting to implement plotting and visualisation facilities directly.

In order to allow for further analysis of the results obtained from the first experiment, the project made use of third-party plotting application called *Veusz* [43]. This tool supported an impressive number of useful facilities including importing (and automatic re-importing) of data files, common statistical analysis functions (including moving average and linear interpolation) and a WYSIWYG interactive graphical plot creation and editing suite.

Within Veusz, it was possible to extract specific result sets from an imported results data file, before plotting them as 2D scatter plots, configuring aspects such as the marker style, axis titles and axis ranges as needed.

For experiments implemented in Python (i.e. all experiments except the first) the project instead made extensive use of the *iPython* framework [34] discussed previously in Section 2.3.1. This library proved to be an invaluable asset to the project investigation, providing not only support for extracting and viewing result data, but an entire ecosystem tailored towards the setup, running and analysis of computational experiments from within a single web-based environment.

A major feature provided within *iPython* is the ability to *interactively* write and run Python code within isolated computational environments known as 'notebooks'[15]. By communicating with the test rig programmatically, it was possible to define all of the required test parameters within Python code, before passing these over to the test rig in order to then start the tests. Should any changes need to be made, this was accomplished within the web-based user interface, without the need for any interaction with the command line whatsoever. By creating multiple notebooks for each experiment, it was even possible to instruct iPython to automatically trigger either a selection, or indeed all of the tests for each experiment to start at the same time.

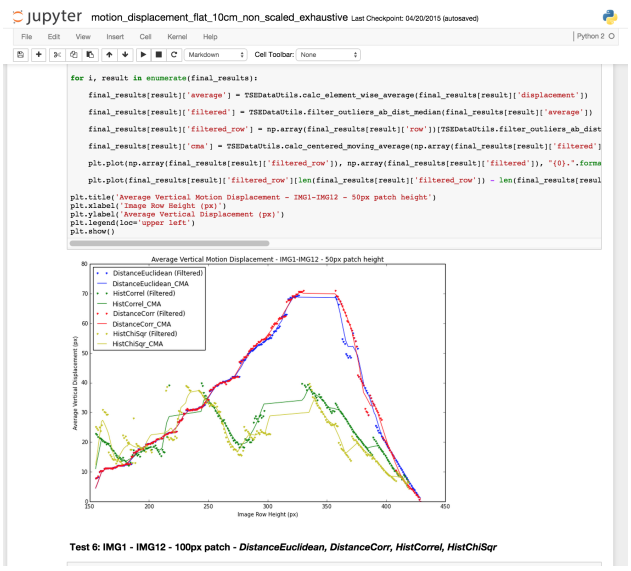

Figure 2.7: Example screenshot of the iPython web-based interface running the Python testing rig and displaying the subsequent results all from within a single 'notebook'.

Through the use of the *Matplotlib* library [21], the results for each experiment were plotted automatically upon the completion of the test rig before being embedded into the same

[15] `http://ipython.org/notebook.html`

'notebook' that contained the Python code used to define and initiate that very same test. This meant in effect, that it had become possible to analyse all of the input and output of a single test, performed as part of specific experiment, all within one "document" (Figure 2.7).

2.3.8 Software Testing

For testing aspects of the software (predominantly the testing rig and implementations of template matching methods), standard software-based testing practices were adopted. This constituted mainly of unit tests written to test for each specific function written in code, and regression tests for confirming that software continued to work as expected following the correction of a bug.

For unit tests written in Python, the *Nose*[16] third-party testing framework was selected in favour of the default testing framework (*unittest*), due to its ability to generate code coverage reports in partnership with the *Coverage.py*[17] library. This was later enhanced by integrating with the *Coveralls.io*[18] web service (Figure 2.8), which provided automatic analysis and notification of changes in the level of code coverage following each new commit to the remote project git repository hosted at Github.

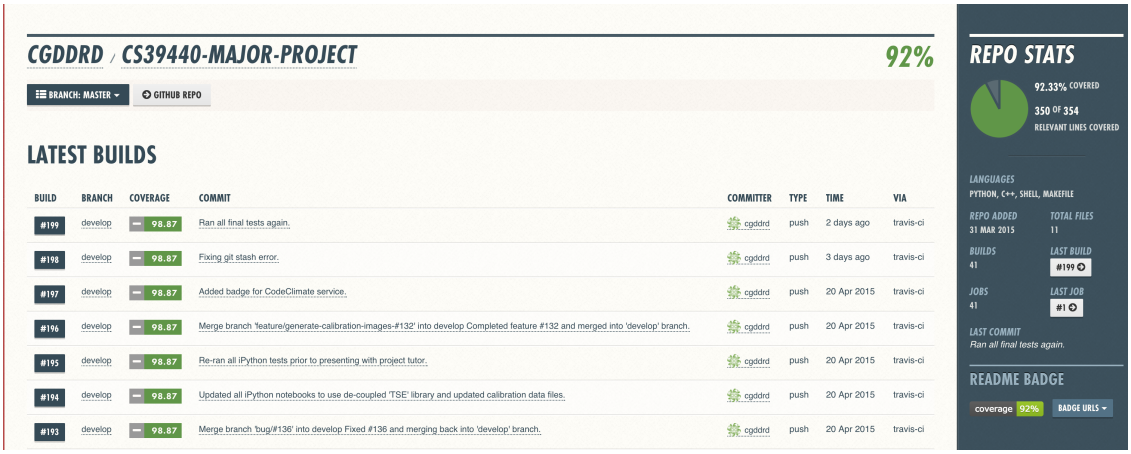

Figure 2.8: Example screenshot of the Coveralls.io[19] web interface providing a breakdown of unit test code coverage.

2.4 Experiment Methods

This section provides an overview the three main experiment methods conducted within the scope of the project investigation, that all relate to the first primary research aim discussed in Section 1.3.1.1.

While not specifically intended, as the investigation progressed the focus began to change

[16] https://nose.readthedocs.org/en/latest/
[17] http://nedbatchelder.com/code/coverage/
[18] https://coveralls.io/r/cgddrd/CS39440-major-project

from exploring a series of reasonably "generalised" research aims, to instead concentrating almost exclusively on one small, but very important aspect; *ensuring that accurate results for appearance-based motion displacement could continue to be achieved, across terrain lacking any significant features and/or demonstrating highly repetitive patterns*.

As a result, each of the three primary experiments detailed in this section, focussed on testing and evaluating subsequent evolutions of the same fundamental approach, with the hope of eventually identifying a method that could provide sufficient accuracy across all types of potential terrain.

2.4.1 Experiment 1: Template Matching (Multiple Small Patches)

This experiment focussed on the extraction and subsequent matching of multiple overlapping square patches in order to provide an appearance-based approach to performing dense optical flow, in the hope that it would be possible to generate a vertical displacement model confirming the predicted behaviour stated in Section 1.2.3.2.

The method followed a *six-stage* process (Figure 2.9)to build the vertical displacement model used later to compare vertical motion displacement for the identification of obstacles and terrain slopes:

1. Import two consecutive images from one of the test datasets, convert both images from RGB to HSV colour space, subsequently "remove" the 'V' channel by setting all the corresponding channel values to 0.
2. Extract a percentage-width region of interest (ROI) from centre of first image (discussed further in Section 1.2.3.3).
3. Extract square patches of a fixed size around the centre each pixel within the extracted ROI. These patches would act as the *templates* when performing the appearance-based template matching.
4. For each patch extracted from image one, move down through a localised search window (column) in image two searching for the best match against the template patch.
5. Identify the patch within the localised search window that provides the "best match" via appearance-based matching.
6. Average all measured vertical displacement values for each pixel along a given row, removing outliers by ignoring any displacements that lie outside of (2 x standard deviation) of the mean, before appending to the vertical displacement model.

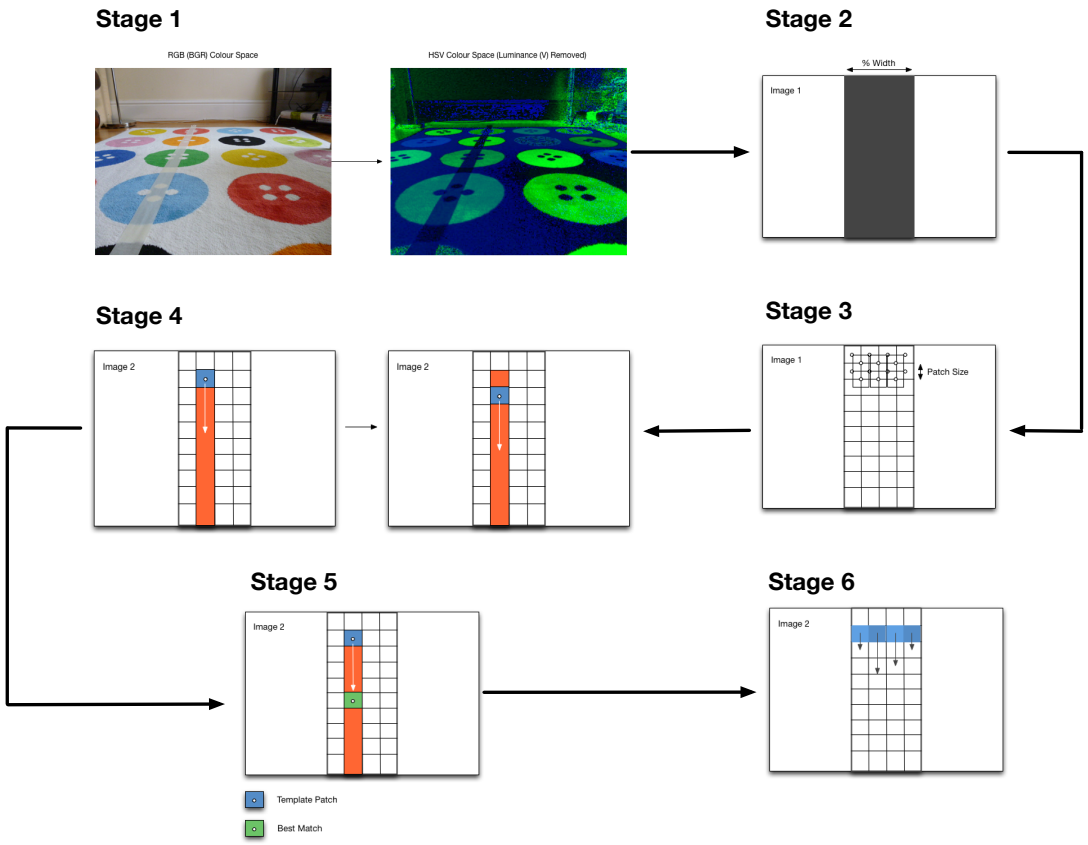

Figure 2.9: Process model for the method undertaken within Experiment 1 to evaluate the use of multiple small-sizes template patches in order to infer the vertical displacement of corresponding features between two consecutive images.

2.4.2 Experiment 2: Template Matching (Full-width Patches - Non-Scaled)

This second experiment focussed on establishing if adopting *larger patches* in fewer numbers provided better appearance-based matching accuracy than the method attempted in the first experiment of using many more, but critically *much smaller* overlapping patches

While the underlying method remained the same as in the first experiment, some crucial changes had been made:

- Extracting a central region of interest relative to the *ground plane*, as opposed to extracting a fixed-width column from the image plane.
- Moving away from multiple overlapping small patches, in favour of adopting a single, *full-width* patch to represent a single row in the image.

2.4.2.1 Exhaustive & Non-Exhaustive Searching

As part of the localised search detailed in stage five of the method from experiment one (Section 2.4.1), the current implementation saw the adoption of *non-exhaustive* searching behaviour for identifying the corresponding region within a localised search window that provided the "best match" overall.

The motivation behind non-exhaustive searching prevented the need to continue searching down the entire localised search window column, if it was believed that the best match for that search window had *already been located*.

The criteria for stopping a search, focussed around the observed scoring behaviour of appearance-based matching metrics, where in the case of Euclidean Distance, the score would demonstrate a uniform decline, until reaching a minima representing the best possible match, before subsequently beginning to rise once more (Figure 2.10). That is to say, the search should continue until such a time as the score no longer continues to become progressively "better", at which point the last score recorded *previously* to this would also represent the best possible score for that search (See Appendix B - Algorithm 2 for an pseudocode outline for this approach).

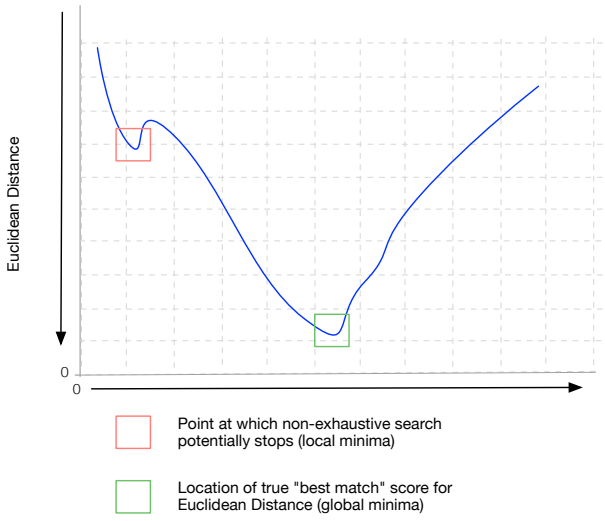

Figure 2.10: Example of potential early termination of non-exhaustive search due to entering local minima caused by noise, rather than global minima which represents to true best match.

While in theory this approach appeared reasonable, it was later identified that it had the potential to suffer greatly upon the onset of image noise. The issue that was presented,

involved the search terminating prematurely as a result of arriving at a local minima caused by noise, rather than terminating at the "global" minima that represented the *true* best match within the search window column (Figure 2.10).

In order to confirm this observation, all of the tests conducted over experiments two and three were run twice, one time using the non-exhaustive search, and the other using an exhaustive search.

2.4.2.2 Perspective Distortion Calibration Tool

A consequence of choosing to isolate a central region-of-interest relative to the ground plane rather than the image plane, described how the shape of the search region would become subject to distortion caused by changes in perspective, where the ground within the image appears to 'converge' towards a central point along the horizon line.

As template patches were now expected to span the *full width* of the region of interest, it became necessary for the width of the extracted template patches to increase as the search moved towards the bottom of the image.

In order to achieve this, the system required knowledge of what the *corresponding width* should be for every *image row* that represented the ground plane (i.e from the central horizon line to the bottom of the image).

To provide this information, a small utility tool was developed that would allow a user to export a calibration data file defining the shape of required region of interest positioned along the ground plane (Figure 2.11).

Figure 2.11: Example output from perspective calibration utility tool highlighting the identified region of interest along the ground plane for dataset one.

The Python application employed the mouse and keyboard listener features included within the *Matplotlib* library [21] to create a minimal graphical user interface that enabled the user to define either one or both of the long-side edges that would form the trapezoidal region of interest along the ground plane of the scene within the image. This was achieved through the use of a special 'calibration' image from the current image dataset containing a guide (i.e. tape along the floor) that would provide a visual indication as to the effects of the observed perspective distortion.

Following the selection of line end-points via mouse clicks, the application made use of the *Bresenham line algorithm* [13] to linearly interpolate the remaining coordinate points in order to arrive at what would become the full long-side edge of the region of interest.

For convenience, the application also included a feature whereby the user could horizon-

tally *reflect* an edge defined on one side of the image over to the other. This helped to ensure that the area in which the region of interest occupied remained *horizontally symmetrical*.

Upon the definition of both long-side edges forming the trapezium-shaped area along the ground plane, the application would finally export calibration data, by calculating the distance in pixels between the two corresponding 'X' coordinates that positioned along both of the opposite edges with respect to each row.

2.4.3 Experiment 3: Template Matching (Full-width Patches - Geometric Scaling)

The final experiment aimed to investigate if it could be possible to further increase the matching accuracy reported in experiment two, by adding the additional step of *geometrically scaling* the current template patch as the search progressed down the trapezoidal-shaped region of interest.

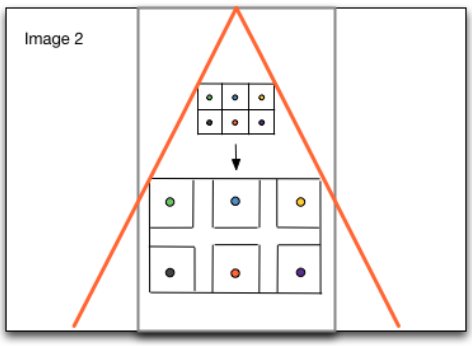

Figure 2.12: Diagram depicting process of geometrically scaling the 2D position coordinates of the pixels from the template patch, in order to subsequently match with the pixels located in the corresponding position within the previously-scaled search window.

The underlying theory adopted for this proposal related back to the idea that, if a camera was to move *closer* to an object on the ground between the capture of two consecutive images, then upon the subsequent visual inspection of both these images, it would be possible to observe that the same object appears *larger* within the second image than in the first. Thus, by geometrically *scaling* the template patch (extracted from the first image), it may then be possible to find a better match within the second image, as a result of the partial scene represented by the template patch subsequently appearing to have become visually larger within the second image due to the effects of perspective distortion (Figure 2.12).

In the context of this experiment, the specific task of geometrically-scaling and subsequently matching the template patch took form of the following stages:

1. Obtain the width and height of the current template patch;
2. Obtain the calibrated *width* of the search window with respect to the current image row;
3. Calculate the independent scale factor for the width and height between the template patch and the search window where:

$$\text{Scale Factor} = \frac{\text{Target Value}}{\text{Current Value}} \tag{8}$$

4. Treating each pixel as a 2D coordinate in geometric space, scale the position of each pixel coordinate within the template patch relative to the position of centre coordinate:

$$\vec{CP} = (P_x - C_x, P_y - C_y) \tag{9}$$

$$\vec{CP}S = \begin{bmatrix} \vec{CP}_1 \\ \vec{CP}_2 \end{bmatrix} \begin{bmatrix} S_{Width} & 0 \\ 0 & S_{Height} \end{bmatrix} = \begin{bmatrix} \vec{CP}_1 S \\ \vec{CP}_2 S \end{bmatrix} \tag{10}$$

$$P_{ScaleTarget} = (C_x + \vec{CP}S_1, C_y + \vec{CP}S_2) \tag{11}$$

   where C refers to the centre position coordinate within the template patch, P refers to the original pixel coordinate position, S_{Width} and S_{Height} refer to the scale factor for the width and height respectively, and $P_{ScaleTarget}$ refers to the final scaled position of P.

5. For each scaled pixel coordinate, extract the value of the pixel at the *corresponding position* with the search window and add to a temporary "image" data structure.

6. Perform "standard" template matching between the original template patch, and the newly created temporary image.

Due to the reliance that appearance-based similarity measures hold on the evaluation of *corresponding* pixel values between two images, a key condition of using such an approach is that both of the images to be compared demonstrate an *equal size* (i.e. there are the *same number of pixels in both images*). One of key challenges associated with the proposed method was how to still uphold this condition, while also wanting to potentially compare images of two different sizes (i.e. original template patch and larger search window).

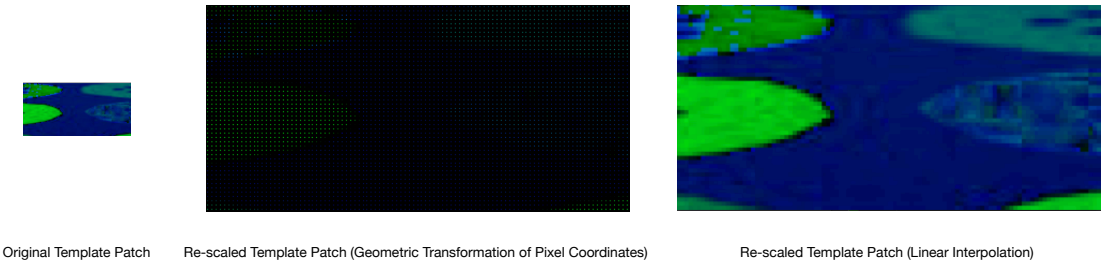

Figure 2.13: Comparison between the various approaches of performing scaling of the original template patch image. In the case linear interpolation, additional information has been added to image as a result of "estimating" the colour that lies between two original pixels.

The solution to this issue lay in the *extraction* of pixels from the larger search window, whose 2D position corresponded to the *scaled* position of each pixel in the original template patch. By subsequently comparing the original template patch, with the collection of newly-extracted corresponding pixels from the search window, it was possible to

continue to use all of the appearance-based similarity measures as had previously been utilised in experiments one and two.

Another important realisation, was that no additional information regarding the scene represented by the two images was being either *lost* or *gained* as a result of utilising pixel 2D position scaling and subsequent extraction. This however, would not have been the case had the method naively attempted to "resize" the template patch in order to have the same dimensions as the larger search window (Figure 2.13).

This is because to enlarge the entire template patch image, it would be necessary to linearly interpolate between each of the original pixels, causing new, and most likely incorrect information to be added prior to perform appearance-based matching, which subsequently may have caused the results to be very different to the actual true similarity between the current template patch and search window.

2.4.3.1 Verification of Geometric Scaling Approach

In order to test that the implemented approach to geometrically scaling template patch pixels could was working as expected, a simple test was devised making use of an artificially generated test image (Figure 2.14) to confirm that the approach could successfully match two identical objects within an image, whereby the only theoretical difference between them was a scale transformation.

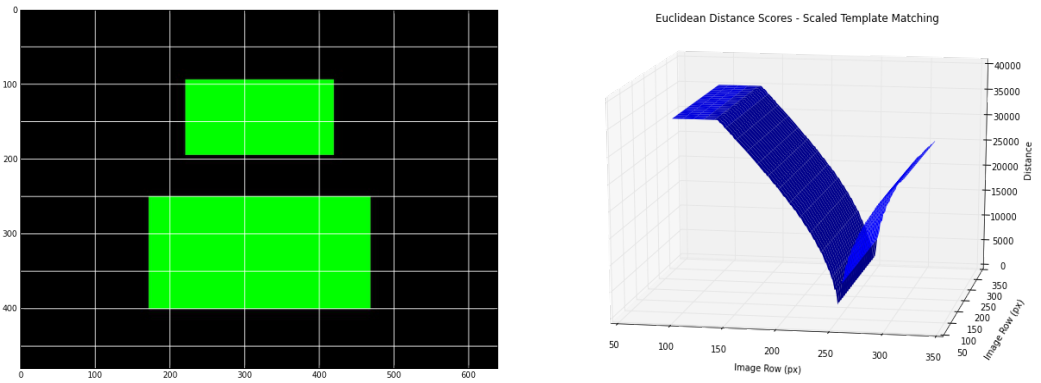

Figure 2.14: Results of the verification test for the approach towards geometric scaling of template patch pixel coordinates. On the left is the calibrated test image (grid applied only for result analysis) where the two rectangles are identical apart from a single scaling transformation. On the right is the plotted Euclidean Distance scores for the geometric search.

The results of this test (Figure 2.14) indicated that the approach was working successfully, given by the clear improvement (i.e. sharp decline) in the Euclidean Distance score as soon as the scaled template patch search arrives at the top of the scaled template patch (Row 250) whereby there is a complete alignment between the scaled version of the top green rectangle, and the bottom rectangle. As the search continues, this alignment moves and the score becomes progressively worse.

Chapter 3

Results and Conclusions

This chapter presents the key results from the three primary experiments conducted under the investigation for this project.

3.1 Investigation Results

3.1.1 Experiment 1: Template Matching (Overlapping Small Patches)

3.1.1.1 Dataset 1: Living Room Carpet (Indoors)

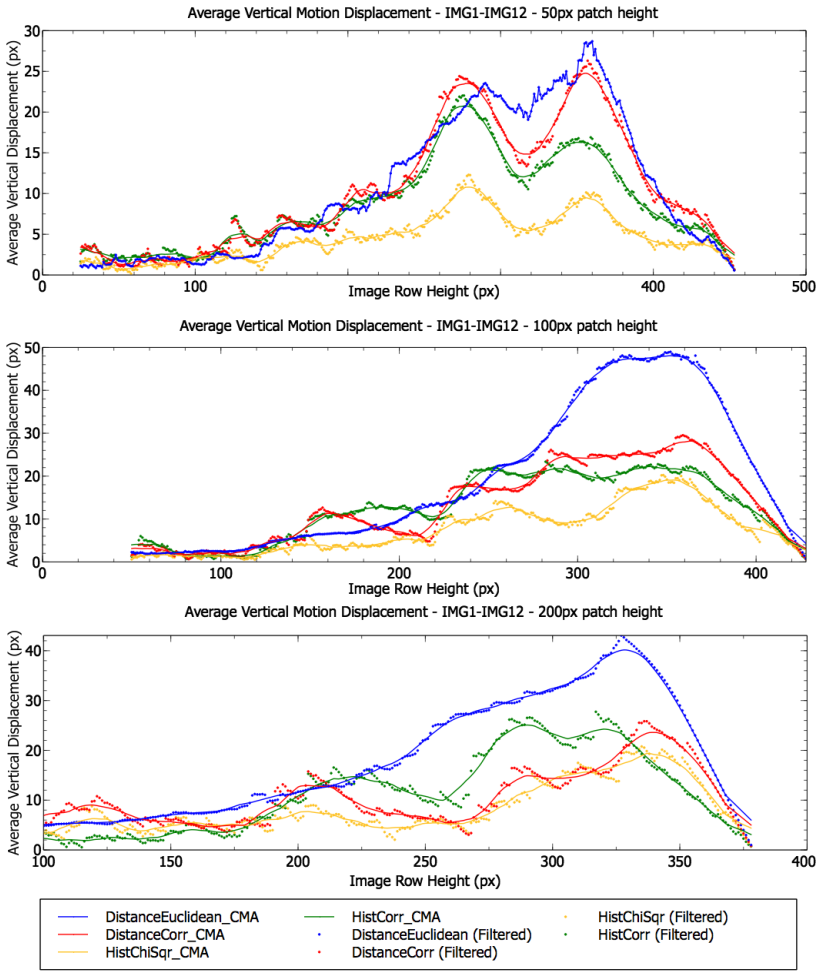

Figure 3.1: Average vertical motion displacement models across all images within "living room carpet" dataset running *non-exhaustive* localised search. Graph 1 (Top) - Overlapping patches of 50px-square; Graph 2 (Middle) Overlapping patches of 100px-square; Graph 3 (Bottom) Overlapping patches of 200px-square. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each image similarity metric.

The results within Figure 3.1 indicate an increase between the position of matched features along the vertical axis of the image, and the subsequent vertical displacement that is demonstrated between two consecutive images. While this trend is visible across all three patch sizes, the maximum displacement is recorded using the 100px patch size. Out of the four similarity metrics, Euclidean Distance appears to show the best performance across all of the patch sizes.

3.1.1.2 Dataset 2: Brick-Paved Road (Outside)

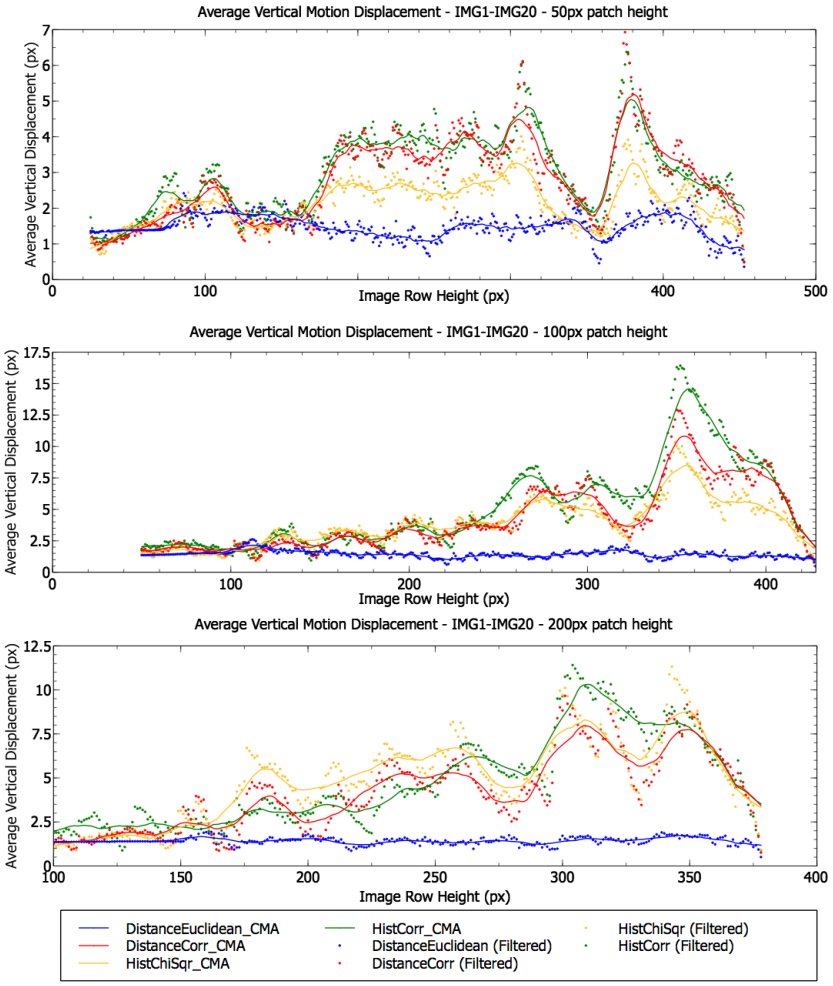

Figure 3.2: Average vertical motion displacement models across all images within "brick-paved road" dataset running *non-exhaustive* localised search. Graph 1 (Top) - Overlapping patches of 50px-square; Graph 2 (Middle) Overlapping patches of 100px-square; Graph 3 (Bottom) Overlapping patches of 200px-square. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

For dataset two, the results show while there does still appear to be an overall positive correlation shown in Figure 3.1, across all three patch sizes it is much weaker and with a significant level of distortion. While the two histogram-based similarity measures and Normalised Cross-Correlation all provided similar results, the Euclidean Distance appears to consistently fail in identifying any significant displacement. Interestingly, Graph 1 shows how all three of the better-performing similarity measures suddenly report a dip in displacement at around row 300, before returning again just before row 400. (3.2).

3.1.1.3 Dataset 3: Asian Rug (Indoors)

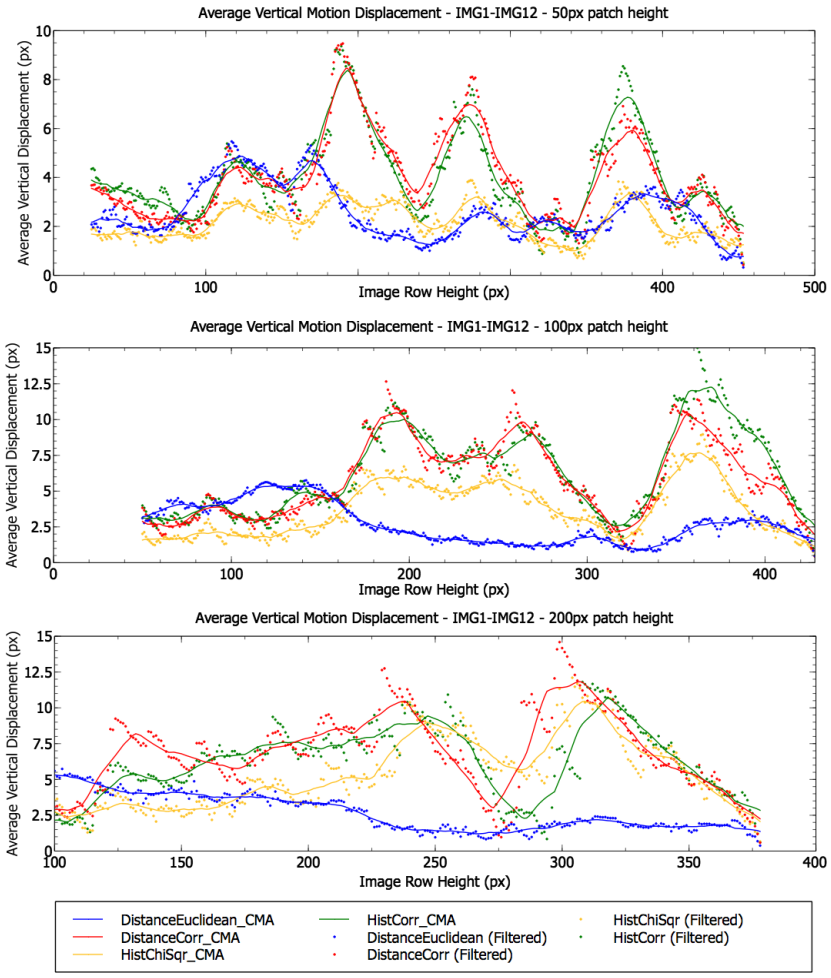

Figure 3.3: Average vertical motion displacement models across all images within "asian rug" dataset running *non-exhaustive* localised search. Graph 1 (Top) - Overlapping patches of 50px-square; Graph 2 (Middle) Overlapping patches of 100px-square; Graph 3 (Bottom) Overlapping patches of 200px-square. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

Within the results for dataset three (Figure 3.3), the vertical displacement shown Graph 1 (50px patch) demonstrates a high level of distortion with no distinct correlation with the vertical height of the row within the image. While the results within Graphs 2 and 3 also demonstrate no particular positive correlation between the row within the image, and the vertical displacement that is demonstrated, they do both appear to identify a sudden decrease in displacement at the same vertical height in the image (approx. 275px for 100px patch and 350px for 200px patch).

3.1.1.4 Dataset 4: Slate Footpath (Outdoors)

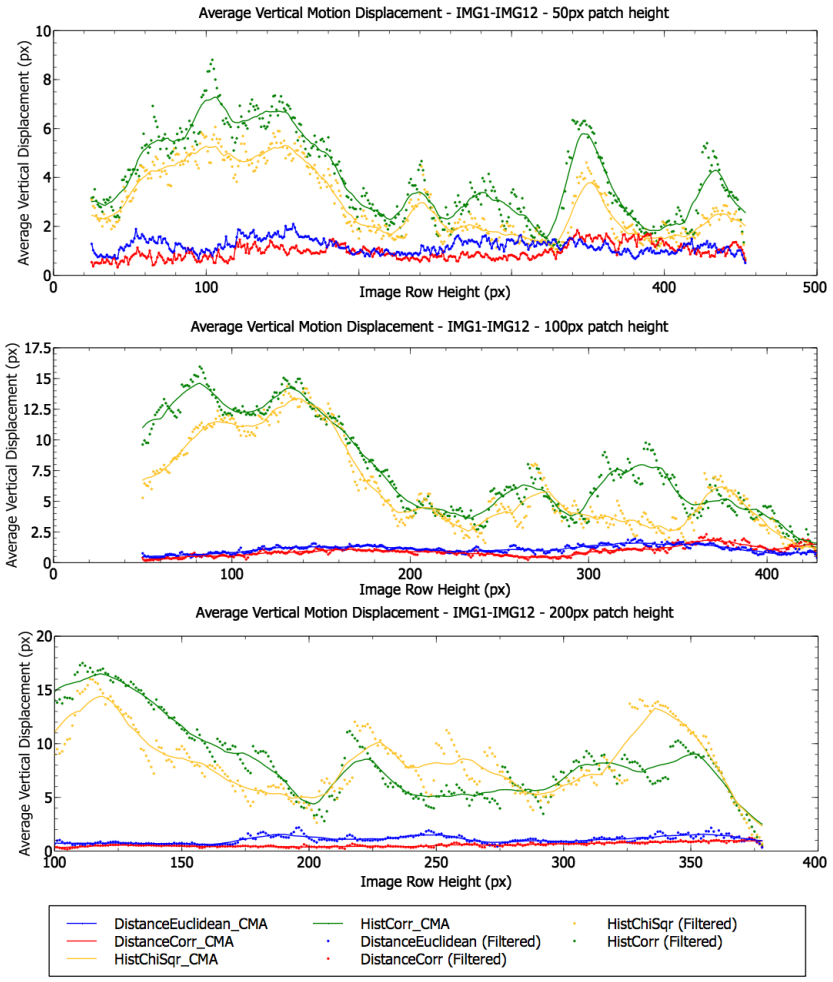

Figure 3.4: Average vertical motion displacement models across all images within "slate footpath" dataset running *non-exhaustive* localised search. Graph 1 (Top) - Overlapping patches of 50px-square; Graph 2 (Middle) Overlapping patches of 100px-square; Graph 3 (Bottom) Overlapping patches of 200px-square. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

The results within Figure 3.4, the tests across all three patch sizes appear to indicate similar results, with a clear divide visible between the performance of the histogram-based, and distance-based similarity measures. Of particular interest is the unexpected negative-correlation trend recorded by the histogram-based similarity measures that appears within all three of the graphs, potentially indicate an issue within the dataset.

3.1.2 Experiment 2: Template Matching (Full-width Patches - Non-Scaled)

3.1.2.1 Dataset 1: Living Room Carpet (Indoors)

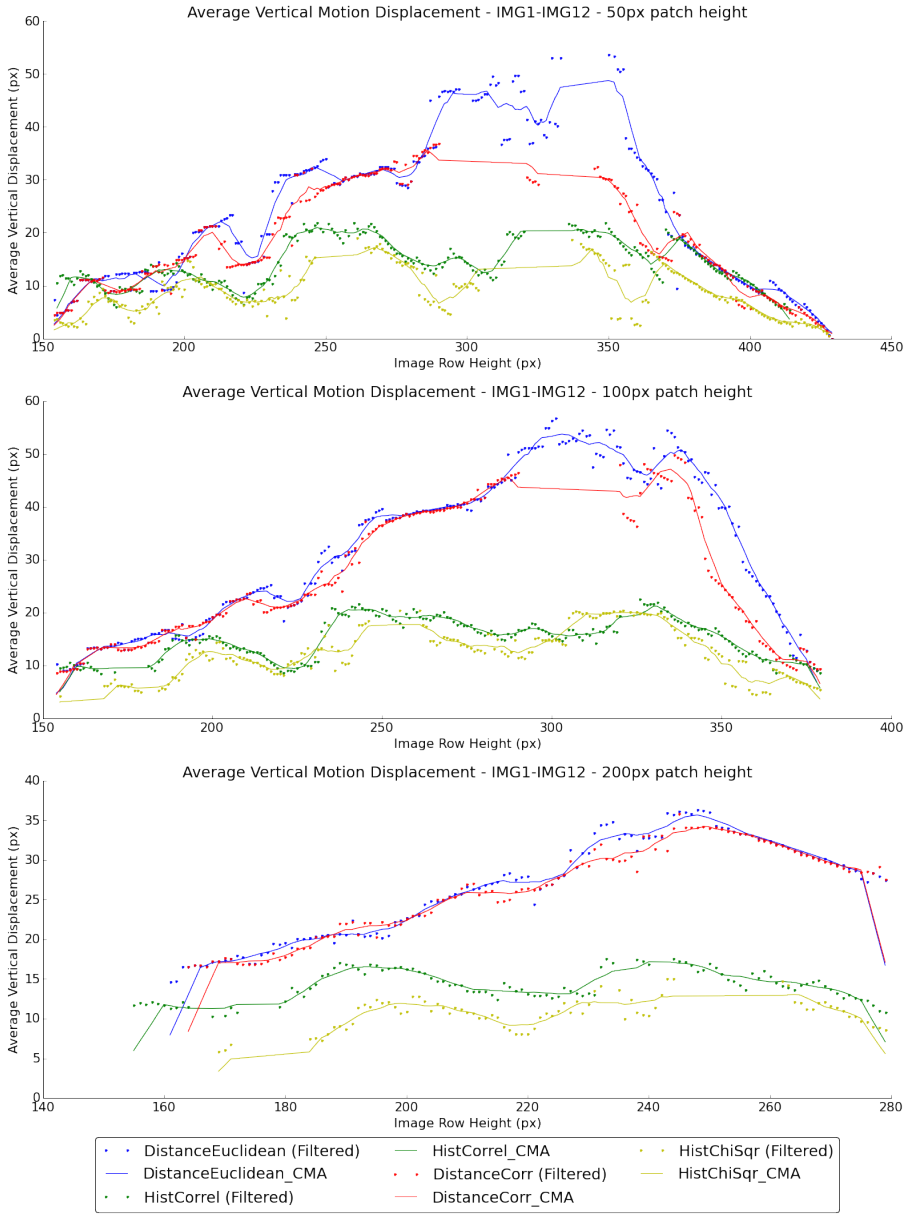

Figure 3.5: Average vertical motion displacement models across all images within "living room carpet" dataset running *non-exhaustive* localised search. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

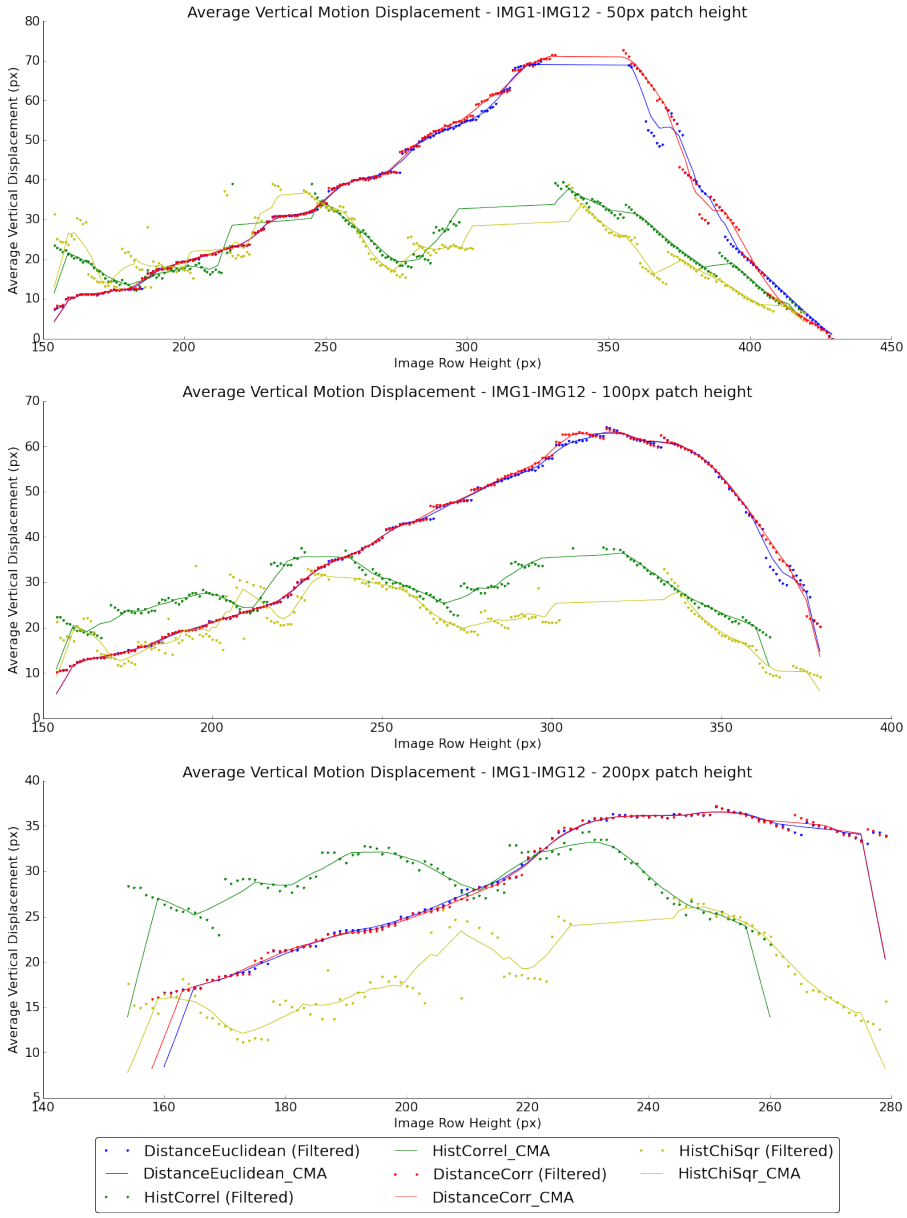

Figure 3.6: Average vertical motion displacement models across all images within "living room carpet" dataset running *exhaustive* localised search. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

Across the non-exhaustive, and exhaustive tests for dataset one (Figures 3.5 and 3.6 respectively) the two distance-based similarity measures provide clear examples of the expected positive relationship between the row within the image, and the level of vertical displacement. However, in the case of all three of the tested patch sizes, the results from the exhaustive search indicate a significantly smoother upwards trend than the non-exhaustive search.

3.1.2.2 Dataset 2: Brick-Paved Road (Outside)

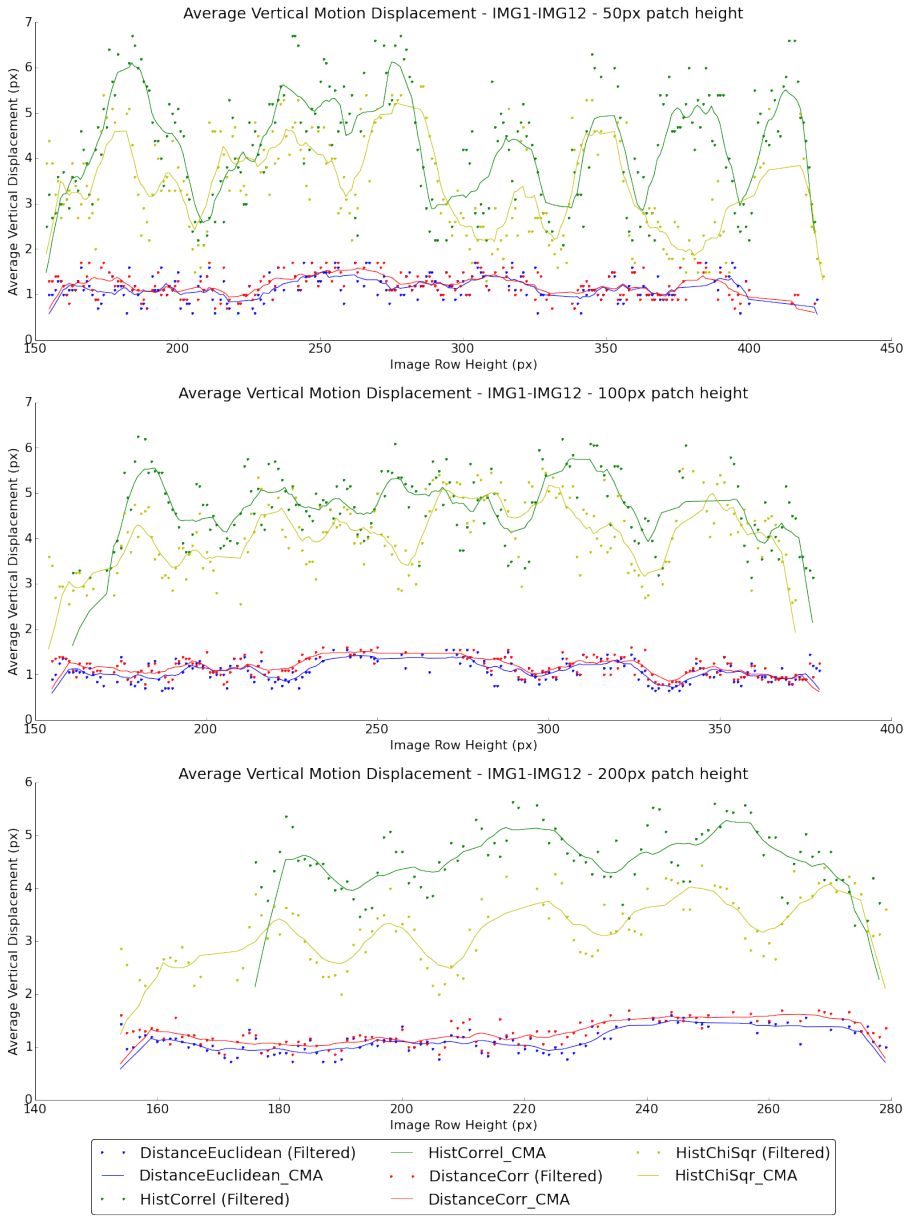

Figure 3.7: Average vertical motion displacement models across all images within "brick-paved drive" dataset running *non-exhaustive* localised search. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

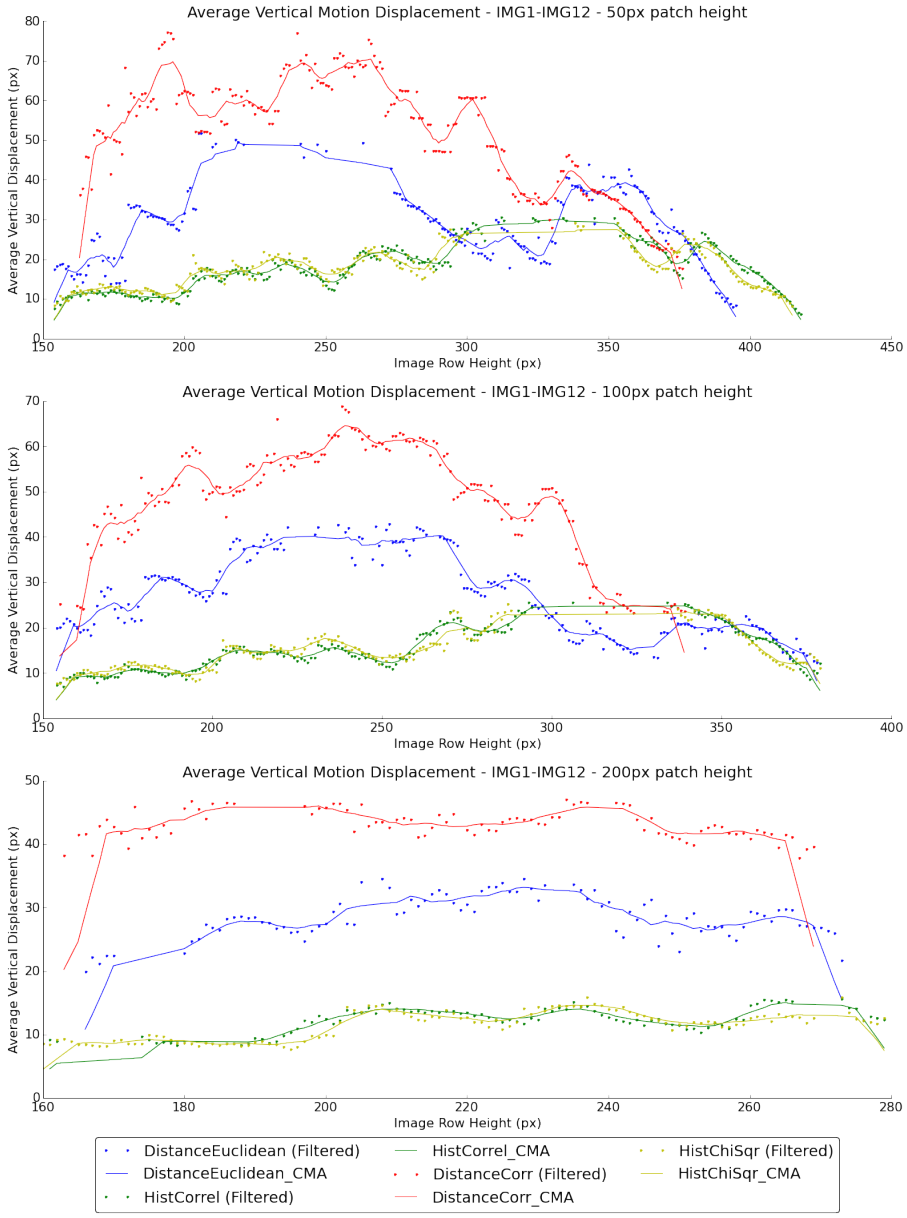

Figure 3.8: Average vertical motion displacement models across all images within "brick-paved drive" dataset running *exhaustive* localised search. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

The results recorded between the non-exhaustive, and exhaustive tests for dataset two (Figures 3.7 and 3.8 respectively) provide a stark contrast in performance between the two categories of similarity measure. Within the results of the non-exhaustive search, the two histogram-based measures indicate proportionally greater levels of displacement than the Normalised Cross-Correlation and Euclidean Distance measures. However this recorded displacement also demonstrates a considerable level of noise. In contrast, the

results for the exhaustive search show greater levels of vertical displacement overall, but in this case it is the Euclidean Distance that provides the highest displacement, with Normalised Cross Correlation coming second.

3.1.2.3 Dataset 3: Asian Rug (Indoors)

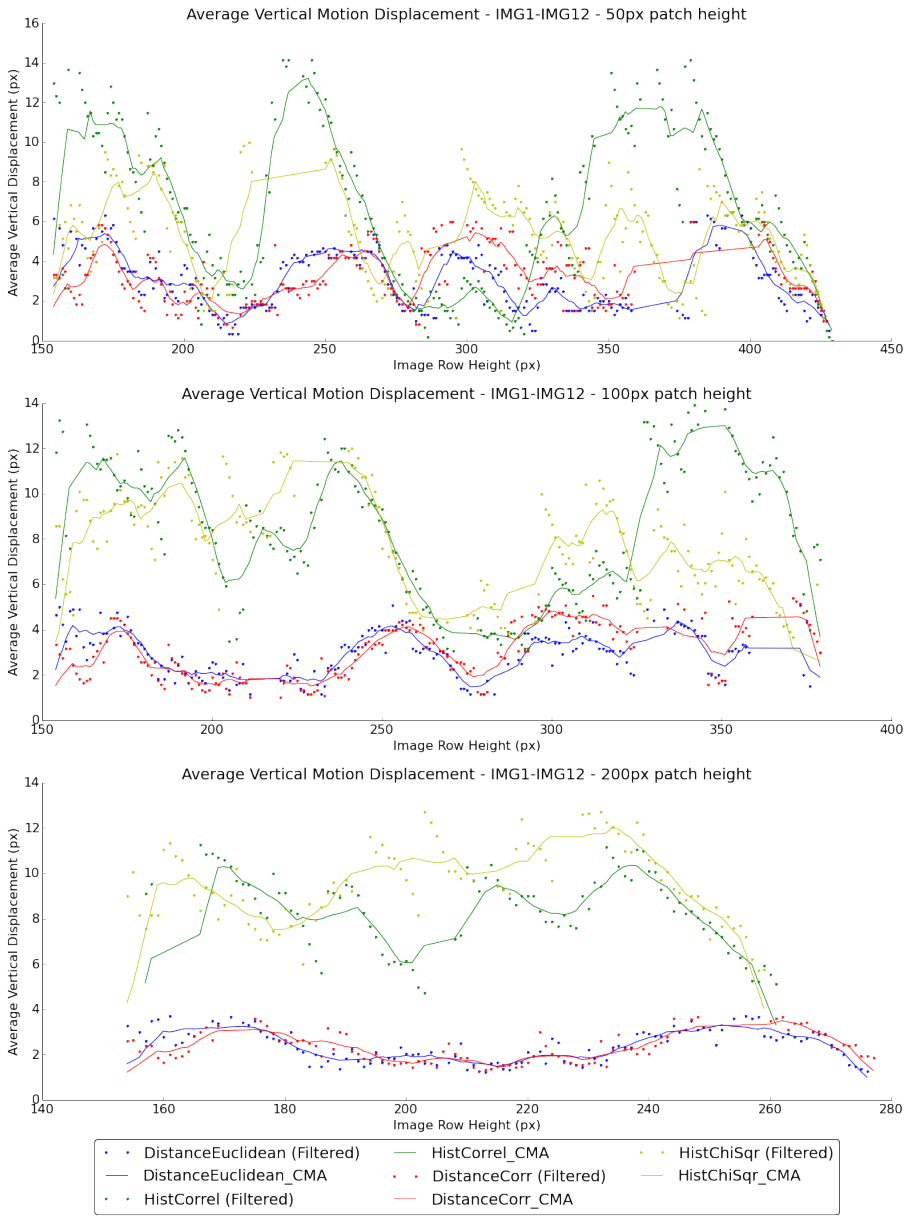

Figure 3.9: Average vertical motion displacement models across all images within "asian rug" dataset running *non-exhaustive* localised search. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

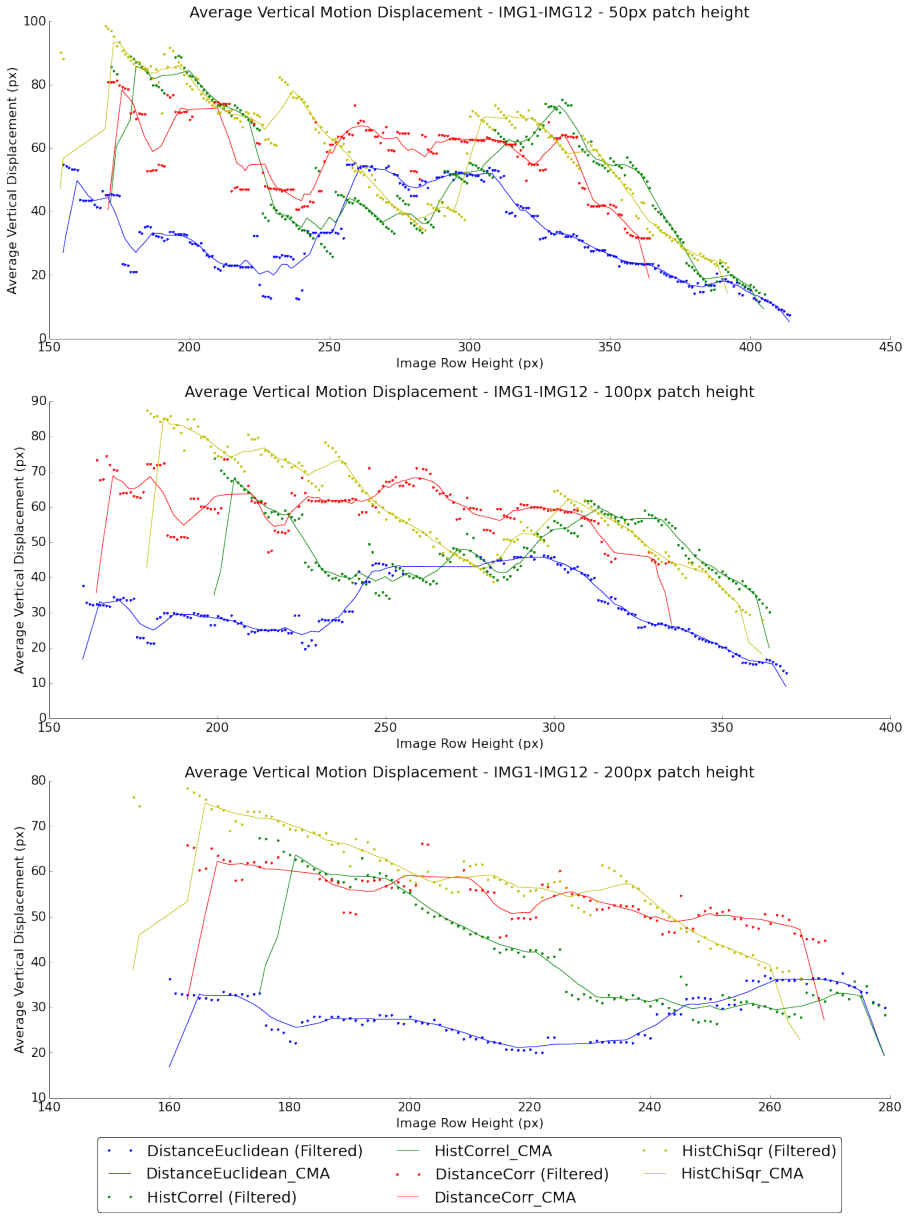

Figure 3.10: Average vertical motion displacement models across all images within "asian rug" dataset running *exhaustive* localised search. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

For dataset three, neither the exhaustive (Figure 3.9), or non-exhaustive (Figure 3.10) localised search approaches provide particularly desirable results, with all of the patch sizes and similarity measures demonstrating either just significant levels of noise with no correlation between image row and vertical displacement demonstrated, or in the case of the exhaustive search, less overall noise but a negative correlation, hence showing the opposite trend of what is expected.

3.1.2.4 Dataset 4: Slate Footpath (Outdoors)

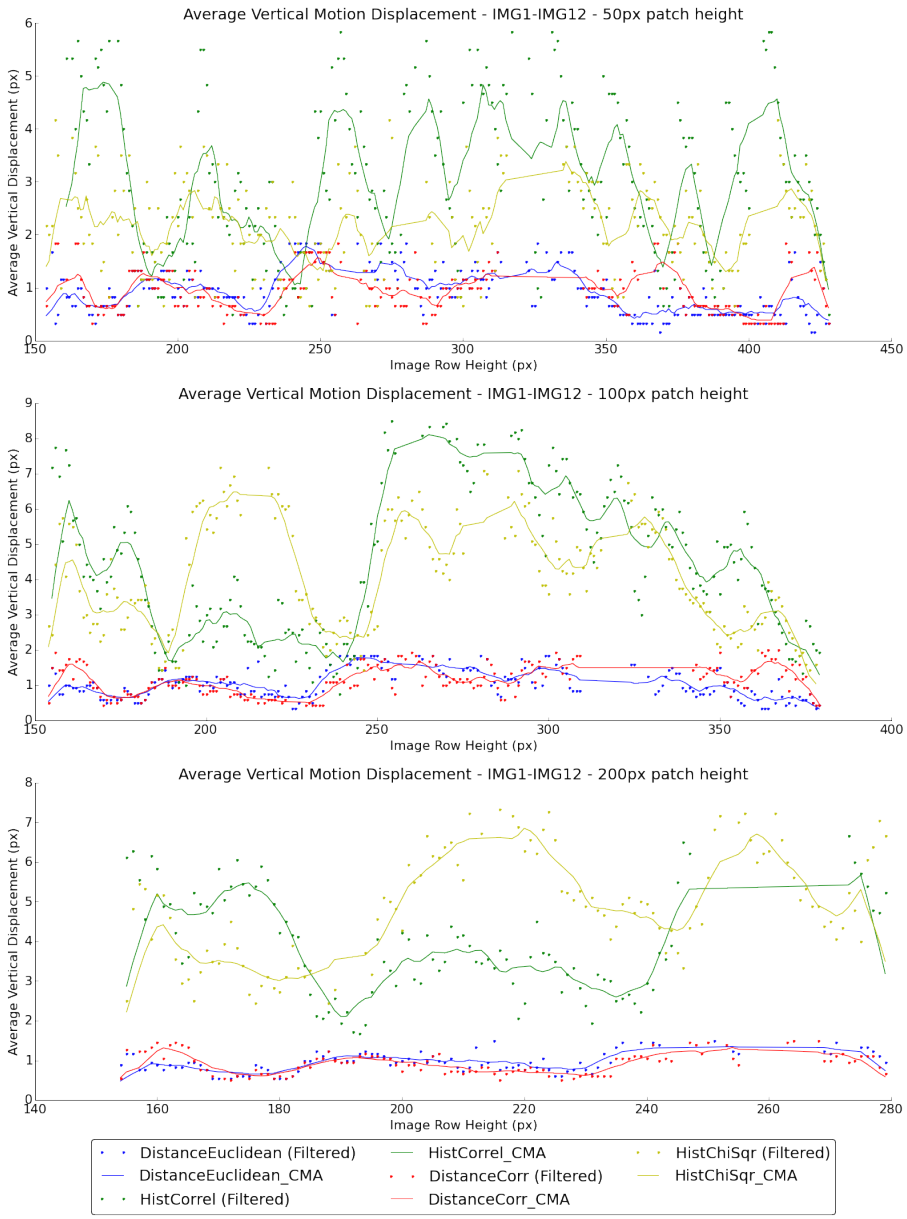

Figure 3.11: Average vertical motion displacement models across all images within "slate footpath" dataset running *non-exhaustive* localised search. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

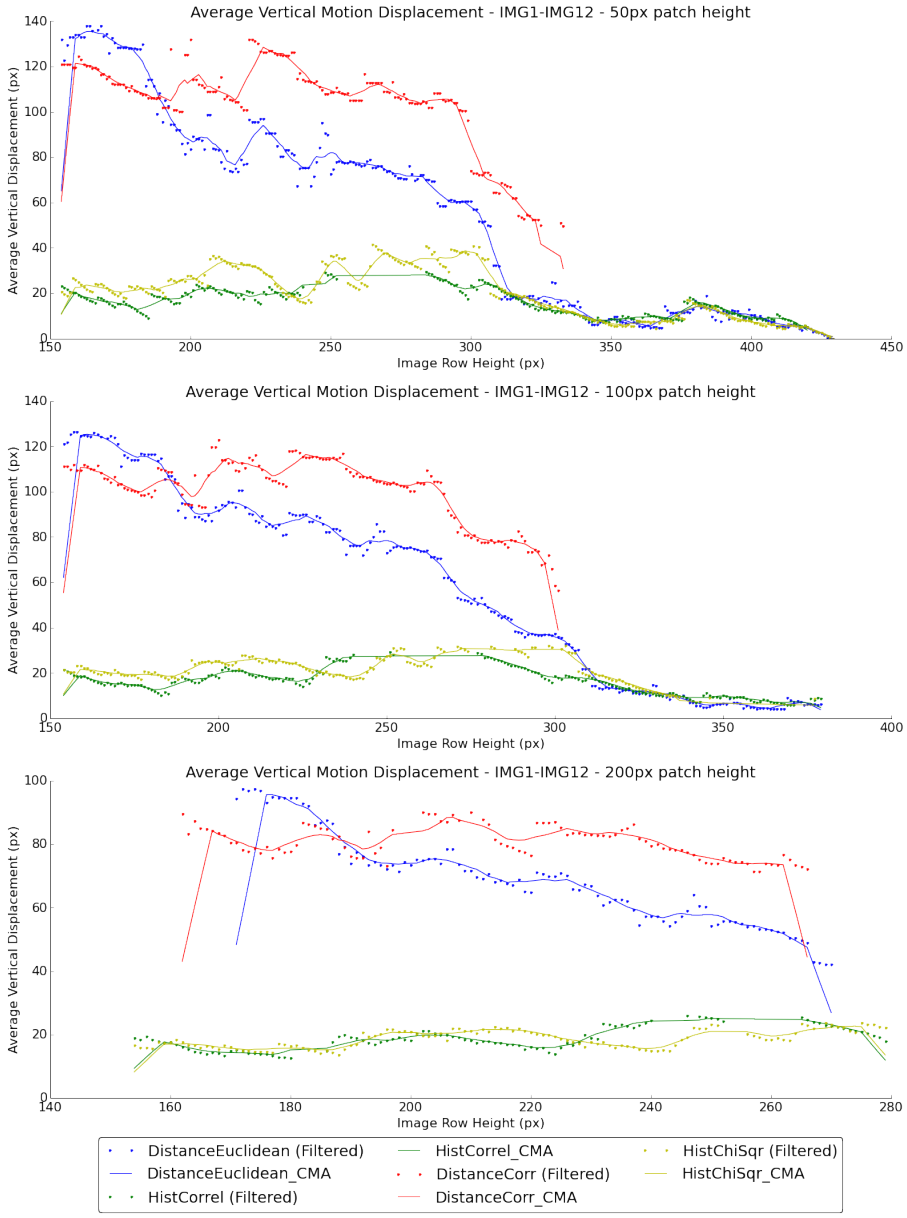

Figure 3.12: Average vertical motion displacement models across all images within "slate footpath" dataset running *exhaustive* localised search. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

The results for dataset four (Figures 3.11 and 3.12) resemble those shown for the previous dataset (Figures 3.9 and 3.10) whereby the non-exhaustive search results indicate high levels of noise and for exhaustive search results, while all patch sizes do indicate a definitive correlation between row height and the demonstrated vertical displacement, it is again a negative, rather than expected positive correlation.

3.1.3 Experiment 3: Template Matching (Full-width Patches - Scaled)

3.1.3.1 Dataset 1: Living Room Carpet (Indoors)

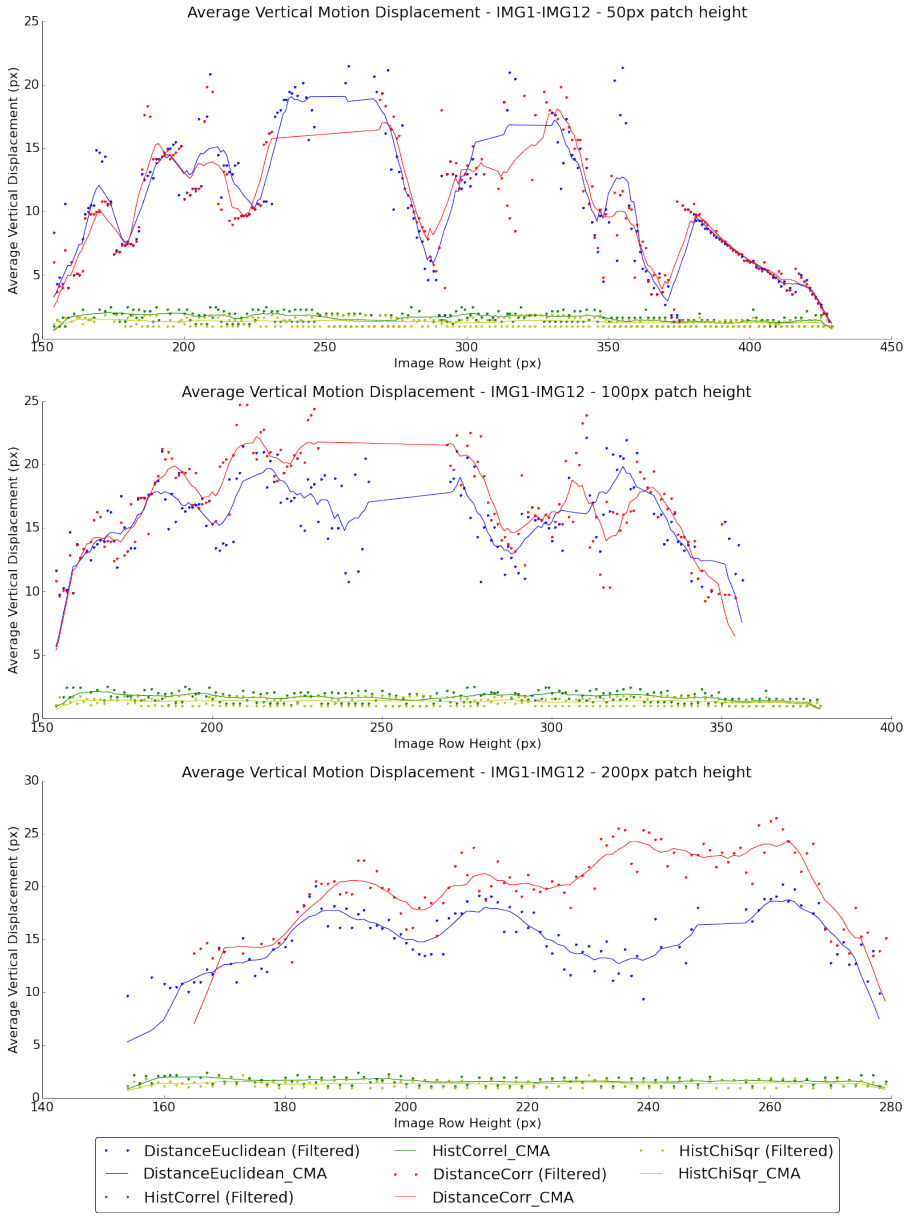

Figure 3.13: Average vertical motion displacement models across all images within "living room carpet" dataset running *non-exhaustive* localised search with geometrically scaled template patches. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

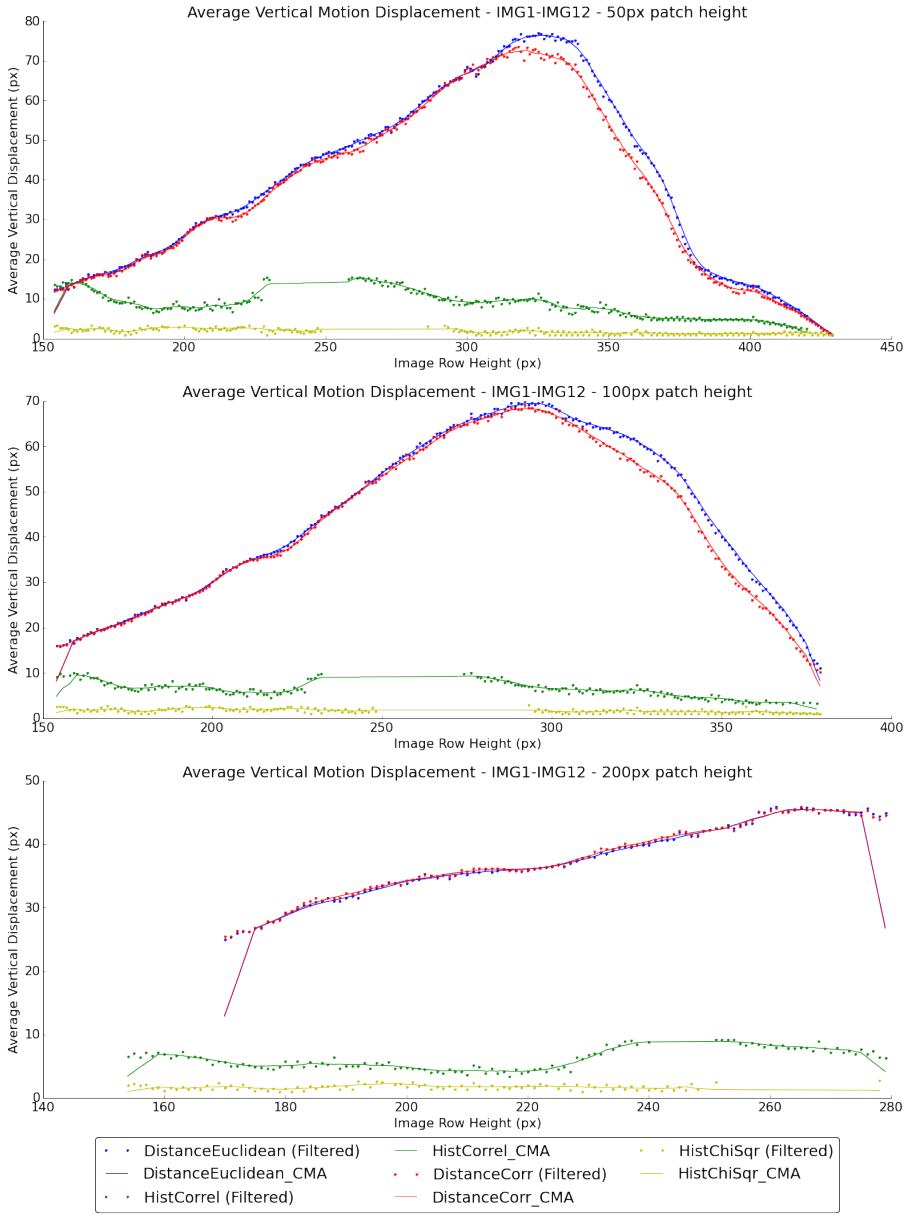

Figure 3.14: Average vertical motion displacement models across all images within "living room carpet" dataset running *exhaustive* localised search with geometrically scaled template patches. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

The results for dataset one appear to indicate a difference between the performance of the non-exhaustive search (Figure 3.13) vs the exhaustive search (Figure 3.14) when performing scaling of the template patch. Within the non-exhaustive results, while the third test using the 200px-height patch (Fig: 3.13 Graph 3) does appear to show the expected positive correlation, the tests involving the smaller patches are more susceptible to noise

(with the smallest patch showing the most distortion). In contrast to this, all three of the exhaustive searches demonstrate a strong and well defined positive correlation for both of the distance-based similarity measures. For all tests performed on dataset one,the histogram-based approaches to record any significant displacement fails.

3.1.3.2 Dataset 2: Brick-Paved Road (Outside)

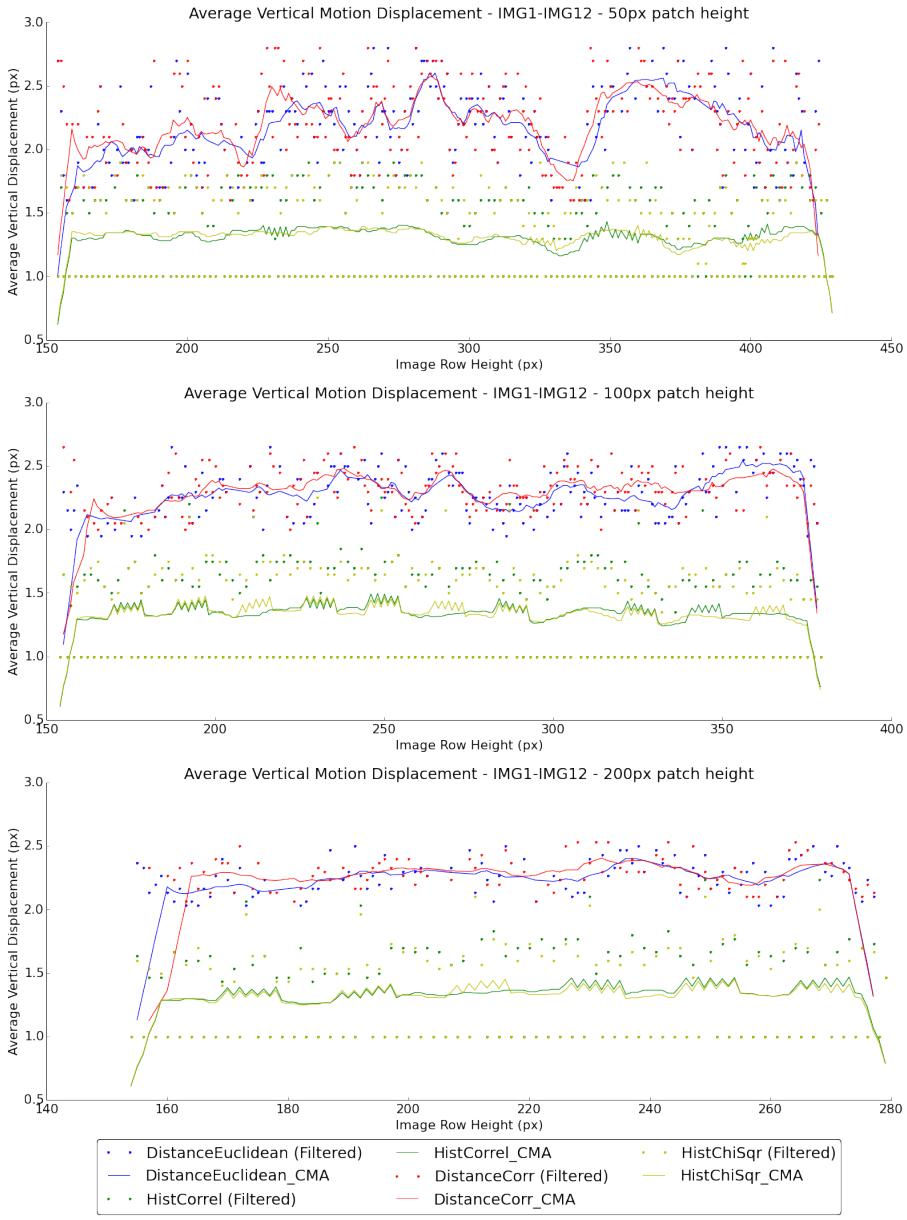

Figure 3.15: Average vertical motion displacement models across all images within "brick-paved drive" dataset running *non-exhaustive* localised search with geometrically scaled template patches. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

Figure 3.16: Average vertical motion displacement models across all images within "brick-paved drive" dataset running *exhaustive* localised search with geometrically scaled template patches. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

Under the non-exhaustive tests for dataset two (Figure 3.15), all four similarity measures appear to indicate an approximately equal level of vertical displacement regardless of the row height, with a small maximum displacement measured (approx. 2-3px). Interestingly, this similar behaviour between the four similarity metrics is demonstrated across all three patches (although the level of noise demonstrated does decrease as the

patch size increases). For the exhaustive search (Figure 3.16), a general positive correlation does begin to emerge for 50px and 100px patch heights under the distance-based similarity measures, however the results for the 200px patch size appear to show a lack of correlation, in the same way as for the corresponding test within the non-exhaustive results (Fig 3.15 Chart 3).

3.1.3.3 Dataset 3: Asian Rug (Indoors)

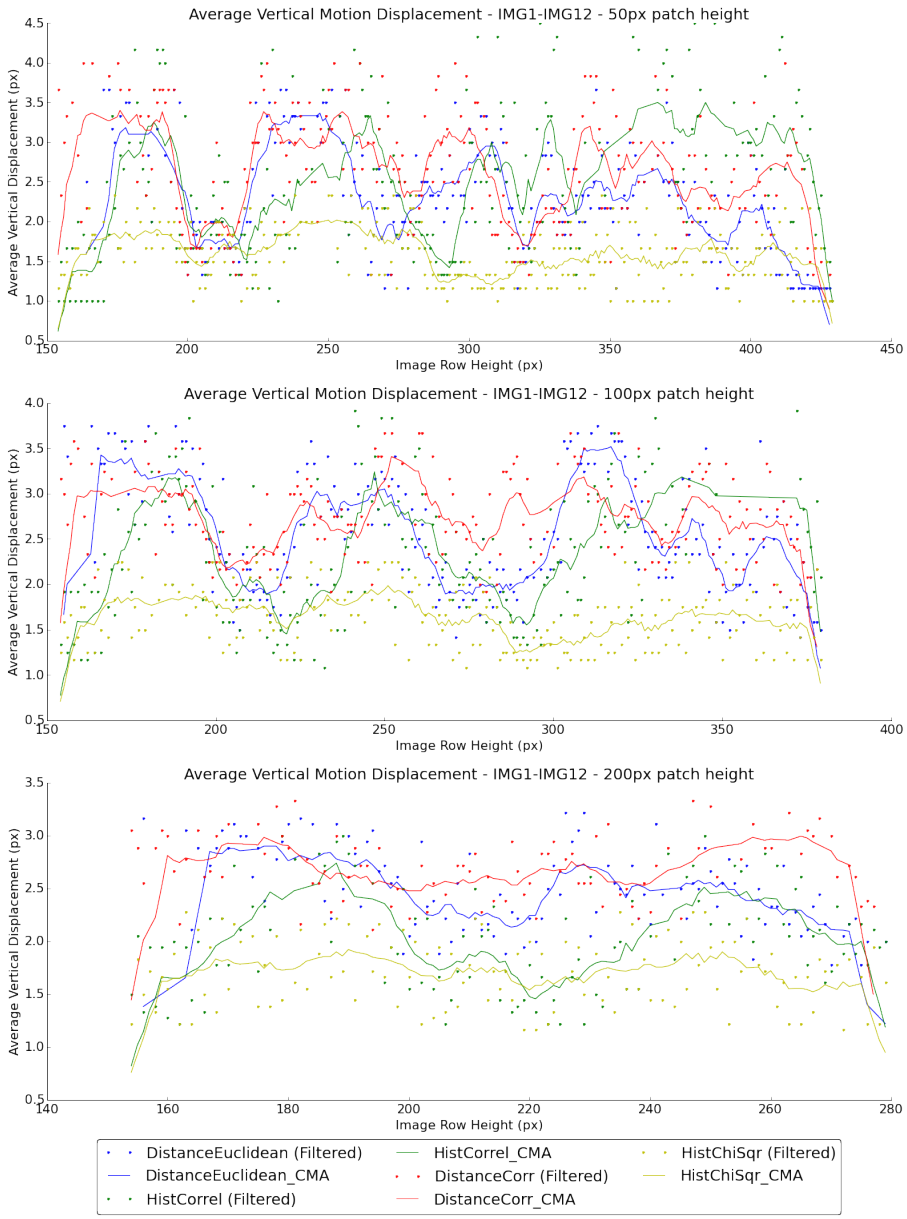

Figure 3.17: Average vertical motion displacement models across all images within "asian rug" dataset running *non-exhaustive* localised searchwith geometrically scaled template patches. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

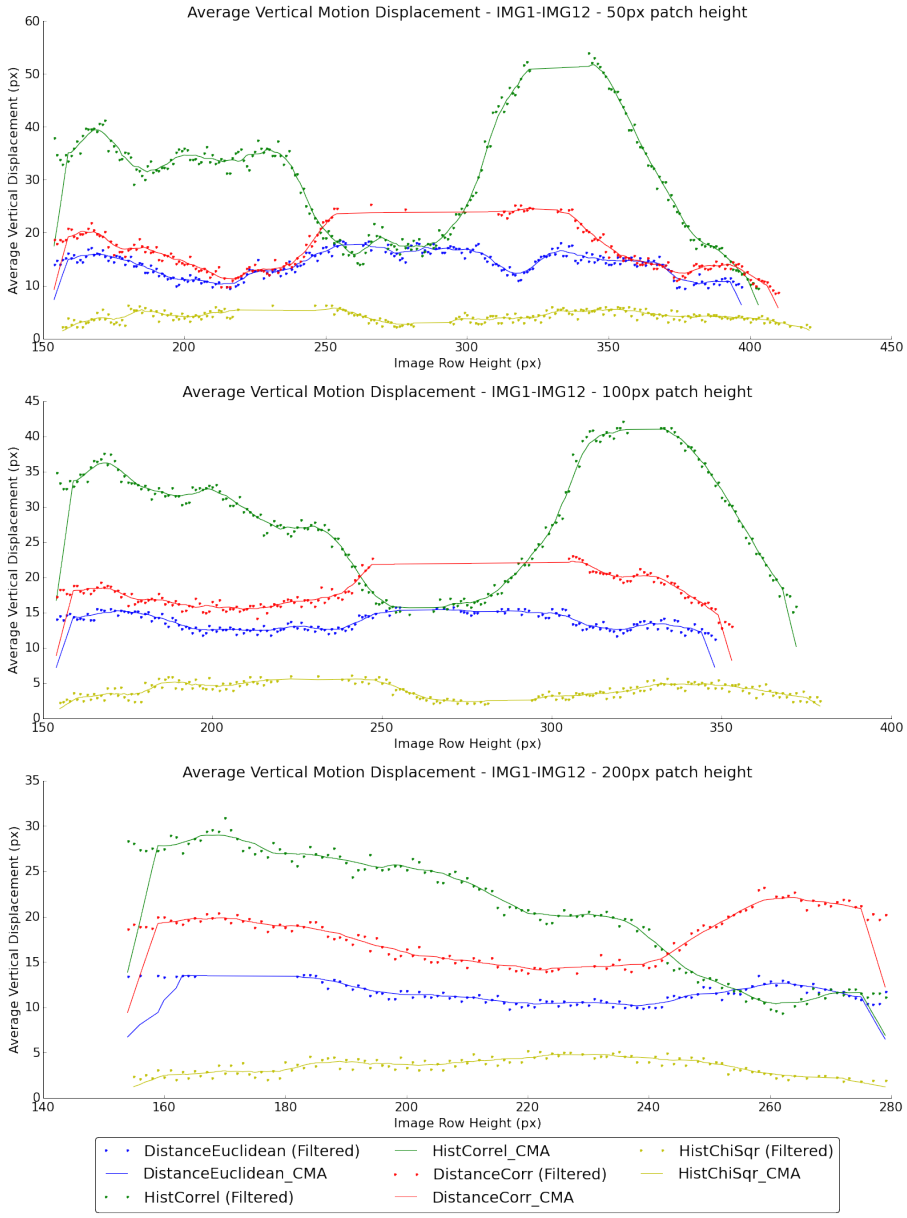

Figure 3.18: Average vertical motion displacement models across all images within "asian rug" dataset running *exhaustive* localised search with geometrically scaled template patches. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

As observed in the experiment two results for dataset three (Figure 3.9), the non-exhaustive search (Figure 3.17) appear to consist predominantly of noise, and with no correlation. Within the exhaustive search results (Figure 3.18), the different categories of similarity measure appear to at times, demonstrate opposite correlation behaviours. In the particular case of the 200px patch-height test (Graph 3), the histogram-based Correlation metric

shows an overall negative correlation, while the two distance-based metrics demonstrate a weak positive correlation.

3.1.3.4 Dataset 4: Slate Footpath (Outdoors)

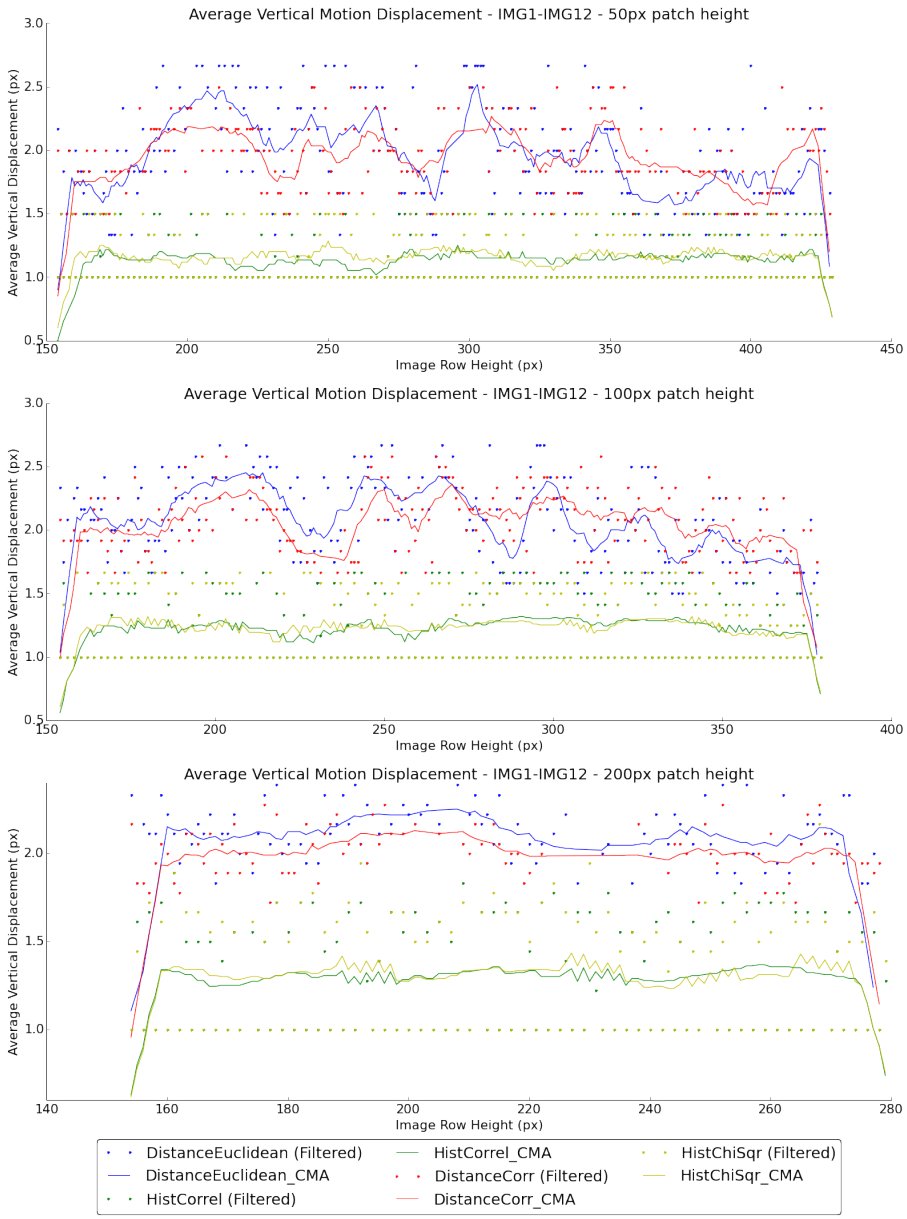

Figure 3.19: Average vertical motion displacement models across all images within "slate footpath" dataset running *non-exhaustive* localised search with geometrically scaled template patches. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

Figure 3.20: Average vertical motion displacement models across all images within "slate footpath" dataset running *exhaustive* localised search with geometrically scaled template patches. Graph 1 (Top) - Full-width patch with fixed height of 50px; Graph 2 (Middle) Full-width patch with fixed height of 100px; Graph 3 (Bottom) Full-width patch with fixed height of 200px. Solid line indicates centred moving average (10-pixel interval) calculated from filtered results for each appearance-based template matching similarity metric.

The results for dataset four show that for the non-exhaustive tests (Figure 3.19), all of the similarity measures demonstrate significant of levels of noise, with the two distance-based metrics showing a greater level of variability in the results than the two histogram-based metrics. Within the exhaustive results (Figure 3.20), a clear performance difference between the two categories of similarity measure is apparent, with the Normalised

Cross-Correlation metric showing better overall correlation behaviour than the Euclidean Distance.

3.2 Discussion of Results

Across all three experiments and within all four datasets tested, the results obtained appear to be mixed in terms of the producing the expected performance, however the relative successes and failures of each experiment can be decomposed into a number of different aspects.

3.2.1 Fixed Dimension of Patches

The first of these aspects to consider is the variety of patch sizes, these have remained consistent for each experiment and dataset.

In terms of robustness to noise, it was the 200px patch size that demonstrated the best overall resilience, in its results caused by erratic matching of incorrect corresponding patches. This is particularly visible in Figures 3.19, 3.8 and 3.10, where it is possible to observe a clear smoothing of the results between the 50px and 100px patch sizes, and those of the 200px patch size. Overall it is the 50px patch size that demonstrates the greatest level of noise in results across all experiments. This is most likely caused by the patch size being too small, resulting in the patch failing to extract a large enough region within the image from which it is possible to identify at least one unique feature that can be used to subsequently locate the same unique feature within the second image. At a patch size of 100px, the results obtained indicate a general level of noise that falls between the two extremities observed in the 50px and 200px patch sizes.

While the results for the 200px patch size may be less prone to noise, they do also indicate a lack of sensitivity to observed vertical displacement. As can be seen in Figures 3.7, 3.8, 3.15, 3.16 and 3.19, while a level of vertical displacement continues to be recorded for the 200px patch size (Graph 3), when compared to the 50px and 100px patch sizes, the level of recorded displacement appears to remain almost constant across all rows within the image. This behaviour would not be expected, and does appear to be isolated to the use of the 200px patch size. The most likely cause is the patch size extracts a template region covering too large an area of the image scene, resulting in little variability of displacement being identified on account of the images only ever showing a relatively small level of motion change between each other.

It should be noted that this behaviour only appears to begin following the introduction of full-width patches within experiments two and three, and in the results for experiment one, the 200px patch size does appear to show much similar behaviour as the other two patch sizes (Figures 3.1, 3.2). For both the 50px and 100px patch sizes, a much greater level of sensitivity is generally demonstrated, with the 50px patch showing the largest sensitivity to variations in pixel displacement. While clearly some sensitively is required in order to establish a correlation between the image row and vertical displacement, too much has been shown to cause erratic results.

Overall a patch size of 100px (fixed height of 100px for experiments two and three) appears to provide the best compromise between the reduction of noise and sensitivity to displacement.

3.2.2 Appearance-Based Similarity Matching Measures

When evaluating the four appearance-based similarity measures, the results indicate that the performance of a specific measure can vary significantly in relation to both *itself*, and the relative performance of alternative measures, across the various tests conducted within all experiments.

A common observation seen throughout all of the results, was the noticeable decline in vertical displacement recorded towards the maximum row height along the X axis. This was expected, and caused by the relationship between search focus on rows towards the bottom of the image, and the decreasing height of the available search window. As a result, the maximum potential displacement that could be recorded also decreases, with the likelihood that patches towards the very bottom of the image will never find a true match as the corresponding features will have since moved below the vertical field of view of the camera.

In an example from the experiment one results, it can be seen that while the Euclidean Distance (distance-based) measure appears to perform very well across the brightly coloured 'spot' carpet within the first dataset (Figure 3.1 Graph 2), it then suffers a clear decline in performance across the next two datasets, both of which feature *heavily textured terrain* (brick-paved road (dataset two) and asian rug (dataset three)) (Figures 3.2 and 3.4).

This observation of the poor performance of Euclidean Distance in relation to the histogram-based metrics within datasets containing a high level of texture, is indicative of the differences in approach that each metric demonstrates in determining image similarity. For the two histogram-based metrics, the similarity between two image patches is measured at a *global* level, which in this case is the overall frequency and distribution of colour within both images.

As a consequence, all of the image 'noise' that may have existed originally (*including examples of detailed textures*), subsequently becomes lost. Therefore, when comparing the similarity between two colour histograms, all of the finite detail contained within the original image is in effect ignored, thus leading to an increased chance of finding at least some kind of match, even if this subsequently this match is not accurate.

In contrast to this, distance-based measures such as the Euclidean Distance retain the finite detail when comparing the similarity between two images. This allows for a greater level of sensitivity and thus potentially better accuracy, but can also result in a failure to identify a correct match due to an acute misalignment of features within the two patches.

While Normalised Cross-Correlation is a distance-based metric, it appears to perform remarkably similar to the Histogram Correlation metric when tested under the same "heavy texture" terrain conditions as the Euclidean distance (Figures 3.2 and 3.4). The most likely cause for this is due to the normalisation step, whereby a considerable proportion of the finer detail within the original image is potentially lost as a consequence of equalising the range in pixel values can lie.

Within the final dataset for experiment one (Figure 3.4), there was a clear difference in the level of recorded displacement between the histogram-based and distance-based similarity measures. In this situation, the poor performance demonstrated by the two distance-based approaches can be attributed to a lack of distinct features available within the dataset, as a result of the terrain consisting almost exclusively of uniform slate tiles.

3.2.3 Exhaustive vs Non-Exhaustive Search

As part of an enquiry conducted following the end of experiment one into establishing potential causes for the significant reduction in performance observed between the results for dataset one, and the remaining datasets, it was predicted that a most likely partial cause was attributed to an unintended consequence of adopting a *non-exhaustive* approach to searching for matching image patches, as discussed further in Section 2.4.2.1. For experiments two and three, additional tests were conducted in order to confirm if any performance gains could be observed between adopting an *exhaustive* vs. *non-exhaustive* search approach.

Within the tests for experiments two and three, the use of the exhaustive search did appear to provide better overall results in comparison to the non-exhaustive search across all the datasets. While in a selection of cases the gains in the expected performance were significant (Figures 3.5 vs. 3.6 and 3.13 vs 3.14), in the majority of tests exhaustive searching failed to bring about any major improvement in the expected positive correlation behaviour between the image row and recorded vertical displacement. However, it was also observed in these cases that the trend of recorded displacement across the image rows demonstrated behaviour that much less "sporadic" than in the non-exhaustive results (Figures 3.11 vs. 3.12 and 3.17 vs 3.18).

An additional observation from the comparison of results between the exhaustive and non-exhaustive searches highlighted how in a large proportion of cases, the increase in performance for the *distance-based* similarity measures far outweighed the level of improvement demonstrated by the *histogram-based* measures (Figures). Given that in these cases all four similarity measures were tested under the same relative conditions (i.e. all were tested together under either a non-exhaustive or exhaustive search), the most likely explanation for these significant differences in performance gain is simply that under those specific conditions (i.e. current dataset, patch size etc.) the distance-based similarity measures genuinely were able to perform better than the histogram-based measures, but in the results for the non-exhaustive search, this observation was in fact "masked" for the reasons detailed in Section 2.4.2.1.

3.2.4 Conclusion of Results

Following analysis of all the results gathered across the three primary experiments, it is not possible to identify one specific combination of experiment method, patch size and appearance-based similarity metric, that is capable of demonstrating an optimal solution across variations in terrain type.

When exclusively considering the four variations in terrain type selected for use within this investigation, the best solution was provided through the use of geometrically-scaled template patches, evaluated within the method for experiment three.

More specifically, it was found that performing geometric scaling of full-width 100px-high template patches, in conjunction with exhaustive appearance-based template matching using the Euclidean Distance similarity metric, provided the best available example of a well-defined vertical displacement model (Figure 3.14 (Graph 2)) that accurately demonstrated the behaviour predicted in the hypothesis (see Section 1.2.3.3).

While this particular combination happens to demonstrate the best performance with respect to this specific dataset, there are other instances where this exact same combination would fail to produce such a successful outcome. An example of such an instance can be observed within Figure 3.16. This is an observation that appears to be common within a large proportion of the various possible experimental parameters evaluated as part of this study.

As a result of analysing the experiments, the overarching conclusion to be drawn is that the results obtain may be regarded as generally inconclusive. However, while certainly further research is required this investigation and it's results have highlighted that it is unlikely one particular solution will suffice. Therefore, a key focus for any further investigation to develop upon this work, would be to attempt to identify the most appropriate configuration setting for each particular terrain type, thereby hopefully ensuring that the maximum possible accuracy can be obtained, regardless of a robot's surrounding environment.

Chapter 4

Critical Evaluation

4.1 Research Aims

At the initial outset of this project, a number of research aims were proposed based upon the ideas established within the investigation hypothesis (Section 1.3.1). While originally, these relatively "high-level" tasks were deemed to be feasible within the time available, as the project progressed, so to did the oncoming realisation that the actual work required to accomplish these aims had been grossly underestimated by myself.

A crucial example of such underestimation, came as a result of not fully appreciating the difficulties that would subsequently be faced in attempting to accurately generate the vertical displacement model (Section 1.2.3.2) across terrain that, from a computer vision perspective, presented significant challenges both in terms of noise and lack of a sufficient number of "unique" features.

Following the discovery of such issues, priorities regarding the allocation of work effort had to be changed. These changes resulted in the exclusive focus on investigating potential solutions for improving the reliability and accuracy of the approach adopted for tracking the vertical displacement of features within the image using appearance-based matching techniques.

I am disappointed that further progress could not be made with regards to the other primary and secondary research aims. However, I feel that given the expectation upon the vertical displacement model for confirming the main underpinning theory within the hypothesis, the decision to focus on providing the highest possible accuracy towards generation of the model was ultimately the right choice to make.

4.2 Technical Achievement

I believe that in general, the project has demonstrated a good level of technical achievement, showcasing examples of developed software relating to both *scientific* and *engineering* aspects of the project. With respect to the development of the research proposals, I feel that a clear and methodical approach was shown towards the design of the investigation hypothesis, incorporating relevant ideas from related work and providing sufficient

justification for the decisions that were subsequently made.

In terms of later implementing the proposals detailed within the hypothesis, I feel that the software developed for each experiment accurately reflected the intended behaviour. Many of the underlying computer vision algorithms required within the project were utilised from an external library (OpenCV), under the reasoning that these implementations were known to already be optimised and throughly tested. While it would have certainly been beneficial to implement these algorithms myself in order to further enhance my own understanding, I continue to stand by the decision to use the library given the lack of available time, and my limited knowledge of the finer details for such algorithms.

The experiment testing rig is one aspect of the technical work that I believe was a particular success. While the test rig certainly made executing batches of experiments both significantly easier and faster under the command line, it was when combined with the *iPython* computational framework library that the experiment setup really began to come into it's own, providing an effectively 'seamless' ecosystem for the setup, execution and analysis of experiments. While the system was in no doubt faster than a human at conducting experiments, one major criticism would be the fact that it only held the ability to run tests sequentially. Given more time, I would have liked to have addressed this issue by implementing an approach whereby multiple experiments could be conducted simultaneously within isolation.

On reflection, I also believe that the decision to switch primary programming language from C++ to Python was necessary in enabling the project to be progress as it has up to this point. Although choosing to make the switch halfway though the project did prove to be costly at the time, I believe that in addition to learning a new programming language, the decision also resulted in development becoming more efficient and the software more maintainable, due primarily to the wide range of high-level functions already implemented within third-party libraries.

With respect to the choice of external libraries, I believe that sensible decisions were made regarding which ones would provide the most benefit to the project, and how they could be integrated as part of the system design.

4.3 Project Management

Although the project did not accomplish all its original high-level aims, the planning and management of tasks was an area that in general worked well. The adoption of SCRUM as the development methodology seemed to overall be a success, and in particular I found the planning of future sprint and release backlogs to be a very good way of remaining firmly grounded in what tasks still needed to be completed across both the shorter and longer term.

I also found that using the online Kanban board[1] made the task of reorganising work much more efficient, and as such I found myself being much more inclined to actually re-plan my work schedule rather than simply "pressing on" with development work regardless.

[1] `https://waffle.io/cgddrd/cs39440-major-project`

One small criticism would be that on occasion, it did prove to be somewhat problematic trying to map between a planned SCRUM 'release', and the somewhat "ad-hoc" development work conducted as a result of the research. The solution chosen for this, was to create a new release for each of the three primary experiments conducted.

4.4 Final Conclusion

Throughout the entirety of this project, it was a lack of available time that provided the single biggest challenge.

With hindsight, it is of course now easy to identify many potential factors that could have contributed to, what appeared to be at the time, a rapid acceleration towards the project deadline. Most likely of these is a combination between a shortfall of prior experience of working within both the fields of computer vision and indeed scientific research in general, along with the fact that the project was attempting to solve a non-trivial problem, of which there was no definitive solution.

From a personal perspective, I found the project in general to be significantly more difficult than originally anticipated. One aspect in particular was simply the fact that as a *research-based* project, there were no truely *fixed* requirements that could be used to define when a particular task should be deemed as 'complete'. As a developer who, prior to beginning this project, only really had experience working on "standard" software-engineering projects with clearly defined requirements, I found it at times to be very hard to know at which point to stop investigating one research aspect, in order to begin with another. As I became better at planning my work, this did progressively become less of an issue, and is certainly a lesson that I will take away for future.

As well representing my first attempt towards conducting a research-based project, this was also my first experience of working closely with computer vision as a technology. While many of the concepts utilised within the project were familiar to me following the completion of the Computer Vision module in semester one, I found that actually developing experiments and software using these algorithms and techniques could at times prove very challenging (in particular the mathematical concepts). As such a significant proportion of time had to be dedicated to ensure that I fully understood the details of relevant concepts, and is knowledge that I intend to utilise again in the future should the opportunity arise.

To conclude, this project has opened my eyes to aspects of software development that I had never experienced before. I was presented with the opportunity to investigate an aspect of research that was truly of interest to me, and there are great number of lessons that I have taken from as a result.

In retrospect, it would be easy to state that the original aims of the project were certainly over ambitious, if not perhaps even naive. While this maybe true, there is little doubt as to the contribution that this project has made in paving the way towards future research and development work in this area.

Appendices

Appendix A

Third-Party Code and Libraries

OpenCV [10]

Originally developed by Intel, *OpenCV* is one of the most popular open source computer vision and machine learning libraries currently available, providing over 2500 optimised functions that implement some of the most renowned computer vision algorithms including SIFT (Scale Invariant Feature Transform)[1], Viola-Jones face detection[2] and Lucas-Kanade optical flow[3].

Available as open source under the BSD license [22].

Numpy [50]

Third-party extension library providing optimised array-type objects in addition to a wide range of mathematical operations associated with numeric collections including matrix manipulation[4], statistical analysis[5] and Fourier transforms[6].

One of the core data structures provided by *Numpy* is the multidimensional array[7], which happened to be the data structure of selected to represent images within the Python bindings for the *OpenCV* library [10].

Extensively utilised throughout the project to provide statistical analysis of results and in order to work with the Python bindings within OpenCV.

Available as open source under the BSD license [2].

Cython [9]

Provided facilities to translate computationally expensive functions written originally in Python, into optimised C code. The library came with its own dialect of Python, that while being almost identical, enabled developers to add additional 'C-specific instruc-

[1] `http://docs.opencv.org/modules/nonfree/doc/feature_detection.html`
[2] `http://docs.opencv.org/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html`
[3] `http://docs.opencv.org/modules/video/doc/motion_analysis_and_object_tracking.html#calcopticalflowpyrlk`
[4] `http://docs.scipy.org/doc/numpy/reference/routines.linalg.htm`
[5] `http://docs.scipy.org/doc/numpy/reference/routines.statistics.html`
[6] `http://docs.scipy.org/doc/numpy/reference/routines.fft.html`
[7] `http://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html`

tions' (such as specifying type declarations for variables) to allow for full conversion from Python to C without the need for the developer to write any C code themselves. *Cython* source files were distinguished from "true" Python source files through the use of the *".pyx"* file extension (as opposed to the *".py"* extension for Python source files).

Available as open source under the Apache Software license [1].

Matplotlib [21]

Plotting library for Python providing high-level support for producing a wide range of scientific graphs and figures in both 2D and 3D. It was used extensively throughout this project for depicting experiment results and analysing method behaviour (e.g. plotting the behaviour of template matching similarity scores).

Available as open source under the Python Software Foundation and BSD licenses [5].

iPython [34]

Provides an interactive environment for running computational operations where a dedicated graphical user interface may not be appropriate. Provides ability to implement, run and share computational tasks and experiments within a web-based environment (known as an *iPython notebook*[8]).

Provided ability to both run experiment tests and view results within the same testing environment.

Available as open source under the Revised BSD license [3].

[8] `http://ipython.org/notebook.html`

Appendix B

Code samples

Algorithm 1 Generalised approach to performing template matching as implemented within the experiments for this investigation.

```
1:  procedure TEMPLATE_MATCHING(template_image, search_image)
2:      let high_score = −1
3:      let high_score_position = (−1, −1)          ▷ Initialise high score and position.
4:      let template_image_height = len(template_image)
5:      let template_image_width = len(template_image[0])
6:
7:                          ▷ Convolve the template image through the image we are searching.
8:      for i := 0 to (len(search_image) - 1) - template_image_height step 1 do
9:          for j := 0 to (len(search_image[0]) - 1) - template_image_width step 1 do
10:
11:             let current_window = search_image[i][j]
12:             let current_match_score = CHECK_SIMILARITY(template_image, current_window)
13:
14:             if (high_score == −1) or (current_match_score is better than high_score) then
15:                 high_score = current_match_score
16:                 high_score_position = (i, j)
17:             end if
18:
19:         end for
20:     end for
21:
22: return high_score_position
23:
24: end procedure
```

Algorithm 2 Non-Exhaustive Search of Localised Search Window

```
1:  procedure LOCALISED_SEARCH_NONEXHAUSTIVE(template_patch, search_column)
2:      let high_score = −1
3:      let vertical_displacement = 0
4:      let template_patch_height = len(template_patch)
5:
6:      ▷ Localised search window originates relative to the top of the extracted template
    patch within the image.
7:
8:      for i := 0 to (len(search_column) - 1) - template_patch_height step 1 do
9:
10:         let current_match_score = CHECK_SIMILARITY(template_patch, search_column)
11:
12:         if (high_score == −1) or (current_match_score is better than high_score) then
13:
14:             ▷ If no high score has been set (i.e. the search has just begun) or the new
    score is deemed "better" than the previous high score, then we have found the new
    best match.
15:
16:             high_score = current_match_score
17:             vertical_displacement = i
18:         else
19:             ▷ Otherwise, if the new score is worse, then stop the search at this point.
20:             return vertical_displacement
21:         end if
22:     end for
23: return vertical_displacement
24:
25: end procedure
```

Appendix C

Additional Figures

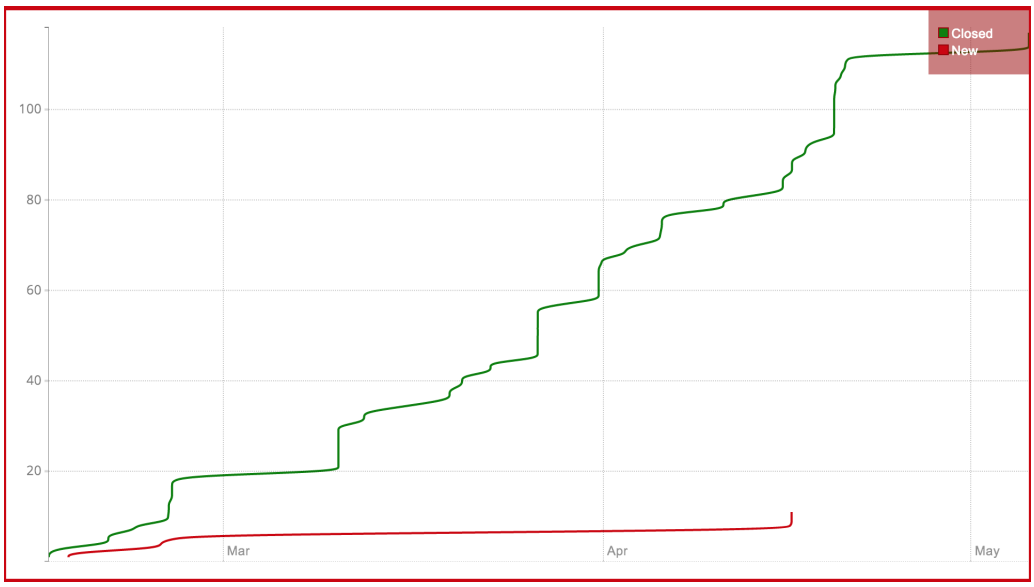

Figure C.1: Burndown chart indicating project progress over the previous 90-day period. Courtesy: `http://burndown.io/#cgddrd/cs39440-major-project`.

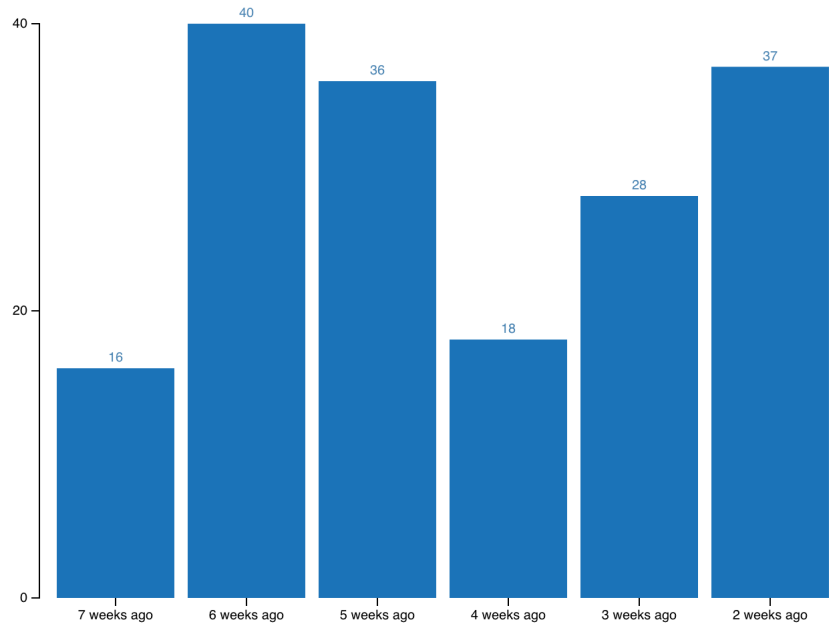

Figure C.2: Throughput chart describing rate of task completion in terms of task weighting over the previous eight weeks. Courtesy: `https://waffle.io/cgddrd/cs39440-major-project/metrics/throughput`.

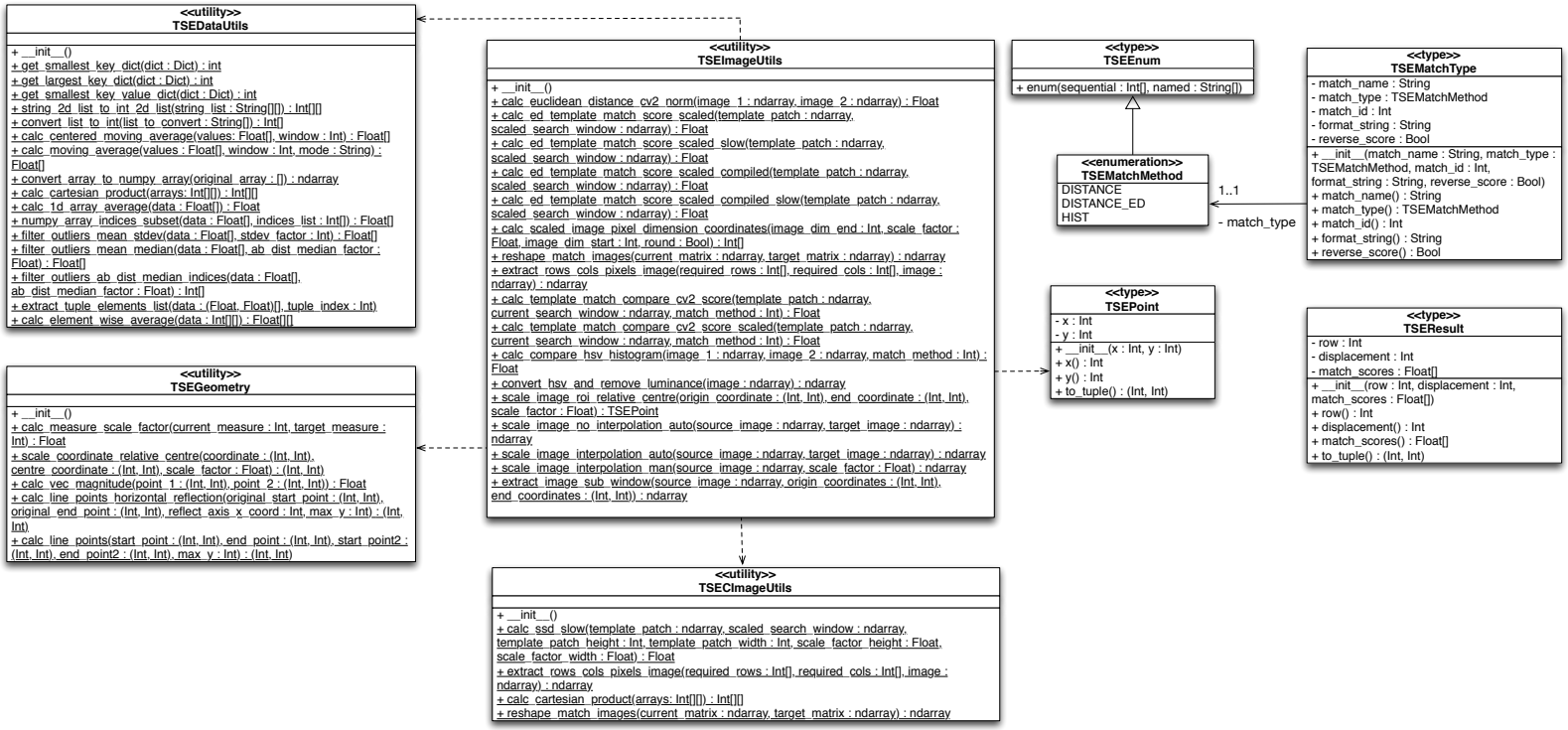

Figure C.3: UML class diagram describing the structure of the 'Terrain Shape Estimation' Python-based library utilised extensively throughout the investigation for providing common functionality shared across experiments.

Annotated Bibliography

[1] "Cython Library - Apache License," http://www.apache.org/licenses/LICENSE-2.0.html, 2004.

Apache license for Cython library.

[2] "Numpy Library - BSD License," http://docs.scipy.org/doc/numpy/license.html, 2005.

BSD License for Numpy mathematical data structures library.

[3] "iPython Library - Revised BSD License," https://ipython.org/ipython-doc/dev/about/license_and_copyright.html, 2011.

Revised BSD License for iPython computational environment library.

[4] "Template Matching - Tutorial (OpenCV)," http://docs.opencv.org/doc/tutorials/imgproc/histograms/template_matching/template_matching.html, 2011.

Tutorial from OpenCV manual used to initially learn how to implement template matching using the OpenCV library.

[5] "Matplotlib Library - Python Software Foundation/BSD License," http://matplotlib.org/users/license.html, 2013.

BSD License for Matplotlib plotting library.

[6] "Collins English Dictionary - Complete & Unabridged 10th Edition," Apr. 2015. [Online]. Available: \url{http://www.collinsdictionary.com/dictionary/english/working-hypotheses}

Provided the definition of the term: working hypothesis, which applied in the context of this investigation.

[7] T. Association of Faculties of Medicine in Canada, *Appraising scientific evidence: qualitative versus quantitative research*. Association of Faculties of Medicine in Canada, The, 2007, ch. 5.

Provided a clear breakdown of the differences between qualitative, and quantitative research.

[8] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision*, vol. 92, no. 1, pp. 1–31, 2011.

One example of an artificial optical flow dataset that includes ground truth data.

[9] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, "Cython: The Best of Both Worlds," *Computing in Science Engineering*, vol. 13, no. 2, pp. 31–39, 2011. [Online]. Available: http://dx.doi.org/10.1109/MCSE.2010.118

Third-party library used to translate computationally-expensive Python functions into more efficient C code.

[10] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

A core third-party library utilised extensively throughout the entire investigation, providing a large number of image processing and computer vision functions.

[11] ——, *Camera Calibration and 3D Reconstruction*, http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html, Feb. 2015.

OpenCV tutorial providing a background into the need to perform calibration of pinhole camera lenses.

[12] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

Reference book for working with OpenCV (C++ bindings). However, within this project this book was primarily used to as a guide to understanding the differences between the various types of histogram-based similarity metric.

[13] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965. [Online]. Available: http://dx.doi.org/10.1147/sj.41.0025

Utilised algorithm within calibration tool to interpolate points along a straight line between two selected points.

[14] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conf. on Computer Vision (ECCV)*, ser. Part IV, LNCS 7577, A, Ed. Springer-Verlag, Oct. 2012, pp. 611–625.

Another example of an artificial optical flow dataset that includes ground truth data.

[15] J. Campbell, R. Sukthankar, and I. Nourbakhsh, "Techniques for evaluating optical flow for visual odometry in extreme terrain," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 4. IEEE, 2004, pp. 3704–3711.

Provides clear information on practical approaches for evaluating the success of vision-based systems. While the paper focusses on the evaluation of optical-flow, many of the points discussed could be applied in the general case (e.g. open loop vs. closed loop testing).

[16] J. Campbell, R. Sukthankar, I. Nourbakhsh, and A. Pahwa, "A robust visual odometry and precipice detection system using consumer-grade monocular vision," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 3421–3427.

Highly informative paper describing an approach for inferring the presence of positive and negative obstacles in a scene, utilising the effects that motion parallax can cause to optical flow vectors of tracked features. This paper was certainly one of the primary sources of information regarding the properties and exploitation of optical flow, and was utilised heavily throughout the entire project.

[17] R. Cipolla, Y. Okamoto, and Y. Kuno, "Robust structure from motion using motion parallax," in *Computer Vision, 1993. Proceedings., Fourth International Conference on*, May 1993, pp. 374–382. [Online]. Available: http://dx.doi.org/10.1109/ICCV.1993.378190

Provides a clear insight into the general use of parallax observed within an image scene in the inference of the ego motion of a robot through an environment.

[18] S. Fazli, H. M. Dehnavi, and P. Moallem, "A robust negative obstacle detection method using seed-growing and dynamic programming for visually-impaired/blind persons," *Optical Review*, vol. 18, no. 6, pp. 415–422, 2011.

Describes a novel approach to improving the accuracy and speed of traditional vision-based negative obstacle avoidance algorithms using of problem-solving models (dynamic programming) with a seed-growing algorithm.

[19] M. S. Guzel and R. Bicker, *Vision based obstacle avoidance techniques*, A. Topalov, Ed. INTECH Open Access Publisher, 2011, 978-953-307-909-7.

Comprehensive survey paper providing a high-level comparison between obstacle detection using both feature-based and appearance-based approaches.

[20] V. Haltakov, C. Unger, and S. Ilic, "Framework for generation of synthetic ground truth data for driver assistance applications," in *Pattern Recognition*. Springer, 2013, pp. 323–332.

Provided a discussion on the difficulties associated with generating reliable ground truth data for optical flow.

[21] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. [Online]. Available: http://scitation.aip.org/content/aip/journal/cise/9/3/10.1109/MCSE.2007.55

Python-based scientific plotting library utilised within the project to plot all results for experiments two and three.

[22] "OpenCV Library - BSD License," http://opencv.org/license.html, Itseez, 2015.

BSD License for OpenCV computer vision library.

[23] H. Y. Kim and S. A. De Araújo, "Grayscale template-matching invariant to rotation, scale, translation, brightness and contrast," in *Advances in Image and Video Technology*. Springer, 2007, pp. 100–113.

Paper proposing highly efficiency template matching that (of particular interest in this project) can provide invariance to rotation.

[24] D. Kondermann, "Ground Truth Generation," http://resources.mpi-inf.mpg.de/conferences/up2013/up2013_files/up2013-abstracts/kondermann/daniel-kondermann.pdf, Heidelberg Collaboratory for Image Processing, 2013.

Another paper discussing the issues associated with generating ground truth data within projects requiring optical flow.

[25] F. Labrosse *et al.*, "Appearance-based heading estimation: the visual compass," Technical Report UWADCS-06-048, Computer Science Department, University of Wales, Aberystwyth, UK, Tech. Rep., 2006.

Discusses an approach to detect change in heading using appearance-based matching measures.

[26] T. Low and G. Wyeth, "Obstacle Detection using Optical Flow," School of Information Technology and Electrical Engineering, University of Queensland, St Lucia, Australia, Tech. Rep., 2011.

[27] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

Original paper proposing the well-known Scale Invariant Feature Transform (SIFT) sparse feature detection method.

[28] Y. Lu, J. Z. Zhang, Q. J. Wu, and Z.-N. Li, "A survey of motion-parallax-based 3-D reconstruction algorithms," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 34, no. 4, pp. 532–548, 2004.

Paper read as part of initial background reading into alternative projects adopting motion parallax as a means of inferring depth information.

[29] O. Mac Aodha, A. Humayun, M. Pollefeys, and G. J. Brostow, "Learning a confidence measure for optical flow," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 5, pp. 1107–1120, 2013.

A third example of an artificial optical flow dataset that includes ground truth data.

[30] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers," *Journal of Field Robotics*, vol. 24, no. 3, pp. 169–186, 2007.

Paper discussing the application of visual odometry upon the MER rovers.

[31] W. McKinney, *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.", 2012.

Textbook used during learning of how to perform computational tasks within the iPython library.

[32] J. S. McLaughlin, "Chi-Square Test."

Read as part of general background reading as to the workings and usages of the Chi-Square test.

[33] N. Nourani-Vatani, P. V. K. Borges, and J. M. Roberts, "A study of feature extraction algorithms for optical flow tracking," in *Proc. Australian Conf. Robotics and Automation*, 2012.

Read as part of background reading into the background and potential applications of sparse optical flow.

[34] F. Pérez and B. E. Granger, "IPython: a system for interactive scientific computing," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 21–29, May 2007. [Online]. Available: http://ipython.org

Third-party Python library providing a single web-based environment from which all of the setup, running and analysis of results from experiments two and three could be performed.

[35] F. Perez, B. E. Granger, and J. D. Hunter, "Python: an ecosystem for scientific computing," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 13–21, 2011.

Provided a detailed discussion on the use of Python within the scientific community for developing experiments requiring computation.

[36] N. Perveen, D. Kumar, and I. Bhardwaj, "An Overview on Template Matching Methodologies and its Applications," *IJRCCT*, vol. 2, no. 10, pp. 988–995, 2013.

Excellent paper providing a clear analysis and comparison of alternative approaches to performing appearance-based template matching.

[37] J. Pillow, "Motion Perception 2," http://homepage.psy.utexas.edu/homepage/faculty/pillow/courses/perception09/slides/Lec13A_Motion_part2.pdf, Oct. 2009.

Read as part of research into the properties regarding Focus of Expansion. Figure 1.6 also originates from this resource.

[38] *Distributing Python Modules - Writing the Setup Script*, https://docs.python.org/2/distutils/setupscript.html, Python Software Foundation, Apr. 2015.

Provided information on how to successfully deploy Python applications using the 'setup.py' script.

[39] A. Ramanujam, "Rapid Prototyping in Python," http://amjith.blogspot.co.uk/2011/09/rapid-prototyping-in-python.html, Sept. 2011.

Provided useful analysis regarding the advantages and disadvantages to using Python to develop rapid prototype applications.

[40] E. Roberts, "Elements of Computer Vision," Sept. 1998.

Provided a clear overview of the characteristics and effects of motion parallax.

[41] B. Rogers, M. Graham, and Others, "Motion parallax as an independent cue for depth perception," *Perception*, vol. 8, no. 2, pp. 125–134, 1979.

Paper discussing the background to the way in which humans can use motion parallax as a visual cue to interpreting depth within a 3D environment.

[42] A. Roongta. (2013, Oct.) Depth Cue Theory. http://akshayroongta.in/notes/depth-cue-theory/.

Provided diagram to visualise motion parallax (Figure 1.1).

[43] J. Sanders. (2015) Veusz - A Scientific Plotting Package. http://home.gna.org/veusz/.

Third-party plotting application used to examine and analyse results recorded from the first experiment.

[44] K. Schwaber and J. Sutherland, "The Scrum guide," 2001.

Official guide on adopting SCRUM as a process methodology.

[45] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pp. 593–600.

Original paper published by Lucas and Kanade in which they proposed their improvement upon the use of Eigenvalues based upon the original Harris Detector.

[46] B. Smith, *The Century Dictionary and Cyclopedia*, v. 11 ed., B. Smith, Ed. New York: Century Company, 1911, pp. 616+. [Online]. Available: \url{http://www.archive.org/stream/centurydictionar11whituoft\#page/616/mode/1up}

Provided a further description of the specific characteristics between a regular hypothesis, and a working hypothesis.

[47] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. New York, NY, USA: Springer-Verlag New York, Inc., 2010.

Excellent textbook covering all major aspects of computer vision. Specifically used within this project to gain more information about the characteristics of appearance-based similarity measures.

[48] T. Tuytelaars and K. Mikolajczyk, "K.: Local invariant feature detectors: A survey," in *FnT Comp. Graphics and Vision*, 2008, pp. 177–280. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.157.4126

Survey paper read in order to obtain an understanding into the relative performance differences between various feature detectors.

[49] I. Ulrich and I. Nourbakhsh, "Appearance-based obstacle detection with monocular color vision," in *AAAI/IAAI*, 2000, pp. 866–871.

Paper that provided a clear discussion into the methods of performing histogram-based obstacle detection. In particular, this paper provided the initial inspiration for making use of the HSV colour space in order to provide some robustness against lighting differences.

[50] S. Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011. [Online]. Available: http://scitation.aip.org/content/aip/journal/cise/13/2/10.1109/MCSE.2011.37

Crucial third-party Python library that provided ability to utilise OpenCV Python bindings and perform statistical analysis upon result data without the need to implement the functions specifically.

[51] T. A. Williamson, "A high-performance stereo vision system for obstacle detection," 1998.

Paper read during background reading into various hardware configurations available within vision-based systems performing optical flow.