

Calculation of Mountain Bike Suspension Settings through Image Analysis

Joe Barrett - 40117680

Submitted in partial fulfilment of the requirements of Edinburgh Napier University for
the Degree of BEng (Hons) Software Engineering

School of Computing

27/03/2017

Authorship Declaration

I, Joe Barrett, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others is always clearly attributed.

Where I have quoted from the work of others, the source is always given. With the exception of such quotations this dissertation is entirely my own work.

I have acknowledged all main sources of help.

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself.

I have read and understand the penalties of academic misconduct.

I also confirm that I have obtained informed consent from all people I have involved in the work in this dissertation following the school's ethical guidelines.

Signed:

Date:

Matriculation Number:

Data Protection Declaration

Under the 1998 Data Protection Act, The University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please sign your name below one of the options below to state your preference.

The University may make this dissertation, with indicative grade, available to others.

The University may make this dissertation available to others, but the grade may not be disclosed.

The University may not make this dissertation available to others.

Abstract

To do (1)

Contents

1	Introduction	1
1.1	Context	1
1.2	Background	1
1.3	Aims and Objectives	2
1.4	Restrictions	2
1.5	Thesis Structure	3
2	Literature Review	3
2.1	Mountain Bike Suspension Concepts	3
2.1.1	Travel and Stroke	3
2.1.2	Front Suspension	3
2.1.3	Rear Suspension	4
2.1.4	Sag	6
2.1.5	Damping	7
2.1.6	Optimal Setup	7
2.2	Image Analysis	8
2.2.1	Digital Camera Operation	8
2.2.2	Lighting Conditions	9
2.2.3	Usages	9
2.2.4	Image Analysis Techniques	11
2.3	Image Analysis in Sports Science	13
2.4	Using Image Analysis for Mountain Bike Suspension	14
2.5	Conclusion	14
3	Methodology	15
3.1	Introduction	15
3.2	Literature Review	15
3.2.1	Source Selection	15
3.3	Platform	16
3.3.1	OpenCV	16
3.3.2	Experimentation	16
3.4	Source Code	17
3.4.1	Pythionic Coding	17
3.4.2	Naming	18
3.4.3	Object Orientation	18
3.5	Testing	19
3.5.1	Unit Testing	19
3.5.2	Project Testing Scope	19
3.6	Project Management	20
3.6.1	Agile Development	20
3.6.2	Organic Development	22
3.6.3	Version Control	22
3.6.4	Milestones	23
3.6.5	Threshold	23
3.6.6	Gantt Chart	23
3.7	Evaluation	23

3.7.1 Validation	24
3.7.2 Reliability and Accuracy	24
3.7.3 Comparison to Alternatives	24
3.7.4 Professional Opinion	25
4 Results	26
4.1 Deviation from Plan	26
4.1.1 Development Approach	26
4.1.2 Use of EXIF Data and Reference Point	26
4.1.3 Colour Quantification	26
4.1.4 Dynamic Measurement Limits	27
4.1.5 Reference Point Measurement Method	28
4.1.6 Pressure Calculation	29
4.2 Application	30
4.2.1 Arguments	30
4.2.2 Process	31
4.3 Critical Evaluation	34
4.3.1 Validation	34
4.3.2 Reliability and Accuracy	34
4.3.3 Comparison to Alternatives	34
4.3.4 Professional Opinion	34
5 Conclusions	35
5.1 Meeting Aims	35
5.2 Comparison to other Products	35
5.3 Future Work	35
5.4 Self Appraisal	35
References	36
Acronyms	39
Glossary	39
A Android Experiments	40
A.1 Table of Android Experiments	40
A.2 Hello CV	41
A.3 15tile	42
A.4 BLOB Analysis	50
A.5 Face Detection	55
B Python Experiments	61
B.1 Table of Python Experiments	61
B.2 find_game.py	62
B.3 thresholding.py	62
B.4 img_ops.py	62
B.5 distance_to_camera.py	63
C Development Log	64

D EXIF Extraction Code	64
E Testing Results	64
F Commit Log	64
G Photos	64

List of Tables

1	Table of common suspension travels and intended disciplines	4
2	Table of pixel data showing an edge	11
3	Table of application arguments	30
4	Table of uncertainty calculation results	34

List of Figures

1	Hardtail and full suspension mountain bikes (Giant Manufacturing Co. Ltd., 2017a, 2017b)	1
2	Diagram showing travel and stroke on a full suspension bike	4
3	Leverage curves of three modern suspension designs	5
4	Maestro suspension	6
5	Diagram of camera and lens operation (Moeslund, 2012)	8
7	Uses of image analysis	10
8	Edge detection applied to an image for number plate recognition	11
9	Contact Instrument Calibration Targets on Mars Rover Curiosity (NASA JPL, 2012)	13
10	Normal image (top) versus quantified colours (bottom)	27
11	Red O-ring found using <code>findContours</code> (left) with <code>boundingBox</code> applied (right)	28
12	Black o-ring found	28
13	Reference point found using <code>HoughCircles</code>	28
14	Reference point found using <code>findContours</code> (left) with <code>boundingCircle</code> applied (right)	29
15	Sag measurements for standard air shock (left) and high volume air shock (right)	30
16	Processed images for finding reference point	31
17	Locating process for black o-ring	32
19	Example of linear equation plot	33

List of Listings

1	An example of Python code	18
2	Examples of bad naming (left) and proper naming (right) taken from Clean Code (Martin, 2009)	18

1 Introduction

1.1 Context

The suspension on a mountain bike plays a vital part in the rider's performance, comfort and overall enjoyment of the sport. With some suspension units costing upwards of £1000 it is vital that they are setup to function correctly. The objective of this thesis is to research the characteristics of mountain bike suspension, look at methods and applications of image analysis, and produce a prototype mobile application which utilises image analysis to aid the user in setting up their suspension.

1.2 Background

A survey carried out by the International Mountain Bike Association shows the average price of mountain bikes owned in Europe to be €2546 (£2206) (IMBA Europe, 2015). Starting at approximately £1000 (Giant Manufacturing Co. Ltd., 2017c), enthusiast level mountain bikes can be purchased with suspension for both the front and rear wheels, known as Full Suspension (FS) bikes whereas Hard Tail (HT) bikes have only front suspension; this difference can be seen in figure 1. Even at this comparably low cost, the suspension units have multiple adjustments available to optimize and personalize how they operate.



Figure 1: Hardtail and full suspension mountain bikes (Giant Manufacturing Co. Ltd., 2017a, 2017b)

To ensure the fork and shock function correctly they must be set up for the rider's weight and intended use of the bike. As this is considered a specialist area, many entry and mid level riders will lack the knowledge of this process or be unsure of how the suspension should operate meaning the rider could use the bike with the suspension set up incorrectly.

It has been proven that using a FS over a HT offers a performance advantage to the rider (Titlestad, Fairlie-Clarke, Davie, Whittaker, & Grant, 2003). However if the suspension fork and/or shock have not been set up, it can be detrimental to the rider's performance and potentially lead to injury. For example, if a shock has too little rebound damping set and the rider goes off a jump, the excessive speed at which the rear of the bike extends can create forwards rotation, causing the rider to go over the

handlebars of the bike.

Additionally, an incorrect suspension setup can cause excessive wear and tear on the bike's frame and components. Suspension which is set too soft can allow for bottoming out which expends excess forces into the frame and potentially cracks the frame's structure. Suspension set too hard forces energy, which it would normally soak up, into the wheels and tires causing denting and warping of the wheel rims. Further effects of suspension setup will be explained in following sections.

Many bicycle retailers will set up the suspension on a newly purchased mountain bike for the customer on delivery. Most of the time this will be enough to avoid incident but due to the extra weight of the equipment riders use, i.e. helmet, hydration pack, body armor which the customer will not be wearing at the time of delivery, this setup is regularly inaccurate. Furthermore, with some manufacturers choosing direct sales over local retailers (Harker, 2010; Staff, 2015), this setup can be circumvented altogether.

Since the birth of the modern smartphone in 2007 brought along by the first generation Apple® iPhone® and introduction of the Android™ mobile operating system, the use of mobile computing in everyday life has grown rapidly. Google™ stated that there were approximately 1.4 billion active Android users worldwide in 2015 (Callaham, 2015).

The introduction of activity tracking devices and mobile applications such as FitBit (Diaz et al., 2015) and Strava (West, 2015) and their growing popularity (Formosa, 2012) shows that individuals are welcome to the idea of using smartphones to aid or augment their participation in hobbies or sports. Due to this popularity and in a bid to give every rider the ability to setup and tune their own suspension, either at home or while out on a ride, companies started producing small devices (Aston, 2016; Hwang, 2016) and mobile applications (Benedict, 2012) which aid riders in the process.

1.3 Aims and Objectives

To do (2) The aim of this project is to create a prototype application capable of providing the user with a suggested suspension setup using image analysis techniques. This will be achieved by meeting the following objectives:

- Complete a literature review of mountain bike suspension and image analysis techniques including uses of image analysis in sports science.
- Design the prototype application
- Implement the prototype application
- Evaluate the success and appropriateness of the produced application

1.4 Restrictions

To do (3)

1.5 Thesis Structure

To do (4)

Introduction Outlines the context of the project and states the major aims and objectives

Literature Review

Methodology

Critical Evaluation

Conclusion

2 Literature Review

The aim of this project is to produce an application which will provide the user with a suggested suspension setup using image analysis techniques. This literature review will identify the various settings relating to mountain bike suspension as well as the intricacies surrounding rear suspension and why it can be difficult to setup for novice riders. It will then go on to highlight how image analysis is carried out and the various techniques which can be used in the context of this project. This review will hopefully shed light on the problem at hand and how the identified application is a viable solution.

2.1 Mountain Bike Suspension Concepts

The purpose of suspension on a mountain bike is to divert energy from bumps and rough features in a trail away from the rider to improve comfort and performance by maintaining contact between the tires and the ground. This requires the use of a spring and damper, collectively known as a shock absorber, which allows the wheel to move away from the feature when it makes contact and make a controlled return once it has been passed.

2.1.1 Travel and Stroke

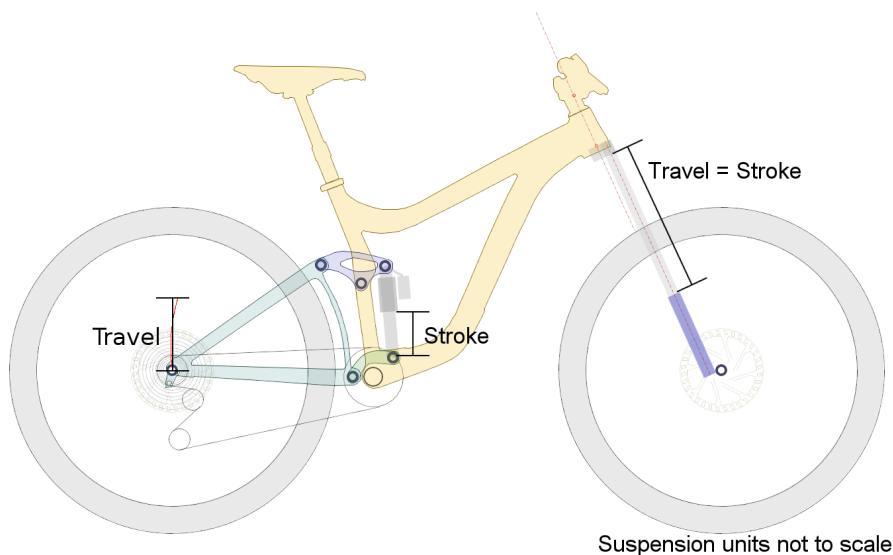
Travel is the distance which the bike's fork or frame allow the wheel to move in and upward direction. Stroke is the distance that the shock absorber can compress before it bottoms out. Travel is measured in inches or millimeters and can range from 80mm to 210mm. The amount of travel which a bike has normally denotes which discipline it was intended for, outlined in Table 1. Typically less suspension is required for endurance oriented riding and more for aggressive or rougher situations.

2.1.2 Front Suspension

Front suspension commonly employs a linear telescoping shock absorber, known as a fork due to its dual sided construction. On nearly all suspension forks the stroke is 1:1 with the potential travel of the wheel. Front suspension is found on all FS and HT bikes.

Table 1: Table of common suspension travels and intended disciplines

Travel (mm)	Cross Country	Trail	Enduro	Downhill
80				
100				
120				
140				
160				
180				
+200				

Figure 2: Diagram showing travel and stroke on a full suspension bike¹

2.1.3 Rear Suspension

Rear suspension uses a shock absorber much smaller than a fork and does not operate on a 1:1 ratio. Bike frames incorporate one or more pivot points and linkages which allow the wheel to move and act as multipliers for the suspension. Rear ratios are expressed as n:1 where n is the distance the rear wheel moves for every 1mm the shock compresses through its stroke. Though this is only the average leverage ratio for the entire travel. The difference between front and rear stroke and travel can be seen in figure 2 noting the separation of rear wheel travel from the stroke of the shock absorber.

To do⁽⁵⁾ As all rear suspension designs are different and the rear wheel must rotate around the main pivot, or in some cases virtual pivot, as opposed to moving linearly and the frame will behave differently through its travel and depending on the type of shock it is using. Because of this the average ratio is normally dismissed in favour of a leverage curve.

Figure 3 shows the leverage curves of three modern suspension designs. Each of these designs has between 150mm and 170mm of travel and uses the 27.5 inch wheel size; however it can be seen that their suspension hosts drastically different characteristics.

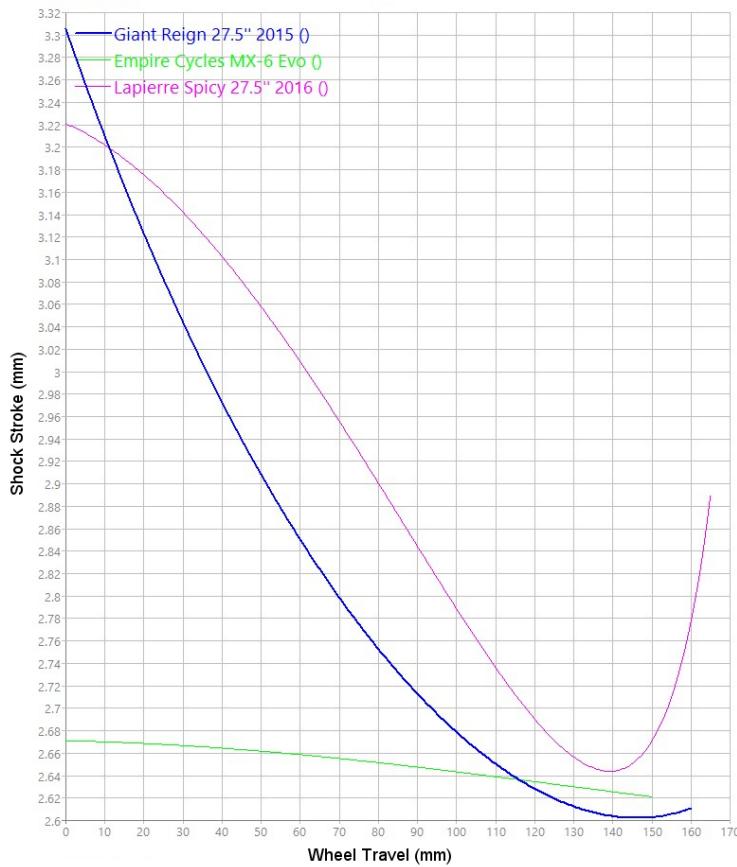


Figure 3: Leverage curves of three modern suspension designs

The Virtual Pivot Point (VPP) design of the Giant Reign (shown blue) has an initial falling rate, meaning the shock can be compressed easily, but slows down and even rises slightly towards the end of its travel; this frame is indicated in figure 2. This means the suspension will feel soft most of the time but feel stiffer on large compressions. This is emphasised by the horst link system of the Lapierre Spicy (shown magenta) which has a large rise at the end of its travel.

In contrast, the curve of the single pivot Empire MX-6 Evo (shown green) is considered linear. This is due to the MX-6 having only one pivot and swinging arm, as opposed to multiple pivots and linkages of the VPP and horst link designs, so there is an almost direct input from the rear wheel to the shock.

For this project the bike used for development and testing of the application will be a 2015 Giant Reign, shown on figure 3 in blue, as there will be constant access to it during the project. The frame uses Giant's Maestro™ suspension system which is a variation of VPP. Like all VPP systems Maestro uses two links, an upper and lower, to create a virtual main pivot point, however unlike other VPP systems, indicated by the red circle in Figure 4, Maestro creates its virtual pivot as close to the rear of the frame as possible.

Despite the complexities relating to rear suspension designs, the average rider and

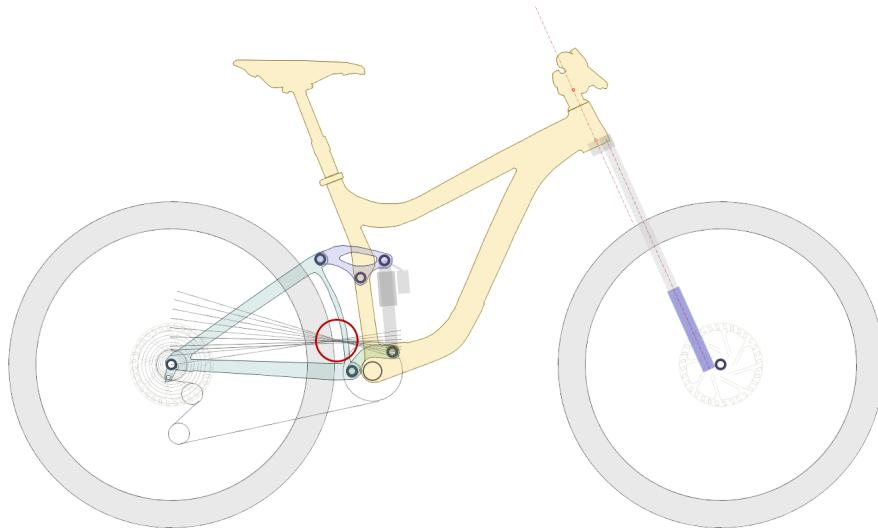


Figure 4: Maestro suspension

even individuals in the industry are unlikely to need this knowledge. Leverage curves are predominantly used by designers in the research and development of new frames to determine the characteristics of how they will use the suspension available before the new bike is released to the market^{To do (6)}. Though an understanding of these characteristics will allow the individual rider to tune the settings described in the following sections themselves, the majority will not notice the difference between a single pivot and VPP design, opting for a more simple "set and forget" approach to their suspension.

In the context of this project and the intended user for the application, this begs the question of how much information should be given to the user? Modern human computer interaction principles aim toward providing information to the user which is relevant and necessary in context^{To do (7)}. Presenting the user with a leverage curve which could require extensive explanation to understand within a mobile application could prove detrimental to the user experience^{To do (8)}. For an application which is intended to remove the difficulties of suspension setup and allow for quick and easy production of a basic setup, providing a single sag setting would be preferable over a multitude of information.

2.1.4 Sag

Sag is the amount that the suspension sits into its travel when the rider is in their neutral position, it is calculated using the rider's weight, available travel, and intended riding style. Sag is required so the suspension has travel available to drop into holes as well as soak up bumps.

To adjust sag, the stiffness of the spring must be adjusted as required. This is done by changing the air pressure when using an air spring or replacing the coil and adjusting the spring pre-load on traditional coil shocks. Depending on discipline and the amount of travel the bike has, sag can vary between 15% and 40% of the available travel though is commonly set between 25% and 35% for the average rider. 15% and 40% are reserved for competitive situations.

2.1.5 Damping

Suspension damping is carried out by forcing oil within the shock absorber through an arrangement of holes in the absorber's damping circuit. Reducing the size or number of holes making the travel of oil through the circuit slower and therefore increases the damping effect making compression or rebound slower.

2.1.5.1 Compression Damping This is applied while the shock absorber is being compressed. More damping forces the wheel to remain in contact with the ground which makes the suspension feel stiffer. Too much compression damping can make the suspension too stiff so it does not soften bumps or rough sections correctly. Too little can cause the suspension to "blow through" its travel prematurely potentially leaving none when it would be required.

2.1.5.2 Rebound Damping This is used to control the speed at which the shock absorber extends once it has been compressed. An optimal setting will allow the suspension to track the ground, returning after a bump as well as dropping into any holes. Too much rebound damping causes the suspension to return slowly and sometimes pack down meaning the absorber gradually runs out of travel. Too little can cause the suspension to buck the rider and lead to an accident.

2.1.5.3 High and Low Speed Damping Depending on the manufacturer and model of the shock absorber, the unit can include up to two adjustable speeds for each damping circuit making four adjustable damping settings in total. High speed adjustments are used in high impact situations such as large jumps or drops, compression tends to be set softer to remove impact and rebound slower so rider has time to recover and the bike is not made unstable.

Low speed adjustments are used against small movements such as rider weight shifts or long, slow compressions. Optimally compression is set stiffer as this type of feature can use a lot of travel and rebound damping set faster to deal with multiple features in quick succession.

2.1.6 Optimal Setup

Although setups will vary between rider, suspension system, and discipline there are some key aspects which all riders should aim to achieve. Sag should be set to an appropriate measurement by adjusting the air pressure on air shocks or spring rating on coil shocks. Compression damping should feel soft and soak up bumps efficiently without excessive bottoming out. Rebound damping should be set to return as fast as possible without bucking the rider, this is normally in the middle of the two extremes of setting with a slight bias to the fast option.

Attaining this optimal setup can be difficult for beginner and intermediate riders; due to their lack of experience and provided information they may not know how different frames react while being ridden and have not dealt with in depth suspension setup. Knowing which measurements to make and calculations required to produce a sag

setting are commonly unknown to this level of rider without prior training or investigation.

2.2 Image Analysis

Image Analysis (IA) is the use of various techniques such as pattern recognition, geometry calculations, and signal processing to extract information from digital images for later use. Image processing however is the application of various processes on an image to change or improve the way it looks. The processing stage normally comes before the analysis stage in an effort to simplify the analysis processes and improve their success.

2.2.1 Digital Camera Operation

A digital camera operates by capturing visible light reflected by objects onto the camera's sensor. The light must travel from the object through a convex focusing lens which refracts the light onto a suitable point on the sensor.

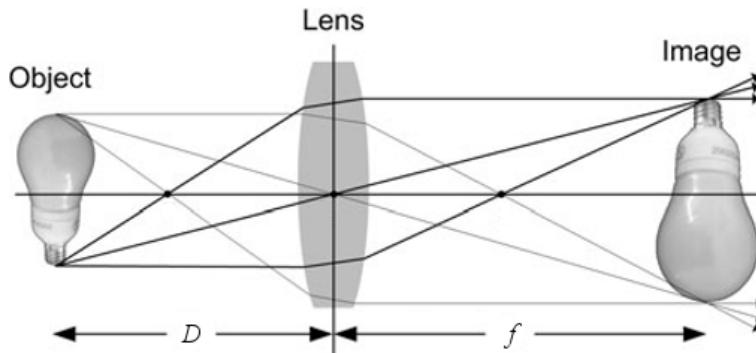


Figure 5: Diagram of camera and lens operation (Moeslund, 2012)

The distance between where light enters the lens and the point at which it is no longer diffused is known as the focal length, marked f in figure 5. This point can be adjusted by changing the lens optics so that the refracted light hits the sensor creating an in focus image.

A camera's sensor is made up of an array of photosensitive cells capable of collecting light and generating an integer value based on the brightness and colour of the received light. The microprocessor inside the camera takes these values and converts them into the image data, sometimes taking an average of the surrounding values to better understand the light which was captured. Each of the individual sensor cells equates to one pixel on the image produced, for example if a camera produces a 1920x1080 image, the sensor has 1920 cells across by 1080 down, otherwise known as a 2 mega-pixel sensor. The larger the sensor in physical size, the more cells it can contain so the better quality images it can produce.

These integer values in the image data can be manipulated to change the visual image

or the values themselves analysed or compared to neighbouring cells in order to locate objects. The processes are known as image processing and image analysis.

2.2.2 Lighting Conditions

When capturing images to be analysed, it is important that the lighting conditions are correct so features are not lost (Moeslund, 2012). If the image is captured in unsuitable conditions then this can seriously affect the outcome of any analysis techniques applied. Some image processing techniques can fix bad lighting conditions though this can prove difficult. The effects of different lighting angles can be seen in figure 6.

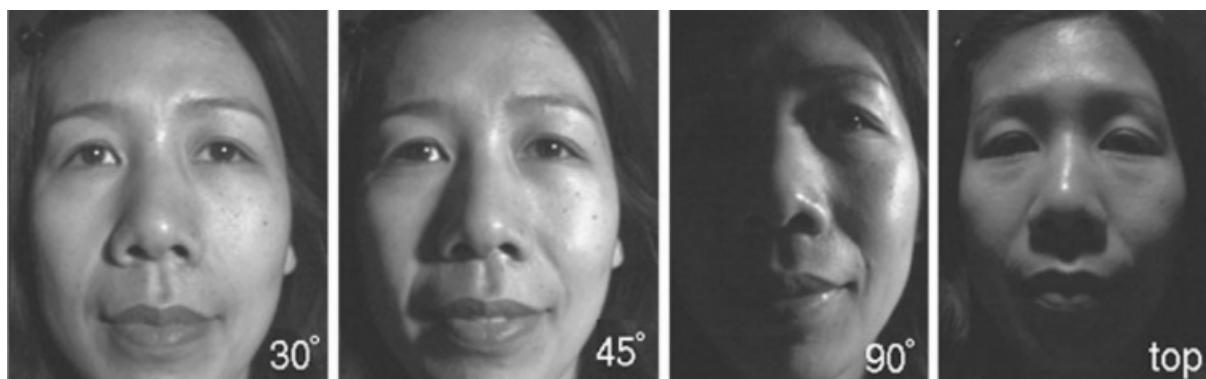


Figure 6: The effects of different lighting conditions on a face (Moeslund, 2012)

It can be seen that the situation which highlights the most features is direct illumination; this reduces the amount of image processing required before analysing. Many cameras have a built in flash which can be used to correctly light the subject of the image and in the context of this project many smartphones also have a flash function. This solves the need for the user to correctly light their image capture as the flash can be applied automatically if needed.

2.2.3 Usages

Image processing and analysis has been applied to multiple areas with its value and effectiveness rapidly improving alongside camera technology and computing power. These applications range from recognising faces in social media uploads (Zuckerberg, Sittig, & Marlette, 2011) to the utilisation of satellite imagery for tracking the changing shape of coastlines (Potter, 2013).

2.2.3.1 Medical Arguably one of the most important uses of IA, advances in medical imaging have reduced costs, diagnosis time, recovery time, and improved the ability to localise and personalise treatments (European Science Foundation, 2007). Major uses of IA in medical applications are Magnetic Resonance Imaging (MRI) and Computerised Topography Scanning (CT Scan) to create detailed images of the human body and identify illness before most symptoms arise. This is shown top left in Figure 7.

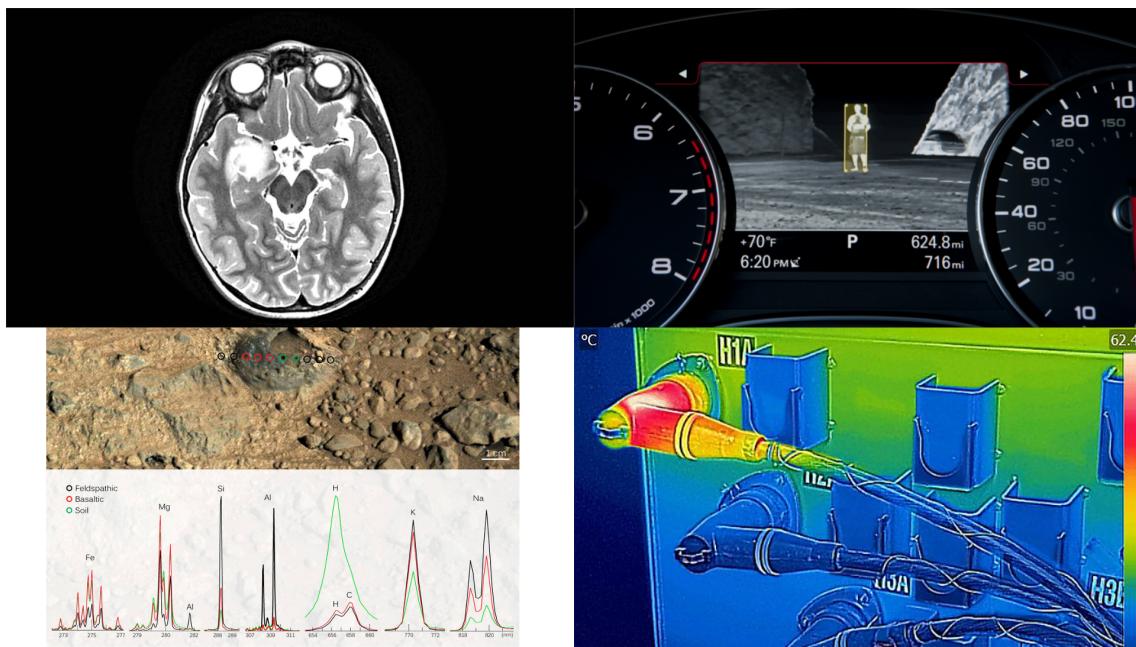


Figure 7: Uses of image analysis, from top left clockwise: An MRI brain scan, automotive night vision with pedestrian recognition, infrared image taken with a smartphone, chemical rock analysis from mars

2.2.3.2 Transport Image analysis has been included in the consumer automotive market on various models since 2004 when Honda introduced an thermographic night vision camera with pedestrian detection on the Legend (Honda Motor Co., 2004), Audi's implementation can be seen top right in figure 7. Since this initial introduction many vehicle manufacturers have included image analysis and recognition features as options such as speed limit sign recognition, lane departure warning systems, and automatic braking systems based on hazard recognition.

2.2.3.3 Engineering The use of image analysis in engineering has pushed to create more stable and efficient structures by looking at the materials used in their construction (Masad, Muhunthan, Shashidhar, & Harman, 1999) and monitoring their stresses and potential weak areas (Kim & Kim, 2013). Using image analysis by engineers on site has become more common with the advances in mobile computing and some manufacturers aiming their products at an engineering demographic with features like improved durability and built in infrared imaging (Liszewski, 2016), the output of which can be seen lower right in figure 7.

2.2.3.4 Space While some industries make use of satellite imagery to monitor changes on our own planet, agencies such as NASA and ESA make use of image analysis to look at other planets and celestial bodies. The Martian rover, Curiosity, uses multiple cameras for navigation, hazard avoidance, and scientific imaging the products of which are streamed back to Earth for analysis. Major uses for the various types of images returned include identification of geological formations and compositions (Blake et al., 2013; Garvin, Malin, & Miniti, 2014) and chemical location and identification using the "ChemCam" (Schröder et al., 2015). Chem cam analysis can be seen lower left in figure 7.

2.2.4 Image Analysis Techniques

There are multiple techniques which can be applied to imagery to extract information which include detection of edges or objects and using known data to take measurements. These methods tend to compare data from neighbouring pixels to spot differences which can indicate features.

2.2.4.1 Edge Detection This is the application of mathematical algorithms to locate and highlight the edges of features in an image. There are multiple algorithms which can be used for edge detection including Sobel, Roberts, Canny, and fuzzy logic though all utilise the concept of comparing side by side pixel data to find "steps" from one brightness to another.

Table 2: Table of pixel data showing an edge

5	7	6	4	152	148	149

Table 2 represents possible pixel values of an edge indicated by the large difference between 4 and 152. The applied algorithm will pick up on this discrepancy and it will be indicated on the resulting image. A common application for edge detection is text recognition such as in automatic number plate recognition (ANPR) (Ahmad, Boufama, Habashi, Anderson, & Elamsy, 2015) as the process can remove unwanted background data and highlight the block shapes of the number plate. Figure 8 shows an image which has had thresholding applied with the number plate of the car clearly visible.



Figure 8: Edge detection applied to an image for number plate recognition

2.2.4.2 Hough Transform ^{To do (9)} Originally created in 1962 by Paul Hough, the Hough transform methods offer techniques for the discovery of features within images which includes features with imperfections. The original algorithms were created for the detection of lines but have since been improved and adapted to find circles, ellipses, and other basic shapes.

Edge detection is suitable to highlight points in an image clearly though issues with the base image such as incorrect lighting or being out of focus and issues with the edge detector creating gaps in lines. These artefacts can make rudimentary feature detection techniques difficult. The Hough techniques circumvent these issues by considering points as a part of an object through a voting procedure.

Linear Hough Transform To locate straight lines in an image, the linear Hough transform utilises Hesse normal form $r = x\cos\theta + y\sin\theta$. Lines, or planes, in an image are expressed as pairs of (r, θ)

Circular Hough Transform

2.2.4.3 Taking Measurements To measure an object in an image, certain data about the camera and camera's location are required. By using digital imagery the majority of this information is provided as each image contains metadata or EXchangeable Image Format (EXIF) data which includes information such as camera manufacturer, focal length, image size, and location.

$$H_o = \left(\frac{f \times \left(\frac{H_i}{H_s} \right)}{D - f} \right) \times H_c \quad (1)$$

where

- f is the focal length of the camera lens
- D is the distance to the object
- H_o is the height of the object
- H_i is the height of the image in pixels
- H_s is the height of the camera sensor
- H_c is the height of the camera from the ground

The process to calculate the height of an object in an image is shown in equation 2. f , H_i , and H_s can be acquired from EXIF data, however D and H_c are not collected by the camera and must be measured. When dealing with image analysis this means that this data must have been collected when the image was taken.

It may be questioned that the distance of the object being measured from the ground is required however this is not the case. As the size of the object is being related to its size on the camera's sensor, the elevation of the object is not required. To test the equation, data was created from known object heights which was then re-used to see if the calculated height was equal to the known. For example: ^{To do (10)}

To circumvent the need for the height of the camera a reference object of known size can be included in the image, this allows a comparison between the object to be measured and the reference object. The previously mentioned Mars rover, Curiosity, carries a United States penny and charts to calibrate its cameras against; this is shown in figure 9.



Figure 9: Contact Instrument Calibration Targets on Mars Rover Curiosity (NASA JPL, 2012)

The reference object must be located in the image using a technique such as BLOB analysis or Hough transforms. Once the object is found, its width or height in pixels in the image must be collected, as the real world size of this object is known then its width in pixels equates to this real world measurement in the image. For example, if 2mm wide object measures 424px across in an image then it can be concluded that 212px in the image are equal to 1mm. Any other objects that can be measured in pixels their real world measurement can be estimated if they are close to the same distance from the camera as the reference object.

This process will be applied during this project in some manner, either using a reference object or measuring camera height, as knowing measurements is vital to suspension setup as previously stated. Implementation of equation 2 in an application will be relatively simple though collection of the relevant data with as little user interaction as possible will be difficult.

2.3 Image Analysis in Sports Science

The benefits of image analysis in sports science come to light when used in biomechanical situations. In most cases, an individual's performance in a sport depends on their physical fitness and technique; including the controversial topic of driver-athletes in motorsport . Despite being surrounded by the engineering of their vehicle or equipment, if the driver is unable to cope with the stresses and strains of race conditions then they will not succeed.

A car, motorcycle, and mountain bike are all predictable and measurable as they have been designed and manufactured by engineers so can easily be fitted with sensors, analysed, and adjusted accordingly. Fitting sensors to humans while maintaining their full mobility is not so simple. To provide the best testing ground for fitness and tech-

nique, the participant should be unhindered and able to perform tasks without data capturing equipment getting in the way.

Image analysis can play a large part in this as utilising various techniques can allow for stable and repeatable test situations while also providing a platform for review in the captured images. A study into cricket bowling technique used a mixture of manual point picking and automated measuring from images to produce data such as angle and speed of bowling deliveries as well as ball spin (Cork, Justham, & West, 2012). While the dataset was limited due to the player's training schedules, the data collected was useful as it was then used for replication on a pitching machine.

A second study made use of similar techniques to Cook, Justham, and West but these were applied to training horses for jumping competitions (Wejer, Lendo, & Lewczuk, 2013). The angles of limbs were recorded and compared over a period of four months to analyse improvements made from training. Using passive image analysis for this study would have been preferable over attaching sensors to the horse as this could have frightened the animal and almost certainly cause it to not have performed effectively.

2.4 Using Image Analysis for Mountain Bike Suspension

Though image analysis has been used for calculations in many engineering and sports applications (Kim & Kim, 2013; Masad et al., 1999), it is relatively unused with mountain bike suspension with only Fox Racing Shox using the technology for setup purposes (Benedict, 2012). The mobile application which Fox created is locked to forks and shocks that the company manufacture though image analysis techniques are adaptable enough to be used on any suspension unit.

By harnessing the image capturing and computing power of modern smartphones, image analysis can be applied to mountain bike suspension to produce a simple method of calculating a baseline setup for any suspension unit. The measurements and calculations required can be removed from the user's responsibility creating a simple and efficient method of generating a safe and reliable suspension setup.

2.5 Conclusion

By identifying the key aspects and settings relating to mountain bike suspension and the common image analysis techniques this literature review has highlighted the processes which this project can utilise to complete its aims. By using this knowledge, the following sections will identify distinct methods which the project will use to produce the application and cover the results. ^{To do (11)}

3 Methodology

3.1 Introduction

To do (12) This chapter will give an outline of the methodologies used to complete the work identified in the previous chapters as well as the reasons for using these methods and alternatives which could have been used. The effectiveness of these methods can determine the success of the project so the chosen methods were required to be reliable and manageable. A technical approach was identified for the most appropriate way to produce a solution to the problem identified and a project management approach was decided on so that the project could remain on track and meet the required deadline.

3.2 Literature Review

The purpose of the literature review presented in section 2 of this dissertation is to outline, investigate, and clarify the subject areas in which this project is involved. This aids the reader and author in understanding these subject areas as an explanation as to how the project presents a viable solution to the problem area. To carry out the review a methodology had to be chosen.

There are multiple types of literature review though they do not have the same aims. A traditional or narrative review can be carried out to critique a body of literature and potentially locate inconsistencies To do (13); this type is normally used when the research question is well defined. Alternatively a systematic literature review can be produced; this requires a more rigorous process. Systematic reviews aim to locate studies which are within the same or similar subject areas and use them to answer the questions presented by the research To do (14).

For this project a loose combination of the two has been used. A systematic approach was applied to examine image analysis uses and techniques to answer how they could be used in the context of mountain bike suspension. A traditional method has been applied to examine the uses of image analysis in sports science to determine if the techniques would transfer to mountain bike suspension and give an idea of how effective it could be.

3.2.1 Source Selection

For both methods, sources were required to be found which warrants a methodical approach to ensure they are suitable. Multiple services are available for viewing academic papers online; for this project the Edinburgh Napier University library and Google Scholar were utilised as between them they have an expansive library to select from and allow for the use of advanced search operators.

Due to the emerging and more corporate subject of applying technology to mountain bike suspension, there were no academic sources to use; because of this multiple mountain bike websites and blogs have been cited. In a context where the research is more established this would be frowned upon as this type of source tends to be opinionated and unproven though due to the age of this subject this was unavoidable. To

counteract this, multiple sources have been cited on the same subject as cross reinforcement.

To determine if any source is suitable, a methodical and critical approach must be taken when reading the paper or article. Initially a decision can be made before reading by using the publication date to ensure the subject and research is up to date, steps to use research which is the most up to date as possible were taken during this project. Following this, the number of times a paper has been cited elsewhere is provided in online libraries. From this metric it can be determined that if a paper has been cited a higher number of times then it is more suitable and legitimate; this view should be taken with reservations. Finally when reading a paper the abstract should be read first followed by conclusions and if it is still considered good research, the body can be read. This reduces the amount of reading required as the abstract and conclusions provide a good indication of a suitable paper.

3.3 Platform

As a proof of concept, the final product of this software project could take a variety of forms. In line with current products on the market an application for Android could be produced which would demonstrate the capabilities of image analysis on a mobile device. Alternatively, the image analysis algorithm can be produced in the Python programming language as a script creating a simpler prototype but clarifying the solution to the problem.

3.3.1 OpenCV

OpenCV is an open-source computer vision library created for a variety of platforms. Originally released by Intel in 1999^{To do (15)} the library was intended as a research project to aid in CPU intensive visual applications. In 2012 the library was taken over by OpenCV.org^{To do (16)}, a non-profit organisation who maintain support and the documentation. Though the library is written in C and C++, modules are available for Python, Fortran, and the Android platform.

This project will use OpenCV as it is widely accepted as the best, free computer vision library available^{To do (17)} and provides functions for the various processes this project will require. Additionally it is well supported and documented which will aid in the development process.

3.3.2 Experimentation

To further understand what may be required to produce both the Android and Python based approaches and aid in deciding which method to use some basic experiments were carried out to compare the two. For the Android experiments some example applications which are bundled with the OpenCV library were hand copied and run on a mobile device. This allowed their output and functionality to be seen as well as experience in using OpenCV on Android. For the Python based approach, tutorials from the site www.pyimagesearch.com created by Adrian Rosebrock and www.pyimagesearch.com creators name unknown were followed and run on a desktop computer.

Appendix A shows the experiments which were carried out on the android platform. It should be noted that, while all applications do not show compilation errors and were adjusted to work with the updated version of Android which the device was using, only two of the four experiments work successfully. Although stack traces of the errors were produced they are not descriptive about the cause of the error. This is due to how OpenCV works on the Android system.

Alongside the Android Software Development Kit (SDK), Google provide a Native Development Kit (NDK) to allow modules which were not written in Java to be run on Android devices. The NDK understands various programming languages and applied a Java wrapper around them which is capable of extracting their functionality; this must be applied when using OpenCV on Android as it is written in C++. An artefact of using the NDK is that it cannot convert the stack trace produced by errors in the C++ module and after research into this issue it was found that this is difficult to rectify.

The two working Android experiments do not operate as expected either. Shown in appendix A the issues with each can be seen. Although steps were taken to rectify the orientation in the Hello CV experiment and the full-screen issue in both, any fix which was applied was not accepted by the system. Research and debugging of these faults did not produce any conclusions. Appendix B shows the experiments which were carried out using Python. These were much more successful than the Android experiments as they are all functional and work as expected.

Included in appendix A and B as a comparison between the two methods are the number of files and lines of code required to create the program as well as the relevant source code for each program. There is a clear difference between the two methods. The Android method taking an average of 295 lines of code over 3 files to produce arguably less complex and less functional applications than is produced by Python's 21 lines of code over 1 file.

Due to the difficulties when using OpenCV on Android and as the operating system is already a proven platform for mobile applications the decision has been made to produce a proof of concept for the image analysis in Python. This will allow the project to maintain focus on the analysis and algorithmic side of the problem as opposed to the portability. The decision also allows for more time to be spent making the program functionally stable which will serve as a better demonstration of the solution.

3.4 Source Code

3.4.1 Pythonic Coding

A metric of software quality is the conciseness and descriptiveness of the source code. The aim of the Python programming language is to complete the same tasks as other object oriented languages but in fewer lines and in a more readable manner^{To do (18)}. By removing brackets and instead using indentation to define the boundaries of classes, functions, and conditionals as well as using worded operators, Python creates source code which reads like a list of instructions for people rather than computers. The source

code for this project will be written in a pythonic way where it makes sense to do so.

```

1 long_string = 'This is a very long string'
2 if 'long' in long_string:
3     print 'Match found'
```

Listing 1: An example of Python code

3.4.2 Naming

As well as the readability which comes from Python, further efforts will be made to ensure all classes, methods, and variables will be named as descriptively as possible. This makes the system and its algorithms easily understandable from an outside perspective and if any parts of the code need to be revisited at a later date. This is demonstrated in listing 2, although the left code looks cleaner, when reading through it is extremely generic and could be part of any system. In contrast the right code is much more descriptive of its function.

<pre> 1 public List<int[]> getThem() { 2 List<int[]> list1 = new 3 → ArrayList<int[]>(); 4 for (int[] x : theList) 5 if (x[0] == 4) 6 list1.add(x); 7 return list1; }</pre>	<pre> 1 public List<int[]> getFlaggedCells() { 2 List<int[]> flaggedCells = new 3 → ArrayList<int[]>(); 4 for (int[] cell : gameBoard) 5 if (cell[STATUS_VALUE] == FLAGGED) 6 flaggedCells.add(cell); 7 }</pre>
---	---

Listing 2: Examples of bad naming (left) and proper naming (right) taken from Clean Code (Martin, 2009)

3.4.3 Object Orientation

Object orientation is the use of classes which contain their specific variables and methods to protect functionality from other parts of a system. An object oriented approach will be taken when creating the system to ensure its functions and data are protected should it ever be used as a module elsewhere. The use of private variables and methods with one or two public methods provides an API with which other developers can use the system.

Python does not provide the private and public keywords as found in the Java or C++ languages. It instead identifies methods and variables prefixed by one or two underscores as protected. This means these items will be hidden from view when the system is used though are accessible should the developer require them. The creators of Python chose this method as they enforce responsibility over restriction.

3.5 Testing

To verify the functionality and quality of a software application it must be tested. This can be carried out at a variety of levels from testing a single class method to an entire system and by knowing how an application works in white-box testing or being unaware of the functionality in black-box testing.

3.5.1 Unit Testing

Unit testing is carried out on individual units of source code to ensure they are functioning correctly. Normally carried out in the scope of an individual class or module these unit tests are typically written by the same software developer who produced the class itself. A single unit test comprises of a set-up process where data and objects are initialised, the test itself including an assertion on a variable determining pass or fail, and a tear-down process where any changes the test has made are cleared up.

As unit tests are simple in structure and quick to run then they are commonly executed when changes are made to the source code. This verifies that the changes made have not broken other sections of code and can be committed into the master branch successfully. If a test failure does occur then each unit test should be concise and descriptive enough to aid the debugging process by indicating what has failed and where.

A metric of quality for unit testing is code coverage. To confirm that a class is of good quality and functional, every possible path through the code must be tested. Testers should aim for 100% coverage of a module meaning every possible piece of functionality has an associated test, a coverage of 85% for example means 15% of a module is not under test and could cause undetected problems if any changes are made. Many IDEs or testing suites provide coverage checking functionality to make this process simple.

3.5.1.1 Python Unit Testing Due to the package based and open-sourced nature of the Python programming languages there are many implementations of unit testing frameworks to choose from. Each with their own advantages and disadvantages, there is common argument over which framework is the best to use. A good unit testing framework should provide the necessary functions to create simple unit tests for each path in the source code and be substantial enough to ensure the class or module is correctly tested.

3.5.2 Project Testing Scope

Due to the small size of the application which will be produced in this project, testing will be limited to unit testing as there is no integration to be carried out. This small size also means that 100% code coverage will be simple to achieve. From this, stability and functionality of the application can be verified improving the quality of the overall product.

Once the basic functionality has been added to the application, unit tests will be created covering all source code produced including normal operation and sections which can

potentially produce errors. This means any changes made thereafter can be tested to see if they have affected functionality. Any new functionality will have tests added to the testing suite to keep the 100% coverage.

The Python unit testing framework chosen is unittest2. Unittest is the default framework used by the PyCharm IDE though this has been updated since the inception of Python3. Unittest2 provides a backport of this functionality for use in Python2 meaning extra testing functions are provided to allow more suitable tests to be created. PyCharm also provides the ability to run tests with code coverage producing a clear indication of which sections of source code are covered and which are not; this will be greatly beneficial in the aim for full coverage.

3.6 Project Management

3.6.1 Agile Development

To do⁽¹⁹⁾ Introduced in 2001 with the writing of the Agile manifesto (Beck et al., 2001), agile development methodologies focus on high quality software products over the rigorous design based structure of traditional waterfall methods. By utilising various techniques such as stand-up meetings, a strong customer focus, and sprint cycles, agile has become widely adopted in industry and has been proven to produce successful projects (VersionOne, 2015).

There are multiple agile methodologies (XP, Scrum, DSDM) all with their own principles and techniques however it is commonplace for a company to create their own development method which picks items from each to tailor to their needs^{To do (20)}. As this is a solo project with no customer then one set methodology will not be used, instead a variety of methods which will aid the management of the project and increase the efficiency of the development process have been selected. These will be outlined in the following sections.

3.6.1.1 Requirements Analysis Used in some form by the majority of agile methodologies^{To do (21)}, requirements analysis or requirements engineering is a variety of processes used to create the conditions that a project must meet. These requirements take into account stakeholders, users, and the development team or teams. Each requirement should be documented, actionable, measurable, testable, and traceable to aid in its understanding and completion^{To do (22)}.

The techniques used to produce requirements include stakeholder identification and interviews which are used to identify what the stakeholders of the project require upon completion. These interviews may be followed by Joint Requirements Development Sessions which bring stakeholders together to further discuss the requirements of the project. A more traditional approach is to produce a contract-style requirements list though these are commonly extensive and incomplete due to their production without collaboration. The most used requirements analysis technique in agile development are use cases and user stories. These are either written or diagrammatic descriptions of how the system will interact with users or other systems and provide an indication of what will be required from the product.

Due to the prototype nature of the application being produced and as the project has no defined stakeholders the only requirements analysis technique which will be used is production of use cases. These will aid in identifying how the application will be operated by users and what the project will need to produce. These requirements will then be prioritised using the MoSCoW format and inserted into a tracking system for the development process; these are described in the following sections.

3.6.1.2 MoSCoW First used in the DSDM agile framework^{To do (23)}, MoSCoW analysis or prioritisation is the process of taking the requirements of a software product and placing them into one of four categories; must, should, could, and won't have. These deliverables may be prioritised for the entire project or for individual sprint cycles depending on the size and manageability of the project.

MoSCoW was created to provide customers a better understanding of software requirements. Opposed to using high, medium, and low priorities MoSCoW is more descriptive of what the prioritisation means for the software project. From a development point of view the prioritisation process allows developers to focus on the core requirements of the project first, creating a viable software solution early on with extra features being added later if the resources are available.

This project will use MoSCoW to prioritise the requirements identified using the requirements analysis technique allowing the development process to complete the goals in the correct order. This will ensure that the core aspects of the solution are implemented first creating a successful project early and allowing it to improve as time allows.

3.6.1.3 Sprint Cycles Used by Scrum development teams^{To do (24)}, a sprint is a time-boxed effort of work scheduled to take between one week and one month. At the start of a sprint goals are chosen from the project requirements which are to be completed by the end of the sprint. When a sprint is complete a retrospective is carried out by the development team to discuss what went well, what didn't, and how this can be rectified for the next sprint.

This project will use sprints in the same manner as an agile development team as this will allow easier completion of requirements and management of the development process. Using sprints will ensure the time allotted for development is used effectively which will lead to a higher quality product at the end of the project.

^{To do (25)} As mentioned previously, MoSCoW prioritised requirements will be selected at the start of each sprint cycle to set objectives the for work which will be carried out. Each cycle will aim to complete all of the must have requirements and the majority of should and could haves. At the end of each cycle a retrospective will be carried out which will re-prioritise the uncompleted requirements based on how successful the sprint cycle was. This means requirements may be promoted or demoted respectively.

3.6.2 Organic Development

An alternative method to agile development could be to take a more organic approach to the development process. This method would take the step by step ethic of agile and reduce the effort applied for analysis, documentation, and structuring of the working schedule.

To do (26) To initiate the development process without analysis of requirements would be to begin producing the basic steps or functionality which the system will go through from an vision of the end product. Once started, development would take a natural direction identified by what is needed for the system or to solve any issues encountered.

To document the development process, a log of any work completed would be kept per day whenever development is carried out. This would allow for referral at later dates and allow the project to remain on track by identifying when features are completed.

There are advantages and disadvantages to using this method over an agile process. Due to the size of the software being produced the amount of management work required of an agile process could mean the development is over managed

To do (27)

3.6.3 Version Control

Large software projects produce multiple files of code and documentation which are vital and must be kept safe. Were the files to be lost then the project would be delayed or drawn to a close as the time and resources may not be available to recreate the lost data. To combat this any data relating to the project must be backed up, preferably on a cloud based system, to avoid loss and allow the project to continue should anything happen to the local copy of the data.

For this project the Git Version Control System (VCS) will be used. Git provides use of a cloud hosted repository to store any files relating to a project and allows for work to be carried out locally by cloning the repository on a computer. However VCSs also enable the management of the previous versions of files including information such as the individual changes made, when those changes were made, and who made the changes. This is a powerful tool as it means the various sections of the project can be experimented on without the risk of damaging the project; if a change is unsuccessful then the repository can be reverted to a functional point.

The web service used to host the repository will be github.com. This site was chosen as it provides unlimited free repositories as well as simple repository management tools. The website also provides issue tracking functionality; though they are not issues, each feature will be added as an entry in the issue tracker. The reason for this is that all features will have a unique identification and description which commits to the repository can be placed against. This is useful for project management reasons as then each feature's stage and completeness can be monitored to ensure the project's success.

Alternative VCSs are available, for example Microsoft's Team Foundation Server or Perforce, however these are limited under free licences or locked to certain development environments. The accessibility and ease of use provided by Git makes it the optimum VCS to use for this project.

3.6.4 Milestones

Milestones are used in project management to mark significant events or points within a project's timeline. This allows a further breakdown of the project as milestones can be used as supplementary deadlines. The use of milestones allows the project manager, management team, or development team to keep track of the project's status and priorities at any given time.

This project has and will use milestones for the same reason. Current examples include the week nine review session and deadline for the completion of this document. As more milestones are identified through breakdown of the development process they will be documented and acted upon.

3.6.5 Threshold

Tasks using more than their allotted time in a common cause of projects using more of their resources. To ensure all tasks can be completed within the allotted time for this project, each task will be allocated a threshold. Adding a threshold is a common technique in project management and provides extra time should anything unexpected occur which impacts the project.

3.6.6 Gantt Chart

A Gantt chart is a method of plotting a project's schedule which was invented by Henry Gantt in the 1910s. By breaking down a project into tasks these can then be added to the chart as a bar. Commonly starting at a project's inception date and ending with its completion date, tasks are allocated an estimated duration for their completion and placed within the timeline. These tasks can then be allocated dependencies to indicate their prerequisites.

This project will make full use of the Gantt chart technique by reducing the project into multiple stages for both writing and development. Previously mentioned milestones will also be added for the known fixed points. Each sprint cycle for the development process will be indicated with further notes on the tasks to be carried out. To create the chart Microsoft Project 2016 will be used.

3.7 Evaluation

To ensure that the application produced is extensively evaluated, multiple methods will be used. Each of the different methods will evaluate a single aspect of the application from varying points of view.

3.7.1 Validation

The is a measure of how well a design or solution is appropriate for its purpose and performs as expected. It involves checking all required functionality is present, each output is correct and as expected, and the solution performs as expected.

In this project, validation will be provided through the implemented unit tests. Ensuring the application has near 100% coverage and all tests are passing then it can be confirmed that the application is operating as expected.

3.7.2 Reliability and Accuracy

Assessing the reliability and accuracy of the application proves that the results produced are consistent and correct; to assess these an uncertainty metric will be produced. When taking multiple measurements there is typically a "true" value which is the actual measurement that falls within the range of produced measurements. Uncertainty is a prediction of how close any produced measurement will be to this "true" value, expressed as $\bar{x} \pm U$. The method for producing uncertainty is as follows:

1. Produce a suitable number of repeat measurements $\{x_0 \dots x_n\}$
2. Calculate the average value of these measurements $\bar{x} = \frac{\sum\{x_0 \dots x_n\}}{n}$
3. Find the differences between the measurements and the average $\{d_0 \dots d_n\} = \{(x_0 - \bar{x}) \dots (x_n - \bar{x})\}$
4. Calculate the average of these differences squared $\overline{d_s} = \frac{\sum\{d_0^2 \dots d_n^2\}}{n}$
5. Produce the uncertainty or standard deviation of these results which is the square root of the average differences $U = \sqrt{\overline{d_s}}$

This process will be carried out a number of times altering different variances possible depending on what the application is capable of once produced. These variances will be outlined once the evaluation has been completed.

3.7.3 Comparison to Alternatives

By comparing the process of using the application with other methods of producing a sag setting it can be evaluated on how simple the application is to use and whether it is more effective than other methods. For this comparison a trial and error process will be used which would be a common practice for a beginner or intermediate rider setting their suspension for the first time. This process involves manually calculating the desired sag measurement, pressurising the shock to the manufacturers recommended pressure, loading the suspension with the rider's weight, and measuring. If the manufacturers setting is incorrect, pressure is then increased or removed by set increments. Comparison to this process will indicate that the application presents a benefit over the manual method.

3.7.4 Professional Opinion

The final method of evaluation will be to seek the opinion of the application from professionals within the mountain bike and cycling industry. This will provide a further indication of how well the application achieves its goal, how the application would be accepted were it to be released as a product, and may present areas for future work which may not have been seen without outside consultation.

To carry out professional consultation the Mountain Bike Research Centre of Scotland² will be contacted, once the application can be demonstrated, to arrange a meeting with an individual who they deem suitable. General evaluative questions will be posed though the meeting will take a semi-formal approach. Additional to this the application will be demonstrated to staff at a local bike shop to also gain their opinion, this will take the same approach as the meeting.

²<http://www.napier.ac.uk/about-us/our-schools/school-of-applied-sciences/research/mountain-bike-centre-of-scotland>

4 Results

This chapter will cover how the project was created and evaluated. It will begin by describing any deviations from the planned development process including any problems which were encountered and how they were addressed. It will then continue to describe how the final version of the application operates with a pseudocode description. Finally it will present the previously outlined critical evaluation covering validation, reliability, alternative comparison, and professional opinion.

4.1 Deviation from Plan

4.1.1 Development Approach

As described in section 3.6 it was unsure which approach would be undertaken during the development stages of the project. Once the development stage was reached, an organic approach was taken. Although less structured than an agile methodology it was concluded that removal of the sprint process would allow development of the application to progress quicker as items would not need to be delayed for the next sprint cycle. Allowing problems to be addressed immediately proved useful as the application was not left in an unusable state while other features were produced.

The log produced by the development stage can be seen in appendix C. This shows that suitable work was carried out within the time-frame allocated despite using a less structured process and the application includes suitable functionality to complete its aims.

4.1.2 Use of EXIF Data and Reference Point

Mentioned in section 2.2.4.3, the process which the application would use to produce measurements was undecided and required experimentation. Initially utilising Exchangeable Image File Format (EXIF) data from the image was used; this was successful until collection of the sensor size was required. Due to this being uncommon in EXIF data, particularly in images taken with smartphone cameras, this method was dropped for utilisation of a reference point. The code produced during this experimentation period can be seen in appendix D, the function `get_sensor_size` is incomplete as this is where the process was dropped.

The reference point chosen is a red circular sticker as these are cheap and easy to find in stores or online and a circle was selected over a square or other shape as the orientation of the object does not need to be taken into account. This sticker is applied to the shock so it can be picked up from the image by its colour; locating the reference point was simple though the measuring process produced issues which will be discussed in a following section.

4.1.3 Colour Quantification

For initial experimentation with using a reference point, the red circle was added to images manually using an image processing application. This allowed for tuning of the

colour masking process and to develop the process before using a real reference point on the shock.

When images with a real reference point were used, the application could not find it. Because the colour range for the masking process was using perfect red (RGB 255,0,0) the non-uniform reds of the sticker in the image were not within this range. These boundaries were expanded to suit the range in the image though this was not enough.

To solve this issue a further image processing technique was implemented known as colour quantification which reduces an image to 2, 4, 8, 16, or 32 bit colour spaces. By reducing the range of colours in an image detail is lost and objects become more apparent. The difference between an original image and quantified colours is shown in figure 10. Quantifying the colours to 8 bit makes the reference point a flat tone of red while maintaining its shape allowing for the masking process to function correctly.

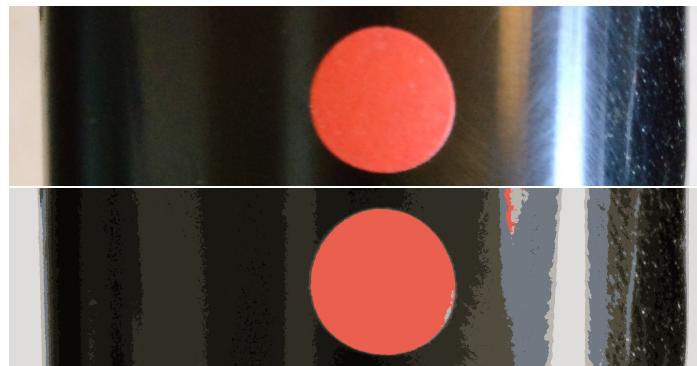


Figure 10: Normal image (top) versus quantified colours (bottom)

4.1.4 Dynamic Measurement Limits

It was decided that the application should utilise the distance between the shock wiper seal and the marker o-ring for providing the measurement. This was deemed suitable as the manual process to calculate sag also uses the o-ring and it should be present on most shocks; the only reason an o-ring would not be present is if it has been removed by the owner. Initial versions of the application assumed the o-ring was positioned around 2/3 of the image height though once multiple images were used from two different shocks this assumption was no longer correct.

So the application is able to analyse different images, a dynamic method to find this o-ring was implemented. Much like finding the reference point an o-ring of specific colour is found, if present, using OpenCV's `findContours`. When located, a bounding box is drawn round the contour from which the y coordinate and height are used to set the limit of measurement. This can be seen in figure 11.

As some shocks do not have a coloured o-ring, this process was adapted to locate a black o-ring though it was unsuccessful. In response, an alternative method was applied where thresholding is applied to the image and contours produced by reflection

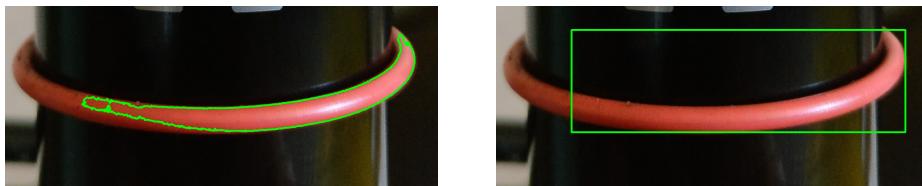


Figure 11: Red O-ring found using `findContours` (left) with `boundingBox` applied (right)

on the shock's shaft are selected. The same bounding box is applied and used as the measurement limit. This process was successful as indicated in figure 12.



Figure 12: Black o-ring found

4.1.5 Reference Point Measurement Method

To measure the circular reference point, the first process applied was Hough Circle Transform. This method was chosen as it sets out to specifically find circles in an image and is simple to implement. Though this did produce a measurement the numbers were regularly incorrect when compared to manually measuring the image and were unreliable, varying between too small and too big. This can be seen in figure 13 where it is clear that the identified circle is much smaller than the actual reference point.

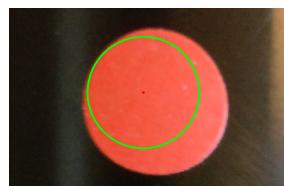


Figure 13: Reference point found using `HoughCircles`

To circumvent this issue the same process for locating the o-ring was used. A contour is found which encompasses the full reference point and a bounding circle is drawn around this contour; this is shown in figure 14. Though the bounding circle appears larger than the reference point, after experimentation with a variety of images, this method produced measurements which were within tolerances of what is suitable for the pixel per millimetre calculation.

This larger appearance is caused by the reference point not being a uniform circle which is also the reason the Hough Circle method was unreliable. Although the threshold of the Hough Circle method can be adjusted, the reference point was still too misshapen to be detected correctly.



Figure 14: Reference point found using `findContours` (left) with `boundingCircle` applied (right)

4.1.6 Pressure Calculation

The original method for producing a suggested pressure was to calculate a psi per millimetre metric from the measurement of the shock shaft and the current pressure in the shock supplied by the user. The equation for this metric is as follows:

$$P_s = \left(\frac{P_c}{M_c} \right) M_s \quad (2)$$

where

P_s is the pressure to produce the desired sag setting

P_c is the pressure currently in the shock

M_s is the measurement to produce the desired sag setting

M_c is the current measurement of the shock shaft

Though this hypothesised method had been tested manually on paper with successful results, when implemented within the application it was unable to produce correct settings from any variation of image, reference point, and desired sag even when these values were hard-coded. Through further investigation and discussion it was suggested that the non-linearity of air springs was the cause of the problem.

When an air spring is compressed the spring rate increases through its travel as the particles become compressed together^{To do (28)}; while this allows tuning of the shock characteristics it presents difficulty when attempting to apply a linear equation in a non-linear situation. However as the range of measurement in this project is small compared to the full stroke of an air shock it was considered that the spring rate is linear within this range.

To investigate this, the sag of a shock was measured at pressures between 100 and 250 psi in increments of 10 psi. This was carried out for two shocks, one with a normal air can and another with a high volume air can then the collected data was then plotted and a linear trend line applied. The results of this can be seen in figure 15.

From this data it could be concluded that, within the range of measurements, the compression of the air spring is linear. The data-points in figure 15 occasionally vary though

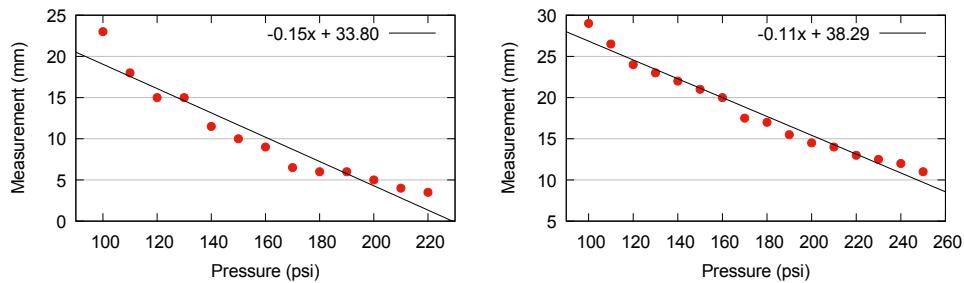


Figure 15: Sag measurements for standard air shock (left) and high volume air shock (right)

this is due to a non sterile test. Both shocks were well used potentially containing age related defects and loading the bike was carried out using body weight.

The plan with this knowledge was to implement an ideal gas equation but through investigation into producing charts using python, a method of calculating a linear equation computationally was discovered. This led to the application's process being rearranged so that it takes in two images, one of the shock loaded at 100psi and another at 150psi. From this data it can produce a virtual plot and linear equation with which the suggested setting can be calculated.

4.2 Application

4.2.1 Arguments

The application operates from a command line interface allowing the user to pass in data through arguments; these are described in Table 3. This system uses Python's Argparse package and a full command to run the application would be as follows:

```
program -i user/images/100.jpg user/images/150.jpg -s 30 -t 57 -c red
```

Table 3: Table of application arguments

Argument	Description	Usage
<code>[-i,--image]<path1 path2></code>	The paths to the two images to be used for analysis	<code>-i path/to/img/1 path/to/img/2</code>
<code>[-s,--sag]<percentage></code>	The desired sag percentage	<code>-s 30</code>
<code>-t,--stroke</code>	The stroke length of the shock being analysed	<code>-t 57</code>
<code>-c,--colour</code>	The colour of the marker o-ring	<code>-c red</code>
<code>-d,--debug</code>	This produces outputs from each stage to the terminal for debugging purposes	

4.2.2 Process

4.2.2.1 Reference Point The first process which the application must complete is to find the reference point for measurement. This is done by first colour quantifying the image to 8 bit colour so that the reference point becomes uniform in colour. The result of this process is shown in Figure 16a. So the application can find the red colour of the reference point within the entire image, a mask is applied for any colour within a range. The output of this process is shown in Figure 16b. If an alternative colour reference point were to be used, this colour range could be easily adapted.

The contours, or shapes, in this image are then found and sorted by their area in decreasing order. By doing this it can be assumed with a high level of confidence that the largest will be the reference point. The found contour can be seen in Figure 16c. A bounding circle is created around the reference point contour, shown in Figure 16d, from which the diameter can be produced which allows the pixels per millimetre metric.

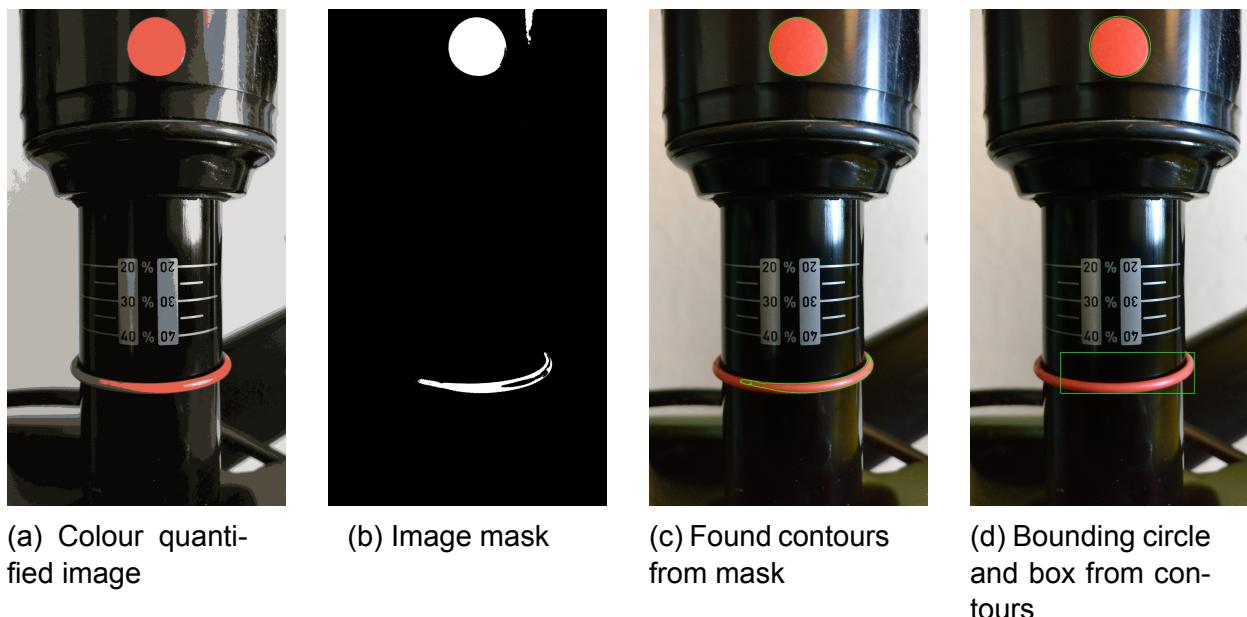
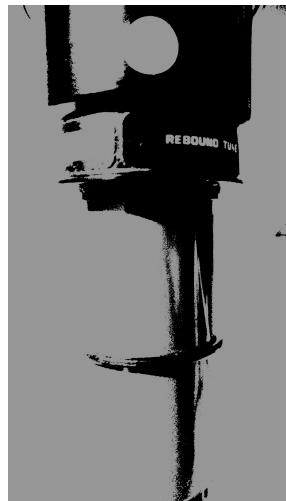


Figure 16: Processed images for finding reference point

4.2.2.2 O-Ring If a red coloured o-ring is specified then the same process to find the reference point is used. The second largest contour found will be the o-ring so a bounding box can be drawn round it from which a measurement limit can be collected. The red o-ring mask, contour, and bounding box can be seen in Figure 16.

If a black o-ring is specified, an alternative process is used. Thresholding is applied to the image which highlights any shadows or highlights on the shock shaft. As these are intersected by the o-ring it is simple to select a contour which ends at the o-ring to produce a measurement limit from. This process is shown in Figure 17.

4.2.2.3 Measurement With the pixels per millimetre metric produced and the measurement limit found, the sag under the current pressure is measured. First, edge



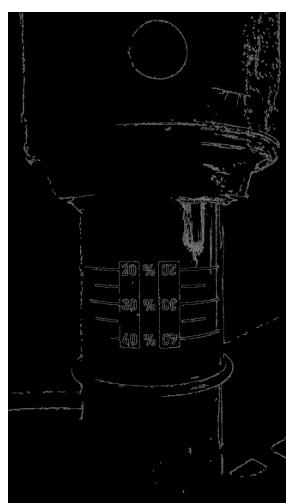
(a) Image thresholding



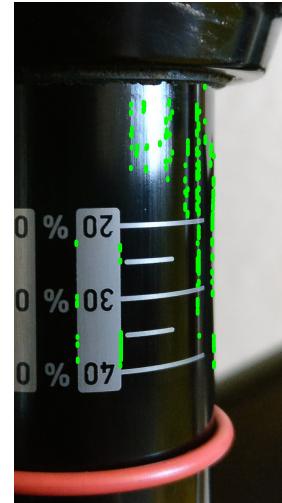
(b) O-ring indicator

Figure 17: Locating process for black o-ring

detection is carried out on the original input image to highlight the lines of the shock, this output is shown in Figure 18a. Hough line transformation is then applied to locate vertical lines between the top of the shock shaft and the measurement limit. The difference in pixels between the highest and lowest points of these lines is the current sag measurement. This can be converted to millimetres using the px/mm metric.



(a) Edge detected image



(b) Vertical lines in measurement area

4.2.2.4 Equation Once the measurements for the shock pressurised to 100psi and 150psi are produced, they can then be utilised to create a linear equation. This is carried out using Python's Scipy package and its linregress method which accepts two arrays of data and returns the slope and intercept of the linear equation. The equation is visually described in Figure 19; it should be noted that this plot is only produced virtually and never output to the user.

4.2.2.5 Produce Setting

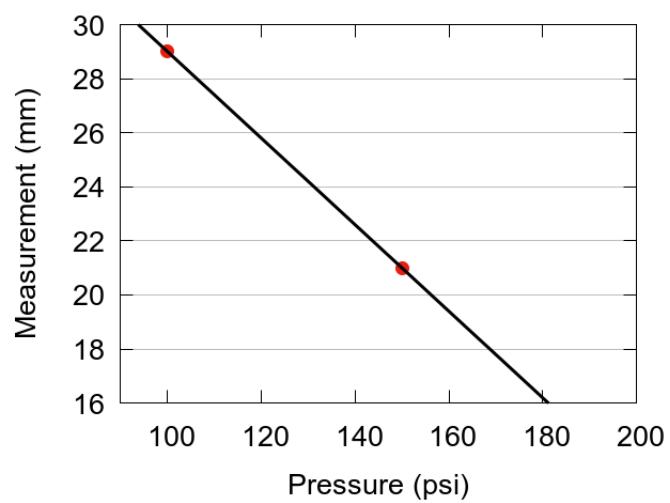


Figure 19: Example of linear equation plot

4.3 Critical Evaluation

4.3.1 Validation

4.3.2 Reliability and Accuracy

Table 4: Table of uncertainty calculation results

	Rockshox		Fox	
	25%	30%	25%	30%
Measurements	175.02	160.97	133.31	120.54
	174.84	160.52	133.28	120.47
	174.95	160.41	133.42	120.48
	174.83	160.65	133.42	120.46
	174.94	160.75	133.29	120.53
	174.86	160.50	133.38	120.51
	175.63	160.48	133.35	120.50
	174.78	160.77	133.36	120.46
	175.29	160.57	133.36	120.46
	174.84	160.74	133.42	120.50
Average	175.00	160.64	133.36	120.49
Standard Deviation	0.27	0.17	0.05	0.03
Uncertainty	175 ± 0.27	160.64 ± 0.17	133.36 ± 0.05	120.49 ± 0.03

4.3.3 Comparison to Alternatives

4.3.4 Professional Opinion

5 Conclusions

5.1 Meeting Aims

5.2 Comparison to other Products

5.3 Future Work

5.4 Self Appraisal

References

- Ahmad, I. S., Boufama, B., Habashi, P., Anderson, W., & Elamsy, T. (2015, Dec). Automatic license plate recognition: A comparative study. In *2015 ieee international symposium on signal processing and information technology (isspit)* (p. 635-640). doi: 10.1109/ISSPIT.2015.7394415
- Aston, P. (2016). Sussmybike data acquisition - eurobike 2016. Retrieved 24/09/2016, from <http://www.pinkbike.com/news/sussmybike-data-acquisition-eurobike-2016.html>
- Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... others (2001). Manifesto for agile software development.
- Benedict, T. (2012). Fox ird smartphone app sets ups shocks, forks for you. Retrieved 24/09/2016, from <http://www.bikerumor.com/2012/08/29/fox-ird-smartphone-app-sets-ups-shocks-forks-for-you/>
- European Science Foundation. (2007). *Medical imaging for improved patient care* [Policy Briefing]. Retrieved 03/10/2016, from http://www.esf.org/fileadmin/links/EMRC/ESF_POLICY28_V09_HD.pdf
- Giant Manufacturing Co. Ltd. (2017a). *Giant fathom* [Product Listing]. Retrieved from <https://www.giant-bicycles.com/gb/fathom-1>
- Giant Manufacturing Co. Ltd. (2017b). *Giant reign* [Product Listing]. Retrieved from <https://www.giant-bicycles.com/gb/reign-1>
- Giant Manufacturing Co. Ltd. (2017c). *Giant stance* [Product Listing]. Retrieved from <https://www.giant-bicycles.com/en-gb/bikes/model/stance/28553/99239/>
- Honda Motor Co. (2004). Honda develops world's first intelligent night vision system able to detect pedestrians and provide driver cautions-available on legend model to be released in fall 2004 [Press Release]. Retrieved 03/10/2016, from <http://world.honda.com/news/2004/4040824a-eng.html>
- IMBA Europe. (2015). *Imba european mountain bike survey* (Survey). International Mountain Bike Association. Retrieved 24/09/2015, from http://www.imba-europe.com/sites/default/files/IMBA_INFOGRAPHIC_final.pdf
- Blake, D. F., Morris, R. V., Kocurek, G., Morrison, S. M., Downs, R. T., Bish, D., ... Sarrazin, P. (2013). Curiosity at gale crater, mars: Characterization and analysis of the rocknest sand shadow. *Science*, 341(6153). Retrieved from <http://science.sciencemag.org/content/341/6153/1239505> doi: 10.1126/science.1239505
- Callaham, J. (2015). Google says there are now 1.4 billion active android devices worldwide. Retrieved 25/09/2016, from <http://www.androidcentral.com/google-says-there-are-now-14-billion-active-android-devices-worldwide>
- Cork, A., Justham, L., & West, A. (2012). Three-dimensional vision analysis to measure the release characteristics of elite bowlers in cricket. *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, 1754337112447264.
- Diaz, K. M., Krupka, D. J., Chang, M. J., Peacock, J., Ma, Y., Goldsmith, J., ... Davidson, K. W. (2015). Fitbit®: An accurate and reliable device for wireless physical activity tracking. *International Journal of Cardiology*, 185, 138-140. Retrieved 24/09/2016, from <http://dx.doi.org.ezproxy.napier.ac.uk/10.1016/j.ijcard.2015.03.038>
- Formosa, N. (2012). Social fitness apps storm cycling world. *Bicycle Retailer and Industry News*, 21, 28-29. Retrieved 24/09/2016,

- from http://search.proquest.com.ezproxy.napier.ac.uk/docview/1018189163?rfr_id=info%3Axri%2Fsid%3Aprimo
- Garvin, J. B., Malin, M. C., & Minitti, M. E. (2014, March 17). *Sedimentology of certain gravels from mardi twilight imaging: Techniques* (Tech. Rep. No. GSFC-E-DAA-TN13878). Retrieved 04/10/2016, from <http://hdl.handle.net/2060/20150001290>
- Harker, J. (2010). German brand rose hopes to bloom in britain. Retrieved 24/09/2016, from <http://www.bikebiz.com/news/read/german-brand-rose-hopes-to-bloom-in-britain/08815>
- Hwang, J. C. B. (2016, 05 19). *Shockwiz* (Trademark Application No. 87042899). 1000 West Fulton Market, 4th Floor, Chicago, Illinois, 60607, USA. Retrieved 25/09/2016, from <https://trademarks.justia.com/870/42/shockwiz-87042899.html>
- Kim, S.-W., & Kim, N.-S. (2013). Dynamic characteristics of suspension bridge hanger cables using digital image processing. *NDT & E International*, 59, 25 - 33. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0963869513000807> doi: <http://dx.doi.org/10.1016/j.ndteint.2013.05.002>
- Liszewski, A. (2016). Caterpillar's new s60 is the first smartphone with flir thermal imaging built right in. Retrieved 04/10/2016, from <http://gizmodo.com/caterpillars-new-s60-is-the-first-smartphone-with-flir-1759685817>
- Martin, R. C. (2009). *Clean code: a handbook of agile software craftsmanship*. Pearson Education.
- Masad, E., Muhunthan, B., Shashidhar, N., & Harman, T. (1999). Internal structure characterization of asphalt concrete using image analysis. *Journal of Computing in Civil Engineering*, 13(2), 88-95. Retrieved from [http://dx.doi.org/10.1061/\(ASCE\)0887-3801\(1999\)13:2\(88\)](http://dx.doi.org/10.1061/(ASCE)0887-3801(1999)13:2(88)) doi: 10.1061/(ASCE)0887-3801(1999)13:2(88)
- Moeslund, T. B. (2012). *Introduction to video and image processing: Building real systems and applications*. Springer.
- NASA JPL. (2012). *Contact instrument calibration targets on mars rover curiosity*. Retrieved from <images-assets.nasa.gov/image/PIA15284/PIA15284~orig.jpg>
- Potter, C. (2013). Ten years of land cover change on the california coast detected using landsat satellite image analysis: part 1—marin and san francisco counties. *Journal of Coastal Conservation*, 17(4), 697–707. Retrieved from <http://dx.doi.org/10.1007/s11852-013-0255-2> doi: 10.1007/s11852-013-0255-2
- Schröder, S., Meslin, P.-Y., Gasnault, O., Maurice, S., Cousin, A., Wiens, R., ... Vaniman, D. (2015). Hydrogen detection with chemcam at gale crater. *Icarus*, 249, 43 - 61. Retrieved from <http://www.sciencedirect.com/science/article/pii/S001910351400445X> (Special Issue: First Year of MSL) doi: <http://dx.doi.org/10.1016/j.icarus.2014.08.029>
- Staff, B. (2015). Yt industries launches consumer-direct sales in north america, oceania. Retrieved 24/09/2015, from <http://www.bicycleretailer.com/industry-news/2015/01/30/yt-industries-launches-consumer-direct-sales-north-america-oceania>
- Titlestad, J., Fairlie-Clarke, T., Davie, M., Whittaker, A., & Grant, S. (2003). Experimental evaluation of mountain bike suspension systems. *Acta Polytechnica*, 43, 15-20.
- VersionOne. (2015). *9th annual state of agile survey*.
- Wejer, J., Lendo, I., & Lewczuk, D. (2013). The effect of training on the jumping pa-

- rameters of inexperienced warmblood horses in free jumping. *Journal of Equine Veterinary Science*, 33(6), 483–486.
- West, L. R. (2015). Strava: challenge yourself to greater heights in physical activity/cycling and running. *British Journal of Sports Medicine*, 49, 1024. Retrieved 24/09/2016, from <http://bjsm.bmjjournals.org.ezproxy.napier.ac.uk/content/49/15/1024.full>
- Zuckerberg, M., Sittig, A., & Marlette, S. (2011, May 17). *Tagging digital media*. Google Patents. Retrieved from <https://www.google.com/patents/US7945653> (US Patent 7,945,653)

Acronyms

FS Full Suspension.

HT Hard Tail.

IA Image Analysis.

VCS Version Control System.

VPP Virtual Pivot Point.

Glossary

Fork The front suspension unit on a mountain bike.

Full Suspension A mountain bike with both front and rear suspension.

Hard Tail A mountain bike with only front suspension.

Horst Link A suspension system utilising a pivot point located near to and below the rear wheel axle.

Rebound Damping Controls the speed at which a suspension unit extends once it has been compressed. Less damping means the unit extends faster.

Sag The distance which the suspension sits into its travel when the rider is in their neutral position.

Shock The rear suspension unit. Only found on full suspension mountain bikes.

Single Pivot A suspension system which rotates around a physical main pivot point.

Stroke The distance a shock absorber can compress before bottoming out.

Travel The distance a wheel can move before bottoming out.

Virtual Pivot Point A rear suspension utilising an upper and lower link to create a simulated pivot a physical one would be impossible.

A Android Experiments

A.1 Table of Android Experiments

Experiment	Purpose	Functional	Issues	Files	LOC
Hello CV	<ul style="list-style-type: none">Introduction to the android OpenCV libraryDisplays camera feed with fps	Yes	<ul style="list-style-type: none">Not fullscreenIncorrect orientation	2	101
15Tile	<ul style="list-style-type: none">Sliding tile gameUses camera feed as puzzle	Yes	<ul style="list-style-type: none">Not fullscreen	3	492
Blob Detection	<ul style="list-style-type: none">Demonstrates blob detectionRuns blob detection on tapped area from camera	No	<ul style="list-style-type: none">Crash on screen tap	3	311
Face Detection	<ul style="list-style-type: none">Detects faces in camera viewPuts boundary around detected faces	No	<ul style="list-style-type: none">Crash on load	3	279

A.2 Hello CV

```

1 package com.example.joe.base_app;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5 import android.util.Log;
6 import android.view.SurfaceView;
7 import android.widget.TextView;
8 import org.opencv.android.*;
9 import org.opencv.core.*;
10
11 public class MainActivity extends AppCompatActivity implements
12     CameraBridgeViewBase.CvCameraViewListener2{
13
14     private static String TAG = "Main Activity";
15     private CameraBridgeViewBase mOpenCvCameraView;
16
17     private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
18         @Override
19         public void onManagerConnected(int status) {
20             switch (status){
21                 case LoaderCallbackInterface.SUCCESS:{
22                     Log.i(TAG, "Open CV loaded successfully");
23                     mOpenCvCameraView.enableView();
24                 }break;
25                 default:{
26                     super.onManagerConnected(status);
27                 }break;
28             }
29         };
30
31     @Override
32     public void onResume(){
33         super.onResume();
34         OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0, this,
35             mLoaderCallback);
36     }
37
38     @Override
39     protected void onCreate(Bundle savedInstanceState) {
40         Log.i(TAG, "called OnCreate");
41         super.onCreate(savedInstanceState);
42         setContentView(R.layout.activity_main);
43         mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.HelloOpenCvView);
44         mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);
45         mOpenCvCameraView.setCvCameraViewListener(this);
46     }
47
48     @Override
49     public void onPause(){
50         super.onPause();
51         if (mOpenCvCameraView != null){
52             mOpenCvCameraView.disableView();
53         }
54     }
55 }
```

```

54
55     @Override
56     public void onDestroy(){
57         super.onDestroy();
58         if (mOpenCvCameraView != null){
59             mOpenCvCameraView.disableView();
60         }
61     }
62
63     @Override
64     public void onCameraViewStarted(int width, int height) {
65
66     }
67
68     @Override
69     public void onCameraViewStopped() {
70
71     }
72
73     @Override
74     public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
75         return inputFrame.rgba();
76     }
77 }
```

A.3 15tile

```

1 package com.example.joe.a15tile;
2
3 import android.annotation.SuppressLint;
4 import android.app.ActionBar;
5 import android.app.Activity;
6 import android.os.Bundle;
7 import android.os.Handler;
8 import android.util.Log;
9 import android.view.Menu;
10 import android.view.MenuItem;
11 import android.view.MotionEvent;
12 import android.view.View;
13 import android.view.WindowManager;
14
15 import org.opencv.android.BaseLoaderCallback;
16 import org.opencv.android.CameraBridgeViewBase;
17 import org.opencv.android.JavaCameraView;
18 import org.opencv.android.LoaderCallbackInterface;
19 import org.opencv.android.OpenCVLoader;
20 import org.opencv.core.Mat;
21
22 public class MainActivity extends Activity implements
23     CameraBridgeViewBase.CvCameraViewListener, View.OnTouchListener {
24     private static final boolean AUTO_HIDE = true;
25     private static final int AUTO_HIDE_DELAY_MILLIS = 3000;
26     private static final int UI_ANIMATION_DELAY = 300;
27     private final Handler mHideHandler = new Handler();
28     private View mContentView;
29     private final Runnable mHidePart2Runnable = new Runnable() {
```

```

29     @SuppressLint("InlinedApi")
30     @Override
31     public void run() {
32         // Delayed removal of status and navigation bar
33
34         // Note that some of these constants are new as of API 16 (Jelly Bean)
35         // and API 19 (KitKat). It is safe to use them, as they are inlined
36         // at compile-time and do nothing on earlier devices.
37         // mContentView.setSystemUiVisibility(View.SYSTEM_UI_FLAG_LOW_PROFILE
38         // | View.SYSTEM_UI_FLAG_FULLSCREEN
39         // | View.SYSTEM_UI_FLAG_LAYOUT_STABLE
40         // | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY
41         // | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
42         // | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION);
43     }
44 };
45 private View mControlsView;
46 private final Runnable mShowPart2Runnable = new Runnable() {
47     @Override
48     public void run() {
49         // Delayed display of UI elements
50         ActionBar actionBar = getActionBar();
51         if (actionBar != null) {
52             actionBar.show();
53         }
54         mControlsView.setVisibility(View.VISIBLE);
55     }
56 };
57 private boolean mVisible;
58 private final Runnable mHideRunnable = new Runnable() {
59     @Override
60     public void run() {
61         hide();
62     }
63 };
64 private final View.OnTouchListener mDelayHideTouchListener = new
65     View.OnTouchListener() {
66     @Override
67     public boolean onTouch(View view, MotionEvent motionEvent) {
68         if (AUTO_HIDE) {
69             delayedHide(AUTO_HIDE_DELAY_MILLIS);
70         }
71         return false;
72     }
73 };
74
75 private static final String TAG = "MainActivity";
76 private CameraBridgeViewBase mOpenCvCameraView;
77 private PuzzleProcessor mPuzzleProcessor;
78 private MenuItem mItemHideNumbers;
79 private MenuItem mItemStartNewGame;
80 private int mGameWidth;
81 private int mGameHeight;
82
83 private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
84     @Override
85     public void onManagerConnected(int status) {

```

```

86         switch (status) {
87             case LoaderCallbackInterface.SUCCESS: {
88                 Log.i(TAG, "OpenCV loaded successfully");
89                 mOpenCvCameraView.setOnTouchListener(MainActivity.this);
90                 mOpenCvCameraView.enableView();
91             }
92             break;
93         default: {
94             super.onManagerConnected(status);
95         }
96         break;
97     }
98 }
99 }
100 };
101
102 @Override
103 protected void onCreate(Bundle savedInstanceState) {
104     super.onCreate(savedInstanceState);
105     getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
106     Log.d(TAG, "onCreate: Creating and setting view");
107     mOpenCvCameraView = new JavaCameraView(this, -1);
108     setContentView(mOpenCvCameraView);
109     mOpenCvCameraView.setVisibility(CameraBridgeViewBase.VISIBLE);
110     mOpenCvCameraView.setCvCameraViewListener(this);
111     mPuzzleProcessor = new PuzzleProcessor();
112     mPuzzleProcessor.prepareNewGame();
113     mVisible = true;
114     //mControlsView = findViewById(R.id.fullscreen_content_controls);
115     //mContentView = findViewById(R.id.fullscreen_content);
116     // Set up the user interaction to manually show or hide the system UI.
117     //mContentView.setOnClickListener(new View.OnClickListener() {
118     //    @Override
119     //    public void onClick(View view) {
120     //        toggle();
121     //    }
122     //});
123     // Upon interacting with UI controls, delay any scheduled hide()
124     // operations to prevent the jarring behavior of controls going away
125     // while interacting with the UI.
126     //findViewById(R.id(dummy_button).setOnTouchListener(mDelayHideTouchListener);
127 }
128
129 @Override
130 public void onPause() {
131     super.onPause();
132     if (mOpenCvCameraView != null) mOpenCvCameraView.disableView();
133 }
134
135 @Override
136 public void onResume() {
137     super.onResume();
138     if (!OpenCVLoader.initDebug()) {
139         Log.d(TAG, "onResume: Internal OpenCV library not found. Using OpenCV
140             ↪ manager for initialization");
141         OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0, this,
142             ↪ mLoaderCallback);
143     } else {

```

```
142         Log.d(TAG, "onResume: OpenCV lib found inside package. Using it!");
143         mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
144     }
145 }
146
147 @Override
148 public void onDestroy() {
149     super.onDestroy();
150     if (mOpenCvCameraView != null) mOpenCvCameraView.disableView();
151 }
152
153 @Override
154 protected void onPostCreate(Bundle savedInstanceState) {
155     super.onPostCreate(savedInstanceState);
156
157     // Trigger the initial hide() shortly after the activity has been
158     // created, to briefly hint to the user that UI controls
159     // are available.
160     delayedHide(100);
161 }
162
163 @Override
164 public boolean onCreateOptionsMenu(Menu menu) {
165     Log.i(TAG, "onCreateOptionsMenu: called");
166     mItemHideNumbers = menu.add("Show/hide the tile numbers");
167     mItemStartNewGame = menu.add("Start a new game");
168     return true;
169 }
170
171 @Override
172 public boolean onOptionsItemSelected(MenuItem item) {
173     Log.i(TAG, "onOptionsItemSelected: Menu item selected" + item);
174     if (item == mItemStartNewGame) {
175         mPuzzleProcessor.prepareNewGame();
176     } else if (item == mItemHideNumbers) {
177         mPuzzleProcessor.toggleTileNumbers();
178     }
179     return true;
180 }
181
182 public void onCameraViewStarted(int width, int height) {
183     mGameWidth = width;
184     mGameHeight = height;
185     mPuzzleProcessor.prepareGameSize(width, height);
186 }
187
188 public void onCameraViewStopped() {
189 }
190
191 public boolean onTouch(View view, MotionEvent event) {
192     int xpos, ypos;
193
194     xpos = (view.getWidth() - mGameWidth) / 2;
195     xpos = (int) event.getX() - xpos;
196
197     ypos = (view.getHeight() - mGameHeight) / 2;
198     ypos = (int) event.getY() - ypos;
199 }
```

```

200     if (xpos >= 0 && xpos <= mGameWidth && ypos >= 0 && ypos <= mGameHeight)
201         mPuzzleProcessor.deliverTouchEvent(xpos, ypos);
202
203     return false;
204 }
205
206 public Mat onCameraFrame(Mat inputFrame) {
207     return mPuzzleProcessor.puzzleFrame(inputFrame);
208 }
209
210 private void toggle() {
211     if (mVisible) {
212         hide();
213     } else {
214         show();
215     }
216 }
217
218 private void hide() {
219     // Hide UI first
220     ActionBar actionBar = getActionBar();
221     if (actionBar != null) {
222         actionBar.hide();
223     }
224     //mControlsView.setVisibility(View.GONE);
225     mVisible = false;
226
227     // Schedule a runnable to remove the status and navigation bar after a delay
228     mHideHandler.removeCallbacks(mShowPart2Runnable);
229     mHideHandler.postDelayed(mHidePart2Runnable, UI_ANIMATION_DELAY);
230 }
231
232 @SuppressLint("InlinedApi")
233 private void show() {
234     // Show the system bar
235     mContentView.setSystemUiVisibility(View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
236             | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION);
237     mVisible = true;
238
239     // Schedule a runnable to display UI elements after a delay
240     mHideHandler.removeCallbacks(mHidePart2Runnable);
241     mHideHandler.postDelayed(mShowPart2Runnable, UI_ANIMATION_DELAY);
242 }
243
244 /**
245 * Schedules a call to hide() in [delay] milliseconds, canceling any
246 * previously scheduled calls.
247 */
248 private void delayedHide(int delayMillis) {
249     mHideHandler.removeCallbacks(mHideRunnable);
250     mHideHandler.postDelayed(mHideRunnable, delayMillis);
251 }
252 }

```

```

1 package com.example.joe.a15tile;
2
3 /**

```

```

4  * Created by joe on 07/11/16.
5  */
6
7  import android.util.Log;
8
9  import org.opencv.core.*;
10 import org.opencv.imgproc.Imgproc;
11
12 public class PuzzleProcessor {
13
14     private static final int GRID_SIZE = 4;
15     private static final int GRID_AREA = GRID_SIZE * GRID_SIZE;
16     private static final int GRID_EMPTY_INDEX = GRID_AREA - 1;
17     private static final String TAG = "PuzzleProcessor";
18     private static final Scalar GRID_EMPTY_COLOR = new Scalar(0x33, 0x33, 0x33, 0xFF);
19
20     private int[] mIndexes;
21     private int[] mTextWidths;
22     private int[] mTextHeights;
23
24     private Mat mRgba15;
25     private Mat[] mCells15;
26     private boolean mShowTileNumbers = true;
27
28     public PuzzleProcessor() {
29         mTextWidths = new int[GRID_AREA];
30         mTextHeights = new int[GRID_AREA];
31         mIndexes = new int[GRID_AREA];
32
33         for (int i = 0; i < GRID_AREA; i++) mIndexes[i] = i;
34     }
35
36     public synchronized void prepareNewGame() {
37         do {
38             shuffle(mIndexes);
39         } while (!isPuzzleSolvable());
40     }
41
42     public synchronized void prepareGameSize(int width, int height) {
43         mRgba15 = new Mat(height, width, CvType.CV_8UC4);
44         mCells15 = new Mat[GRID_AREA];
45
46         for (int i = 0; i < GRID_SIZE; i++) {
47             for (int j = 0; j < GRID_SIZE; j++) {
48                 int k = i * GRID_SIZE + j;
49                 mCells15[k] = mRgba15.submat(i * height / GRID_SIZE,
50                     (i + 1) * height / GRID_SIZE,
51                     j * width / GRID_SIZE,
52                     (j + 1) * width / GRID_SIZE);
53             }
54         }
55
56         for (int i = 0; i < GRID_AREA; i++) {
57             Size s = Imgproc.getTextSize(Integer.toString(i + 1), 3, 1, 2, null);
58             mTextHeights[i] = (int) s.height;
59             mTextWidths[i] = (int) s.width;
60         }
61     }

```

```

62
63     public synchronized Mat puzzleFrame(Mat inputPicture) {
64         Mat[] cells = new Mat[GRID_AREA];
65         int rows = inputPicture.rows();
66         int cols = inputPicture.cols();
67
68         rows = rows - rows % 4;
69         cols = cols - cols % 4;
70
71         for (int i = 0; i < GRID_SIZE; i++) {
72             for (int j = 0; j < GRID_SIZE; j++) {
73                 int k = i * GRID_SIZE + j;
74                 cells[k] = inputPicture.submat(i * inputPicture.rows() / GRID_SIZE,
75                     (i + 1) * inputPicture.rows() / GRID_SIZE,
76                     j * inputPicture.cols() / GRID_SIZE,
77                     (j + 1) * inputPicture.cols() / GRID_SIZE);
78             }
79         }
80
81         rows = rows - rows % 4;
82         cols = cols - cols % 4;
83
84         for (int i = 0; i < GRID_AREA; i++) {
85             int idx = mIndexes[i];
86             if (idx == GRID_EMPTY_INDEX)
87                 mCells15[i].setTo(GRID_EMPTY_COLOR);
88             else {
89                 cells[idx].copyTo(mCells15[i]);
90                 if (mShowTileNumbers) {
91                     Imgproc.putText(mCells15[i],
92                         Integer.toString(1 + idx),
93                         new Point((cols / GRID_SIZE - mTextWidths[idx]) / 2,
94                             (rows / GRID_SIZE + mTextHeights[idx]) / 2),
95                         3,
96                         1,
97                         new Scalar(255, 0, 0, 255),
98                         2);
99                 }
100            }
101        }
102
103        for (int i = 0; i < GRID_AREA; i++) cells[i].release();
104
105        drawGrid(cols, rows, mRgba15);
106
107        return mRgba15;
108    }
109
110    public void toggleTileNumbers() {
111        mShowTileNumbers = !mShowTileNumbers;
112    }
113
114    public void deliverTouchEvent(int x, int y) {
115        int rows = mRgba15.rows();
116        int cols = mRgba15.cols();
117
118        int row = (int) Math.floor(y * GRID_SIZE / rows);
119        int col = (int) Math.floor(x * GRID_SIZE / cols);

```

```

120
121     if (row < 0 || row >= GRID_SIZE || col < 0 || col >= GRID_SIZE) {
122         Log.e(TAG, "deliverTouchEvent: Touch event outside of image not expected");
123         return;
124     }
125
126     int idx = row * GRID_SIZE + col;
127     int idxToSwap = -1;
128
129     if (idxToSwap < 0 && col > 0)
130         if (mIndexes[idx - 1] == GRID_EMPTY_INDEX)
131             idxToSwap = idx - 1;
132     if (idxToSwap < 0 && col < GRID_SIZE - 1)
133         if (mIndexes[idx + 1] == GRID_EMPTY_INDEX)
134             idxToSwap = idx + 1;
135     if (idxToSwap < 0 && row > 0)
136         if (mIndexes[idx - GRID_SIZE] == GRID_EMPTY_INDEX)
137             idxToSwap = idx - GRID_SIZE;
138     if (idxToSwap < 0 && row < GRID_SIZE - 1)
139         if (mIndexes[idx + GRID_SIZE] == GRID_EMPTY_INDEX)
140             idxToSwap = idx + GRID_SIZE;
141
142     if (idxToSwap >= 0) {
143         synchronized (this) {
144             int touched = mIndexes[idx];
145             mIndexes[idx] = mIndexes[idxToSwap];
146             mIndexes[idxToSwap] = touched;
147         }
148     }
149 }
150
151 private void drawGrid(int cols, int rows, Mat drawMat) {
152     for (int i = 1; i < GRID_SIZE; i++) {
153         Imgproc.line(drawMat,
154                     new Point(0, i * rows / GRID_SIZE),
155                     new Point(cols, i * rows / GRID_SIZE),
156                     new Scalar(0, 255, 0, 255),
157                     3);
158         Imgproc.line(drawMat,
159                     new Point(i * cols / GRID_SIZE, rows),
160                     new Point(i * cols / GRID_SIZE, rows),
161                     new Scalar(0, 255, 0, 255),
162                     3);
163     }
164 }
165
166 private static void shuffle(int[] array) {
167     for (int i = array.length; i > 1; i--) {
168         int temp = array[i - 1];
169         int randIx = (int) (Math.random() * i);
170         array[i - 1] = array[randIx];
171         array[randIx] = temp;
172     }
173 }
174
175 private boolean isPuzzleSolvable() {
176     int sum = 0;
177     for (int i = 0; i < GRID_AREA; i++) {

```

```

178         if (mIndexes[i] == GRID_EMPTY_INDEX) sum += (i / GRID_SIZE) + 1;
179     else {
180         int smaller = 0;
181         for (int j = i+1; j < GRID_AREA; j++){
182             if (mIndexes[j] < mIndexes[i]) smaller++;
183         }
184         sum += smaller;
185     }
186 }
187 return sum % 2 == 0;
188 }
189
190 }
```

A.4 BLOB Analysis

```

1 package com.example.joe.blob_analysis;
2
3 import android.app.Activity;
4 import android.os.Bundle;
5 import android.util.Log;
6 import android.view.MotionEvent;
7 import android.view.SurfaceView;
8 import android.view.View;
9 import android.view.Window;
10 import android.view.WindowManager;
11
12 import org.opencv.android.BaseLoaderCallback;
13 import org.opencv.android.CameraBridgeViewBase;
14 import org.opencv.android.LoaderCallbackInterface;
15 import org.opencv.android.OpenCVLoader;
16 import org.opencv.core.Core;
17 import org.opencv.core.CvType;
18 import org.opencv.core.Mat;
19 import org.opencv.core.MatOfPoint;
20 import org.opencv.core.Rect;
21 import org.opencv.core.Scalar;
22 import org.opencv.core.Size;
23 import org.opencv.imgproc.Imgproc;
24
25 import java.util.List;
26
27 /**
28 * Created by joe on 12/11/16.
29 */
30
31 public class ColorBlobDetectionActivity extends Activity implements
32     View.OnTouchListener, CameraBridgeViewBase.CvCameraViewListener2 {
33     private static final String TAG = "Activity: ";
34
35     private boolean mIsColorSelected = false;
36     private Mat mRgba;
37     private Scalar mBlobColorRgba;
38     private Scalar mBlobColorHsv;
39     private ColorBlobDetector mDetector;
        private Mat mSpectrum;
```

```

40     private Size SPECTRUM_SIZE;
41     private Scalar CONTOUR_COLOR;
42
43     private CameraBridgeViewBase mOpenCvCameraView;
44
45     private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
46         @Override
47         public void onManagerConnected(int status) {
48             switch (status) {
49                 case LoaderCallbackInterface.SUCCESS: {
50                     Log.i(TAG, "onManagerConnected: OpenCV Loaded");
51                     mOpenCvCameraView.enableView();
52                     mOpenCvCameraView.setOnTouchListener(ColorBlobDetectionActivity.this);
53                 }
54                 break;
55             default: {
56                 super.onManagerConnected(status);
57             }
58             break;
59         }
60     };
61
62
63     public ColorBlobDetectionActivity() {
64         Log.i(TAG, "ColorBlobDetectionActivity: Instantiated new");
65     }
66
67     @Override
68     public void onCreate(Bundle savedInstanceState) {
69         Log.i(TAG, "onCreate: Called");
70         super.onCreate(savedInstanceState);
71         requestWindowFeature(Window.FEATURE_NO_TITLE);
72         getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
73         setContentView(R.layout.color_blob_detection_surface_view);
74         mOpenCvCameraView = (CameraBridgeViewBase)
75             findViewById(R.id.color_blob_detection_activity_surface_view);
76         mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);
77         mOpenCvCameraView.setCvCameraViewListener(this);
78     }
79
80     @Override
81     public void onPause() {
82         super.onPause();
83         if (mOpenCvCameraView != null) {
84             mOpenCvCameraView.disableView();
85         }
86     }
87
88     @Override
89     public void onResume() {
90         super.onResume();
91         if (!OpenCVLoader.initDebug()) {
92             Log.d(TAG, "onResume: OpenCV not found. Using Manager");
93             OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0, this,
94                 mLoaderCallback);
95         } else {
96             Log.d(TAG, "onResume: OpenCV found in package.");
97             mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
98         }
99     }
100 }
```

```

96
97    }
98
99    public void onDestroy() {
100        super.onDestroy();
101        if (mOpenCvCameraView != null) {
102            mOpenCvCameraView.disableView();
103        }
104    }
105
106    public void onCameraViewStarted(int width, int height) {
107        mRgba = new Mat(height, width, CvType.CV_8UC4);
108        mDetector = new ColorBlobDetector();
109        mSpectrum = new Mat();
110        mBlobColorRgba = new Scalar(255);
111        mBlobColorHsv = new Scalar(255);
112        SPECTRUM_SIZE = new Size(200, 64);
113        CONTOUR_COLOR = new Scalar(255, 0, 0, 255);
114    }
115
116    public void onCameraViewStopped() {
117        mRgba.release();
118    }
119
120    public boolean onTouch(View v, MotionEvent event){
121        int cols = mRgba.cols();
122        int rows = mRgba.rows();
123
124        int xOffset = (mOpenCvCameraView.getWidth() - cols) / 2;
125        int yOffset = (mOpenCvCameraView.getHeight() - rows) / 2;
126
127        int x = (int)event.getX() - xOffset;
128        int y = (int) event.getY() - yOffset;
129
130        Log.i(TAG, "onTouch: Touch coordinates: (" + x + ", " + y + ")");
131
132        if ((x < 0) || (y < 0) || (x > cols) || (y > rows)) return false;
133
134        Rect touchedRect = new Rect();
135
136        touchedRect.x = (x > 4) ? x - 4 : 0;
137        touchedRect.y = (y > 4) ? y - 4 : 0;
138
139        touchedRect.width = (x + 4 < cols) ? x + 4 - touchedRect.x : cols -
140            ↳   touchedRect.x;
140        touchedRect.height = (y + 4 < rows) ? y + 4 - touchedRect.y : rows -
141            ↳   touchedRect.y;
141
142        Mat touchedRegionRgba = mRgba.submat(touchedRect);
143
144        Mat touchedRegionHsv = new Mat();
145        Imgproc.cvtColor(touchedRegionRgba, touchedRegionHsv,
146                         ↳   Imgproc.COLOR_RGB2HSV_FULL);
146
147        mBlobColorHsv = Core.sumElems(touchedRegionHsv);
148        int pointCount = touchedRect.width * touchedRect.height;
149        for (int i = 0; i < mBlobColorHsv.val.length; i++) {
150            mBlobColorHsv.val[i] /= pointCount;

```

```

151     }
152
153     mBlobColorRgba = convertScalarHsv2Rgba(mBlobColorHsv);
154
155     Log.i(TAG, "onTouch: Touched rgba color: (" + mBlobColorRgba.val[0] + ", " +
156           ↵ mBlobColorRgba.val[1] + ", " + mBlobColorRgba.val[2] + ", " +
157           ↵ mBlobColorRgba.val[3] + ")");
158
159     mDetector.setHsvColor(mBlobColorHsv);
160
161     Imgproc.resize(mDetector.getSpectrum(), mSpectrum, SPECTRUM_SIZE);
162     mIsColorSelected = true;
163
164     touchedRegionRgba.release();
165     touchedRegionHsv.release();
166
167     return false;
168 }
169
170 public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
171     mRgba = inputFrame.rgba();
172     if (mIsColorSelected) {
173         mDetector.process(mRgba);
174         List<MatOfPoint> contours = mDetector.getContours();
175         Log.e(TAG, "onCameraFrame: Contours count: " + contours.size());
176         Imgproc.drawContours(mRgba, contours, -1, CONTOUR_COLOR);
177         Mat colorLabel = mRgba.submat(4, 68, 4, 68);
178         colorLabel.setTo(mBlobColorRgba);
179         Mat spectrumLabel = mRgba.submat(4, 4 + mSpectrum.rows(), 70, 70 +
180             ↵ mSpectrum.cols());
181         mSpectrum.copyTo(spectrumLabel);
182     }
183
184     return mRgba;
185 }
186
187 private Scalar convertScalarHsv2Rgba(Scalar hsvColor) {
188     Mat pointMatRgba = new Mat();
189     Mat pointMatHsv = new Mat(1, 1, CvType.CV_8UC3, hsvColor);
190     Imgproc.cvtColor(pointMatHsv, pointMatRgba, Imgproc.COLOR_HSV2RGB_FULL, 4);
191     return new Scalar(pointMatRgba.get(0, 0));
192 }

```

```

1 package com.example.joe.blob_analysis;
2
3 import org.opencv.core.Core;
4 import org.opencv.core.CvType;
5 import org.opencv.core.Mat;
6 import org.opencv.core.MatOfPoint;
7 import org.opencv.core.Scalar;
8 import org.opencv.imgproc.Imgproc;
9
10 import java.util.ArrayList;
11 import java.util.Iterator;
12 import java.util.List;

```

```

13
14 /**
15 * Created by joe on 12/11/16.
16 */
17
18 public class ColorBlobDetector {
19
20     private Scalar mLowerBound = new Scalar(0);
21     private Scalar mUpperBound = new Scalar(0);
22
23     private static double mMinContourArea = 0.1;
24
25     private Scalar mColorRadius = new Scalar(25, 50, 50, 0);
26     private Mat mSpectrum = new Mat();
27     private List<MatOfPoint> mContours = new ArrayList<MatOfPoint>();
28
29     Mat mPyrDownMat = new Mat();
30     Mat mHsvMat = new Mat();
31     Mat mMask = new Mat();
32     Mat mDilatedMask = new Mat();
33     Mat mHierarchy = new Mat();
34
35     public void setColorRadius(Scalar radius) {
36         mColorRadius = radius;
37     }
38
39     public void setHsvColor(Scalar hsvColor) {
40         double minH = (hsvColor.val[0] >= mColorRadius.val[0]) ? hsvColor.val[0] -
41             → mColorRadius.val[0] : 0;
42         double maxH = (hsvColor.val[0] + mColorRadius.val[0] <= 255) ? hsvColor.val[0]
43             → + mColorRadius.val[0] : 255;
44
45         mLowerBound.val[0] = minH;
46         mUpperBound.val[0] = maxH;
47
48         mLowerBound.val[1] = hsvColor.val[1] - mColorRadius.val[1];
49         mUpperBound.val[1] = hsvColor.val[1] + mColorRadius.val[1];
50
51         mLowerBound.val[2] = hsvColor.val[2] - mColorRadius.val[2];
52         mUpperBound.val[2] = hsvColor.val[2] + mColorRadius.val[2];
53
54         mLowerBound.val[3] = 0;
55         mUpperBound.val[3] = 255;
56
57         Mat spectrumHsv = new Mat(1, (int)(maxH-minH), CvType.CV_8UC3);
58
59         for (int j = 0; j < maxH-minH; j++) {
60             byte[] tmp = {(byte)(minH+j), (byte)255, (byte)255};
61             spectrumHsv.put(0, j, tmp);
62         }
63
64         Imgproc.cvtColor(spectrumHsv, mSpectrum, Imgproc.COLOR_HSV2RGB_FULL);
65     }
66
67     public Mat getSpectrum() {
68         return mSpectrum;
69     }

```

```

69     public void setMinContourArea(double area){mMinContourArea = area;}
70
71     public void process(Mat rgbaImage){
72         Imgproc.pyrDown(rgbaImage, mPyrDownMat);
73         Imgproc.pyrDown(mPyrDownMat, mPyrDownMat);
74
75         Imgproc.cvtColor(mPyrDownMat, mHsvMat, Imgproc.COLOR_RGB2HSV_FULL);
76
77         Core.inRange(mHsvMat, mLowerBound, mUpperBound, mMask);
78         Imgproc.dilate(mMask, mDilatedMask, new Mat());
79
80         List<MatOfPoint> contours = new ArrayList<MatOfPoint>();
81
82         Imgproc.findContours(mDilatedMask, contours, mHierarchy,
83             → Imgproc.RETR_EXTERNAL, Imgproc.CHAIN_APPROX_SIMPLE);
84
85         double maxArea = 0;
86         Iterator<MatOfPoint> each = contours.iterator();
87         while (each.hasNext()) {
88             MatOfPoint wrapper = each.next();
89             double area = Imgproc.contourArea(wrapper);
90             if (area > maxArea) {
91                 maxArea = area;
92             }
93         }
94
95         mContours.clear();
96         each = contours.iterator();
97         while (each.hasNext()) {
98             MatOfPoint contour = each.next();
99             if (Imgproc.contourArea(contour) > mMinContourArea*maxArea) {
100                 Core.multiply(contour, new Scalar(4,4), contour);
101                 mContours.add(contour);
102             }
103         }
104
105     public List<MatOfPoint> getContours(){return mContours;}
106
107 }
```

A.5 Face Detection

```

1 package com.example.joe.face_recognition;
2
3 import org.opencv.core.Mat;
4 import org.opencv.core.MatOfRect;
5
6 /**
7  * Created by joe on 08/11/16.
8 */
9
10 public class DetectionBasedTracker {
11     public DetectionBasedTracker(String cascadeName, int minFaceSize) {
12         mNativeObj = nativeCreateObject(cascadeName, minFaceSize);
13     }

```

```

14
15     public void start() {
16         nativeStart(mNativeObj);
17     }
18
19     public void stop() {
20         nativeStop(mNativeObj);
21     }
22
23     public void setMinFaceSize(int size) {
24         nativeSetFaceSize(mNativeObj, size);
25     }
26
27     public void detect(Mat imageGray, MatOfRect faces) {
28         nativeDetect(mNativeObj, imageGray.getNativeObjAddr(),
29                     faces.getNativeObjAddr());
30     }
31
32     public void release() {
33         nativeDestroyObject(mNativeObj);
34         mNativeObj = 0;
35     }
36
37     private long mNativeObj = 0;
38
39     private static native long nativeCreateObject(String cascadeName, int minFaceSize);
40     private static native void nativeDestroyObject(long thiz);
41     private static native void nativeStart(long thiz);
42     private static native void nativeStop(long thiz);
43     private static native void nativeSetFaceSize(long thiz, int size);
44     private static native void nativeDetect(long thiz, long inputImage, long faces);
}

```

```

1 package com.example.joe.face_recognition;
2
3 import android.app.Activity;
4 import android.content.Context;
5 import android.os.Bundle;
6 import android.util.Log;
7 import android.view.Menu;
8 import android.view.MenuItem;
9 import android.view.WindowManager;
10
11 import org.opencv.android.BaseLoaderCallback;
12 import org.opencv.android.CameraBridgeViewBase;
13 import org.opencv.android.LoaderCallbackInterface;
14 import org.opencv.android.OpenCVLoader;
15 import org.opencv.core.Mat;
16 import org.opencv.core.MatOfRect;
17 import org.opencv.core.Rect;
18 import org.opencv.core.Scalar;
19 import org.opencv.core.Size;
20 import org.opencv.imgproc.Imgproc;
21 import org.opencv.objdetect.CascadeClassifier;
22
23 import java.io.File;
24 import java.io.FileOutputStream;

```

```

25 import java.io.IOException;
26 import java.io.InputStream;
27
28 /**
29 * Created by joe on 08/11/16.
30 */
31
32 public class FrActivity extends Activity implements
33     CameraBridgeViewBase.CvCameraViewListener2 {
34
35     private static final String TAG             = "FrActivity";
36     private static final Scalar FACE_RECT_COLOR = new Scalar(0,255,0,255);
37     private static final int    JAVA_DETECTOR   = 0;
38     private static final int    NATIVE_DETECTOR = 1;
39
40     private MenuItem mItemFace50;
41     private MenuItem mItemFace40;
42     private MenuItem mItemFace30;
43     private MenuItem mItemFace20;
44     private MenuItem mItemType;
45
46     private Mat          mRgba;
47     private Mat          mGray;
48     private File         mCascadeFile;
49     private CascadeClassifier mJavaDetector;
50     private DetectionBasedTracker mNativeDetector;
51
52     private int          mDetectorType = JAVA_DETECTOR;
53     private String[]     mDetectorName;
54
55     private float        mRelativeFaceSize = 0.2f;
56     private int          mAbsoluteFaceSize = 0;
57
58     private CameraBridgeViewBase mOpenCvCameraView;
59
60     private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
61         @Override
62         public void onManagerConnected(int status) {
63             switch (status){
64                 case LoaderCallbackInterface.SUCCESS:{
65                     Log.i(TAG, "onManagerConnected: OpenCV loaded correctly");
66                     System.loadLibrary("detection_based_tracker");
67                     try{
68                         InputStream is =
69                             getResources().openRawResource(R.raw.lbpcascade_frontalface);
70                         File cascadeDir = getDir("cascade", Context.MODE_PRIVATE);
71                         mCascadeFile = new File(cascadeDir,
72                             "lbpcascade_frontalface.xml");
73                         FileOutputStream os = new FileOutputStream(mCascadeFile);
74
75                         byte[] buffer = new byte[4096];
76                         int bytesRead;
77                         while ((bytesRead = is.read(buffer)) != -1){
78                             os.write(buffer, 0, bytesRead);
79                         }
80                         is.close();
81                         os.close();
82                     }
83                 }
84             }
85         }
86     }

```

```

80             mJavaDetector = new
81                 ↪ CascadeClassifier(mCascadeFile.getAbsolutePath());
82             if (mJavaDetector.empty()){
83                 Log.e(TAG, "onManagerConnected: Failed to load cascade
84                     ↪ classifier");
85                 mJavaDetector = null;
86             }else{
87                 Log.i(TAG, "onManagerConnected: Loaded cascade classifier
88                     ↪ from "+mCascadeFile.getAbsolutePath());
89             }
90             mNativeDetector = new
91                 ↪ DetectionBasedTracker(mCascadeFile.getAbsolutePath(), 0);
92             cascadeDir.delete();
93         }catch (IOException e){
94             e.printStackTrace();
95             Log.e(TAG, "onManagerConnected: Failed to load cascade.
96                     ↪ Exception thrown: "+e);
97         }
98     }
99 }
100 }
101 };
102
103 public FrActivity(){
104     mDetectorName = new String[2];
105     mDetectorName[JAVA_DETECTOR] = "Java";
106     mDetectorName[NATIVE_DETECTOR] = "Native (tracking)";
107     Log.i(TAG, "FrActivity: Instantiated new" + this.getClass());
108 }
109
110 @Override
111 public void onCreate(Bundle savedInstanceState){
112     Log.i(TAG, "onCreate: Called");
113     super.onCreate(savedInstanceState);
114     getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
115     setContentView(R.layout.face_detect_surface_view);
116     mOpenCvCameraView = (CameraBridgeViewBase)
117         ↪ findViewById(R.id.fd_activity_surface_view);
118     mOpenCvCameraView.setVisibility(CameraBridgeViewBase.VISIBLE);
119     mOpenCvCameraView.setCvCameraViewListener(this);
120 }
121
122 @Override
123 public void onPause(){
124     super.onPause();
125     if (mOpenCvCameraView != null) mOpenCvCameraView.disableView();
126 }
127
128 @Override
129 public void onResume(){
130     super.onResume();
131     if (!OpenCVLoader.initDebug()){

```

```

131         Log.d(TAG, "onResume: Internal OpenCV lib not found. Using OpenCV manager
132             for initialization");
133
134             → OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0, this, mLoaderCallback);
135     }else{
136         Log.d(TAG, "onResume: OpenCV lib found inside package.");
137         mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
138     }
139
140     public void onDestroy(){
141         super.onDestroy();
142         mOpenCvCameraView.disableView();
143     }
144
145     public void onCameraViewStarted(int width, int height){
146         mGray = new Mat();
147         mRgba = new Mat();
148     }
149
150     public void onCameraViewStopped(){
151         mGray.release();
152         mRgba.release();
153     }
154
155     public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame){
156         mRgba = inputFrame.rgba();
157         mGray = inputFrame.gray();
158
159         if (mAbsoluteFaceSize == 0){
160             int height = mGray.rows();
161             if (Math.round(height * mRelativeFaceSize)>0){
162                 mAbsoluteFaceSize = Math.round(height * mRelativeFaceSize);
163             }
164             mNativeDetector.setMinFaceSize(mAbsoluteFaceSize);
165         }
166
167         MatOfRect faces = new MatOfRect();
168
169         if (mDetectorType == JAVA_DETECTOR){
170             if (mJavaDetector != null){
171                 mJavaDetector.detectMultiScale(mGray,
172                     faces,
173                     1.1,
174                     2,
175                     new Size(mAbsoluteFaceSize, mAbsoluteFaceSize),
176                     new Size());
177             }
178         }else if (mDetectorType == NATIVE_DETECTOR){
179             if (mNativeDetector != null){
180                 mNativeDetector.detect(mGray, faces);
181             }
182         }else{
183             Log.e(TAG, "onCameraFrame: Detection method is not selected");
184         }
185
186         Rect[] facesArray = faces.toArray();

```

```
187     for (Rect aFacesArray : facesArray)
188         Imgproc.rectangle(mRgba, aFacesArray.tl(), aFacesArray.br(),
189                           FACE_RECT_COLOR, 3);
190
191     return mRgba;
192 }
193
194 @Override
195 public boolean onCreateOptionsMenu(Menu menu){
196     Log.i(TAG, "onCreateOptionsMenu: called");
197     mItemFace50 = menu.add("Face size 50%");
198     mItemFace40 = menu.add("Face size 40%");
199     mItemFace30 = menu.add("Face size 30%");
200     mItemFace20 = menu.add("Face size 20%");
201     mItemType = menu.add(mDetectorName[mDetectorType]);
202     return true;
203 }
204
205 private void setMinFaceSize(float faceSize){
206     mRelativeFaceSize = faceSize;
207     mAbsoluteFaceSize = 0;
208 }
209
210 private void setDetectorType(int type){
211     if (mDetectorType != type){
212         mDetectorType = type;
213         if (type == NATIVE_DETECTOR){
214             Log.i(TAG, "setDetectorType: Detection based tracker enabled");
215             mNativeDetector.start();
216         }else{
217             Log.i(TAG, "setDetectorType: Cascade detector enabled");
218             mNativeDetector.stop();
219         }
220     }
221 }
```

B Python Experiments

B.1 Table of Python Experiments

Experiment	Purpose	Functional	Issues	Files	LOC
Find Game	<ul style="list-style-type: none">Identify red game cartridge out of 3Displays boundary around correct part of image	Yes	N/A	1	18
Threshold Methods	<ul style="list-style-type: none">Demonstrates various thresholding methods on an imageOriginal image text unreadable but clear after techniques are applied	Yes	N/A	1	14
Image Operations	<ul style="list-style-type: none">Moves parts of an image to other locations using arraysDemonstrates how pixel data is stored	Yes	N/A	1	15
Distance to Camera	<ul style="list-style-type: none">Calculates distance to the camera from an identified objectDisplays various distances using 3 images	Yes	N/A	1	38

B.2 find_game.py

```

1 import numpy as np
2 import cv2
3
4 image = cv2.imread('games.jpg')
5
6 upper = np.array([65, 65, 255])
7 lower = np.array([0, 0, 200])
8 mask = cv2.inRange(image, lower, upper)
9
10 cnts, _ = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
11 c = max(cnts, key=cv2.contourArea)
12
13 peri = cv2.arcLength(c, True)
14 approx = cv2.approxPolyDP(c, 0.05 * peri, True)
15
16 cv2.drawContours(image, [approx], -1, (0,255,0), 4)
17 cv2.imshow('Image', image)
18 cv2.waitKey(0)

```

B.3 thresholding.py

```

1 import cv2
2 import numpy as np
3
4 img = cv2.imread('bookpage.jpg')
5 grayscaled = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
6 _, threshold = cv2.threshold(img, 12, 255, cv2.THRESH_BINARY)
7 _, gs_threshold = cv2.threshold(grayscaled, 10, 255, cv2.THRESH_BINARY)
8 adaptive = cv2.adaptiveThreshold(grayscaled, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
9     cv2.THRESH_BINARY, 115, 1)
10 cv2.imshow('original', img)
11 cv2.imshow('threshold', threshold)
12 cv2.imshow('grayscale', gs_threshold)
13 cv2.imshow('adaptive', adaptive)
14 cv2.waitKey(0)
15 cv2.destroyAllWindows()

```

B.4 img_ops.py

```

1 import cv2
2 import numpy as np
3
4 img = cv2.imread('watch.jpg', cv2.IMREAD_COLOR)
5 px = img[55, 55]
6 img[55, 55] = [255, 255, 255]
7 px = img[55, 55]
8 print(px)
9 px = img[100:150, 100:150]
10 print(px)
11
12 watch_face = img[37:111, 107:194]

```

```

13 img[0:74, 0:87] = watch_face
14 cv2.imshow('image', img)
15 cv2.waitKey(0)
16 cv2.destroyAllWindows()

```

B.5 distance_to_camera.py

```

1 import numpy as np
2 import cv2
3
4
5 def find_marker(image):
6     gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
7     gray = cv2.GaussianBlur(gray, (5, 5), 0)
8     edged = cv2.Canny(gray, 35, 125)
9
10    (cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
11    c = max(cnts, key=cv2.contourArea)
12
13    return cv2.minAreaRect(c)
14
15
16 def distance_to_camera(knownWidth, focalLength, perWidth):
17     return (knownWidth * focalLength) / perWidth
18
19
20 KNOWN_DISTANCE = 24.0
21 KNOWN_WIDTH = 11.0
22 IMAGE_PATHS = ['2ft.png', '3ft.png', '4ft.png']
23
24 image = cv2.imread(IMAGE_PATHS[0])
25 marker = find_marker(image)
26 focalLength = (marker[1][0] * KNOWN_DISTANCE) / KNOWN_WIDTH
27
28 for imagePath in IMAGE_PATHS:
29     image = cv2.imread(imagePath)
30     marker = find_marker(image)
31     inches = distance_to_camera(KNOWN_WIDTH, focalLength, marker[1][0])
32
33     box = np.int0(cv2.cv.BoxPoints(marker))
34     cv2.drawContours(image, [box], -1, (0, 255, 0), 2)
35     cv2.putText(image, '%.2fft' % (inches/12),
36                 (image.shape[1] - 200, image.shape[0] - 20), cv2.FONT_HERSHEY_SIMPLEX,
37                 2.0, (0, 255, 0), 3)
38     cv2.imshow('image', image)
39     cv2.waitKey(0)

```

C Development Log

D EXIF Extraction Code

```

36 def get_sensor_size(exif):
37     # (Resolution in pixels / Focal plane resolution in dpi) X 25.4(mm / in) = size in
38     # → mm
39     # Do for hor and ver
40     horizontal_measurement = None
41     vertical_measurement = None
42
43 def get_focal_length(exif):
44     """
45
46     :param exif:
47     :return:
48     """
49     measurement, divisor = exif['FocalLength']
50     return float(measurement)/float(divisor)
51
52
53 def get_exif(image_path):
54     """
55
56     :param image_path:
57     :return:
58     """
59     image = PIL.Image.open(image_path)
60     exif = {PIL.ExifTags.TAGS[k]: v
61             for k, v in image._getexif().items()
62             if k in PIL.ExifTags.TAGS}
63
64     return exif

```

E Testing Results

F Commit Log

G Photos

To do...

- 1 (p. i): Write abstract
- 2 (p. 2): Revise aims and objectives
- 3 (p. 2): Write project restrictions section
- 4 (p. 3): Finish structure section
- 5 (p. 4): Fix paragraph because it's awful
- 6 (p. 6): cite
- 7 (p. 6): cite
- 8 (p. 6): cite?
- 9 (p. 11): complete hough transform section
- 10 (p. 12): What is going on here?
- 11 (p. 14): revise
- 12 (p. 15): change to suit
- 13 (p. 15): cite
- 14 (p. 15): cite
- 15 (p. 16): cite
- 16 (p. 16): cite
- 17 (p. 16): cite?
- 18 (p. 17): cite
- 19 (p. 20): Remove statements saying this will be used
- 20 (p. 20): cite
- 21 (p. 20): cite
- 22 (p. 20): cite
- 23 (p. 21): cite
- 24 (p. 21): cite

- 25 (p. 21): revise in light of dev methods
- 26 (p. 22): Weird
- 27 (p. 22): segway
- 28 (p. 29): cite