# 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

10/04/2017

# 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

With some mountain bike suspension units costing upwards of £1000, it is vital that they are correctly set up to the rider's weight and riding style. An improper setup can result in damage to the suspension unit and/or injury to the rider. The most basic setting which should be set for every rider is sag, this is the amount the suspension sits into its travel under normal load. Calculated from a percentage of the shock's stroke, setting sag requires knowledge of suspension and repeated measurements which beginner mountain bikers may not know about.

Image analysis techniques are capable of measuring objects in images computationally without interfering with the subject. By researching the concepts of mountain bike suspension along with various image analysis techniques and their uses, this project devised an application which could calculate a sag setting for rear suspension with minimal knowledge and input from the rider.

The application was written in Python as it allows for simple use of the chosen imaging library, OpenCV. By supplying two images of the rear shock at different pressures, the desired sag percentage, and the shock's stroke length this application is capable of outputting a pressure setting for a baseline setup. By using various evaluative techniques it was proven that the application functions correctly, the results are reliable, and it would be well received by the industry. This application also presents the groundwork for additional projects to extend the functionality and migrate to a mobile platform.

# Contents

# List of Tables

# List of Figures

# List of Listings

# Acknowledgements

I would like to thank my supervisor, Brian Davison, for his invaluable aid, insight, suggestions, and support. His vital knowledge of physics and the academic process revitalised this project in times when it was running thin.

Thanks to Professor Geraint Florida-James of the Mountain Bike Centre of Scotland for his support and time taken to arrange the evaluation meeting covered in section 4.3 of this document. Additionally to Mark Ravilious for agreeing to the evaluation meeting and providing his feedback.

Thanks to all my colleagues at Napier for their suggestions and criticism. I hope I have provided as much value to their work as they have to mine.

Thanks to Michaela Scott, for providing the second bike for testing whenever it was needed, and to James and Ryan of Pedals bike shop for their time and suggestions.

Finally special thanks to my parents, for their unrelenting support not only during this project but throughout my university career. I would not be where I am today without them.

# 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 set up to function correctly. The objective of this dissertation is to examine how image analysis can be used in the setting up of mountain bike suspension, and produce a prototype application that helps beginner and intermediate riders to carry out an initial set up.

## 1.2 Background

A survey carried out by the International Mountain Bike Association (IMBA) found that the average price paid for a mountain bike was €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 only have front suspension units; this difference can be seen in Figure 1. The price of higher specification mountain bikes can run to many thousands of pounds, but even entry level models are equipped with adjustable suspension units that need to be correctly configured to the rider's weight and the characteristics of the terrain.

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

It has been proven that using a FS bike as opposed to a HT gives the rider a measurable performance advantage (Titlestad, Fairlie-Clarke, Davie, Whittaker, & Grant, 2003). However, if the suspension fork and/or rear shock have not been correctly configured it can be detrimental to the rider's performance and potentially lead to injury. For example, if a shock has too little rebound damping applied 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. However many enthusiasts lack the specialist knowledge required to optimize the setup of their bike's suspension units and as a consequence compromise their performance and riding experience.

Additionally, an incorrect suspension setup can cause excessive wear and tear on the

bike's frame and components. For instance, suspension which is set too soft will cause "bottoming out" where the unit reaches the end of its available range so that excess forces are transferred to the frame. This causes stress and potentially dramatic fracture that could result in injury. Conversely, suspension that is set too hard forces energy, that would normally be absorbed, into the wheels resulting in denting and warping of the rims.

Many bicycle retailers will configure the suspension of a newly purchased mountain bike for the customer at the time of collection. Most of the time this will be enough to avoid incident but as the customer is unlikely to be wearing weighty protection equipment and accessories such as helmet, body armour and hydration pack, this initial setup is rarely accurate. Furthermore, with some manufacturers choosing direct sales over supplying to local retailers (Harker, 2010; Staff, 2015), even this rudimentary setup can be omitted altogether.

Since the birth of the modern smartphone in 2007, brought about 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 many individuals like to use their smartphones to aid or augment their performance and enjoyment of their chosen hobby or sport. On the back of this increase, a number of companies have launched small devices (Aston, 2016; Hwang, 2016) and mobile applications (Benedict, 2012) designed to help riders set up and fine tune their own suspension, either at home or while out on a ride. Each of the devices produced cost more than £100 and require the suspension to have an initial setup in place and the only mobile application capable of producing an initial setup is tied to a single suspension manufacturer.

## 1.3 Aims and Objectives

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 how image analysis is currently used in sports science
- Implement the prototype application using identified and researched methods
- Evaluate the success and appropriateness of the produced application
- Present conclusions about the project's successes and downfalls

## 1.4 Dissertation Structure

This dissertation will document the project by taking the following structure:

1. **Introduction** - Outlines the context of the project and states the major aims and objectives
2. **Literature Review** - Presents a literature review to further describe the project context and provide information on mountain bike suspension concepts and image analysis techniques
3. **Methodology** - Discusses the methodologies available which could be used to complete the project and their advantages
4. **Results** - Describes the outcome of the project including any deviations from the plan, the operation of the application, and a critical evaluation of the application
5. **Conclusion** - Presents conclusions and learnings from completion of the project including potential future work and a self appraisal

# 2 Literature Review

The aim of this project is to produce an application that makes use of image analysis techniques to provide the user with a suggested setup for their suspension units. This literature review will identify the intricacies involved in correctly configuring a rear suspension unit and how these make it difficult for riders to do it themselves. It will then go on to highlight how image analysis is carried out and identify the various techniques that are relevant to this project. This review will shed light on the problem at hand and help identify a viable solution.

## 2.1 Mountain Bike Suspension Concepts

The purpose of suspension on a mountain bike is to absorb the energy created from riding over features, such as bumps and rough terrain encountered along a trail, improving comfort for the rider and allowing them to go faster by maintaining better 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 on 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 an upward direction while stroke is the distance that the shock absorber can compress before it bottoms out. Travel is measured in millimetres or inches and can range from 80mm to 210mm or 4in to 9in. Bikes designed for different disciplines require differing amounts of travel with those designed for cross-country riding typically requiring less travel than those designed for more aggressive disciplines such as downhill racing. See Table 1 for typical specifications.

Table 1: Table of common suspension travels and intended disciplines

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

### 2.1.2 Front Suspension

Front suspension commonly employs a linear telescoping shock absorber, known as a fork due to it's 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.

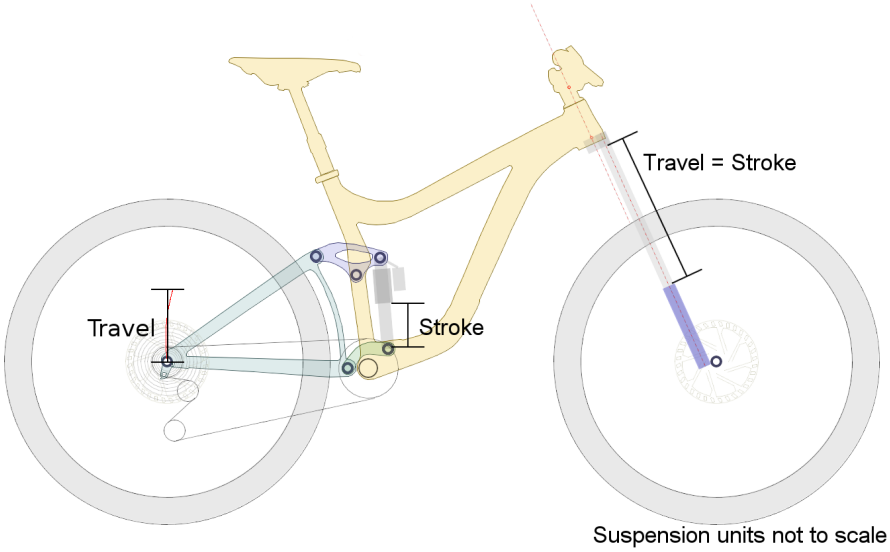



Figure 2: Diagram showing travel and stroke on a full suspension bike

### 2.1.3 Rear Suspension

Rear suspension uses a shock absorber that is much shorter than a fork so that it can fit neatly into frame designs. This means it cannot operate on a 1:1 ratio while still allowing the full amount of travel required. Full suspension 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 average distance the rear wheel moves for every 1mm the shock compresses throughout its stroke; the average is used as the leverage ratio changes across the compression cycle.

The difference between front and rear stroke and travel can be seen in Figure 2 which shows the separation of rear wheel travel from the stroke of the shock absorber. Although manufacturers design their rear suspension differently, the rear wheel always rotates around the main pivot, (or in some cases a virtual pivot), as opposed to moving linearly as front forks do. As a result, the frame behaves differently through its travel, depending on the number and location of pivot points and the type of shock that is being used.

Because of this the average ratio is normally dismissed in favour of a leverage curve that plots the ratio n:1 throughout the compression cycle. 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 is evident that varying the location of pivot points produces suspension with drastically different characteristics.

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 as seen in Figure 3. This means the suspension will feel soft most of the time but stiffens as compression increases. This is emphasised by the Horst link system of the Lapierre Spicy (shown magenta) where the leverage curve rises significantly towards the end of its travel. In contrast, the curve of

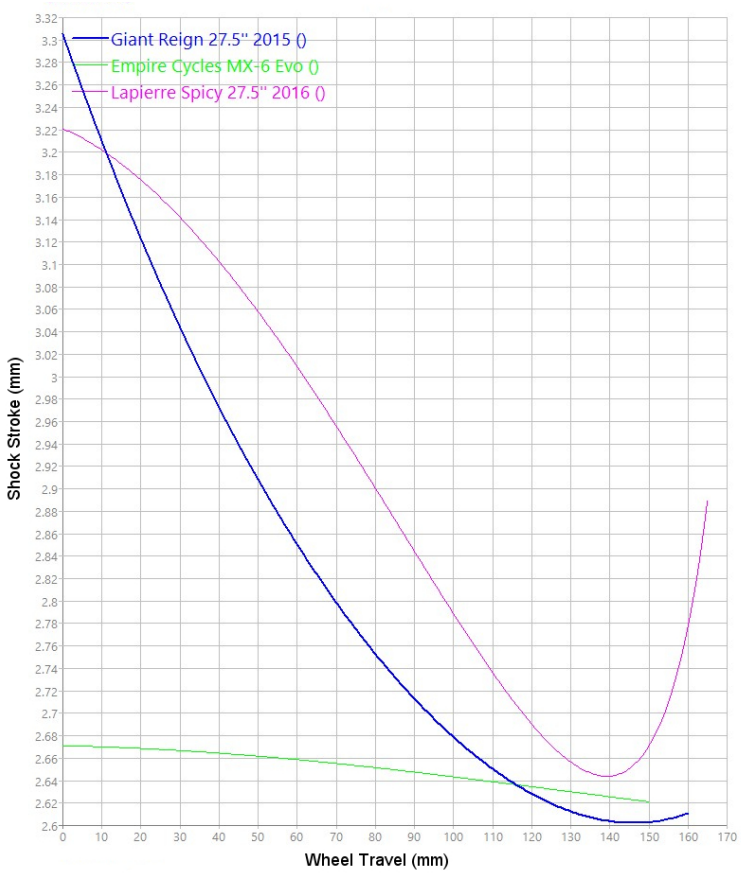
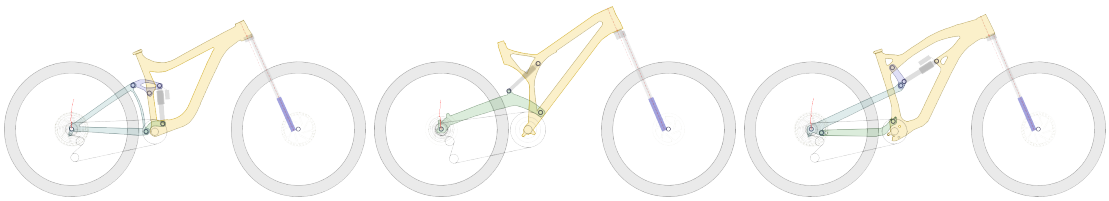



Figure 3: Leverage curves of three modern suspension designs

the single pivot Empire MX-6 Evo (shown green) is effectively linear. This is due to the MX-6 having only one pivot and swinging arm, as opposed to the 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. The physical differences in each of these frames can be seen in Figure 4 which was created using the Linkage[1] computer program.



(a) Giant Reign (b) Empire MX6 Evo (c) Lapierre Spicy

Figure 4: Full suspension frame comparison

For this project two bikes will be used to develop and test the application. The first is a 2015 Giant Reign, shown on Figure 3 in blue, as it will be constantly available throughout 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, Maestro creates its virtual pivot as close to the rear of the frame as possible as shown by the red circle

[1] http://www.bikechecker.com/

in Figure 5. The second will be a 2011 Orange 5 which has the same suspension design as the Empire MX6 Evo in Figure 3. This bike was selected as it has a completely different design to the Giant Reign and will also be easily accessible throughout the project.

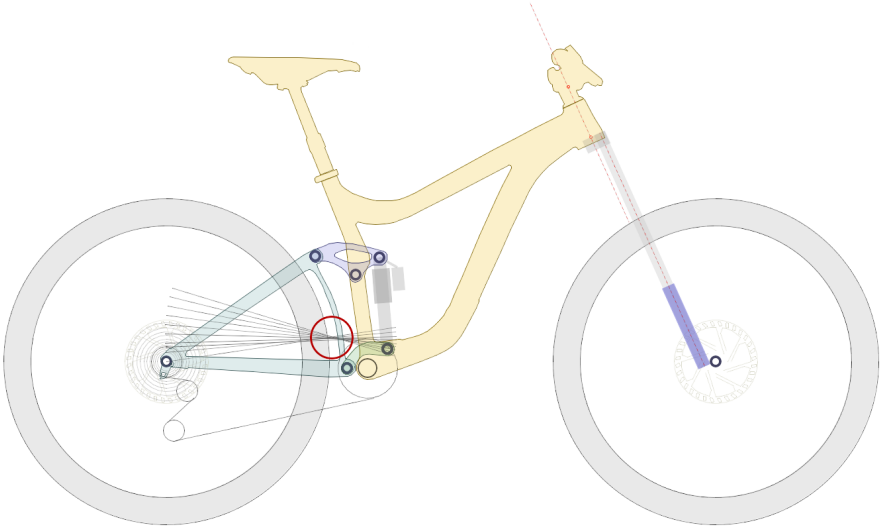


Figure 5: Maestro suspension

Although rear suspension designs are complex neither the average rider nor professionals, such as mechanics working in the industry, necessarily need this specialist knowledge. Leverage curves are predominantly used by designers to determine how a particular suspension unit will behave when researching and designing new frames (Cane Creek, 2016). Though having an understanding of leverage curves will help the individual rider to fine tune their suspensions settings themselves, the majority will take little interest in the differences between a single pivot and VPP design, choosing a simpler ”set and forget” approach to their suspension.

In the context of this project and the intended target audience for the application, this begs the question of how much information should be provided to the user? Modern human computer interaction principles aim toward providing information to the user which is relevant and necessary in context (Shneiderman, 2010). Presenting the user with a leverage curve which may require extensive interpretation before they understand the implications within the limitations of a mobile application may prove detrimental to the user experience. 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 plethora of technical data.

### 2.1.4 Sag

Sag is the amount that the suspension sits into its travel when the rider is in a neutral position, as described in Figure 6. It is required so that the suspension has travel available to extend as well as just compress and is calculated based on the rider’s weight, available travel, and intended riding style.

The amount of sag depends on the stiffness of the suspension unit which can be adjusted by changing the air pressure of an air spring or replacing the spring and adjusting the preload of a traditional coil shock. Depending on the discipline and the amount of travel the bike has, sag can vary between 15% and 40% of the available travel though it is typically set at between 25% and 35% for the average rider with greater variances only encountered in competition.

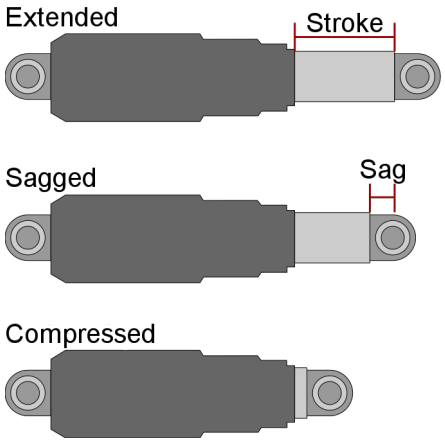



Figure 6: Diagram of shock states indicating sag

Sag is set by first calculating the distance that the shock should sit into its travel by producing the desired percentage of the shock's stroke. For an air shock the manufacturer's recommended pressure is then pumped into the shock, and the rider weights the bike. On air shocks there is a marker O-ring which is pushed down the shock shaft as it is compressed. The position this O-ring ends at can be measured and the shock pressure adjusted accordingly until it sits at the target measurement. For coil shocks, the total length of the shock must be measured while the bike is under load.

### 2.1.5 Damping

In a spring and mass system (visualised in Figure 7), when the spring is elongated it will exert a returning force on the mass, labelled $F_k$. According to Hooke's law (Rychlewski, 1984), this force is directly proportionate to the distance which the spring has been pulled. In suspension this force is explained as two separate forces. If an individual were to push down on mountain bike suspension they would feel a resistance, this is known as compression, and when they let go the suspension will return to its neutral state, this is known as rebound. Both of these can be controlled and are referred to as compression damping and rebound damping respectively.

Suspension damping works by forcing oil within the shock absorber through a series of holes in the absorber's damping circuit. Reducing the size or number of holes reduces the speed at which the oil can flow through the circuit. This increases the damping effect and makes compression require more effort and rebound slower.

**2.1.5.1 Compression Damping** This is applied while the shock absorber is being compressed to control the amount of effort required. Increasing damping forces the wheel to stay in contact with the ground for longer making the suspension feel stiffer.

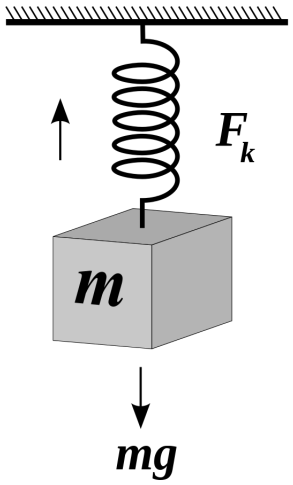



Figure 7: A spring and mass system

However too much compression damping can make the suspension overly stiff so it does not effectively absorb the impact of bumps and rough terrain. Conversely, too little compression damping can cause the suspension to ”blow through” all of the available travel prematurely with nothing left to soak up impact when it is most required.

**2.1.5.2 Rebound Damping** This is used to control the speed at which the shock absorber recovers to its normal riding position after compression. An optimal setting will allow the suspension to track the ground quickly and smoothly, returning to the correct position after a bump or hole is passed.

Too much rebound damping causes the suspension unit to recover slowly and sometimes “pack down” meaning the shock absorber remains compressed for too long leaving the rider fully exposed to the next impact. Too little rebound damping can cause the suspension to “buck” the rider, like a horse, and potentially cause an accident.

**2.1.5.3 High and Low Speed Damping** The ways of adjusting compression and rebound damping differ by manufacturer and model with better specified bikes having two adjustable speeds for each damping circuit giving four damping settings. High speed adjustments are used in high G-force situations such as riding large jumps or drops where compression needs to be set softer to absorb impacts and rebound set slower so the rider has time to recover without the equilibrium of the bike being upset.

Low speed damping set ups are used against lower G-force movements such as rider weight shifts or long, slow compressions. Optimally compression is set stiffer as this type of feature can deceptively use a lot of travel from successive compressions. Rebound damping is 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 principles that 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, which is normally somewhere in the middle of the two extremes available with a slight bias towards the faster option.

Attaining this optimal setup can be difficult for both beginner and intermediate riders as they lack experience and in-depth knowledge of suspension units and may not know how different frames react while being ridden. Knowing which measurements to make and the calculations required to correctly configure sag, compression, and rebound settings are typically beyond the capabilities of this level of rider unless they have been previously trained by a professional or investigated the topic in detail for themselves.

## 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. Image processing is the application of various processes on an image to change or enhance its appearance. In practice, the processing stage normally comes before the analysis stage as a way of simplifying the image prior to analysis in order to maximize the likelihood of obtaining usable data.

### 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 corresponding point on the sensor.

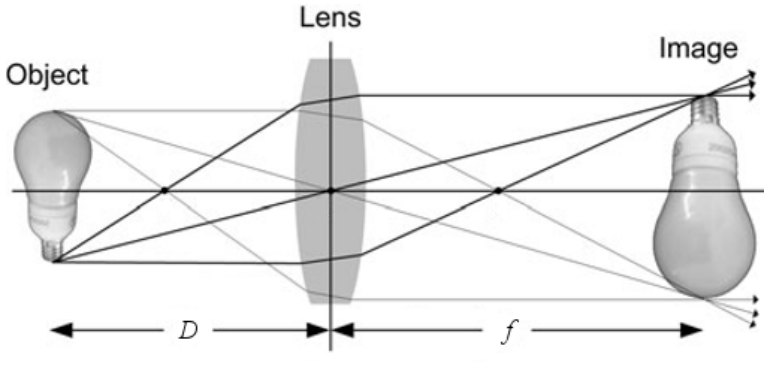



Figure 8: Diagram of camera and lens operation (Moeslund, 2012)

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 8. This point can be adjusted by changing the optical characteristics of the lens so that the refracted light hits the sensor at the correct point to create a sharply focused 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 as it was captured. Each of the cells in the camera's sensor equates to a single pixel in the final image. For example, a camera with a sensor made up of 1920 cells across by 1080 down (otherwise known as a 2 megapixel sensor) produces an image 1920 pixels wide and 1080 high. The larger the physical size of the sensor, the more cells it can contain resulting in a higher quality image.

Image processing is the manipulation of these integer values which adjusts the visual appearance of the image for either artistic or scientific purposes. Image analysis is the comparison of these values either to their neighbours or in clusters to locate features, produce measurements, or for other purposes within the context of the image.

### 2.2.2 Lighting Conditions

When capturing images to be analysed, it is important to have good lighting conditions so that none of the required detail is lost (Moeslund, 2012). If the image is captured in unsuitable conditions, such as low light, then this can seriously affect the outcome of subsequent analysis.

Certain image processing techniques, such post-processing images captured as Camera RAW files, can help ameliorate poor lighting conditions though this is not foolproof and it is always better to capture a well-lit image. Figure 9 shows the effects that different lighting angles have on a subject.

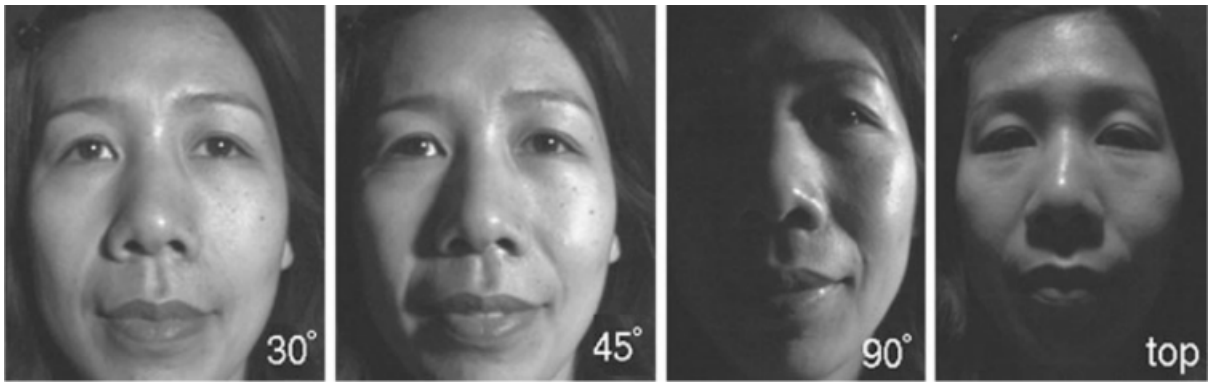



Figure 9: The effects of different lighting conditions on a face (Moeslund, 2012)

It can be seen that the situation which highlights the most detail is when the light source is close to being in front of the subject matter. Such lighting reduces the amount of image processing required before analysing and produces the largest amount of usable data. In the context of this project many smartphones also have an integral flash function which means the user can correctly light their image in sub-optimal lighting conditions where required detail may otherwise be lost.

### 2.2.3 Usages

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

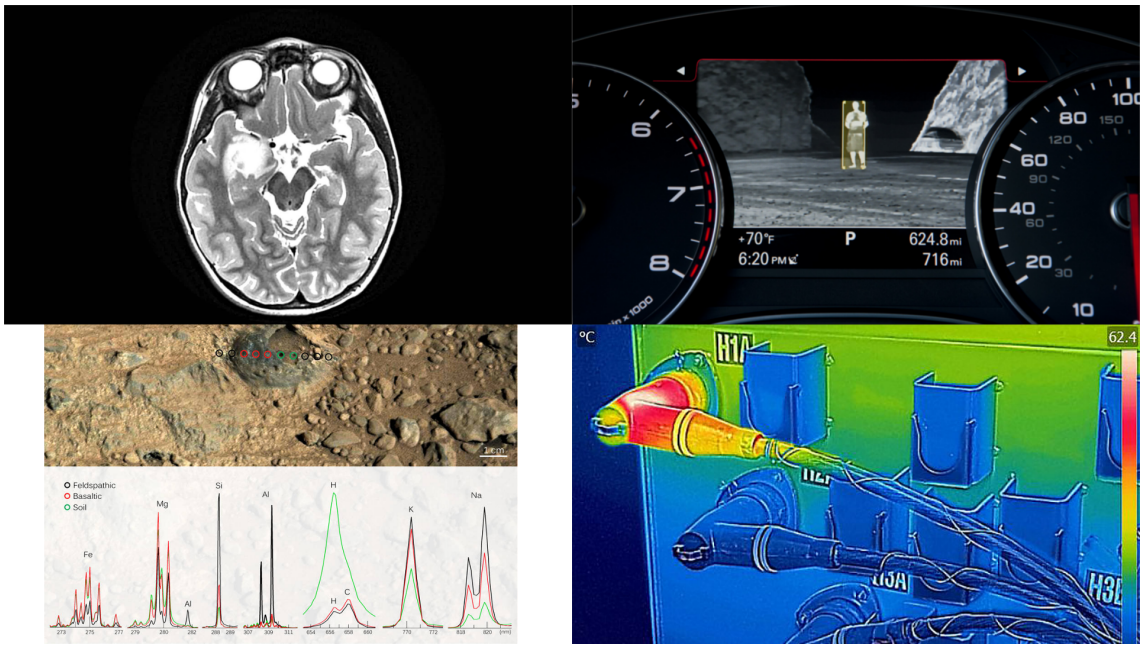



Figure 10: Uses of image analysis, from top left clockwise: An MRI brain scan, automotive night vision with pedestrian recognition, infrared image taken with a smartphone, chemical rock analysis from mars

**2.2.3.1 Medical** This is arguably one of the most important uses of IA. Advances in medical imaging have reduced costs, diagnosis time, and patient recovery time while improving the ability to localise and personalise treatments (European Science Foundation, 2007). Major uses of IA in medical applications are the use of Magnetic Resonance Imaging (MRI) and Computerised Topography Scanning (CT Scan) to create detailed images of the human body and identify illness before obvious symptoms appear. This is shown top left in Figure 10.

**2.2.3.2 Transport** Image analysis has been included in the consumer automotive market on various models since 2004 when Honda introduced a thermographic night vision camera with automatic pedestrian detection on their Legend model (Honda Motor Co., 2004). Other manufacturers, including Audi, have since introduced similar technology and their implementation can be seen top right in Figure 10. Since this initial use many other vehicle manufacturers have included functionality based on image analysis expanding its use into areas such as the automatic recognition of speed limit signs, 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 helped to create more stable and efficient structures, in bridges and skyscrapers for example, by looking at the materials used in their construction (Masad, Muhunthan, Shashidhar, & Harman, 1999) and monitoring their stresses and potential areas of weakness (Kim & Kim, 2013). Today advances in mobile computing have allowed engineers to routinely use IA while on site and some application vendors target their products at an engineering sector by improving their durability and integrating features such as infrared imaging (Liszewski, 2016), the output of which can be seen lower right in Figure 10.

**2.2.3.4 Space** While some industries make use of satellite imagery to monitor changes to our own planet, agencies such as NASA and ESA use image analysis to look at other planets and celestial bodies. The Martian rover, Curiosity, uses multiple cameras for navigation, hazard avoidance, and scientific imaging with the images streamed back to Earth for detailed analysis. Major uses of such extraterrestrial images include the identification of geological formations and their likely composition (Blake et al., 2013; Garvin, Malin, & Minitti, 2014) and the location and identification of chemicals using the analysis tools such as ”ChemCam” (Schröder et al., 2015), which can be seen lower left in Figure 10.

### 2.2.4 Image Analysis Techniques

There are a number of analysis techniques which can be applied to imagery to extract information. Perhaps the most common is analysing the pixels in an image to identify discontinuities in their data values which suggests the edge of an object, structure, or land mass. Such functionality is readily available on desktop image processing packages designed for amateur photographers (Adobe, 2017). But scientific and engineering applications of edge detection provide far greater levels of sensitivity helping isolate objects that can be subsequently measured with a high level of accuracy (MatLab, 2017).

**2.2.4.1 Thresholding** Image thresholding is the most basic form of image segmentation (Haralock & Shapiro, 1991). At its simplest this would entail taking a greyscale image, examining the intensity of each pixel and setting those below a certain threshold to black and those above it to white as shown in Figure 11 below. By thresholding an image, objects lose detail but their silhouettes become clearer making them more visible to object detection techniques.

Figure 11: A copy of Figure 1 with thresholding applied

**2.2.4.2 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, although all utilise the same basic concept of comparing the data values of adjacent pixels to find discontinuities from one value 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 this discontinuity which will be highlighted 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) where edge detection is applied to highlight the block shapes of the vehicle registration plate. Figure 12 shows an image which has been edge detected with the number plate of the car clearly visible.


Figure 12: Edge detection applied to an image for number plate recognition

**2.2.4.3 BLOB Analysis** Binary Large OBject (BLOB) analysis is a method of feature detection used in image analysis (Moeslund, 2012). Typically carried out after thresholding, the pixels in an image are inspected and compared to their neighbours based on colour and intensity. If they are considered similar, the pixel locations are logged within some data structure which once complete will define all pixels within a single object.

Once all BLOBs are identified they can then be analysed based on some metric, which may be shape, colour, or size for example. What is done with the results of this is dependant on the project context though some uses include identification of pedestrians as shown in Figure 10, finding a reference point for measurements as explained in section 2.2.4.5, or reading characters in type (Patil & Dhanvijay, 2015).

**2.2.4.4 Hough Transform** Originally created in 1962 by Paul Hough, the Hough transform methods offer techniques for the discovery of imperfect objects in images. 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.

While edge detection is suitable for highlighting critical points in an image, it can be problematic if the base image is not well lit or out of focus causing an edge detector to produce imperfect lines. Such artefacts can make rudimentary feature detection techniques difficult but the Hough Transformation techniques circumvent such issues by considering points as a part of an object through a voting procedure.

The collection of Hough transform algorithms are separated by the objects which they are capable of finding, such as linear Hough transform or circular Hough transform, and have been well described by Leavers (1992). The Hough transform methods could be used in this project for the detection of suspension units in images or sections of the whole bike depending on the chosen process for the application.

**2.2.4.5 Taking Measurements** To measure the dimensions of an object in an image, certain data about the camera and its location are required. By using digital imagery, the majority of this information is automatically 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 if available.

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

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 1. Necessary data, such as $f$, $H_i$, 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 has to be collected at the time the image is taken. To briefly test this equation, data was manufactured for each parameter and passed through the equation. This process was then reversed using the now known object height to see if the result matched the manufactured one, which it did.

To circumvent using this equation, a reference object of a known size can be included in the image. This removes the need to consult EXIF data meaning objects can be measured, irrespective of the camera used, though IA techniques. The previously mentioned Mars rover, Curiosity, carries a United States penny and charts as reference objects so that its cameras can be precisely calibrated as shown in Figure 13.

Figure 13: Contact Instrument Calibration Targets on Mars Rover Curiosity (NASA JPL, 2012)

First, 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 is collected and compared against its actual size. This gives a ratio for applying to the pixel count of any object in the image to calculate its real world height. For example, if a reference object that is 2mm wide measures 200 pixels across in an image then it can be concluded that 100 pixels in the image corresponds to 1mm in the real world. If another object in the image measures 300 pixels across then it must be 3mm in the real world. This can only be concluded if the reference object and measurable object are of similar distance from the camera as perspective commonly causes issues when comparing the size of objects.

This process will be applied during this project in some manner. Either a reference object will be used or the equation to calculate the height of an object will be applied. While incorporating Equation 1 in an application will be relatively simple, collecting the necessary data without too much user interaction may be more difficult.

## 2.3 Image Analysis in Sports Science

The benefits of image analysis in sports science are most prevalent in the area of biomechanics. In most sports, an individual's performance depends on a combination of physical fitness and acquired skills. In certain sports, such as motorsports, equipment is undeniably important although unless "driver-athletes" can cope with the stresses and strains of race conditions they are unlikely to achieve success regardless of the technical superiority of their vehicle (Klarica, 2001).

In such sports, the performance of cars, motorcycles, and even mountain bikes are predictable and measurable. They have all been designed and manufactured to be as fast as possible within the rules laid down by the sport's governing body so it is im-

portant to continually analyse and fine tune their setup by changing tires or adjusting suspension. Doing so seeks out the marginal gains that make the difference between gaining a place on the podium or not. As a result, performance cars and bikes are fitted with an array of sensors that constantly monitor critical aspects of performance such as engine temperatures or suspension (Segers, 2008).

Fitting sensors to humans without impairing their mobility is not quite so simple. To provide the best testing ground for fitness and technique, the participant should be unhindered and able to perform tasks without data capturing equipment getting in their way. Image analysis can get around such issues by utilising various techniques that can allow for stable and repeatable test situations while also providing a platform for reviewing the captured images. A study into bowling techniques in cricket used a mix of manual point picking and automated measuring from images to produce data such as the angle and speed of bowling deliveries as well as ball spin (Cork, Justham, & West, 2012). While the dataset was limited due to conflicts with the players' training schedules, the results proved useful and were subsequently replicated in a pitching machine so that batsmen practised against more lifelike deliveries.

A second study on the use of IA in showjumping (Wejer, Lendo, & Lewczuk, 2013) made use of techniques similar to those used by Cook, Justham, and West. In this instance, the angles of the horse's limbs were recorded and compared over a period of four months to analyse whether different training techniques delivered measurable improvements. Here, passive image analysis was chosen as the preferred technique as attaching sensors to the horse could have frightened the animal and almost certainly caused it to perform below par.

## 2.4 Image Analysis for Optimizing 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 the specialists at Fox Racing Shox using the technology (Benedict, 2012). The mobile application that Fox has produced is locked to the forks and shocks that the company manufacture although the image analysis techniques they use are adaptable enough to be used on suspension units from other manufacturers.

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 to create 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 that this project can utilise to achieve its aims. By using this knowledge, the

following sections will identify distinct methods which the project could use and cover the results of how the application performed under test conditions.

# 3 Methodology

## 3.1 Introduction

This chapter gives an outline of the methodologies which could be used to complete the work identified in the previous chapters as well as the reasons for using these methods and rejecting their alternatives. The effectiveness of these methods determines the success of the project so those chosen need to be both reliable and manageable. A technical approach is identified for the most appropriate way to produce a solution to the problem identified and project management approaches are considered so that the project can 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 with which this project is involved. This aids both the reader and author in understanding these subject areas and helps to explain how the project presents a viable solution to the problem area.

There are numerous forms a literature review can take although not all have the same aims. A traditional or narrative review can be carried out to critique a body of literature and potentially locate inconsistencies (Adams, Khan, Raeside, & White, 2007). This type of review is normally used when the research question is well defined. Alternatively a systematic literature review can be undertaken although this requires a more rigorous process and takes more time. Systematic reviews aim to locate previous studies that are within the same or similar subject areas and gain insight to expedite answers to the questions presented by the research (Kitchenham et al., 2009).

For this project a combination of both methods has been used. A systematic approach was applied to investigate how image analysis is used and ascertain which particular techniques were relevant to the use of IA 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 whether previously used techniques would transfer to mountain bike suspension and provide some idea of how effective they could be.

### 3.2.1 Source Selection

Regardless of which type of review was chosen, it was necessary to adopt a methodological approach to identify sources relevant to the project. 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 provide an expansive library to select from, which is made easier by using a variety of advanced search filters.

As the application of advanced technologies to mountain bike suspension is in its infancy and commercially driven, much of the research is undertaken by the manufacturers themselves so there were no academic sources to use. As a result, numerous multiple mountain bike websites and blogs have been cited. In a subject area where

the research is more established this would be frowned upon as such sources tend to be biased and often unproven. However it was entirely unavoidable. To overcome author bias, multiple sources have been cited wherever possible as a way of providing different viewpoints on the same topic.

To determine whether a source is suitable requires a methodical and critical approach. Even before reading a paper or article, it is possible to decide whether the subject matter and research is up to date by referring to its publication date. Such an approach was adopted for this project in order to identify the most recent research available. As a second step, online libraries provide metrics about the number of times a paper has been cited elsewhere; the inference being that papers that have been cited most are both more relevant and trustworthy. However, applying this rule of thumb needed care as some papers are cited so frequently purely because they are the most controversial. Finally when reading a paper, the abstract should be read first followed by the conclusions. This takes little time and taken together, the abstract and conclusions provide a good indication of whether the paper contains relevant and reliable research. Only then should the body of the paper be read in its entirety.

## 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 core of 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 (Bradski & Kaehler, 2008) 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 (OpenCV.org, 2017), a non-profit organisation who provide support and documentation. Although 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 and provides functions for the various processes this project requires. 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 while simultaneously gaining experience in using OpenCV on Android. For the Python based approach, tutorials from the www.pyimagesearch.com website created by Adrian Rosebrock (Rosebrock, 2017) and the www.pythonprogramming.net website (Pythonprogramming, 2016) were followed and run on a desktop computer.

Appendix C 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 worked successfully. Although a stack trace of the errors was produced they do not explain 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 originally written in Java to be run on Android devices. The NDK understands various programming languages and applied a Java wrapper around them that 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 following research into this issue it was found that this is difficult to rectify.

The two working Android experiments do not operate as expected either. Appendix C shows the issues with each that were encountered in the test. 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 could not rectify these issues to produce an acceptable outcome. Appendix D shows the experiments that were carried out using Python. These were much more successful than the Android experiments as they are all functioned and worked exactly as anticipated.

Appendices C and D show a comparison between the two methods by including 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 encountered when using OpenCV on Android, 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 solution as opposed to its portability which has already been proven by many applications. 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 (Kuhlman, 2009). 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, as shown in Listing 1. The source code for this project will be written in a pythonic way where it makes sense to do so.

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

Listing 1: An example of Python code

### 3.4.2 Naming

As well as the readability which comes with 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 easier for an outsider to understand if any parts of the code need to be revisited at a later date. This is demonstrated in Listing 2, where although the left-hand code looks cleaner, when read through looks extremely generic and could be part of any system. In contrast the right-hand code is much more descriptive of its function.

```
public List<int[]> getThem() {
  List<int[]> list1 = new
   ↪  ArrayList<int[]>();
  for (int[] x : theList)
    if (x[0] == 4)
      list1.add(x);
  return list1;
}
```

```
public List<int[]> getFlaggedCells() {
  List<int[]> flaggedCells = new
   ↪  ArrayList<int[]>();
  for (int[] cell : gameBoard)
    if (cell[STATUS_VALUE] == FLAGGED)
      flaggedCells.add(cell);
  return flaggedCells;
}
```

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 Application Programming Interface

(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. Instead it identifies methods and variables prefixed by one or two underscores as protected. This means these items will be hidden from view in autocompletion systems or documentation but remain 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 always 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 to data or memory the test has made are cleaned up.

As unit tests are simple in structure and quick to run 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 and completeness 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 that 15% of a module is not under test and could cause undetected problems if any changes are made. Many integrated developers environments (IDEs) or testing suites provide coverage checking 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 language there are many implementations of unit testing frameworks to choose from. Each has its own advantages and disadvantages and there is much debate 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 subsequent changes can be tested to see if they have affected functionality. Any new functionality will have tests added to the testing suite to maintain 100% coverage.

The Python unit testing framework chosen is unittest2. Although it has been superseded by unittest2 in Python3.x, Unittest is the default framework used by the PyCharm IDE. The unittest2 module provides a backport of unittest's functionality for use in Python2.x, 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 to achieving full coverage.

## 3.6 Project Management

### 3.6.1 Agile Development

Introduced in 2001 with the writing of the Agile manifesto (Beck et al., 2001), agile development methodologies focus on the rapid delivery of high quality software rather than the rigorous design-based structure of traditional waterfall methods. By utilising various techniques such as stand-up meetings, a strong customer focus, and sprint development cycles, agile has become widely adopted in industry and has been proven to produce successful projects that are both closely tailored to the customers' needs and delivered in shorter time scales (VersionOne, 2015).

There are numerous agile methodologies (XP, Scrum, DSDM) each with their own principles and techniques. However it is commonplace for a company to create their own agile approach picking and choosing items from different methodologies to suit their particular needs (Aydin, Harmsen, Van Slooten, & Stegwee, 2004). As this is a solo project with no customer then a single set methodology will not be used. Instead a variety of methods may be used to help manage the project and drive the efficiency of the development process. These potential methods will be outlined in the following sections.

**3.6.1.1 Requirements Analysis** Used in some form by the majority of agile methodologies (Cao & Ramesh, 2008), 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 trace-

able to aid in its understanding and completion.

The techniques used to produce requirements include stakeholder identification and interviews which are used to identify what the stakeholders of the project require once it is completed. 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, although these are typically extensive and incomplete as considerable collaboration is involved in producing them. The requirements analysis techniques that are most frequently used in agile developments 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.

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

**3.6.1.2 MoSCoW** First used in the DSDM agile framework (Bittner, 2002), 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 then be prioritised either for the entire project or for individual sprint cycles depending on the size and magnitude of the project.

MoSCoW was created to give customers a better understanding of software requirements. Unlike categorizing priorities as high, medium and low, MoSCoW is more descriptive of what the prioritisation means for the software project. From a development point of view, this prioritisation process allows developers to focus on the core requirements of the project first, creating a viable software solution early in the development cycle with less important features being added later if the resources are available.

This project could use MoSCoW to prioritise the requirements identified using the requirements analysis technique. 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 (Rising & Janoff, 2000), a sprint is a timeboxed 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 review is carried out by the development team to discuss what went well, what didn't, and how this can be rectified in the next sprint.

This project could use sprints in the same manner as an agile development team as this would allow easier completion of requirements and tighter management of the de-

velopment process. Using sprints would ensure the time allotted for development is used effectively which may lead to a higher quality product at the end of the project.

As mentioned previously, MoSCoW prioritised requirements are selected at the start of each sprint cycle to set objectives for the work which will be carried out. Each cycle aims to complete all of the must have requirements and the majority of should and could haves. At the end of each cycle, a retrospective review is carried out which will re-prioritise any 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 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 a vision of the end product. Once started, development would take a natural direction identified by what is needed for the system at the time or to solve any issues encountered.

To document the development process, a daily log of any work completed would be kept whenever any development is carried out. This would allow for referral at later dates and keep the project 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 inherent in an agile process could mean the development process is over managed. In contrast the lack of structure and defined direction by using an organic approach could cause the project to run out of time and not produce a suitable solution. The development process is still undecided and will be chosen closer to the start of development.

### 3.6.3 Version Control

Large software projects produce multiple files of code, data, and documentation. These are vital to the overall success of a project and should be kept safe. If these files are lost then the project could easily be delayed or halted as the time and resources may not be available to recreate the lost data. Action against this can be taken by backing up any data, preferably on a cloud based system, to avoid loss and allow the project to continue should the local copy of the data be lost.

For this project the Git Version Control System (VCS) will be used. Git uses 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, details of when those changes were made, and who made them This is a powerful tool as it means the various sections of the project can be worked

on without the risk of damaging the project and should a change prove unsuccessful then the repository can be reverted to a functional point.

For this project 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 which could be used to log each new feature as it is added, although strictly speaking they are not issues at all. The reason for this is that all features logged in the repository will have a unique identification and description against which commits are recorded. 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, such as Microsoft's Team Foundation Server or Perforce, are available. However these are limited under free licences or locked to certain development environments whereas the accessibility and ease of use provided by Git makes it the optimal 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 which can then be used as intermediary 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 exactly the same reason. Current examples include the week nine review session and the deadline for the completion of this document. As more milestones are identified through breaking down the development process into development sprints and other ancillary tasks, they too will be documented and acted upon.

### 3.6.5 Threshold

Tasks taking up more than their allotted time is a common cause of projects consuming more resource than initially allocated. 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 timely delivery of the project.

### 3.6.6 Gantt Chart

A Gantt chart is a method of breaking down a project into tasks and graphically representing them as bars on a chart, first used by Henry Gantt in the second decade of the 20th century, as a way of ensuring schedules are met. Commonly starting at a project's inception date and ending with it's 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 and graphically show the project's critical path.

This project will make full use of the Gantt chart technique by deconstructing 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. Microsoft Project 2016 will be used to create the Gantt chart.

## 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

This is a measure of how well a design or solution meets its original specifications and fulfils its intended purpose. It involves checking that all required functionality is present, each output is correct, 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 that it has successfully passed each test, will confirm 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 it produces are both consistent and correct. To assess these an uncertainty metric will be produced as described below. When taking multiple measurements there is typically a ”true” value which is the actual measurement that falls somewhere within the range of produced measurements. Uncertainty is a prediction of how close any produced measurement will be to this ”true” value, expressed as $\overline{x} \pm U$. The method for producing a value for uncertainty is as follows:

1. Produce a suitable number of repeat measurements $\{x_0...x_n\}$
2. Calculate the average value of these measurements $\overline{x} = \frac{\sum\{x_0...x_n\}}{n}$
3. Find the differences between the measurements and the average $\{d_0...d_n\} = \{(x_0 - \overline{x})...(x_n - \overline{x})\}$
4. Calculate the average of these differences squared $\overline{d_s} = \frac{\sum\{{d_0}^2...{d_n}^2\}}{n}$
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 all possible variables depending on what the application is capable of. These variances will be outlined once the evaluation has been completed.

### 3.7.3 Comparison to Alternatives

Comparing the process of using the application to produce a sag setting against other available methods will show how simple the application is to use and whether it is more effective than other methods. For this comparison, outputs from the application will be compared with a trial and error process that is a common practice for a beginner or intermediate rider setting their suspension for the first time as described in section 2.1.4. Comparison to this process will indicate whether the application presents a benefit over the manual approach and further determine if it is outputting suitable results.

### 3.7.4 Professional Opinion

The final method of evaluation will be to seek the opinion of professionals working within the mountain bike and cycling industry. This will provide further indication of how well the application achieves its goal and how it might be received were it to be commercialized. This process could also suggest areas for future work which may not have been seen without outside consultation.

The Mountain Bike Centre of Scotland[2] will be approached to recommend an individual who they deem suitable to provide such expert assessment once the application can be demonstrated. Although it is anticipated that the meeting will take a semi-formal approach, general evaluative questions will be posed with responses recorded. In addition to this meeting, the application will also be demonstrated to staff at a local bike shop to gain their feedback. This will take the same approach as the first meeting.

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

# 4 Results

This section 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. 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 there was some uncertainty about which approach would be followed during the development stages of the project. As it happened, once the development stage was reached, an organic approach was taken. Although less structured than an agile methodology it was decided 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 F. This shows that an adequate amount of work was carried out within the time-frame allocated despite using a less structured process and that the final application includes complete functionality to meet its stated aims.

### 4.1.2 Use of EXIF Data and Reference Point

As mentioned in section 2.2.4.5, the process which the application would use to produce measurements was undecided and required experimentation. Initially Exchangeable Image File Format (EXIF) data from the image was used, which was successful until collection of data about the sensor size was required. Due to this being uncommon in EXIF data, particularly in images taken with smartphone cameras, this method was abandoned in favour of using a reference point. The code produced during this experimentation phase can be seen in Appendix E where the function `get_sensor_size` is incomplete as this is the point where this approach was dropped.

The reference point chosen is a red circular sticker as these are cheap, freely available in store or online, and being round do not need to be orientated to the object as a square or rectangular sticker would. This sticker is applied directly to the shock unit where it can be easily picked out in the image due to its prominent colour. But although locating the reference point was simple, issues were encountered in the measuring process and these will be discussed in a following section.

### 4.1.3 Colour Quantification

For initial experimentation with using a reference point, a similar sized red circle was manually added to images using an image processing application. This allowed for tuning of the colour masking process and refining the process before using a real reference point stuck on the shock. However when images with a real reference point were used, the application could not detect it. This was because the colour range for the masking process was using perfect red (RGB 255,0,0) and the non-standard reds of the sticker as captured in the image were not within this range.

A first attempt to resolve this issue involved expanding the boundaries of acceptable colours that the application would recognize in the image as being red. This proved to be insufficient so a further image processing technique known as colour quantification was also used. This reduces an image to 2, 4, 8, 16, or 32 bit colour spaces which sacrifices the range of colours contained in an image but makes objects more prominent so they are easier to detect as they become shapes of solid colours. The difference between an original image and one using quantified colours is shown in Figure 14. 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.

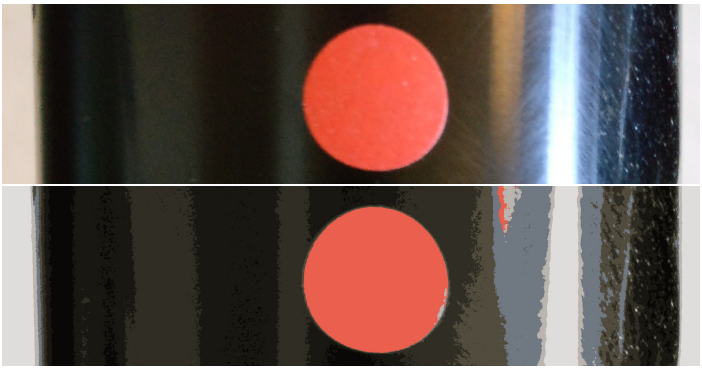


Figure 14: Normal image (top) versus quantified colours (bottom)

### 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 sag measurement. This was deemed suitable as the manual process to calculate sag also uses the O-ring which is a prominent feature on most shocks, unless removed by the owner. Initial versions of the application assumed it would find the O-ring somewhere about two thirds of the way up the image height although once multiple images were used from two different makes of shocks this assumption proved to be incorrect.

To address this and ensure the application is always able to find the O-ring no matter where it appears in the image, it was necessary to use a dynamic method to detect it. In much the same way that using quantified colours solved the issues of detecting the reference point, OpenCV's `findContours` can be used to find an O-ring of a known colour as long as it is present in the image. When located, a bounding box is drawn around the contour as shown in Figure 15 and this can be used to determine the lowest

point at which the O-ring appears in the image to establish the limit of measurement.

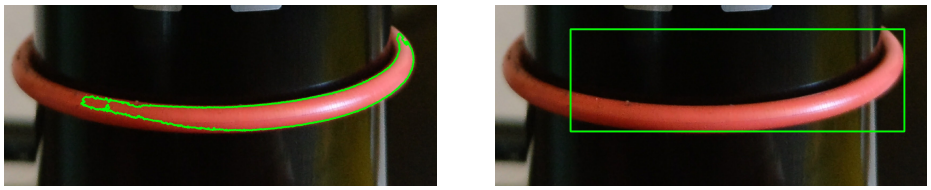


Figure 15: Red O-ring found using `findContours` (left) with `boundingBox` applied (right)

As some shocks do not have a coloured O-ring, this process was adapted to locate a black O-ring. This was not successful as black is much less prominent and an alternative method was used. Thresholding is applied to the image and contours produced from the reflections on the telescopic shaft of the shock unit. This made it easy for the application to detect the discontinuity between the O-ring and the shaft so the same bounding box could be applied and used to establish the lower limit for where the application can measure. This result is shown in Figure 16.

Figure 16: Black O-ring found using thresholding and `findContours`

### 4.1.5 Reference Point Measurement Method

To measure the dimensions of the circular reference point, the first process applied was Hough Circle Transform as described in 2.2.4.4. This method was chosen as it is designed specifically to find circles within an image and is simple to implement. Although this did produce measurements, once these values were compared with values derived by manually measuring the image, they proved to be unreliable with variances both above and below the actual value. This effect is shown in Figure 17 where it is clear that the circle identified using Hough Circle Transform is much smaller than the actual reference point.

To overcome this issue it was necessary to again use OpenCV's `findContours` in exactly the same way it used to successfully locate the O-ring. Once a contour is found that encompasses the entire reference point, a bounding circle can be drawn around it as shown in Figure 18. Although the bounding circle appears to be slightly larger than the reference point, experimentation with a variety of images showed that this method

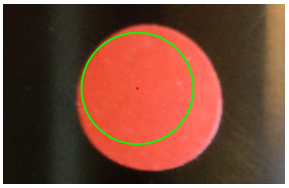
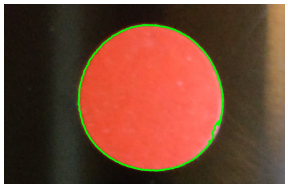


Figure 17: Reference point found using Hough Circle Transform

produced measurements that were within acceptable tolerances for the pixel per millimetre calculation.

The reason the Hough Circle method produced a bounding circle that is slightly different to the reference point is because the reference point captured in the image is not a uniform circle. Although the thresholds of the Hough Circle method can be adjusted, applying the sticker to the cylindrical body of the shock unit means it is too misshapen to be detected correctly without the use of OpenCV's `findContours`.



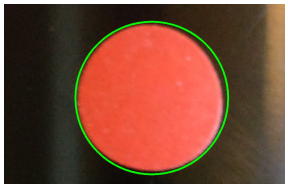


Figure 18: Reference point found using `findContours` (left) with `boundingCircle` applied (right)

### 4.1.6 Pressure Calculation

The original method for producing a suggested pressure that would give the optimal amount of sag was to calculate a pressure per millimetre metric in pounds per square inch (PSI) from the measurement of the shock shaft and the current pressure of the shock supplied by the user. The assumption being that for every 1 PSI of pressure added to the shock, this would reduce the movement by a fixed amount proportionate to the original pressure. The equation for this metric is as follows:

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

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 produced reliable results when tested manually, it failed to do so in the application both when different images, reference points, and desired sag values were tested and even when these values were hard-coded. Further investigation and discussion suggested the cause of this problem was the non-linearity

of rear suspension, previously discussed in 2.1.3, and non-linearity of air springs.

When an air spring is compressed the spring rate increases through its travel as the particles of air become increasingly compressed (Goodyear, 2014). While this effect still allows the shock unit to be fine tuned, applying a linear equation to a non-linear phenomenon is clearly going to produce some degree of error. However as the range of measurement in this project is small compared to the full stroke of an air shock, it was theorised that the spring rate can be considered linear within the limited range of compression in question.

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 plotted and compared against a trend line generated by linear regression. The results of this can be seen in Figure 19.

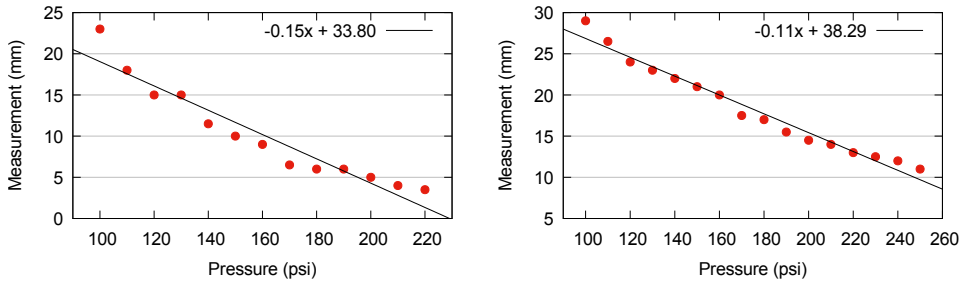



Figure 19: Sag measurements for standard air shock (left) and high volume air shock (right)

From this data it could be concluded that, within the range of measurements, the compression rate of the air spring is linear. The data-points in Figure 19 do occasionally vary though this thought to be due to it being a non-sterile test environment. Both shocks were well-used potentially containing age-related defects and loading the bike was done simply by using body weight.

Having established that the relationship between the pressure in the shock and movement from weighting the shock could be treated as if it were linear, the plan was to implement an ideal gas equation (Burdette & Thompson, 2012). However investigation into producing charts using Python uncovered a method of computationally producing a linear equation. This led to the application's process being rearranged so that it takes in two images, one of the shock loaded at 100 PSI and another at 150 PSI. Using this data it can produce a virtual plot and linear equation with which to calculate a value for the optimal sag setting.

## 4.2 Application

### 4.2.1 Images

The application requires two images to generate the measurements needed for the calculation. The application can process images in different lighting conditions and ones

where a flash has been used though there are some criteria which must be adhered to. There must be a red, circular, 7.5mm diameter sticker on the shock, preferably sited just above the shock seal. The image should also be taken so that the shock seal is around 1/3 down in the image. Examples of the images used are shown in Figure 20.

The user must supply one image of the shock which has been pressurised to 100 PSI and a second with the shock pressurised to 150 PSI both loaded under the normal riding weight, i.e. complete with helmet, body armour and hydration pack. Each time the marker O-ring should be returned to the top of the shock shaft before weight is applied to the shock.

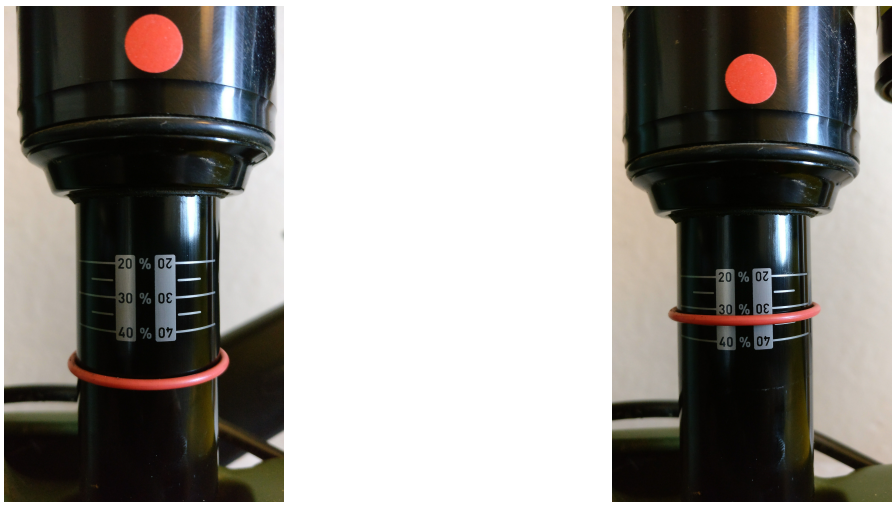



Figure 20: Images suitable for processing

### 4.2.2 Arguments

The application operates from a command line interface allowing the user to supply 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

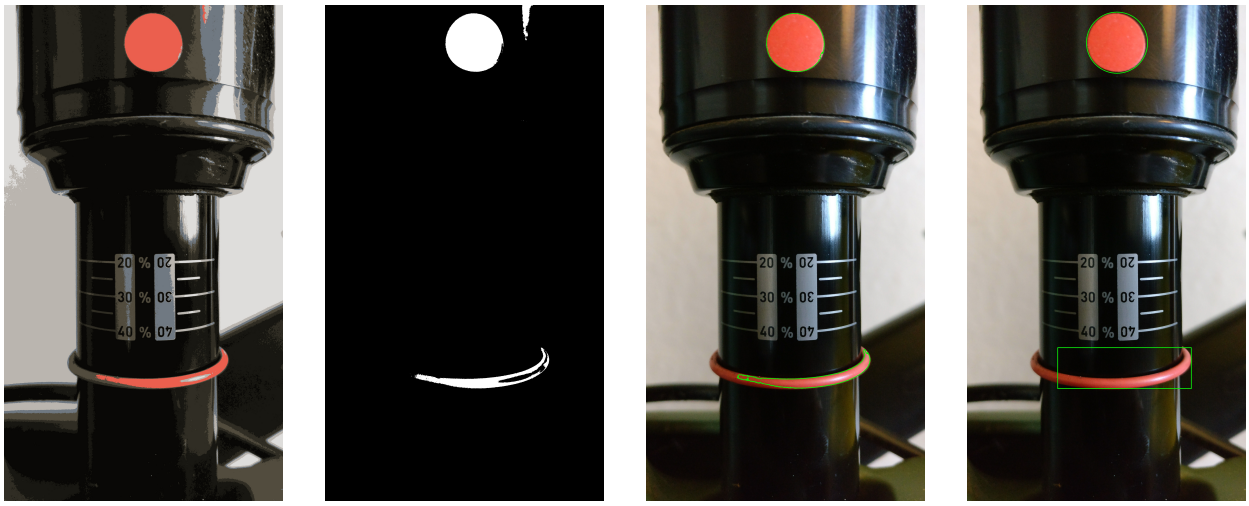
| Argument | Description | Usage |
|---|---|---|
| `-i,--image <path1 path2>` | The paths to the two images to be used for analysis | `-i path/to/img/1 path/to/img/2` |
| `-s,--sag <percentage>` | The desired sag percentage | `-s 30` |
| `-t,--stroke` | The stroke length of the shock being analysed |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

### 4.2.3 Processing

All source code for the application can be seen in appendix J.

**4.2.3.1 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 21a. 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 21b. 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 21c. A bounding circle is created around the reference point contour, as shown in Figure 21d, and this can be used to calculate the pixel per millimetre metric.



(a) Colour quantified image

(b) Red image mask

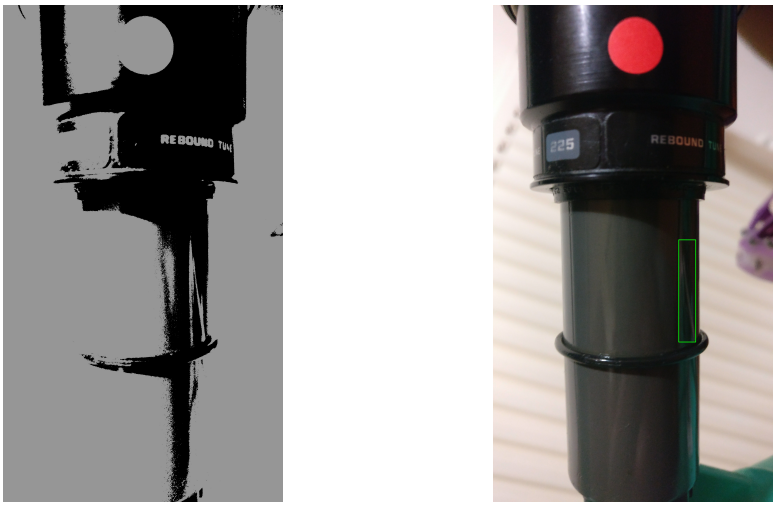
(c) Found contours from mask

(d) Bounding circle and box applied from contours

Figure 21: Processed images for finding reference point

**4.2.3.2 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 21.

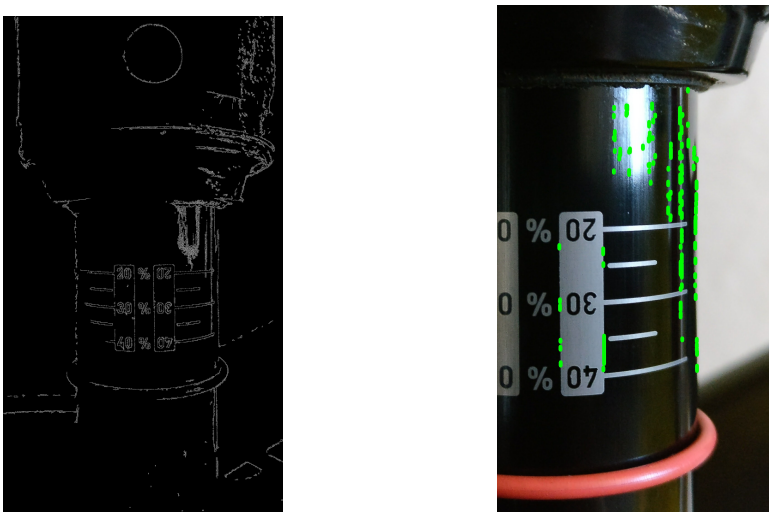
If a black O-ring is specified, an alternative process is used. Thresholding is applied to the image which emphasises any shadows or highlights on the shock shaft. As these are intersected by the O-ring it is simple to select a contour which ends at the O-ring to use as a measurement limit. This process is shown in Figure 22.



(a) Image thresholding (b) O-ring indicator

Figure 22: Locating process for black O-ring

**4.2.3.3 Measurement** With the pixels per millimetre metric produced and the measurement limit found, the sag under the current pressure is measured. First, edge detection is carried out on the original input image to highlight the lines of the shock, as shown in Figure 23a. Hough line transformation is then applied to locate vertical lines between the top of the shock shaft and the measurement limit. The difference in pixel count between the highest and lowest points of these lines represents the current sag measurement in the image and this can be converted to millimetres using the px/mm metric.



(a) Edge detected image (b) Vertical lines in measurement area

Figure 23: Measurement process using edge detection and Hough lines

**4.2.3.4 Equation** Once the measurements for the shock pressurised to 100 PSI and 150 PSI 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 code

for this can be seen in appendix J.3 lines 32-38. An example equation which is visually described in the chart shown in Figure 24 is only produced virtually and never output to the user.

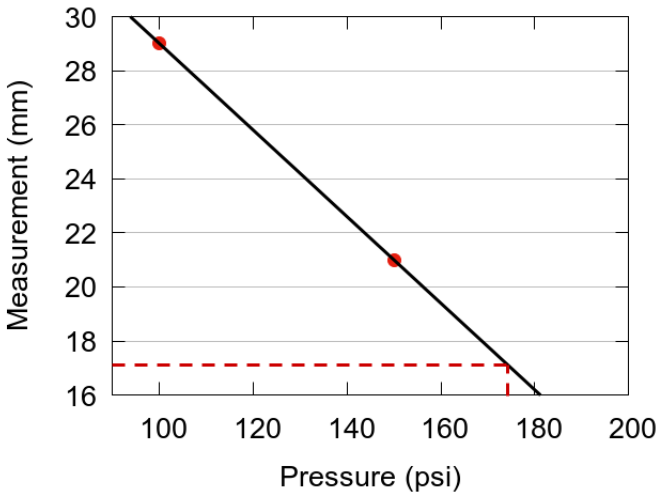



Figure 24: Example of linear equation plot

**4.2.3.5 Produce Setting** To produce the final setting the optimal distance that the shock should move to give the desired sag setting is first calculated using data input by the user. For example if the shock stroke is 57mm and the desired sag is 30% then the desired movement is 17.1mm. This and the equation produced in the previous section are used to calculate the optimal pressure. For example if the equation is $-5:017x + 246.426$ then the pressure required to give optimal sag is 161 PSI.

## 4.3 Evaluation

The following section will describe the results of applying the evaluation techniques outlined in section 3.7 to the final prototype application. The outcome of the evaluation gives an indication of how successful the application is at meeting its intended goals and the success of the methods used in the project.

### 4.3.1 Validation

Figure 25 shows an extract from the testing report produced by the Pycharm IDE. It includes each unit test for both classes and clearly indicates that the final prototype application successfully passed them all. These included checks to see whether the outputs produced by the application are within an expected range. By doing so this validates not only that the application works but that it also produces results within the anticipated range. The test code can be seen in appendix J.

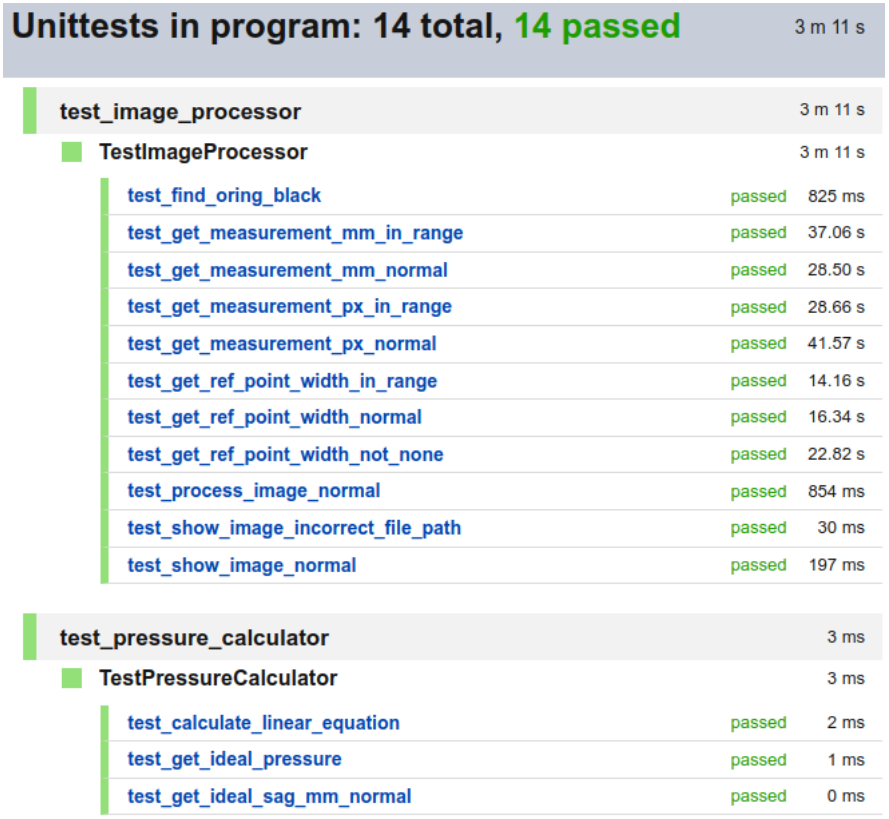



Figure 25: Unit test results

Figure 26 shows the test coverage report for the application. As stated in section 3.5.2 the goal for this project was 100% test coverage of the application. This has not been achieved but has come close with 97% of the application being covered.

The `main.py` file has no coverage as it is a script as opposed to a testable Class. This is not an issue as it contains the argument parsing and calls to other classes which are not vital to test as the Argparse module is well tested already and the lower level classes have their own testing. The `image_processor.py` class does not have full coverage as this includes lines which check if outputs provided by OpenCV are valid and

close the application if something has gone wrong. The conditions under which these lines are hit could not be recreated due to the irregularity with which problems occur. However testing these lines would not validate the application's normal functionality.

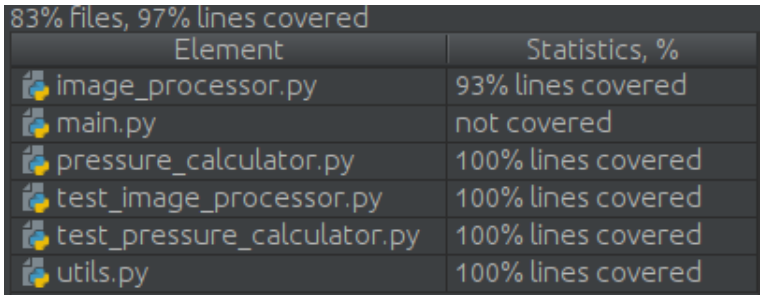



Figure 26: Test coverage results

### 4.3.2 Reliability and Accuracy

Table 4 shows the results of calculating uncertainty for the application under different conditions. The calculations were carried out for two different shocks with two different coloured O-rings, and for two sag settings on each. For the tests the application was run 10 times using the same images and arguments with every output collected, and used in the calculation.

Table 4: Table of uncertainty calculation results

| | **High Volume, Red O-Ring** | | **Low Volume, Black O-Ring** | |
|---|---|---|---|---|
| | **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** | $175 \pm 0.27$ | $160.64 \pm 0.17$ | $133.36 \pm 0.05$ | $120.49 \pm 0.03$ |

For each test the variances are low, (the highest being $\pm 0.27$), which indicates that the application is capable of consistently producing correct sag settings. If these uncertainty measures had been much higher then the settings produced by the application could not have been considered reliable. The calculated setting produced by the application is also rounded to the nearest whole number as increments of less than 1 PSI in a mountain bike shock have negligible effect. Even the most advanced, digital shock pumps do not display fractions of PSI and professional race teams do not make changes of less than 1 PSI. This knowledge further reinforces the capabilities of the

application.

The results for the shock with a red O-ring are slightly higher than those for the black O-ring. This is because the method for finding and applying a boundary to the red O-ring produces a larger variance in results than the method for finding and applying a boundary to the black O-ring. However, the minor difference between the results is insignificant as both colours of O-ring produce accurate results.

### 4.3.3 Comparison to Alternative Methods

As outlined in section 3.7.3, the results generated by the application were compared with the manual, trial and error, method of setting up sag. For this experiment the target sag was 30% of a 57mm shock stroke equalling a target measurement of 17.1mm. To begin the rider was weighed wearing their normal riding clothing and equipment with the result being approximately 160lbs. Following the manufacturer's recommended process, the shock was then pressurised to 160 PSI and loaded by the rider. Each of the subsequent steps followed in the trial and error process are shown in Table 5.

Table 5: Stages of manual sag setting process

| Stage | Pressure(PSI) | Outcome | Action |
|---|---|---|---|
| 1 | 160 | Too little | +5 |
| 2 | 165 | Too little | +20 |
| 3 | 185 | Too little | +5 |
| 4 | 190 | Too little | +10 |
| 5 | 200 | Hit target | N/A |

To produce the correct sag setting using the manual process required 5 separate adjustments each including pressurisation of the shock, weighting of the bike, and measurement of the marker O-ring. For this particular shock, small changes in pressure made no noticeable difference to the amount of movement so an additional 20 PSI was added in stage 2. Even though there is a chance that the manufacturer's recommended setting will be correct or very close, the majority of the time a manual set-up needs 4 or more adjustments. Conversely, using the application only involves 3 easy to follow stages and produced reliable results meaning optimal setup can be achieved quicker and with far less effort. The application also removes both the need to weigh the rider and to refer to the manufacturer's recommended pressure setting which further simplifies the process and saves time.

It should be noted that there is disparity between the pressure that results from following the manual process (seen in Table 5) and that produced by the application (seen in Table 4). As the images used during development of the application were of a shock that had not been weighted while the rider was wearing correct clothing and equipment, new images were produced to match the conditions during the manual process and the application retested. This produced a result of 176 PSI which was closer to, but still lower than, the manually produced result. However, when the shock is pressurised to this setting, the sag appears correct.

A manual setting was produced for the low volume shock to further investigate this issue. It was found that under these conditions both the manual and application's results were extremely close at approximately 120 PSI. As high volume shocks require much higher pressures than low volume shocks, it is believed that as the manual pressure of 200 PSI lies above the two pressures which the shock is pressurised to when using the application, this creates inaccurate results. For comparison, the low volume target pressure of 120 PSI lies between the 100 and 150 PSI required by the application. A solution to this will be discussed in section 5.3.

### 4.3.4 Professional Opinion

As outlined in section 3.7.4, once the application was complete the Mountain Bike Centre of Scotland were contacted to arrange a meeting with an individual from the industry to demonstrate the application and collect their feedback. The meeting was arranged with Geraint Florida-Jones of the Centre and Mark Ravilious, a mechanical engineer who has ventured into the design and manufacture of his own full suspension frame. Both Mark and Geraint are familiar with the current products that are similar to the application produced by this project such as Shockwiz (Quarq, 2017) and the Fox IRD application (Benedict, 2012).

Once Mark and Geraint were familiar with the context of the project and operation of the application, they were asked for their feedback. They both stated that the project had been well executed and the application was simple to operate. Both verified that the setting produced appeared correct for the given shock, bike, and rider's weight. Furthermore, both agreed that they would recommend the application to the target audience of beginner and intermediate riders though a more professional setup was required for those at the competitive end of the sport.

Both said that the application produced in this project should be released as a mobile application as this would make it much more user friendly compared to the prototype command line application. Additionally, both stated that more features should be added to provide a more rounded user experience and their suggestions will be explored in section 5.3. Mark questioned the application of a linear equation to a non-linear air shock though once the experiments into linearity, outlined in section 4.1.6, were explained he agreed that the shock could be considered linear within the range of its stroke that sag can be adjusted. A written statement from Geraint can be seen in appendix I.

The application was also demonstrated to the staff at Pedals[3] bike shop in Edinburgh. Here it received much the same feedback as it got during the meeting with Mark and Geraint; namely that it is a good execution of the concept and with more features would make a mobile application that is entirely suitable for the target demographic.

The positive reception that the application received from professionals within the cycling industry demonstrates that it is a suitable prototype which produces correct settings in an easy to use manner. The only issues suggested were the lack of additional features and functionality which is addressed in section 5.3.

[3] http://pedalsbikecare.co.uk/

# 5 Conclusions

This section will conclude the project by evaluating how well the original aims have been met to determine the success of the project, compare the prototype application to current products on the market, and outline any future work which can be potentially completed in subsequent projects. Finally a self appraisal will be carried out to discuss this author's performance during the project.

## 5.1 Meeting Aims

As a measure of success the original project aims, which were explained in section 1.3, can be examined to see if they have been met now the project is complete.

### 5.1.1 Aim 1: Literature Review

The first aim was to complete a literature review of mountain bike suspension and image analysis techniques including how image analysis is currently used in sports science. This has been met in section 2. This literature review presented the fundamentals of mountain bike suspension to provide insight into its operation as a way of building knowledge of the project context. Research into the various techniques of image analysis and their uses proved useful in deciding how the application would operate. This aided the development process, particularly when issues were encountered in finding the reference point and subsequently in gaining accurate measurements.

### 5.1.2 Aim 2: Prototype Application

The second aim was to implement the prototype application using methods identified and researched as part of the literature review. This aim has been met using methods selected from those outlined in section 3 with the resulting application documented in section 4.

The development approach chosen allowed for the project to run smoothly and produce the application on time. Although adopting an agile approach may have resulted in the earlier identification issues encountered during development by incorporating ongoing design and analysis techniques, it was felt that this would have put unnecessary pressure on the project in the form of excessive documentation and structure. The looser organic approach taken allowed for issues to be resolved, as they arose, to maintain momentum in the process.

Adopting an organic approach also allowed the project to continue despite not knowing the exact algorithmic approach the application would take. Given the limited time available it was felt that it was best to begin development as soon as possible and allow the project to adapt accordingly as time went on rather than rigorously structure the process. This lead to a change in the image analysis techniques used for reference point finding and a complete change, from trying to create a pressure per millimetre metric, to utilising two images and the linear equation. It is believed that had creative freedom been curtailed by including extensive analysis and exhaustive design into the process,

these solutions may not have been discovered. For larger projects with multiple stakeholders this would not be suitable but in this experimental context the approach worked well.

Using the Git version control system outlined in section 3.6.3 allowed for the project's documents and source code to be easily backed up and tracked. Indicated by the commit logs shown in appendix G, the system was well used with regular commits and numerous branches. The Gantt charts, described in section 3.6.6 and shown in appendix H, were regularly checked and updated throughout the project to track each stage and its completion. Multiple versions were produced as the chart was updated whenever the timeline was significantly adjusted, this included putting the project on hold for a period of 2 weeks and moving the deadline forward.

By producing a Python application as opposed to an Android mobile application, the project was able to produce a product with greater capabilities than it would have on the Android platform. Not having to produce a user interface and handle the difficulties of using OpenCV on Android allowed more time to research and develop more advanced image analysis techniques and greater sophistication in the underlying calculation. By implementing unit tests it could be confirmed that the application was functioning correctly throughout development. A push could have been made for 100% test coverage but as described in section 4.3.1, this was not vital.

### 5.1.3 Aim 3: Evaluation

The third aim was to evaluate the success and appropriateness of the produced application, this has been completed and is documented in section 4.3. By evaluating a selection of metrics it has been proven that the application functions as expected, produces reliable results, and is well received by industry professionals. Additionally this evaluation process has shown that the project was capable of producing the original concept for the application and has created a basis for future work.

### 5.1.4 Aim 4: Conclusions

The final aim was to present conclusions about the project's successes and shortcomings which is shown in the present section.

## 5.2 Comparison to similar Products

Due to the high price and limited availability of the Shockwiz device was not available for comparison. Additionally as the Fox IRD application is only offered on Apple's iOS, it was not available for testing and comparison. However a comparison can still be made by using information about each product and anecdotal experience from users collected from various online discussion groups.

### 5.2.1 Shockwiz

The Shockwiz data logging device (Quarq, 2017) collects and analyses data about front and rear suspension units while the bike is being ridden and provides users with

recommended adjustments through a mobile application. In comparison to the application produced during this project it provides rebound and compression settings in conjunction with the basic air pressure. Each adjustment works on a sliding scale as opposed to set numbers which is helpful when tuning suspension.

Due to the number of settings and level of detail provided to the user, alongside the £359 price, it is clear that the Shockwiz system is aimed at intermediate to expert riders wanting to tune their suspension for various trails. The target audience for this project is beginner riders wanting to produce a baseline setup with minimal effort.

### 5.2.2 Fox IRD

The Fox Intelligent Ride Dynamics application Fox Factory Inc. (2015) uses a smartphone's camera to produce a sag setting for a fork or shock. This is carried out in a similar way to the application delivered by this project. Additionally the IRD application can suggest a rebound setting dynamically, although there is no documentation available as to how this is produced.

While this project's application has been tested on shocks from two manufacturers, the Fox's application is tied to their own suspension units. A further caveat is that it is restricted to their 2013 model year. This means that the application cannot be used with either older and newer suspension which has resulted in complaints from customers. The locking of the application to one manufacturer and model year suggests that the application was created as an experiment rather than a viable product.

## 5.3 Future Work

### 5.3.1 Mobile Application

A clear section of work would be to create a smartphone application based around the one produced in this project. This would move the application closer to the original vision of this project and allow for a simpler user experience by using the smartphone to capture the images, complete the analysis, and provide the feedback.

This could be completed by including the source code from this application by means of a wrapper (Kivy, 2015) or scripting layer (Kohler, 2015). However there are issues with both of these methods. Development by the original team on the Python-for-Android wrapper has stopped which means remaining bugs with the system will not be fixed. This is also the case with the Android Scripting Layer which was also never taken further than an alpha level application so it is likely to be unstable and incapable of providing the required functionality.

An alternative method would be to use the OpenCV package for Android and recreate the algorithms from this project natively, as previously described in section 3.3.2, though this also presents difficulties. With enough time and resources available the issues encountered in this project may be rectified with a correctly setup and configured environment. Alternatively, the other image analysis libraries available for Android could be considered.

### 5.3.2 Expanded Functionality

The functionality of the application could also be expanded in line with the feedback provided by the industry experts. Currently it can process images of rear shocks but this should be expanded to include the fork, allowing users to set up both suspension units. This would be a case of adapting the current image analysis to work on the different images. The application has also been tested on two varieties of shock, this should be expanded to other manufacturers and models in a variety of situations and adapted accordingly to assure its capabilities.

Secondly the application should be adapted to work with coil shocks as opposed to just air suspension. This could be done by locating the bolts which hold the shock into the frame and measuring the distance between them as coil shocks do not possess an O-ring. This could be simple as they are circular though these bolts are commonly concealed from view by other components. A further difficulty arises as coil shocks must be measured under load due to the lack of O-ring, so the rider would have to be seated on the bike when the image is taken.

As discussed in section 4.3.3, the disparity between the manually produced and computationally produced pressures should be resolved. To do so the application should be altered to allow the user to specify which pressures the images are taken at, as opposed to being locked to 100PSI and 150PSI. This would make the application adaptable for higher volume shocks where pressures can be as high as 350PSI.

Finally the application could be expanded to suggest compression and rebound damping settings alongside the current sag setting. This would allow the user to setup every aspect of their suspension before riding for the first time. One method would be to dynamically predict a suitable setting using the produced sag setting, intended riding style, and current suspension design. Alternatively, an online database of suggested settings could be collated and queried with the user's data. This database could be included as part of a wider scoped project including a website allowing users to look up settings from those contributed by other users rather than just using the application.

## 5.4 Self Appraisal

This has been the largest project I have ever undertaken in terms of timeline and amount of research required and at the outset I was unsure as to whether I could complete it to the required standard. Many factors have interfered with the amount of time available to work on this project such as other university commitments and repeatedly taking time out to attend job interviews. Such interruptions made it difficult to maintain focus particularly once the development stage started. However keeping a mental check on progress and routinely using documentation and project management tools to constantly juggle priorities and the amount of time available meant I was always able to maintain good momentum.

Midway through the development stage, I had to refer to the Git commit logs to trace the source of an issue in the application. While doing this, I found that the commit messages were not sufficiently descriptive of the work that had already been completed

which made it difficult and time consuming to locate the issues. Had it been someone other than myself carrying out this investigation, this lack of description would have been a serious issue. Subsequently I made commit messages considerably more descriptive so the detail was always available should I need to refer to them again. The same issues apply to the level of detail recorded in project documentation which although suitable for a small, solo project would not be appropriate in larger projects involving more dependencies and a larger number of stakeholders.

Overall I believe I have performed well in completing this project and successfully delivering an application that meets its requirements. Constant interruptions and a string of issues, such as the initially inability to detect the reference point, meant time was sometimes scarce even when it was most needed in researching and testing new approaches to problems. However, in retrospect, I am pleased to have been able to apply the methods and techniques I have acquired during my course to overcome the many issues encountered during this project.

# References


Adams, J., Khan, H. T., Raeside, R., & White, D. I. (2007). *Research methods for graduate business and social science students*. SAGE publications India.

Adobe. (2017). *Adobe photoshop.* Retrieved 02/04/2017, from http://www.adobe.com/uk/products/photoshop.html

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

Aydin, M. N., Harmsen, F., Van Slooten, K., & Stegwee, R. (2004). An agile information systems development method in use. *Turkish Journal of Electrical Engineering & Computer Sciences*, *12*(2), 127–138.

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

Bittner, K. (2002). *Use case modeling*. Addison-Wesley Longman Publishing Co., Inc.

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

Bradski, G., & Kaehler, A. (2008). *Learning opencv: Computer vision with the opencv library*. " O'Reilly Media, Inc.".

Burdette, D., & Thompson, T. R. (2012). Ideal gas law.

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

Cane Creek. (2016). *How to choose the right rear shock.* Retrieved 02/04/2017, from https://www.canecreek.com/culture/blog-news/how-to-choose-the-right-rear-shock

Cao, L., & Ramesh, B. (2008). Agile requirements engineering practices: An empirical study. *IEEE software*, *25*(1).

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

Fox Factory Inc. (2015). *Fox ird application.* Retrieved 05/04/2017, from http://www.ridefox.com/content.php?c=app

Garvin, J. B., Malin, M. C., & Minitti, M. E. (2014, March 17). *Sedimentology of martain gravels from mardi twilight imaging: Techniques* (Tech. Rep. No. GSFC-E-DAA-TN13878). Retrieved 04/10/2016, from http://hdl.handle.net/2060/20150001290

Goodyear. (2014). Basic principles of air springs. Retrieved 02/04/2017, from http://www.lhtech.com/pdf/automation/pneumatic/Goodyear-Air-Spring-Data.pdf

Haralock, R. M., & Shapiro, L. G. (1991). *Computer and robot vision*. Addison-Wesley Longman Publishing Co., Inc.

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

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology*, *51*(1), 7–15.

Kivy. (2015). *Python-for-android.* Retrieved 05/04/2017, from https://github.com/kivy/python-for-android

Klarica, A. (2001). Performance in motor sports. *British journal of sports medicine*, *35*(5), 290–291.

Kohler, D. (2015). *Android scripting layer.* Retrieved 05/04/2017, from https://github.com/damonkohler/sl4a

Kuhlman, D. (2009). *A python book: Beginning python, advanced python, and python*

*exercises*. Dave Kuhlman.
Leavers, V. F. (1992). *Shape detection in computer vision using the hough transform*. Springer-Verlag.
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) doi: 10.1061/(ASCE)0887-3801(1999)13:2(88)
MatLab. (2017). *Edge detection.* Retrieved 02/04/2017, from https://uk.mathworks.com/discovery/edge-detection.html
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
OpenCV.org. (2017). *Opencv.* Retrieved 02/04/2017, from http://opencv.org/about.html
Patil, A., & Dhanvijay, M. (2015). Blob detection technique using image processing for identificationu of machine printed characters. *International Journal of Innovations in Engineering Research and Technology*, *2*.
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
Pythonprogramming. (2016). *Pythonprogramming.net.* Retrieved 02/04/2017, from www.pythonprogramming.net
Quarq. (2017). *Shockwiz.* Retrieved 05/04/2017, from https://www.quarq.com/shockwiz/
Rising, L., & Janoff, N. S. (2000). The scrum software development process for small teams. *IEEE software*, *17*(4), 26–32.
Rosebrock, A. (2017). *Pyimagesearch.* Retrieved 02/04/2017, from www.pyimagesearch.com
Rychlewski, J. (1984). On hooke's law. *Journal of Applied Mathematics and Mechanics*, *48*(3), 303–314.
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
Segers, J. (2008). *Analysis techniques for racecar data acquisition* (Tech. Rep.). SAE Technical Paper.
Shneiderman, B. (2010). *Designing the user interface: strategies for effective human-computer interaction*. Pearson Education India.
Staff, B. (2015). Yt industries launches consumer-direct sales in north amer-

ica, 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 parameters 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.bmj.com.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

**API** Application Programming Interface.

**BLOB** Binary Large OBject.

**EXIF** Exchangeable Image File Format.

**FS** Full Suspension.

**HT** Hard Tail.

**IA** Image Analysis.

**IDE** Integrated Developers Environments.

**NDK** Native Development Kit.

**SDK** Software Development Kit.

**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.

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

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

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

**Travel** The distance a wheel can move before bottoming out.

# A Initial Project Overview

Joe Barrett 40117680

## Initial Project Overview
## SOC10101 Honours Project (40 Credits)

### Title of Project:

Calculation of Mountain Bike Suspension Setup through Mobile Image Processing

### Overview of Project Content and Milestones

Due to the lack of knowledge and complexity regarding rear suspension setup on mountain bikes, many riders have an improper setup which can potentially lead to injury. The purpose of this project will be to design and produce a mobile application capable of providing a suggested rear suspension setup for the user based on images of the bike frame and user provided information.

The project will consist of research into current image processing uses, techniques, and mobile applications which use image processing. Design and implementation of the prototype mobile application. Finally, an evaluation of the prototype application.

### The Main Deliverable(s):

- A literature review of image processing techniques
- An analysis of currently available applications and products in the related area
- A prototype mobile application capable of suggesting a rear suspension setup for the user
- A critical evaluation of the prototype application against currently available applications

### The Target Audience for the Deliverable:

Entry level to intermediate mountain bikers with little to no knowledge of the rear suspension setup process.

### The Work to be Undertaken:

- Investigation into image processing techniques, uses, and current applications
- Produce a literature review of image processing techniques
- Produce a design for the prototype mobile application
- Implement the design into a working prototype application
- Evaluate the prototype

Joe Barrett 40117680

## Additional Information / Knowledge Required:

- Knowledge of OpenCV library or alternative image processing libraries
- Improved knowledge of Android™ framework and programming
- Improved knowledge of mountain bike rear suspension setup

## Information Sources that Provide a Context for the Project:

- OpenCV Image Processing Library for Android™, www.opencv.org
- Introduction to Video and Image Processing Building Real Systems and Applications, Thomas B Moeslund, 2012
- SussMyBike Data Acquisition - Eurobike 2016, www.pinkbike.com/news/sussmybike-data-acquisition-eurobike-2016

## The Importance of the Project:

Correct suspension setup on a mountain bike improves its ride-ability and reduces excessive wear and tear on components leading to further enjoyment of the sport. After conversations with The Mountain Bike Research Centre of Scotland it has been shown that a drastically incorrect setup can cause the rider to crash which may lead to injury and can potentially be fatal.

## The Key Challenge(s) to be Overcome:

- Personal understanding of image processing techniques
- Use of image processing to calculate valuable metrics
- Carrying out a critical and scientific evaluation of the prototype

# B Week 9 Report

## SOC10101 Honours Project (40 Credits)

## Week 9 Report

**Student Name:** Joe Barrett

**Supervisor**: Brian Davison

**Second Marker**: Simon Wells

**Date of Meeting**: 02.11.2016

Can the student provide evidence of attending supervision meetings by means of project diary sheets or other equivalent mechanism? **yes**

If not, please comment on any reasons presented

Please comment on the progress made so far

**Good progress made. Lit. review is reasonably good but there is plenty of room for more academic references to underpin the contribution made and to fix the position of this project in relation to other uses of computer vision technology.**

Is the progress satisfactory? **yes**

Can the student articulate their aims and objectives? **yes**

If yes then please comment on them, otherwise write down your suggestions.

**The objectives stem from the students own project ideas and are achievable. There is some element of risk related to the application of CV technologies but these can be managed.**

Does the student have a plan of work? **yes**

If yes then please comment on that plan otherwise write down your suggestions.

**The plan is good, comprehensive and easily achievable to a high quality if Joe continues to work at the current pace.**

Does the student know how they are going to evaluate their work? **yes**

If yes then please comment otherwise write down your suggestions.

**Functional testing & expert evaluation with mountain bike centre**

Any other recommendations as to the future direction of the project

* Please circle one answer; if **no** is circled then this **must** be amplified in the space provided

Signatures: Supervisor Second Marker SW

Student

Please give the student a photocopy of this form immediately after the review meeting; the original should be lodged in the School Office with Leanne Clyde

* Please circle one answer; if **no** is circled then this **must** be amplified in the space provided

# C Android Experiments

## C.1 Table of Android Experiments

| Experiment | Purpose | Works | Issues | Files | LOC |
|---|---|---|---|---|---|
| Hello CV | Introduction to the android OpenCV library. Displays camera feed with fps. | ✓ | • Not fullscreen<br>• Incorrect orientation | 2 | 101 |
| 15Tile | Sliding tile game. Uses camera feed as puzzle. | ✓ | • Not fullscreen | 3 | 492 |
| Blob Detection | Demonstrates blob detection. Runs blob detection on tapped area from camera. | X | • Crash on screen tap | 3 | 311 |
| Face Detection | Detects faces in camera view. Puts boundary around detected faces. | X | • Crash on load | 3 | 279 |

## C.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
   ↪  CameraBridgeViewBase.CvCameraViewListener2{
12
13     private static String TAG = "Main Activity";
14     private CameraBridgeViewBase mOpenCvCameraView;
15
16     private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
17         @Override
18         public void onManagerConnected(int status) {
19             switch (status){
20                 case LoaderCallbackInterface.SUCCESS:{
21                     Log.i(TAG, "Open CV loaded successfully");
22                     mOpenCvCameraView.enableView();
23                 }break;
```

```
                default:{
                    super.onManagerConnected(status);
                }break;
            }
        }
    };

    @Override
    public void onResume(){
        super.onResume();
        OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0, this,
         ↪  mLoaderCallback);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        Log.i(TAG, "called OnCreate");
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.HelloOpenCvView);
        mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);
        mOpenCvCameraView.setCvCameraViewListener(this);
    }

    @Override
    public void onPause(){
        super.onPause();
        if (mOpenCvCameraView != null){
            mOpenCvCameraView.disableView();
        }
    }

    @Override
    public void onDestroy(){
        super.onDestroy();
        if (mOpenCvCameraView != null){
            mOpenCvCameraView.disableView();
        }
    }

    @Override
    public void onCameraViewStarted(int width, int height) {

    }

    @Override
    public void onCameraViewStopped() {

    }

    @Override
    public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
        return inputFrame.rgba();
    }
}
```

## C.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
    ↪  CameraBridgeViewBase.CvCameraViewListener, View.OnTouchListener {
23     private static final boolean AUTO_HIDE = true;
24     private static final int AUTO_HIDE_DELAY_MILLIS = 3000;
25     private static final int UI_ANIMATION_DELAY = 300;
26     private final Handler mHideHandler = new Handler();
27     private View mContentView;
28     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
        ↪  View.OnTouchListener() {
65          @Override
66          public boolean onTouch(View view, MotionEvent motionEvent) {
67              if (AUTO_HIDE) {
68                  delayedHide(AUTO_HIDE_DELAY_MILLIS);
69              }
70              return false;
71          }
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
                ↪  manager for initialization");
140             OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0, this,
                ↪  mLoaderCallback);
141         } 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 ouside 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];
```

```
                mIndexes[idxToSwap] = touched;
            }
        }
    }

    private void drawGrid(int cols, int rows, Mat drawMat) {
        for (int i = 1; i < GRID_SIZE; i++) {
            Imgproc.line(drawMat,
                    new Point(0, i * rows / GRID_SIZE),
                    new Point(cols, i * rows / GRID_SIZE),
                    new Scalar(0, 255, 0, 255),
                    3);
            Imgproc.line(drawMat,
                    new Point(i * cols / GRID_SIZE, rows),
                    new Point(i * cols / GRID_SIZE, rows),
                    new Scalar(0, 255, 0, 255),
                    3);
        }
    }

    private static void shuffle(int[] array) {
        for (int i = array.length; i > 1; i--) {
            int temp = array[i - 1];
            int randIx = (int) (Math.random() * i);
            array[i - 1] = array[randIx];
            array[randIx] = temp;
        }
    }

    private boolean isPuzzleSolvable() {
        int sum = 0;
        for (int i = 0; i < GRID_AREA; i++) {
            if (mIndexes[i] == GRID_EMPTY_INDEX) sum += (i / GRID_SIZE) + 1;
            else {
                int smaller = 0;
                for (int j = i+1; j < GRID_AREA; j++){
                    if (mIndexes[j] < mIndexes[i]) smaller++;
                }
                sum += smaller;
            }
        }
        return sum % 2 == 0;
    }

}
```

## C.4 BLOB Analysis

```
package com.example.joe.blob_analysis;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.MotionEvent;
import android.view.SurfaceView;
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
    ↪  View.OnTouchListener, CameraBridgeViewBase.CvCameraViewListener2 {
32     private static final String TAG = "Activity: ";
33
34     private boolean mIsColorSelected = false;
35     private Mat mRgba;
36     private Scalar mBlobColorRgba;
37     private Scalar mBlobColorHsv;
38     private ColorBlobDetector mDetector;
39     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)
            ↪  findViewById(R.id.color_blob_detection_activity_surface_view);
75          mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);
76          mOpenCvCameraView.setCvCameraViewListener(this);
77      }
78
79      @Override
80      public void onPause() {
81          super.onPause();
82          if (mOpenCvCameraView != null) {
83              mOpenCvCameraView.disableView();
84          }
85      }
86
87      @Override
88      public void onResume() {
89          super.onResume();
90          if (!OpenCVLoader.initDebug()) {
91              Log.d(TAG, "onResume: OpenCV not found. Using Manager");
92              OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0, this,
                ↪  mLoaderCallback);
93          } else {
94              Log.d(TAG, "onResume: OpenCV found in package.");
95              mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
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 -
            ↪ touchedRect.x;
140         touchedRect.height = (y + 4 < rows) ? y + 4 - touchedRect.y : rows -
            ↪ touchedRect.y;
141
142         Mat touchedRegionRgba = mRgba.submat(touchedRect);
143
144         Mat touchedRegionHsv = new Mat();
145         Imgproc.cvtColor(touchedRegionRgba, touchedRegionHsv,
            ↪ 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] + ", " +
            ↪ mBlobColorRgba.val[1] +", " + mBlobColorRgba.val[2] + ", " +
            ↪ mBlobColorRgba.val[3] + ")");
156
157         mDetector.setHsvColor(mBlobColorHsv);
158
159         Imgproc.resize(mDetector.getSpectrum(), mSpectrum, SPECTRUM_SIZE);
160         mIsColorSelected = true;
161
162         touchedRegionRgba.release();
163         touchedRegionHsv.release();
164
165         return false;
166     }
167
168     public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
169         mRgba = inputFrame.rgba();
170         if (mIsColorSelected) {
171             mDetector.process(mRgba);
172             List<MatOfPoint> contours = mDetector.getContours();
173             Log.e(TAG, "onCameraFrame: Contours count: " + contours.size());
174             Imgproc.drawContours(mRgba, contours, -1, CONTOUR_COLOR);
```

```
175             Mat colorLabel = mRgba.submat(4, 68, 4, 68);
176             colorLabel.setTo(mBlobColorRgba);
177             Mat spectrumLabel = mRgba.submat(4, 4 + mSpectrum.rows(), 70, 70 +
                ↪  mSpectrum.cols());
178             mSpectrum.copyTo(spectrumLabel);
179         }
180
181         return mRgba;
182     }
183
184     private Scalar convertScalarHsv2Rgba(Scalar hsvColor) {
185         Mat pointMatRgba = new Mat();
186         Mat pointMatHsv = new Mat(1, 1, CvType.CV_8UC3, hsvColor);
187         Imgproc.cvtColor(pointMatHsv, pointMatRgba, Imgproc.COLOR_HSV2RGB_FULL, 4);
188         return new Scalar(pointMatRgba.get(0, 0));
189     }
190
191 }
```

```
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] -
            ↪  mColorRadius.val[0] : 0;
41          double maxH = (hsvColor.val[0] + mColorRadius.val[0] <= 255) ? hsvColor.val[0]
            ↪  + mColorRadius.val[0] : 255;
42
43          mLowerBound.val[0] = minH;
44          mUpperBound.val[0] = maxH;
45
46          mLowerBound.val[1] = hsvColor.val[1] - mColorRadius.val[1];
47          mUpperBound.val[1] = hsvColor.val[1] + mColorRadius.val[1];
48
49          mLowerBound.val[2] = hsvColor.val[2] - mColorRadius.val[2];
50          mUpperBound.val[2] = hsvColor.val[2] + mColorRadius.val[2];
51
52          mLowerBound.val[3] = 0;
53          mUpperBound.val[3] = 255;
54
55          Mat spectrumHsv = new Mat(1, (int)(maxH-minH), CvType.CV_8UC3);
56
57          for (int j = 0; j < maxH-minH; j++) {
58              byte[] tmp = {(byte)(minH+j), (byte)255, (byte)255};
59              spectrumHsv.put(0, j, tmp);
60          }
61
62          Imgproc.cvtColor(spectrumHsv, mSpectrum, Imgproc.COLOR_HSV2RGB_FULL);
63      }
64
65      public Mat getSpectrum() {
66          return mSpectrum;
67      }
68
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,
            ↪  Imgproc.RETR_EXTERNAL, Imgproc.CHAIN_APPROX_SIMPLE);
83
84          double maxArea = 0;
85          Iterator<MatOfPoint> each = contours.iterator();
86          while (each.hasNext()) {
87              MatOfPoint wrapper = each.next();
88              double area = Imgproc.contourArea(wrapper);
89              if (area > maxArea) {
90                  maxArea = area;
91              }
92          }
93
```

```
94          mContours.clear();
95          each = contours.iterator();
96          while (each.hasNext()) {
97              MatOfPoint contour = each.next();
98              if (Imgproc.contourArea(contour) > mMinContourArea*maxArea) {
99                  Core.multiply(contour, new Scalar(4,4), contour);
100                 mContours.add(contour);
101             }
102         }
103     }
104
105     public List<MatOfPoint> getContours(){return mContours;}
106
107 }
```

## C.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(),
           ↪  faces.getNativeObjAddr());
29     }
30
31     public void release() {
32         nativeDestroyObject(mNativeObj);
33         mNativeObj = 0;
34     }
35
36     private long mNativeObj = 0;
37
38     private static native long nativeCreateObject(String cascadeName, int minFaceSize);
```

```
39      private static native void nativeDestroyObject(long thiz);
40      private static native void nativeStart(long thiz);
41      private static native void nativeStop(long thiz);
42      private static native void nativeSetFaceSize(long thiz, int size);
43      private static native void nativeDetect(long thiz, long inputImage, long faces);
44  }
```

```
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
    ↪  CameraBridgeViewBase.CvCameraViewListener2 {
33
34      private static final String TAG             = "FrActivity";
35      private static final Scalar FACE_RECT_COLOR = new Scalar(0,255,0,255);
36      private static final int    JAVA_DETECTOR   = 0;
37      private static final int    NATIVE_DETECTOR = 1;
38
39      private MenuItem mItemFace50;
40      private MenuItem mItemFace40;
41      private MenuItem mItemFace30;
42      private MenuItem mItemFace20;
43      private MenuItem mItemType;
44
45      private Mat                     mRgba;
46      private Mat                     mGray;
47      private File                    mCascadeFile;
48      private CascadeClassifier       mJavaDetector;
49      private DetectionBasedTracker   mNativeDetector;
```

```
50
51      private int          mDetectorType = JAVA_DETECTOR;
52      private String[]     mDetectorName;
53
54      private float   mRelativeFaceSize = 0.2f;
55      private int     mAbsoluteFaceSize = 0;
56
57      private CameraBridgeViewBase mOpenCvCameraView;
58
59      private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
60          @Override
61          public void onManagerConnected(int status) {
62              switch (status){
63                  case LoaderCallbackInterface.SUCCESS:{
64                      Log.i(TAG, "onManagerConnected: OpenCV loaded correctly");
65                      System.loadLibrary("detection_based_tracker");
66                      try{
67                          InputStream is =
                             ↪  getResources().openRawResource(R.raw.lbpcascade_frontalface);
68                          File cascadeDir = getDir("cascade", Context.MODE_PRIVATE);
69                          mCascadeFile = new File(cascadeDir,
                             ↪  "lbpcascade_frontalface.xml");
70                          FileOutputStream os = new FileOutputStream(mCascadeFile);
71
72                          byte[] buffer = new byte[4096];
73                          int bytesRead;
74                          while ((bytesRead = is.read(buffer))!=-1){
75                              os.write(buffer, 0, bytesRead);
76                          }
77                          is.close();
78                          os.close();
79
80                          mJavaDetector = new
                             ↪  CascadeClassifier(mCascadeFile.getAbsolutePath());
81                          if (mJavaDetector.empty()){
82                              Log.e(TAG, "onManagerConnected: Failed to load cascade
                                 ↪  classifier");
83                              mJavaDetector = null;
84                          }else{
85                              Log.i(TAG, "onManagerConnected: Loaded cascade classifier
                                 ↪  from "+mCascadeFile.getAbsolutePath());
86                          }
87                          mNativeDetector = new
                             ↪  DetectionBasedTracker(mCascadeFile.getAbsolutePath(), 0);
88                          cascadeDir.delete();
89                      }catch (IOException e){
90                          e.printStackTrace();
91                          Log.e(TAG, "onManagerConnected: Failed to load cascade.
                             ↪  Exception thrown: "+e);
92                      }
93
94                      mOpenCvCameraView.enableView();
95                  }break;
96                  default:{
97                      super.onManagerConnected(status);
98                  }break;
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)
            ↪  findViewById(R.id.fd_activity_surface_view);
117         mOpenCvCameraView.setVisibility(CameraBridgeViewBase.VISIBLE);
118         mOpenCvCameraView.setCvCameraViewListener(this);
119     }
120
121     @Override
122     public void onPause(){
123         super.onPause();
124         if (mOpenCvCameraView != null) mOpenCvCameraView.disableView();
125     }
126
127     @Override
128     public void onResume(){
129         super.onResume();
130         if (!OpenCVLoader.initDebug()){
131             Log.d(TAG, "onResume: Internal OpenCV lib not found. Using OpenCV manager
                ↪  for initialization");
132
                ↪  OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0,this,mLoaderCallback);
133         }else{
134             Log.d(TAG, "onResume: OpenCV lib found inside package.");
135             mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
136         }
137     }
138
139     public void onDestroy(){
140         super.onDestroy();
141         mOpenCvCameraView.disableView();
142     }
143
144     public void onCameraViewStarted(int width, int height){
145         mGray = new Mat();
146         mRgba = new Mat();
147     }
148
149     public void onCameraViewStopped(){
150         mGray.release();
151         mRgba.release();
152     }
153
154     public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame){
155         mRgba = inputFrame.rgba();
```

```
156             mGray = inputFrame.gray();
157
158             if (mAbsoluteFaceSize == 0){
159                 int height = mGray.rows();
160                 if (Math.round(height * mRelativeFaceSize)>0){
161                     mAbsoluteFaceSize = Math.round(height * mRelativeFaceSize);
162                 }
163                 mNativeDetector.setMinFaceSize(mAbsoluteFaceSize);
164             }
165
166             MatOfRect faces = new MatOfRect();
167
168             if (mDetectorType == JAVA_DETECTOR){
169                 if (mJavaDetector != null){
170                     mJavaDetector.detectMultiScale(mGray,
171                             faces,
172                             1.1,
173                             2,
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(),
                    ↪  FACE_RECT_COLOR, 3);
189
190             return mRgba;
191         }
192
193         @Override
194         public boolean onCreateOptionsMenu(Menu menu){
195             Log.i(TAG, "onCreateOptionsMenu: called");
196             mItemFace50 = menu.add("Face size 50%");
197             mItemFace40 = menu.add("Face size 40%");
198             mItemFace30 = menu.add("Face size 30%");
199             mItemFace20 = menu.add("Face size 20%");
200             mItemType   = menu.add(mDetectorName[mDetectorType]);
201             return true;
202         }
203
204         private void setMinFaceSize(float faceSize){
205             mRelativeFaceSize = faceSize;
206             mAbsoluteFaceSize = 0;
207         }
208
209         private void setDetectorType(int type){
210             if (mDetectorType != type){
211                 mDetectorType = type;
212                 if (type == NATIVE_DETECTOR){
```

```
213                 Log.i(TAG, "setDetectorType: Detection based tracker enabled");
214                 mNativeDetector.start();
215             }else{
216                 Log.i(TAG, "setDetectorType: Cascade detector enabled");
217                 mNativeDetector.stop();
218             }
219         }
220     }
221 }
```

# D Python Experiments

## D.1 Table of Python Experiments

| Experiment | Purpose | Works | Issues | Files | LOC |
|---|---|---|---|---|---|
| Find Game | Identify red game cartridge out of 3. Displays boundary around correct part of image. | ✓ | N/A | 1 | 18 |
| Threshold Methods | Demonstrates various thresholding methods on an image. Original image text unreadable but clear after techniques are applied. | ✓ | N/A | 1 | 14 |
| Image Operations | Moves parts of an image to other locations using Arrays. Demonstrates how pixel data is stored. | ✓ | N/A | 1 | 15 |
| Distance to Camera | Calculates distance to the camera from an identified object. Displays various distances using 3 images. | ✓ | N/A | 1 | 38 |

## D.2 find_game.py

```
import numpy as np
import cv2

image = cv2.imread('games.jpg')

upper = np.array([65, 65, 255])
lower = np.array([0, 0, 200])
mask = cv2.inRange(image, lower, upper)

cnts, _ = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
c = max(cnts, key=cv2.contourArea)

peri = cv2.arcLength(c, True)
approx = cv2.approxPolyDP(c, 0.05 * peri, True)

cv2.drawContours(image, [approx], -1, (0,255,0), 4)
cv2.imshow('Image', image)
cv2.waitKey(0)
```

## D.3 thresholding.py

```
import cv2
import numpy as np

img = cv2.imread('bookpage.jpg')
grayscaled = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
_, threshold = cv2.threshold(img, 12, 255, cv2.THRESH_BINARY)
```

```
_, gs_threshold = cv2.threshold(grayscaled, 10, 255, cv2.THRESH_BINARY)
adaptive = cv2.adaptiveThreshold(grayscaled, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
    cv2.THRESH_BINARY, 115, 1)
cv2.imshow('original', img)
cv2.imshow('threshold', threshold)
cv2.imshow('grayscaled', gs_threshold)
cv2.imshow('adaptive', adaptive)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## D.4 img_ops.py

```
import cv2
import numpy as np

img = cv2.imread('watch.jpg', cv2.IMREAD_COLOR)
px = img[55, 55]
img[55, 55] = [255, 255, 255]
px = img[55, 55]
print(px)
px = img[100:150, 100:150]
print(px)

watch_face = img[37:111, 107:194]
img[0:74, 0:87] = watch_face
cv2.imshow('image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## D.5 distance_to_camera.py

```
import numpy as np
import cv2


def find_marker(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (5, 5), 0)
    edged = cv2.Canny(gray, 35, 125)

    (cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
    c = max(cnts, key=cv2.contourArea)

    return cv2.minAreaRect(c)


def distance_to_camera(knownWidth, focalLength, perWidth):
    return (knownWidth * focalLength) / perWidth


KNOWN_DISTANCE = 24.0
KNOWN_WIDTH = 11.0
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,
                    ↪  2.0, (0, 255, 0), 3)
37     cv2.imshow('image', image)
38     cv2.waitKey(0)
```

# E 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
       ↪  mm
38     # Do for hor and ver
39     horizontal_measurement = None
40     vertical_measurement = None
41
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     return exif
```

# F Development Log

| 31/01/2017 | |
|---|---|
| **Tasks** | • Created first commit of codebase<br>• Removed practice files |
| **Issues** | N/A |

| 01/02/2017 | |
|---|---|
| **Tasks** | • Added arguments using argparse<br>• Added functions to take EXIF data from image, removed for ref point version<br>• Started ref point finding algorithm using own image with manufactured ref point |
| **Issues** | • Ref point finding not functional |

| 03/02/2017 | |
|---|---|
| **Tasks** | • Completed ref point finding and measuring<br>• Added calculations to produce pressure setting from measurement |
| **Issues** | • Suggested pressure setting incorrect, issue with calculation |

| 06/02/2017 | |
|---|---|
| **Tasks** | • Added debug printing to program<br>• Fixed calculations to produce more acceptable suggested pressrure |
| **Issues** | • Suggested pressure setting incorrect |

| 07/02/2017 | |
|---|---|
| **Tasks** | • Created project in PyCharm for better management |
| **Issues** | N/A |

| 08/02/2017 | |
|---|---|
| | |

| | |
|---|---|
| **Tasks** | • Created unit test for single function |
| **Issues** | • Not fully covered |

| **09/02/2017** | |
|---|---|
| **Tasks** | • Added more unit tests to suite<br>• Fixed issue with calculations |
| **Issues** | • No full coverage |

| **10/02/2017** | |
|---|---|
| **Tasks** | • Completed unit tests for full coverage of class |
| **Issues** | N/A |

| **12/02/2017** | |
|---|---|
| **Tasks** | • Changed test suite to use unittest2 for access to assertRaises and assertIsInstance<br>• Adjusted class to contain protected functions and added API function with test |
| **Issues** | • Unsure that psi/mm method will produce correct settings, probably requires more technical approach. Will discuss in meeting |

| **13/02/2017** | |
|---|---|
| **Tasks** | • Changed program to run as standalone script<br>• Added test for running with no image<br>• Created script to test pressure calculation equation from excel data |
| **Issues** | • Equation method requires testing with alternative bike |

| **15/02/2017** | |
|---|---|
| **Tasks** | • Created Python script to produce linear regression equation from csv<br>• Implemented linear regression production into main program in seperate class |

| | |
|---|---|
| | • Tested using hardcoded movement values from both shocks<br>• Tested finding real reference point |
| **Issues** | • Reference point finding needs tuning for real sticker<br>• Smarter algorithm needs testing with measured values |

| **17/02/2017** | |
|---|---|
| **Tasks** | • Added images with flash for testing<br>• Made code to reduce colors for easier ref point finding<br>• Tested with 100 and 150 flash and no flash images |
| **Issues** | • Can't find circle in 150 flash image, colors off<br>• Measurements wrong, possibly due to imperfect circle |

| **19/02/2017** | |
|---|---|
| **Tasks** | • Added color quantification into program<br>• Added colored debug printing |
| **Issues** | • PX to mm conversion is wrong somehow<br>• Still can't find circle in 150 flash image |

| **21/02/2017** | |
|---|---|
| **Tasks** | • Adjusted parameters of hough circles to improve circle finding<br>• Changed circle finding method from hough to find countours and bounding circle<br>• Adjusted upper and lower bounds for finding reds |
| **Issues** | Unit test fails |

| **23/02/2017** | |
|---|---|
| **Tasks** | • Adjustments to improve measuring process<br>• Fixed tests failing from new ref point process |
| **Issues** | N/A |

| **28/02/2017** | |
|---|---|

| | |
|---|---|
| **Tasks** | • Added images of fox shock |
| **Issues** | • Does not work on fox shock, need to find black O-ring |

| **01/03/2017** | |
|---|---|
| **Tasks** | • Experimented with and implemented ability to find black O-ring limit using thresholding |
| **Issues** | • New code not covered by tests |

| **03/03/2017** | |
|---|---|
| **Tasks** | • Added unit tests for new functionality<br>• Improved how the application fails with catches and better messages |
| **Issues** | N/A |

# G Git Commit Log

```
* 2017-04-05 - Wrote lots of conclusion
* 2017-04-04 - More writing, more proofing
* 2017-04-03 - Completed final eval section
* 2017-04-03 - Started last eval section
* 2017-04-03 - Added source code
* 2017-04-02 - Added reign image
* 2017-04-02 - Ticked off a load of todos
* 2017-04-02 - Added all citations from todo
* 2017-04-02 - Wrote most eval sections
* 2017-04-01 - Final proofing
* 2017-03-31 - Added bit for eval intro
* 2017-03-31 - Finished proofing with current material
* 2017-03-31 - Proofed to moscow
* 2017-03-31 - Proofed lit review
* 2017-03-31 - Proofed ia in sport
* 2017-03-31 - Proofed analysis techniques
* 2017-03-31 - Proofed analysis fundamentals and uses section
* 2017-03-30 - Added images of 3 bikes
* 2017-03-30 - Added testing figures to document
* 2017-03-28 - Removed poster subrepo
* 2017-03-28 - Finished proofing and adding content
* 2017-03-28 - Making changes from proofing, committing before large
 ↪  change
* 2017-03-27 - Cont. writing
* 2017-03-26 - Some fixes
* 2017-03-26 - Cleaned todos
* 2017-03-20 - Added some figures
* 2017-03-20 - Work on results
* 2017-03-17 - Work on results section including figures, Move
 ↪  appendicies into section
* 2017-03-16 - Added images into results section for oring finding and
 ↪  ref point finding
* 2017-03-16 - Created final sections, added uncertainty table
* 2017-03-14 - Finished first draft of evaluation
* 2017-03-13 - Work on eval section
* 2017-03-11 - Updated Plan
* 2017-03-07 - Started organic development method and eval method
* 2017-03-06 - Updated plan following meeting / Wrote about threshold
* 2017-03-06 - Wrote variable naming and OO method
* 2017-03-06 - Finished lit review methodology / Added thresholds to pm
 ↪  method
* 2017-03-05 - Plan, writing, work
* 2017-03-03 - Cleaned up notes / Added notes
* 2017-03-01 - Added pedals feedback
* 2017-02-27 - Started eval plan
* 2017-02-15 - Continued
```

```
* 2017-02-12 - Wrote pythonic bit
* 2017-02-12 - Small addition
* 2017-02-12 - Finished first draft of testing methodology
* 2017-02-12 - Small changes
* 2017-02-11 - Wrote about testing
* 2017-02-10 - ?
* 2017-02-09 - Cont.
* 2017-01-29 - Added methodology parts
* 2017-01-21 - Cont.
* 2017-01-15 - Small change
* 2017-01-15 - Added todos
* 2017-01-14 - Added todos
* 2017-01-14 - Requirements methodology section
* 2017-01-13 - Changes
* 2017-01-13 - Cont.
* 2017-01-11 - Cont.
* 2017-01-11 - Synced pdf
* 2017-01-10 - Changes from meeting
* 2017-01-10 - Added notes for second wave
* 2017-01-10 - Small changes
* 2017-01-09 - Cont.
* 2017-01-09 - Some work
* 2017-01-07 - Cont.
* 2017-01-06 - Cont.
* 2017-01-05 - Cont.
* 2017-01-04 - Cont.
* 2017-01-03 - Cont.
* 2017-01-02 - Cont.
* 2017-01-02 - Started agile bit
* 2017-01-01 - First draft of vcs section
* 2017-01-01 - Started vcs section
* 2016-12-24 - Started methodology
* 2016-12-24 - Updated notes
* 2016-12-13 - Method notes
* 2016-11-30 - Finished first exp log
* 2016-11-30 - removed test file
* 2016-11-30 - testing script
* 2016-11-30 - Added to exp log
* 2016-11-30 - More md messing
* 2016-11-30 - Working on md
* 2016-11-30 - Started experiment log
* 2016-11-15 - Finished referencing figures
* 2016-11-14 - Added some figure references
* 2016-11-14 - Wrote about horses and that
* 2016-11-08 - Added sport science refs
* 2016-10-31 - Added paragraphs to lit review, added declarations
* 2016-10-29 - Cont.
*   2016-10-28 - Fixed merge
```

```
|\
| * 2016-10-27 - Cont.
* | 2016-10-26 - First proof
|/
* 2016-10-24 - Finished? section on cameras
* 2016-10-18 - Continued camera operation
* 2016-10-18 - Added second camera operation image
* 2016-10-18 - Added image of camera operation
* 2016-10-18 - Added combination section, started on image capturing
* 2016-10-16 - Finished analysis techniques section
* 2016-10-13 - cont
* 2016-10-09 - Added to measuring
* 2016-10-08 - Started obj recognition
* 2016-10-08 - Finished edge detection
* 2016-10-07 - I wrote a sentence, woop
* 2016-10-07 - Changed analysis struct, moved figure, added ignore
* 2016-10-06 - Added techniques section file
* 2016-10-05 - Added images and schematics
* 2016-10-05 - Added images for figures
* 2016-10-04 - Added to image analysis bit
* 2016-10-04 - Saving before worktree
* 2016-10-03 - Continued analysis bit
* 2016-10-03 - Added sections dir
* 2016-10-03 - Started imagery bit
* 2016-10-02 - Added aims and structure
* 2016-10-02 - Split document into files, added to intro
* 2016-09-29 - Added to physics
* 2016-09-27 - Added glossary entries where required
* 2016-09-27 - Finished physics
* 2016-09-27 - Added image
* 2016-09-27 - Continued physics
* 2016-09-27 - Continued work on physics
* 2016-09-25 - Started physics section
* 2016-09-25 - Finished(?) context
* 2016-09-25 - Continued context
* 2016-09-24 - Continued context
* 2016-09-24 - Continued work on context
* 2016-09-24 - Started context, added bib
* 2016-09-24 - Added major headings
* 2016-09-21 - Added missing from prev push
* 2016-09-21 - Created title page
* 2016-09-21 - Added template
* 2016-09-21 - Created tex documents for full diss, title, and declaration
| * 2017-04-03 - Added images with kit
| * 2017-03-30 - Re-implemented coloured debug
| * 2017-03-28 - Created unittest results
| * 2017-03-27 - Produced some new images
| * 2017-03-17 - Created plots for dissitation
```

```
| * 2017-03-16 - Created images for dissertation
| * 2017-03-14 - Added check for missing color argument / Calculated
↪  uncertainty for 25% sag
| * 2017-03-09 - Added scatter for demonstration
| * 2017-03-09 - Created images for poster and dissertation
| * 2017-03-06 - Calculated uncertainty for fox and rs
| * 2017-03-06 - Protected class variables
| * 2017-03-03 - Added unit test for black oring / Fixed failing unit
↪  tests due to new structure
| * 2017-03-03 - Reinforced a bit / Added more descriptive messages
| * 2017-03-01 - Added black o-ring functionality / Prepped images for
↪  eval
| * 2017-03-01 - Made script to find black oring
| * 2017-02-28 - Measure tests
| * 2017-02-28 - Added fox images
| * 2017-02-23 - Small adjustments for improved measuring
| * 2017-02-23 - Fixed test fails / Added result checking tests
| * 2017-02-22 - No change
| * 2017-02-21 - Added more images for testing / tweaked red bounds
| * 2017-02-21 - Created method to find ref and oring from contours /
↪  Adapted to measure to limit of oring
| * 2017-02-21 - Added params to houghcircles \ created hough methods for
↪  each image as test
| * 2017-02-20 - Experimentation
| * 2017-02-19 - Added colored debug printing / Added color
↪  quantification into program
| * 2017-02-17 - dded images with flash for testing / Made code to reduce
↪  colors for easier ref point finding / Tested with 100 and 150 flash
↪  and no flash images
| * 2017-02-16 - Moved dev log to code branch
| * 2017-02-16 - Created pressure calculator unit tests
| * 2017-02-15 - Tuned circle detection
| * 2017-02-15 - Cont.
| * 2017-02-15 - Added images
| * 2017-02-15 - Adapted to work with linear regression
| * 2017-02-15 - Added physics folder
| | * 2017-03-20 - Ready for printing
| | * 2017-03-17 - Changed text in method
| | * 2017-03-16 - Unknown change
| | * 2017-03-16 - Added high res background and exported
| | * 2017-03-15 - Added mtb centre qr code
| | * 2017-03-15 - Added mtb centre logo
| | * 2017-03-14 - Spellchecked
| | * 2017-03-13 - Finished first draft
| | * 2017-03-13 - Lots of work on poster
| | * 2017-03-12 - Created hex version of poster
| | * 2017-03-11 - Created background
| | * 2017-03-11 - Continued
```

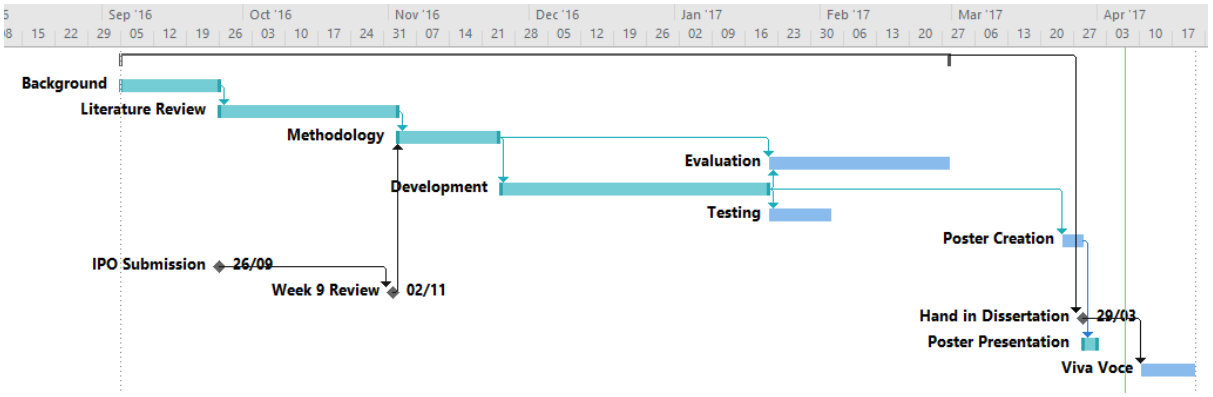
```
| | * 2017-03-11 - Updated plan
| | * 2017-03-09 - Updated plan
| | *   2017-03-08 - Merge branch 'poster' of
↪  github.com:Reignable/Honours-Project into poster
| | |\
| | | * 2017-03-07 - Played with layout
| | * | 2017-03-08 - Added to plan
| | |/
| | * 2017-03-07 - Added to plan, added to poster
| | * 2017-03-05 - work on poster
| | * 2017-03-03 - Started poster
| | | * 2017-02-16 - Removed dev log
| | | * 2017-02-16 - Added dev log
| | | * 2017-02-14 - Added to figures
| | | * 2017-02-13 - Created equation workings spreadsheet
| | | * 2017-02-02 - started collecting requirements
| | | * 2017-02-02 - updated plan
| | | * 2017-01-29 - Updated Plan
| | | * 2017-01-08 - Updated plan
| | | * 2016-10-31 - Added pdf plan
| | | * 2016-10-31 - Adjusted plan, made image
| | | * 2016-10-30 - Updated Plan
| | | * 2016-10-24 - Added project plan
| | | * 2016-09-15 - Added dates
| | | * 2016-09-15 - Created Timeline
| | |_|/
|/| |
| | | * 2017-02-15 - Checking measurement process
| | | * 2017-02-15 - Work on calculating equation
| | |/
| |/|
| * | 2017-02-15 - Unknown change
| * | 2017-02-13 - Added presssure calculation script
| * | 2017-02-13 - Removed redundant script
| * | 2017-02-13 - Changes
| * | 2017-02-12 - Cleaned file, created api function, added test
| * | 2017-02-12 - Adapted all tests to use unittest2
| * | 2017-02-12 - Changed to using unittest2
| * | 2017-02-10 - Finished tests
| * | 2017-02-09 - Fixed math
| * | 2017-02-09 - More unit tests
| * | 2017-02-08 - Made test
| * | 2017-02-07 - Created v4
| * | 2017-02-06 - Added debug printing, fixed maths
| * | 2017-02-03 - Found and measured ref point, added calculations
| * | 2017-02-01 - Started ref point finding
| * | 2017-02-01 - Added stuff
| * | 2017-02-01 - Added arguments
```

```
| * | 2017-01-31 - Start of official code branch
| * | 2017-01-31 - Progress
| * | 2017-01-31 - Added paramemters
| * | 2017-01-29 - Cont.
| * | 2017-01-29 - Added own images
| * | 2017-01-29 - Progress
| * | 2017-01-15 - Started on shock
| * | 2017-01-12 - Found the wheels
| * | 2017-01-12 - Progress
| * | 2017-01-12 - ?
| * | 2017-01-10 - Measure obj
| * | 2016-11-29 - Added distance to camera images
| * | 2016-11-21 - New script
| * | 2016-11-18 - Playing with python opencv
| * | 2016-11-12 - Started user stories
| |/
| | *   2016-11-30 - Fix merge
| | |\
| | | * 2016-11-08 - Finished 15tile game
| | | * 2016-11-07 - Started copying 15tile example
| | * |   2016-11-30 - Fix merge
| | |\ \
| | | * | 2016-11-12 - Copied sample
| | | |/
| | * |   2016-11-30 - Fix conflict
| | |\ \
| | | * | 2016-11-08 - Wrote facial recognition app, doesn't work
| | | |/
| | * | 2016-11-05 - Started hellocv app
| | |/
| | * 2016-11-05 - Created app with opencv installed
| |/
| * 2016-09-28 - Changed title in readme
| * 2016-09-28 - Added lit review document
| * 2016-09-22 - Added book
| * 2016-09-21 - Updated readme
| * 2016-09-21 - Notes post 21/09/2016 lecture
|/
| * 2016-09-21 - Removed hyperlinks
| * 2016-09-21 - Finished IPO after supervisor review
| * 2016-09-20 - Finished first ipo draft
| * 2016-09-20 - Continued work on ipo
| * 2016-09-18 - Continued work on ipo
| * 2016-09-17 - merge stash
| * 2016-09-17 - merging stash
| *   2016-09-17 - Merge branch 'ipo' of
↪  github.com:Reignable/Honours-Project into ipo
| |\
```

```
| | * 2016-09-15 - Small Change
| * | 2016-09-17 - Added content to overview, deliverable, and audience
| |/
| * 2016-09-14 - Completed headings - Added headers and footers
| * 2016-09-14 - Began IPO document
|/
* 2016-09-14 - Initial commit
```

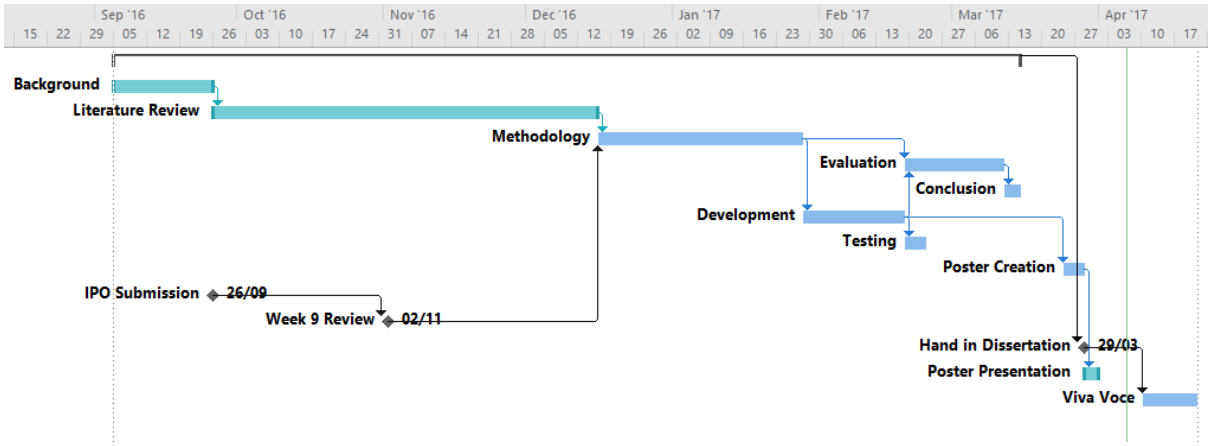
# H Gantt Charts

## H.1 Original



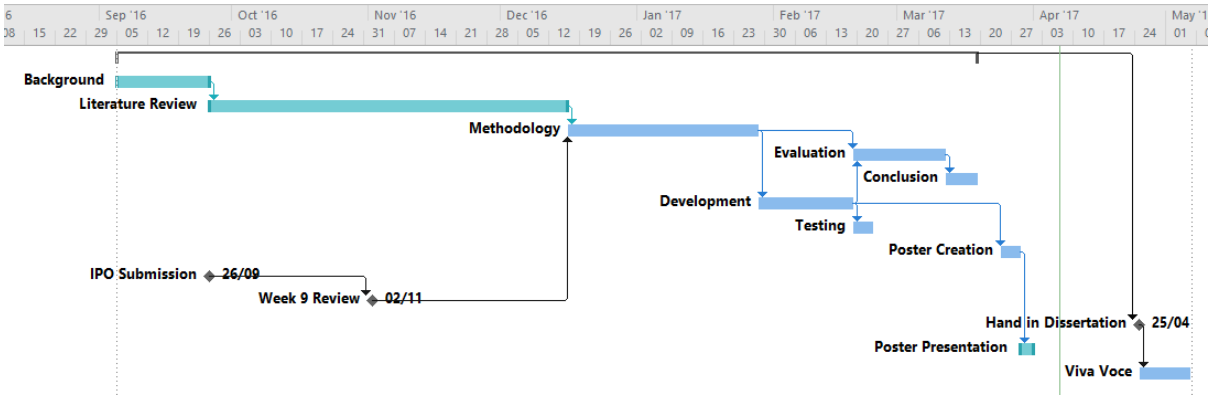

| | Duration | Task Name | Start | Finish | Predecessors |
|---|---|---|---|---|---|
| 1 | **225.5 days** | **Dissertation** | **Mon 05/09/16** | **Tue 28/02/17** | |
| 2 | 28.38 days | Background | Mon 05/09/16 | Sun 25/09/16 | |
| 3 | 52.13 days | Literature Review | Mon 26/09/16 | Wed 02/11/16 | 2 |
| 4 | 30 days | Methodology | Thu 03/11/16 | Thu 24/11/16 | 3,10 |
| 5 | 50 days | Evaluation | Sat 21/01/17 | Tue 28/02/17 | 6,4 |
| 6 | 63.75 days | Development | Fri 25/11/16 | Fri 20/01/17 | 4 |
| 7 | 14 days | Testing | Sat 21/01/17 | Fri 03/02/17 | 6 |
| 8 | 7 days | Poster Creation | Fri 24/03/17 | Tue 28/03/17 | 6 |
| 9 | 0 days | IPO Submission | Mon 26/09/16 | Mon 26/09/16 | |
| 10 | 0 days | Week 9 Review | Wed 02/11/16 | Wed 02/11/16 | 9 |
| 11 | 0 days | Hand in Dissertation | Wed 29/03/17 | Wed 29/03/17 | 1 |
| 12 | 3.38 days | Poster Presentation | Wed 29/03/17 | Fri 31/03/17 | 8 |
| 13 | 15 days | Viva Voce | Mon 10/04/17 | Fri 21/04/17 | 11 |

## H.2 First Revision



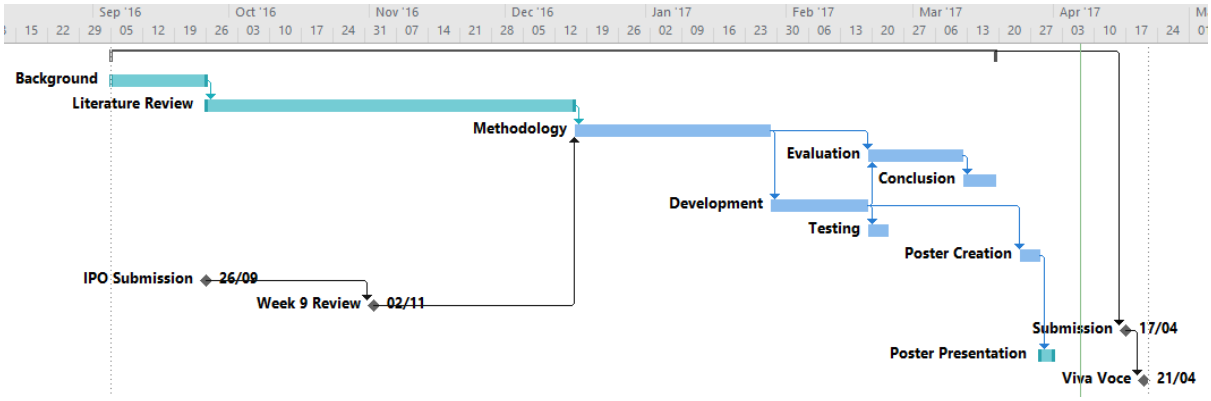

| | Task Name | Duration | Start | Finish | Predecessors |
|---|---|---|---|---|---|
| 1 | **Dissertation** | **244 days** | **Mon 05/09/16** | **Wed 15/03/17** | |
| 2 | Background | 28.38 days | Mon 05/09/16 | Sun 25/09/16 | |
| 3 | Literature Review | 110.63 days | Sun 25/09/16 | Fri 16/12/16 | 2 |
| 4 | Methodology | 40 days | Fri 16/12/16 | Sat 28/01/17 | 3,11 |
| 5 | Evaluation | 30 days | Sun 19/02/17 | Sun 12/03/17 | 7,4 |
| 6 | Conclusion | 5 days | Sun 12/03/17 | Wed 15/03/17 | 5 |
| 7 | Development | 30 days | Sat 28/01/17 | Sun 19/02/17 | 4 |
| 8 | Testing | 7 days | Sun 19/02/17 | Thu 23/02/17 | 7 |
| 9 | Poster Creation | 7 days | Fri 24/03/17 | Tue 28/03/17 | 7 |
| 10 | IPO Submission | 0 days | Mon 26/09/16 | Mon 26/09/16 | |
| 11 | Week 9 Review | 0 days | Wed 02/11/16 | Wed 02/11/16 | 10 |
| 12 | Hand in Dissertation | 0 days | Wed 29/03/17 | Wed 29/03/17 | 1 |
| 13 | Poster Presentation | 3.38 days | Wed 29/03/17 | Fri 31/03/17 | 9 |
| 14 | Viva Voce | 15 days | Mon 10/04/17 | Fri 21/04/17 | 12 |

## H.3 Second Revision




| | Task Name | Duration | Start | Finish | Predecessors |
|---|---|---|---|---|---|
| 1 | **Dissertation** | **249 days** | **Mon 05/09/16** | **Sun 19/03/17** | |
| 2 | Background | 28.38 days | Mon 05/09/16 | Sun 25/09/16 | |
| 3 | Literature Review | 110.63 days | Sun 25/09/16 | Fri 16/12/16 | 2 |
| 4 | Methodology | 40 days | Fri 16/12/16 | Sat 28/01/17 | 3,11 |
| 5 | Evaluation | 30 days | Sun 19/02/17 | Sun 12/03/17 | 7,4 |
| 6 | Conclusion | 10 days | Sun 12/03/17 | Sun 19/03/17 | 5 |
| 7 | Development | 30 days | Sat 28/01/17 | Sun 19/02/17 | 4 |
| 8 | Testing | 7 days | Sun 19/02/17 | Thu 23/02/17 | 7 |
| 9 | Poster Creation | 7 days | Fri 24/03/17 | Tue 28/03/17 | 7 |
| 10 | IPO Submission | 0 days | Mon 26/09/16 | Mon 26/09/16 | |
| 11 | Week 9 Review | 0 days | Wed 02/11/16 | Wed 02/11/16 | 10 |
| 12 | Hand in Dissertation | 0 days | Tue 25/04/17 | Tue 25/04/17 | 1 |
| 13 | Poster Presentation | 3.38 days | Wed 29/03/17 | Fri 31/03/17 | 9 |
| 14 | Viva Voce | 15 days | Tue 25/04/17 | Sat 06/05/17 | 12 |

## H.4 Third Revision




| | Task Name | Duration | Start | Finish | Predecessors |
|---|---|---|---|---|---|
| 1 | **Dissertation** | **249 days** | **Mon 05/09/16** | **Sun 19/03/17** | |
| 2 | Background | 28.38 days | Mon 05/09/16 | Sun 25/09/16 | |
| 3 | Literature Review | 110.63 days | Sun 25/09/16 | Fri 16/12/16 | 2 |
| 4 | Methodology | 40 days | Fri 16/12/16 | Sat 28/01/17 | 3,11 |
| 5 | Evaluation | 30 days | Sun 19/02/17 | Sun 12/03/17 | 7,4 |
| 6 | Conclusion | 10 days | Sun 12/03/17 | Sun 19/03/17 | 5 |
| 7 | Development | 30 days | Sat 28/01/17 | Sun 19/02/17 | 4 |
| 8 | Testing | 7 days | Sun 19/02/17 | Thu 23/02/17 | 7 |
| 9 | Poster Creation | 7 days | Fri 24/03/17 | Tue 28/03/17 | 7 |
| 10 | IPO Submission | 0 days | Mon 26/09/16 | Mon 26/09/16 | |
| 11 | Week 9 Review | 0 days | Wed 02/11/16 | Wed 02/11/16 | 10 |
| 12 | Submission | 0 days | Mon 17/04/17 | Mon 17/04/17 | 1 |
| 13 | Poster Presentation | 3.38 days | Wed 29/03/17 | Fri 31/03/17 | 9 |
| 14 | Viva Voce | 0 days | Fri 21/04/17 | Fri 21/04/17 | 12 |

# I Statements From Evaluation Meeting

## I.1 Geraint Florida-James

”I have spent some time with Joe looking at his work and there is definitely a useful application in development. This as expected is not a full resource to deal with the vast number of variables that constitute mountain bike suspension set up but this application helps the user on the very first, and most important, step of this process. With the evidence I have seen I would encourage further development of the next steps. ”

# J Source Code

## J.1 main.py

```
#! /usr/bin/env python

import argparse
import sys
import warnings
from image_processor import ImageProcessor
from pressure_calculator import PressureCalculator

warnings.filterwarnings('ignore')


def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('-i',
                        '--image',
                        type=str,
                        nargs=2,
                        action='store',
                        metavar='',
                        help='The path to the image you wish to use')
    parser.add_argument('-c',
                        '--colour',
                        type=str,
                        action='store',
                        metavar='',
                        help='The colour of the o-ring on the shock')
    parser.add_argument('-s',
                        '--sag',
                        type=int,
                        action='store',
                        metavar='',
                        help='The desired sag percentage')
    parser.add_argument('-p',
                        '--pressure',
                        type=int,
                        action='store',
                        metavar='',
                        help='The pressure you have already put into your bike')
    parser.add_argument('-t',
                        '--stroke',
                        type=float,
                        action='store',
                        metavar='',
                        help='')
    parser.add_argument('-d',
                        '--debug',
                        action='store_true')

    args = parser.parse_args()
    if args.image is None:
        print 'This program requires an image'
        parser.print_help()
        sys.exit(1)
```

```
54      if args.colour is None:
55          print 'Please specify a colour for the o-ring'
56          parser.print_help()
57          sys.exit(1)
58
59      image_processor = ImageProcessor(args.colour, args.debug)
60      pressure_calculator = PressureCalculator(args.sag, args.stroke, args.debug)
61      pressure_calculator.measurement_100 =
        ↪   image_processor.get_measurement(args.image[0])
62      image_processor = ImageProcessor(args.colour, args.debug)
63      pressure_calculator.measurement_150 =
        ↪   image_processor.get_measurement(args.image[1])
64      #pressure_calculator.measurement_100 = 30.0
65      #pressure_calculator.measurement_150 = 20.0
66      setting = pressure_calculator.calculate()
67      print '\nYour recommended pressure setting is'
68      print str(round(setting, 0)) + ' psi'
69
70  if __name__ == '__main__':
71      main()
```

## J.2 image_processor.py

```
1   import cv2
2   import numpy
3   import sys
4   import utils
5   from sklearn.cluster import MiniBatchKMeans
6
7
8   def show_image(image, wait_time):
9       try:
10          cv2.imshow(str(image), image)
11          cv2.waitKey(wait_time)
12          return 0
13      except cv2.error as e:
14          print e.message
15
16
17  class ImageProcessor:
18      REF_POINT_KNOWN_WIDTH = 9.5
19
20      _image_path = None
21      _edged_image = None
22      _debug = False
23      _ref_point = None
24      _oring = None
25      _colour = None
26
27      def __init__(self, colour='red', debug=False):
28          self._colour = colour
29          self._debug = debug
30
31      def _edge_detect(self):
32          image = cv2.imread(self._image_path)
33          gray_scaled = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
34          blurred = cv2.GaussianBlur(gray_scaled, (5, 5), 0)
35          edged = cv2.Canny(blurred, 10, 90, apertureSize=3)
36          return edged
37
38      def _quantify_colors(self):
39          image = cv2.imread(self._image_path)
40          (h, w) = image.shape[:2]
41          image = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
42          image = image.reshape((image.shape[0] * image.shape[1], 3))
43          clt = MiniBatchKMeans(n_clusters=8)
44          labels = clt.fit_predict(image)
45          quantified = clt.cluster_centers_.astype('uint8')[labels]
46          quantified = quantified.reshape((h, w, 3))
47          return cv2.cvtColor(quantified, cv2.COLOR_LAB2BGR)
48
49      def _find_red(self):
50          image = self._quantify_colors()
51          upper_bound = numpy.array([100, 100, 255])
52          lower_bound = numpy.array([0, 0, 130])
53          mask = cv2.inRange(image, lower_bound, upper_bound)
54          marker = cv2.bitwise_and(image, image, mask=mask)
55          marker = cv2.cvtColor(marker, cv2.COLOR_BGR2GRAY)
56          contours, _ = cv2.findContours(marker.copy(), cv2.RETR_EXTERNAL,
            ↪  cv2.CHAIN_APPROX_SIMPLE)
57          return sorted(contours, key=cv2.contourArea, reverse=True)[:2]
58
59      def _find_ref(self):
60          try:
61              self._ref_point = self._find_red()[0]
62          except (ValueError, IndexError):
63              print 'Could not find reference point\n' \
64                    'Is the the correct colour given and is one present in the image?\nIf
                      ↪  ' \
65                    'yes, please try again.'
66              sys.exit(1)
67
68      def _find_oring(self):
69          if self._colour == 'red':
70              try:
71                  self._oring = self._find_red()[1]
72              except IndexError:
73                  print 'Could not find o-ring\n'\
74                        'Is the the correct colour given and is one present in the
                          ↪  image?\nIf ' \
75                        'yes, please try again.'
76          elif self._colour == 'black':
77              image = cv2.imread(self._image_path)
78              (height, width) = image.shape[:2]
79              gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
80              (T, thresh) = cv2.threshold(gray, 30, 150, cv2.THRESH_BINARY)
81              contours, _ = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
                ↪  cv2.CHAIN_APPROX_SIMPLE)
82              contours = sorted(contours, key=cv2.contourArea, reverse=True)[:5]
83              for contour in contours:
84                  _, y, _, _ = cv2.boundingRect(contour)
85                  if y > (height / 3):
86                      self._oring = contour
87
```

```
88      def _get_ref_point_width(self):
89          (_, _), radius = cv2.minEnclosingCircle(self._ref_point)
90          if self._debug:
91              utils.debug_print(self.__class__.__name__, '_get_ref_point_width', radius *
                ↪  2)
92          return radius * 2
93
94      def _get_oring_height(self):
95          _, y, _, h = cv2.boundingRect(self._oring)
96          if self._debug:
97              utils.debug_print(self.__class__.__name__, '_get_oring_height', y+h)
98          return y + h
99
100     def _get_measurement_px(self):
101         min_line_length = 1000
102         max_line_gap = 10
103         image_height, image_width = self._edged_image.shape[:2]
104         y_limit = self._get_oring_height()
105         y_min = image_height
106         y_max = 0
107         lines = cv2.HoughLinesP(self._edged_image,
108                                 1,
109                                 numpy.pi / 180,
110                                 50,
111                                 min_line_length,
112                                 max_line_gap)
113         for x1, y1, x2, y2 in lines[0]:
114             if x1 >= image_width / 2 and \
115                             y1 >= image_height / 3 and \
116                             y2 <= y_limit and \
117                             abs(x1 - x2) <= 1:
118                 y_min = min(y_min, y1)
119                 y_max = max(y_max, y2)
120
121         if self._debug:
122             utils.debug_print(self.__class__.__name__, '_get_measurement_px', y_max -
                ↪  y_min)
123         return y_max - y_min
124
125     def _convert_px_mm(self, measurement_px):
126         pixels_per_mm = self._get_ref_point_width() / self.REF_POINT_KNOWN_WIDTH
127         if self._debug:
128             utils.debug_print(self.__class__.__name__, '_convert_px_mm', pixels_per_mm)
129         return measurement_px / pixels_per_mm
130
131     def get_measurement(self, image_path):
132         self._image_path = image_path
133         if self._debug:
134             utils.debug_print(self.__class__.__name__, 'filepath', self._image_path)
135         self._edged_image = self._edge_detect()
136         self._find_ref()
137         self._find_oring()
138         measurement_px = self._get_measurement_px()
139         measurement_mm = self._convert_px_mm(measurement_px)
140
141         if self._debug:
142             utils.debug_print(self.__class__.__name__, 'get_measurement',
                ↪  measurement_mm)
```

```
143         if measurement_mm <= 0:
144             print 'Incorrect measurement produced from image {i}\n'\
145                   'Please check settings and try again.'.format(i=self._image_path)
146             sys.exit(1)
147         return measurement_mm
```

## J.3 pressure_calculator.py

```
1  import numpy as np
2  import scipy.stats as stats
3  import utils
4
5
6  class PressureCalculator:
7
8      measurement_100 = None
9      measurement_150 = None
10     _debug = None
11     _sag = None
12     _stroke = None
13
14     def __init__(self, sag, stroke, debug=False):
15         self._sag = sag
16         self._stroke = stroke
17         self._debug = debug
18
19     def _get_ideal_sag_mm(self):
20         ideal = float(self._stroke) * (float(self._sag) / 100.0)
21         if self._debug:
22             utils.debug_print(self.__class__.__name__, '_get_ideal_sag_mm', ideal)
23         return ideal
24
25     def _get_ideal_pressure(self):
26         slope, intercept = self._calculate_linear_equation()
27         ideal = (slope * self._get_ideal_sag_mm()) + intercept
28         if self._debug:
29             utils.debug_print(self.__class__.__name__, '_get_ideal_pressure', ideal)
30         return ideal
31
32     def _calculate_linear_equation(self):
33         x = np.array([self.measurement_100, self.measurement_150])
34         y = np.array([100, 150])
35         slope, intercept, __, __, __ = stats.linregress(x, y)
36         if self._debug:
37             utils.debug_print(self.__class__.__name__, '_calculate_equation', '{s}x +
               ↪  {i}'.format(s=slope, i=intercept))
38         return slope, intercept
39
40     def calculate(self):
41         return self._get_ideal_pressure()
```

## J.4 test_image_processor.py

```
import warnings

from unittest2 import TestCase


class TestImageProcessor(TestCase):

    image_processor = None
    processed_test_image = None
    RS_IMAGE_PATH = '100_rs.jpg'
    FOX_IMAGE_PATH = '100_fox.jpg'

    def setUp(self):
        from image_processor import ImageProcessor
        self.image_processor = ImageProcessor('red', True)
        self.image_processor._image_path = self.RS_IMAGE_PATH
        warnings.filterwarnings('ignore')

    def tearDown(self):
        pass

    def test_show_image_normal(self):
        import cv2
        from image_processor import show_image
        image = cv2.imread(self.RS_IMAGE_PATH)
        result = show_image(image, 1)
        self.assertEqual(result, 0)

    def test_show_image_incorrect_file_path(self):
        import cv2
        from image_processor import show_image
        image = cv2.imread('')
        self.assertRaises(cv2.error, show_image(image, 1))

    def test_process_image_normal(self):
        import cv2
        image = cv2.imread(self.RS_IMAGE_PATH)
        gray_scaled = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        blurred = cv2.GaussianBlur(gray_scaled, (5, 5), 0)
        self.processed_test_image = cv2.Canny(blurred, 0, 100, apertureSize=3)
        self.image_processor._image_path = self.RS_IMAGE_PATH
        processed_image = self.image_processor._edge_detect()
        self.assertEqual(processed_image.all(), self.processed_test_image.all())

    def test_get_ref_point_width_normal(self):
        self.image_processor._find_ref()
        self.assertIsInstance(self.image_processor._get_ref_point_width(), float)

    def test_get_ref_point_width_in_range(self):
        self.image_processor._find_ref()
        width = self.image_processor._get_ref_point_width()
        expected = 565.0
        self.assertTrue((expected*0.9) <= width <= (expected*1.1))

    def test_get_ref_point_width_not_none(self):
```

```
56          self.image_processor._find_ref()
57          self.assertIsNotNone(self.image_processor._get_ref_point_width(), float)
58
59      def test_find_oring_black(self):
60          self.image_processor._colour = 'black'
61          self.image_processor._image_path = self.FOX_IMAGE_PATH
62          self.image_processor._find_oring()
63          self.assertIsNotNone(self.image_processor._oring)
64
65      def test_get_measurement_px_normal(self):
66          import numpy
67          self.image_processor._edged_image = self.image_processor._edge_detect()
68          self.image_processor._find_ref()
69          self.image_processor._find_oring()
70          self.assertIsInstance(self.image_processor._get_measurement_px(), numpy.int32)
71
72      def test_get_measurement_px_in_range(self):
73          self.image_processor._edged_image = self.image_processor._edge_detect()
74          self.image_processor._find_ref()
75          self.image_processor._find_oring()
76          measurement = self.image_processor._get_measurement_px()
77          expected = 1749
78          self.assertTrue((expected * 0.95) <= measurement <= (expected * 1.05))
79
80      def test_get_measurement_mm_normal(self):
81          self.assertIsInstance(self.image_processor.get_measurement(self.RS_IMAGE_PATH),
              ↪  float)
82
83      def test_get_measurement_mm_in_range(self):
84          expected = 29.0
85          measurement = self.image_processor.get_measurement(self.RS_IMAGE_PATH)
86          self.assertTrue((expected * 0.9) <= measurement <= (expected * 1.1))
```

## J.5 test_pressure_calculator.py

```
1   from unittest2 import TestCase
2
3
4   class TestPressureCalculator(TestCase):
5
6       pressure_calculator = None
7
8       def setUp(self):
9           from pressure_calculator import PressureCalculator
10          self.pressure_calculator = PressureCalculator(30, 57, True)
11
12      def tearDown(self):
13          pass
14
15      def test_get_ideal_sag_mm_normal(self):
16          self.assertIsInstance(self.pressure_calculator._get_ideal_sag_mm(), float)
17
18      def test_get_ideal_pressure(self):
19          self.pressure_calculator.measurement_100 = 29
20          self.pressure_calculator.measurement_150 = 21
21          self.assertIsInstance(self.pressure_calculator._get_ideal_pressure(), float)
```

```
22
23      def test_calculate_linear_equation(self):
24          self.pressure_calculator.measurement_100 = 29
25          self.pressure_calculator.measurement_150 = 21
26          self.assertIsInstance(self.pressure_calculator.calculate(), (float, float))
```