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

31/03/2017

Authorship Declaration

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

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

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

I have acknowledged all main sources of help.

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

I have read and understand the penalties of academic misconduct.

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

Signed:

Date:

Matriculation Number:

Data Protection Declaration

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

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

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

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

The University may not make this dissertation available to others.

Abstract

To do (1)

Contents

List of Tables

List of Figures

List of Listings

1 Introduction

1.1 Context

The suspension on a mountain bike plays a vital part in the rider's performance, comfort and overall enjoyment of the sport. With some suspension units costing upwards of £1000 it is vital that they are setup to function correctly. The objective of this thesis is to examine how image analysis can be used in the set-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 found that the average price had 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 set and the rider goes off a jump, the excessive speed at which the rear of the bike extends can create forwards rotation, causing the rider to go over the handlebars of the bike. However many enthusiasts lack the specialist knowledge required to optimize the set-up 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 causing stress and potentially dramatic fracture that could result in injury. Conversely, suspension that is set too hard forces energy that would normally be absorbed to 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 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 in the use of smartphone, a number of companies have launched small devices (Aston, 2016; Hwang, 2016) and mobile applications (Benedict, 2012) designed to help riders setup 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 set up 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

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

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

1.4 Restrictions

To do (3)

1.5 Thesis Structure

To do (4)

Introduction	Outlines the context of the project and states the major aims and objectives
Literature Review	
Methodology	
Critical Evaluation	
Conclusion	

2 Literature Review

The aim of this project is to produce an application that makes use of image analysis techniques to provide the user with a suggested set-up 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 having more. See Table 1 for typical specifications.

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.

Table 1: Table of common suspension travels and intended disciplines

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

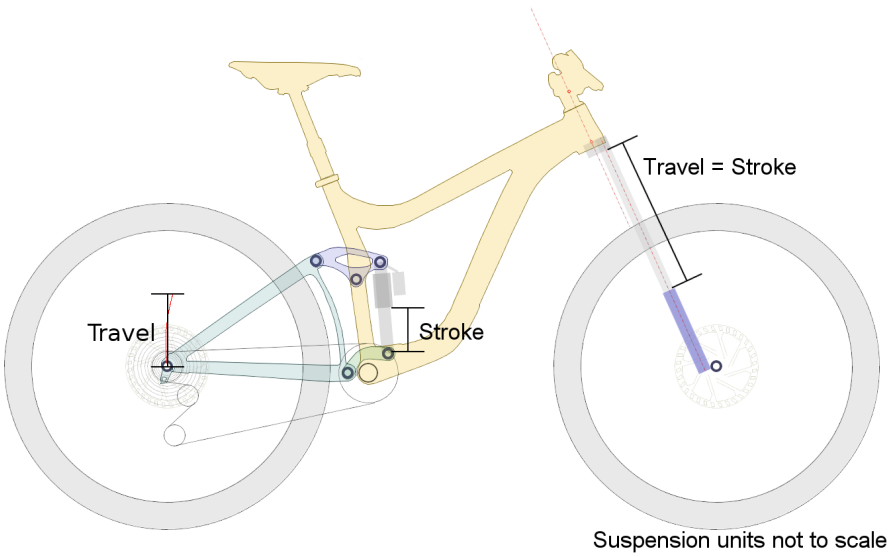


Figure 2: Diagram showing travel and stroke on a full suspension bike[1]

2.1.3 Rear Suspension

Rear suspension uses a shock absorber that is much shorter than a fork so it cannot operate on a 1:1 ratio and still allow the desired travel. 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 leverage ratio changing constantly across the compression cycle.

The difference between front and rear stroke and travel can be seen in Figure 2 noting the separation of rear wheel travel from the stroke of the shock absorber. 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 it 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. **To do** (5)

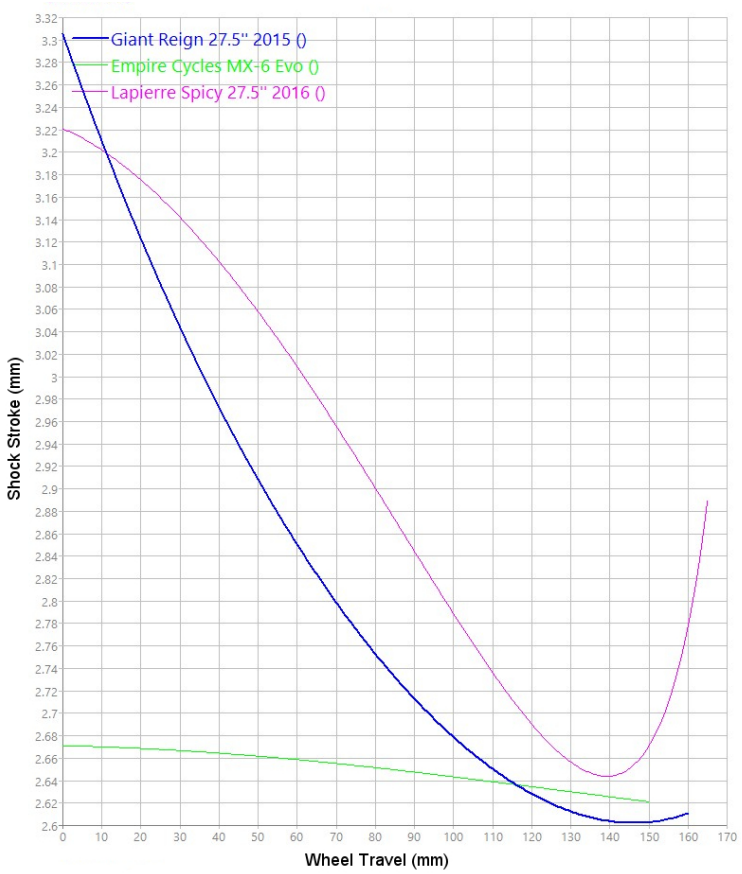
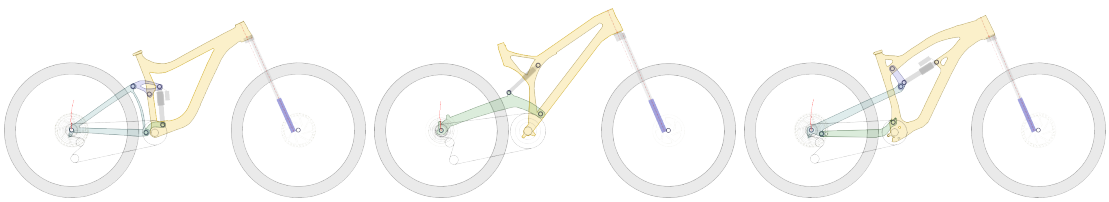


Figure 3: Leverage curves of three modern suspension designs

The Virtual Pivot Point (VPP) design of the Giant Reign (shown blue) has an initial falling rate, meaning the shock can be compressed easily, but slows down and even rises slightly towards the end of its travel as seen in Figure 3. This means the suspension will feel soft most of the time but stiffer 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 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 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.



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

Figure 4: Full suspension frame comparison

For this project two bikes will be used for development and testing of 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 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.

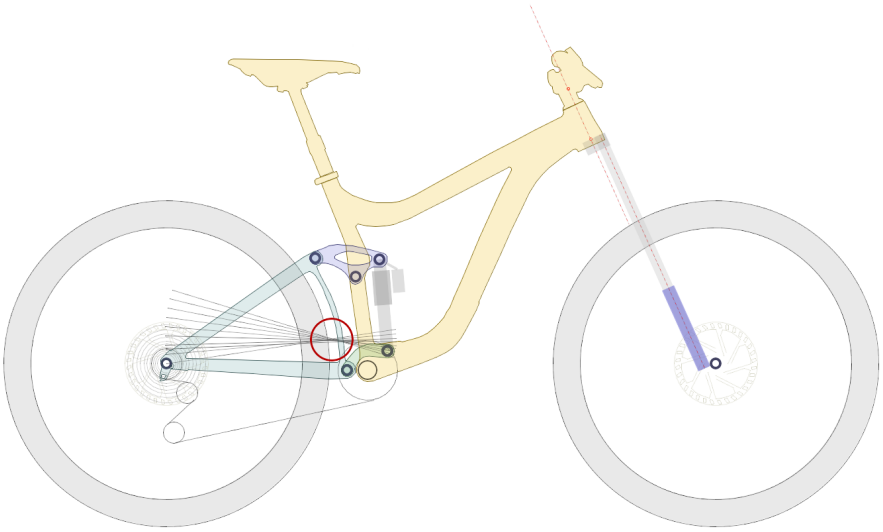


Figure 5: Maestro suspension

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. **To do** (6). Though having an understanding of leverage curves will help the individual rider to fine tune their suspensions settings described in the following sections 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 user 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 **To do** (7). 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 **To do** (8). For an application which is intended to remove the difficulties of suspension setup and allow for quick and easy production of a basic setup, providing a single sag setting would be preferable over a plethora of of technical data.

2.1.4 Sag

To do (9) Sag is the amount that the suspension sits into its travel when the rider is in a neutral position, described in Figure 6. It is required so that the suspension has travel available to extend as well as just compress and is calculated using 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 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.

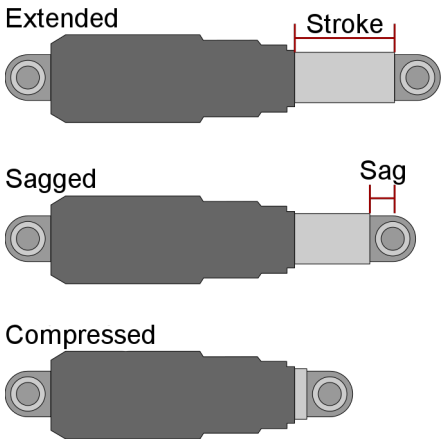


Figure 6: Diagram of shock states indicating sag

Sag is set by first calculating the distance that the shock should sit into its travel by producing the desired percentage of the shocks stroke. For an air shock the manufacturers recommended pressure is then pumped into the shock and the rider weights the bike. On each air shock there is a marker o-ring which is pushed down the shock shaft when the shock 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.

2.1.5 Damping

In a spring and mass system, when the spring is elongated it will exert a returning force on the mass which, according to Hooke's law (Rychlewski, 1984), 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 rebound. Both of these can be controlled which is known as damping.

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 increases resistance, reducing the speed at which the oil flows through the circuit to increase the damping effect by making compression and rebound slower.

2.1.5.1 Compression Damping This is applied while the shock absorber is being compressed to control the effort required to do so. Increasing damping forces the wheel to remain in contact with the ground which makes the suspension feel stiffer. 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 use a lot of travel and rebound damping set faster to deal with multiple features in quick succession.

2.1.6 Optimal Setup

Although setups will vary between rider, suspension system and discipline, there are some key 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 setting available with a slight bias towards the faster option.

Attaining this optimal setup can be difficult for both beginners 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 generating 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.

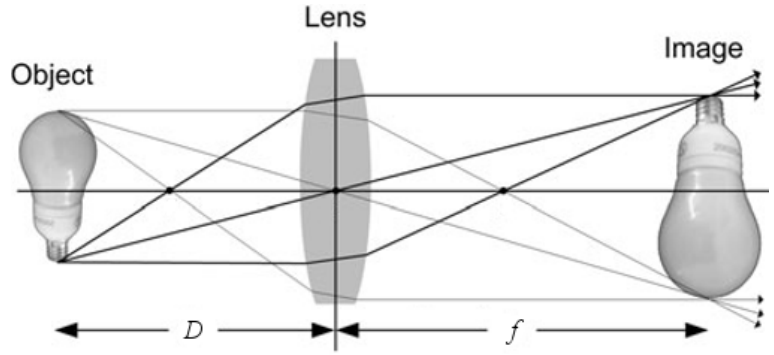


Figure 7: Diagram of camera and lens operation (Moeslund, 2012)

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 7. This point can be adjusted by changing the optical characteristics of the lens so that the refracted light hits the sensor to create a sharply focussed 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 imaged, so 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.

To do (10) These integer values in the image data can be manipulated to change the visual appearance of the image or analysed or compared to neighbouring cells in order to locate objects represented in the image. These processes are known as image processing and image analysis.

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 in camera. Figure 8 shows the effects that different lighting angles have on a subject.

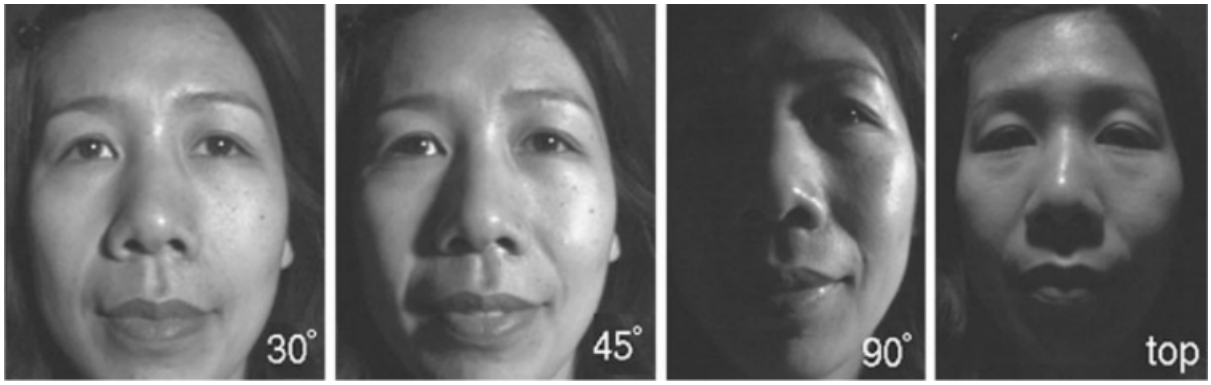


Figure 8: The effects of different lighting conditions on a face (Moeslund, 2012)

It can be seen that the situation which highlights the most detail is when the light source is directly 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 details 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 tracking the changing shape of coastlines (Potter, 2013).

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 most symptoms arise. This is shown top left in Figure 9.

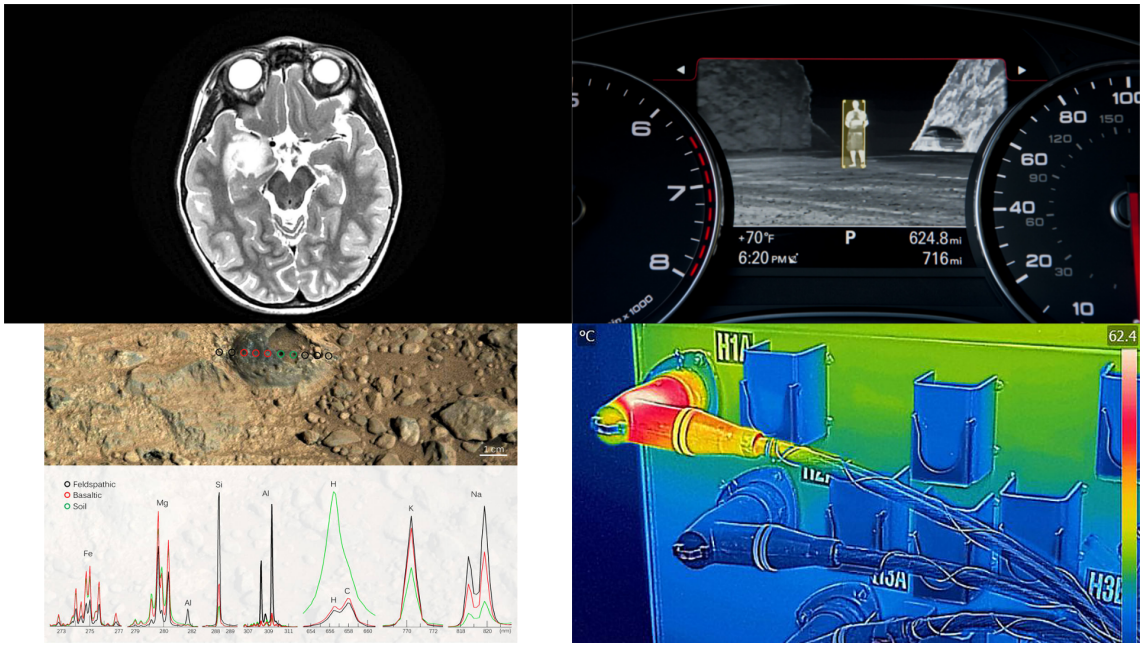


Figure 9: Uses of image analysis, from top left clockwise: An MRI brain scan, automotive night vision with pedestrian recognition, infrared image taken with a smartphone, chemical rock analysis from mars

2.2.3.2 Transport Image analysis has been included in the consumer automotive market on various models since 2004 when Honda introduced 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 9. 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 weak areas (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 9.

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 location and identification of chemicals using the

analysis tools such as ”ChemCam” (Schröder et al., 2015), which can be seen lower left in Figure 9.

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 suggest the edge of and object, structure or land mass. Such functionality is readily available on desktop image processing packages designed for amateur photographers **To do** (11). But scientific and engineering applications of edge detection provide far great levels of sensitivity helping isolate objects that can be subsequently measured with high level of accuracy **To do** (12).

2.2.4.1 Edge Detection This is the application of mathematical algorithms to locate and highlight the edges of features in an image. There are multiple algorithms which can be used for edge detection, including Sobel, Roberts, Canny, and fuzzy logic, although all utilise the same basic concept of comparing the data values of adjacent pixels to find ”steps” from one **To do** (13) brightness to another.

Table 2: Table of pixel data showing an edge

5	7	6	4	152	148	149

Table 2 represents possible pixel values of an edge indicated by the large difference between 4 and 152. The applied algorithm will pick up 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 unwanted background data is automatically excluded using thresholding techniques to highlight the block shapes of the vehicle registration plate. Figure 10 shows an image which has had **To do** (14) thresholding applied with the number plate of the car clearly visible.

Figure 10: Edge detection applied to an image for number plate recognition

2.2.4.2 Hough Transform To do (15) Originally created in 1962 by Paul Hough, the Hough transform methods offer techniques for the discovery of imperfect objects n 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 sharply focussed To do (16) with the edge detector creating gaps in 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.

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

Circular Hough Transform

2.2.4.3 Taking Measurements To measure 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.

However it is not necessary to collect data for H_o - the distance of the object being measured from the ground - as the size of the object is in direct proportion to its size on the camera's sensor so is not required. To test this equation, data was created from known object heights which was then re-used to see if the calculated height was equal to the actual height. For example:To do (17)

To circumvent the need to collect data for H_c - the height of the camera -, a reference object of a known size can be included in the image as this allows a comparison

to be made between the height of the object to be measured and the reference object. The previously mentioned Mars rover, Curiosity, carries a United States penny as a reference object so that its cameras can be precisely calibrated using reference charts as shown in Figure 11.

Figure 11: Contact Instrument Calibration Targets on Mars Rover Curiosity (NASA JPL, 2012)

However, the reference object must be located in the image using a technique such as **To do** (18) 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 to give a ratio for applying to the pixel count of the object to calculate its actual height in real world. For example, if 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. Knowing this ratio, any other objects that can be measured in pixels can have the real world measurement estimated as long as they are close to the same distance from the camera as the reference object.

This process will be applied in some manner in this project. Either a reference object will be used or measurements for H_c (the height of the camera from the ground) will be taken at the time the image is captured as knowing these measurements is vital to achieving reliable results that will ensure that suspension units are correctly configured as previously discussed. But 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 skills. In certain sports, such as motorsports **To do** (19), 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.

In such sports, the performances of cars, motorcycles, and even mountain bikes are all 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 important to continually analyse and fine tune their set – up by changing tires or adjusting suspension to seeking out the margin 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 **To do** (20).

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 play a large part in this as utilising various techniques 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 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 perform below par.

2.4 Using 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 Fox Racing Shox using the technology for setup purposes (Benedict, 2012). The mobile application which Fox created is locked to forks and shocks that the company manufacture though image analysis techniques are adaptable enough to be used on any suspension unit.

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

2.5 Conclusion

By identifying the key aspects and settings relating to mountain bike suspension and the common image analysis techniques this literature review has highlighted the processes which this project can utilise to complete its aims. By using this knowledge, the following sections will identify distinct methods which the project will use to produce the application and cover the results. **To do** (21)

3 Methodology

3.1 Introduction

To do (22) This chapter will give an outline of the methodologies used to complete the work identified in the previous chapters as well as the reasons for using these methods and alternatives which could have been used. The effectiveness of these methods can determine the success of the project so the chosen methods were required to be reliable and manageable. A technical approach was identified for the most appropriate way to produce a solution to the problem identified and a project management approach was decided on so that the project could remain on track and meet the required deadline.

3.2 Literature Review

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

There are multiple types of literature review though they do not have the same aims. A traditional or narrative review can be carried out to critique a body of literature and potentially locate inconsistencies **To do** (23); this type is normally used when the research question is well defined. Alternatively a systematic literature review can be produced; this requires a more rigorous process. Systematic reviews aim to locate studies which are within the same or similar subject areas and use them to answer the questions presented by the research **To do** (24).

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

3.2.1 Source Selection

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

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

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

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

3.3 Platform

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

3.3.1 OpenCV

OpenCV is an open-source computer vision library created for a variety of platforms. Originally released by Intel in 1999**To do** (25) the library was intended as a research project to aid in CPU intensive visual applications. In 2012 the library was taken over by OpenCV.org**To do** (26), a non-profit organisation who maintain support and the documentation. Though the library is written in C and C++, modules are available for Python, Fortran, and the Android platform.

This project will use OpenCV as it is widely accepted as the best, free computer vision library available**To do** (27) and provides functions for the various processes this project will require. Additionally it is well supported and documented which will aid in the development process.

3.3.2 Experimentation

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

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

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

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

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

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

3.4 Source Code

3.4.1 Pythonic Coding

A metric of software quality is the conciseness and descriptiveness of the source code. The aim of the Python programming language is to complete the same tasks as other object oriented languages but in fewer lines and in a more readable manner**To do** (28). By removing brackets and instead using indentation to define the boundaries of classes, functions, and conditionals as well as using worded operators, Python creates source code which reads like a list of instructions for people rather than computers. The source

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

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

```
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 API with which other developers can use the system.

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

3.5 Testing

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

3.5.1 Unit Testing

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

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

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

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

3.5.2 Project Testing Scope

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

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

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

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

3.6 Project Management

3.6.1 Agile Development

To do (29) Introduced in 2001 with the writing of the Agile manifesto (Beck et al., 2001), agile development methodologies focus on high quality software products over the rigorous design based structure of traditional waterfall methods. By utilising various techniques such as stand-up meetings, a strong customer focus, and sprint cycles, agile has become widely adopted in industry and has been proven to produce successful projects (VersionOne, 2015).

There are multiple agile methodologies (XP, Scrum, DSDM) all with their own principles and techniques however it is commonplace for a company to create their own development method which picks items from each to tailor to their needs**To do** (30). As this is a solo project with no customer then one set methodology will not be used, instead a variety of methods which will aid the management of the project and increase the efficiency of the development process have been selected. These will be outlined in the following sections.

3.6.1.1 Requirements Analysis Used in some form by the majority of agile methodologies **To do** (31), requirements analysis or requirements engineering is a variety of processes used to create the conditions that a project must meet. These requirements take into account stakeholders, users, and the development team or teams. Each requirement should be documented, actionable, measurable, testable, and traceable to aid in its understanding and completion**To do** (32).

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

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

3.6.1.2 MoSCoW First used in the DSDM agile framework**To do** (33), MoSCoW analysis or prioritisation is the process of taking the requirements of a software product and placing them into one of four categories; must, should, could, and won't have. These deliverables may be prioritised for the entire project or for individual sprint cycles depending on the size and manageability of the project.

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

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

3.6.1.3 Sprint Cycles Used by Scrum development teams**To do** (34), 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 is carried out by the development team to discuss what went well, what didn't, and how this can be rectified for the next sprint.

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

To do (35) As mentioned previously, MoSCoW prioritised requirements will be selected at the start of each sprint cycle to set objectives the for work which will be carried out. Each cycle will aim to complete all of the must have requirements and the majority of should and could haves. At the end of each cycle a retrospective will be carried out which will re-prioritise the uncompleted requirements based on how successful the sprint cycle was. This means requirements may be promoted or demoted respectively.

3.6.2 Organic Development

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

To do (36) To initiate the development process without analysis of requirements would be to begin producing the basic steps or functionality which the system will go through from an vision of the end product. Once started, development would take a natural direction identified by what is needed for the system or to solve any issues encountered.

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

There are advantages and disadvantages to using this method over an agile process. Due to the size of the software being produced the amount of management work required of an agile process could mean the development is over managed
To do (37)

3.6.3 Version Control

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

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

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

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

3.6.4 Milestones

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

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

3.6.5 Threshold

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

3.6.6 Gantt Chart

A Gantt chart is a method of plotting a project's schedule which was invented by Henry Gantt in the 1910s. By breaking down a project into tasks these can then be added to the chart as a bar. Commonly starting at a project's inception date and ending with 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.

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

3.7 Evaluation

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

3.7.1 Validation

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

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

3.7.2 Reliability and Accuracy

Assessing the reliability and accuracy of the application proves that the results produced are consistent and correct; to assess these an uncertainty metric will be produced. When taking multiple measurements there is typically a ”true” value which is the actual measurement that falls within the range of produced measurements. Uncertainty is a prediction of how close any produced measurement will be to this ”true” value, expressed as $\overline{x} \pm U$. The method for producing 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 different variances possible depending on what the application is capable of once produced. These variances will be outlined once the evaluation has been completed.

3.7.3 Comparison to Alternatives

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

3.7.4 Professional Opinion

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

To carry out professional consultation the Mountain Bike Research Centre of Scotland[2] will be contacted, once the application can be demonstrated, to arrange a meeting with an individual who they deem suitable. General evaluative questions will be posed though the meeting will take a semi-formal approach. Additional to this the application will be demonstrated to staff at a local bike shop to also gain their opinion, this will take the same approach as the meeting.

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

4 Results

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

4.1 Deviation from Plan

4.1.1 Development Approach

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

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

4.1.2 Use of EXIF Data and Reference Point

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

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

4.1.3 Colour Quantification

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

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

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

To solve this issue a further image processing technique was implemented known as colour quantification which reduces an image to 2, 4, 8, 16, or 32 bit colour spaces. By reducing the range of colours in an image detail is lost and objects become more apparent. The difference between an original image and quantified colours is shown in figure 12. 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.

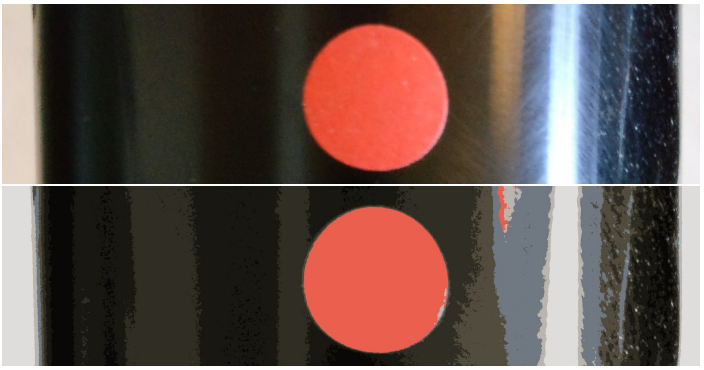


Figure 12: Normal image (top) versus quantified colours (bottom)

4.1.4 Dynamic Measurement Limits

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

So the application is able to analyse different images, a dynamic method to find this o-ring was implemented. Much like finding the reference point an o-ring of specific colour is found, if present, using OpenCV's `findContours`. When located, a bounding box is drawn round the contour from which the y coordinate and height are used to set the limit of measurement. This can be seen in figure 13.

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

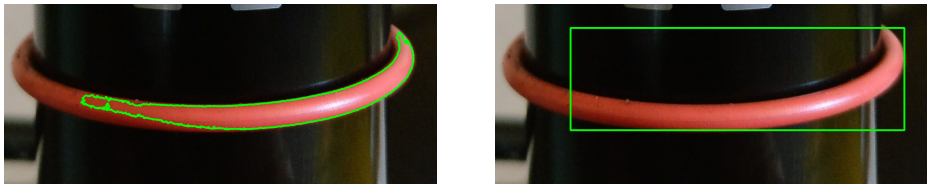


Figure 13: Red O-ring found using `findContours` (left) with `boundingBox` applied (right)

on the shock's shaft are selected. The same bounding box is applied and used as the measurement limit. This process was successful as indicated in figure 14.

Figure 14: Black o-ring found

4.1.5 Reference Point Measurement Method

To measure the circular reference point, the first process applied was Hough Circle Transform. This method was chosen as it sets out to specifically find circles in an image and is simple to implement. Though this did produce a measurement the numbers were regularly incorrect when compared to manually measuring the image and were unreliable, varying between too small and too big. This can be seen in figure 15 where it is clear that the identified circle is much smaller than the actual reference point.

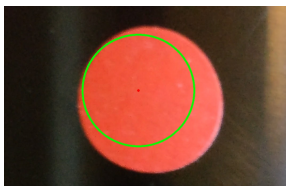


Figure 15: Reference point found using HoughCircles

To circumvent this issue the same process for locating the o-ring was used. A contour is found which encompasses the full reference point and a bounding circle is drawn around this contour; this is shown in figure 16. Though the bounding circle appears larger than the reference point, after experimentation with a variety of images, this method produced measurements which were within tolerances of what is suitable for the pixel per millimetre calculation.

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

Figure 16: Reference point found using findContours (left) with boundingCircle applied (right)

4.1.6 Pressure Calculation

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

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

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

When an air spring is compressed the spring rate increases through its travel as the particles become compressed together[To do (38)]; while this allows tuning of the shock characteristics it presents difficulty when attempting to apply a linear equation in a non-linear situation. However as the range of measurement in this project is small compared to the full stroke of an air shock it was considered that the spring rate is linear within this range.

To investigate this, the sag of a shock was measured at pressures between 100 and 250 psi in increments of 10 psi. This was carried out for two shocks, one with a normal air can and another with a high volume air can then the collected data was then plotted and a linear trend line applied. The results of this can be seen in figure 17.

From this data it could be concluded that, within the range of measurements, the compression of the air spring is linear. The data-points in figure 17 occasionally vary though

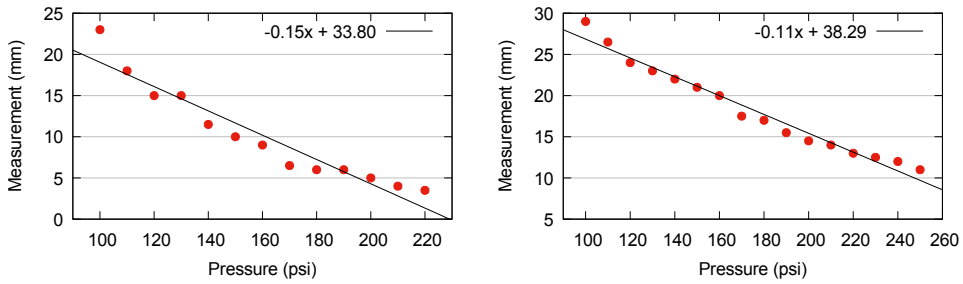


Figure 17: Sag measurements for standard air shock (left) and high volume air shock (right)

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

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

4.2 Application

4.2.1 Images

The application requires two images to measure and produce calculations from. The application can process images in different lighting conditions and ones where a flash has been used though there are some criteria which are needed. There must be a red, circular, 7.5mm diameter sticker on the shock, preferably 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 18.

The user must supply one image of the shock which has been pressurised to 100psi and loaded under the normal riding weight, i.e. with helmet, bag, protection on, and a second with the shock pressurised to 150psi and loaded again. The marker o-ring should be returned to the top of the shock shaft before weight is applied to the shock.

4.2.2 Arguments

The application operates from a command line interface allowing the user to pass in data through arguments; these are described in Table 3. This system uses Python's Argparse package and a full command to run the application would be as follows:
`program -i user/images/100.jpg user/images/150.jpg -s 30 -t 57 -c red`

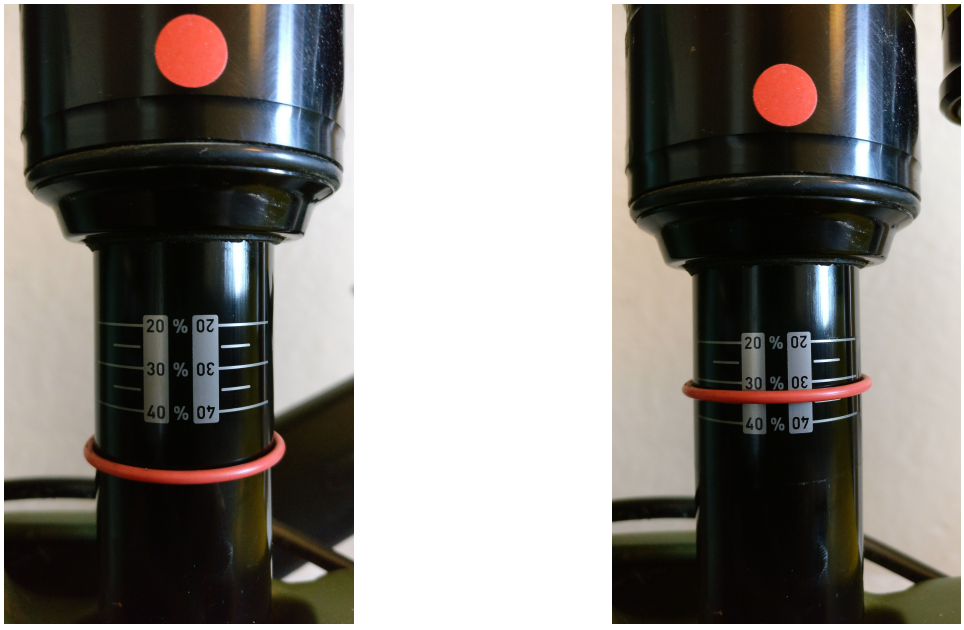


Figure 18: Images suitable for processing

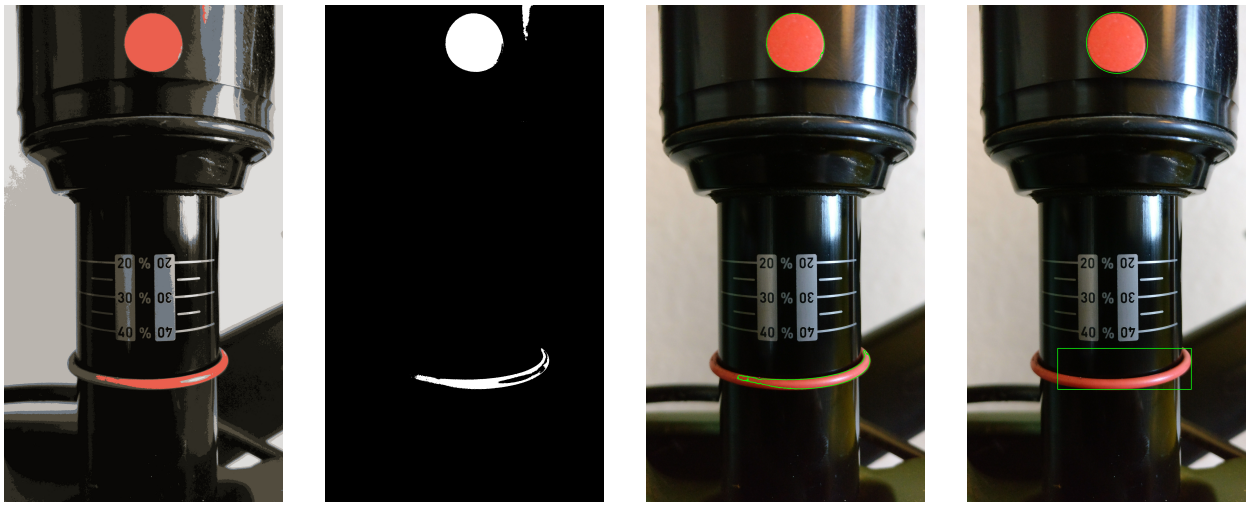
Table 3: Table of application arguments

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

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 19a. 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 19b. 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 19c. A bounding circle is created around the reference point contour, shown in Figure 19d, from which the diameter can be produced which allows the pixels per millimetre metric.

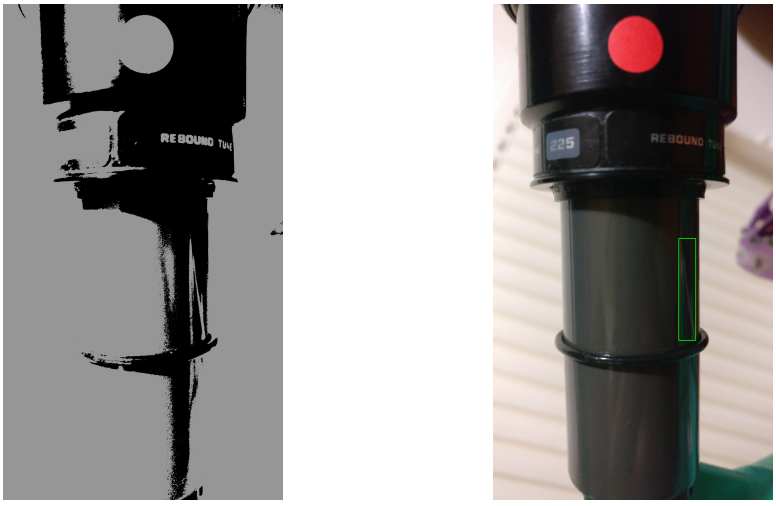


(a) Colour quantified image (b) Image mask (c) Found contours from mask (d) Bounding circle and box from contours

Figure 19: 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 19.

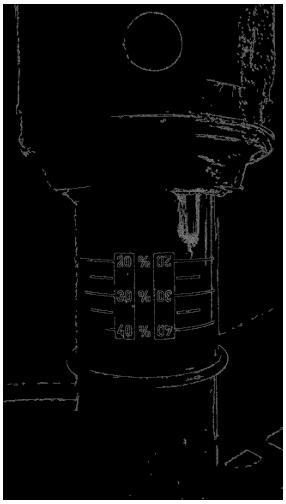
If a black o-ring is specified, an alternative process is used. Thresholding is applied to the image which highlights any shadows or highlights on the shock shaft. As these are intersected by the o-ring it is simple to select a contour which ends at the o-ring to produce a measurement limit from. This process is shown in Figure 20.



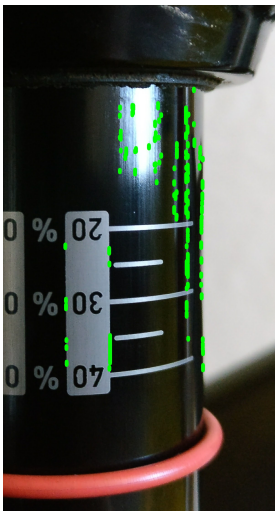
(a) Image thresholding (b) O-ring indicator

Figure 20: 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, this output is shown in Figure 21a. Hough line transformation is then applied to locate vertical lines between the top of the shock shaft and the measurement limit. The difference in pixels between the highest and lowest points of these lines is the current sag measurement. This can be converted to millimetres using the px/mm metric.



(a) Edge detected image



(b) Vertical lines in measurement area

4.2.3.4 Equation Once the measurements for the shock pressurised to 100psi and 150psi are produced, they can then be utilised to create a linear equation. This is carried out using Python's Scipy package and its linregress method which accepts two arrays of data and returns the slope and intercept of the linear equation. The equation is visually described in Figure 22; it should be noted that this plot is only produced virtually and never output to the user.

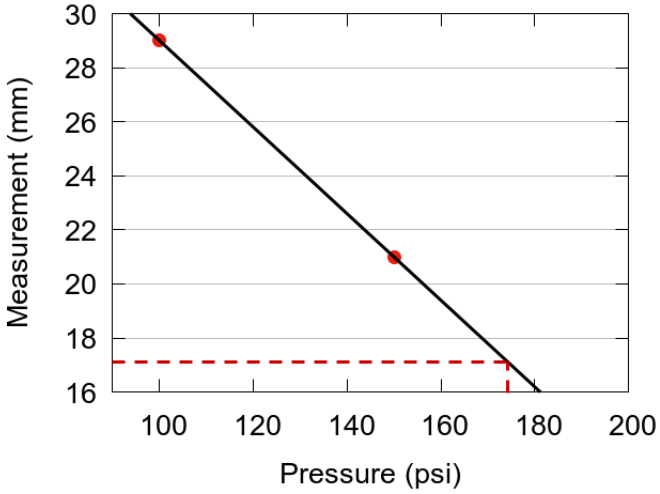


Figure 22: Example of linear equation plot

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 from the user supplied data. For example if the shock stroke is 57mm and the desired sag is 30% then the desired movement is 17.1mm. Using this and the equation produced in the last step, the optimal pressure is calculated; for example if the equation is $-5.017x+246.426$ then the pressure is 161psi.

4.3 Evaluation

4.3.1 Validation

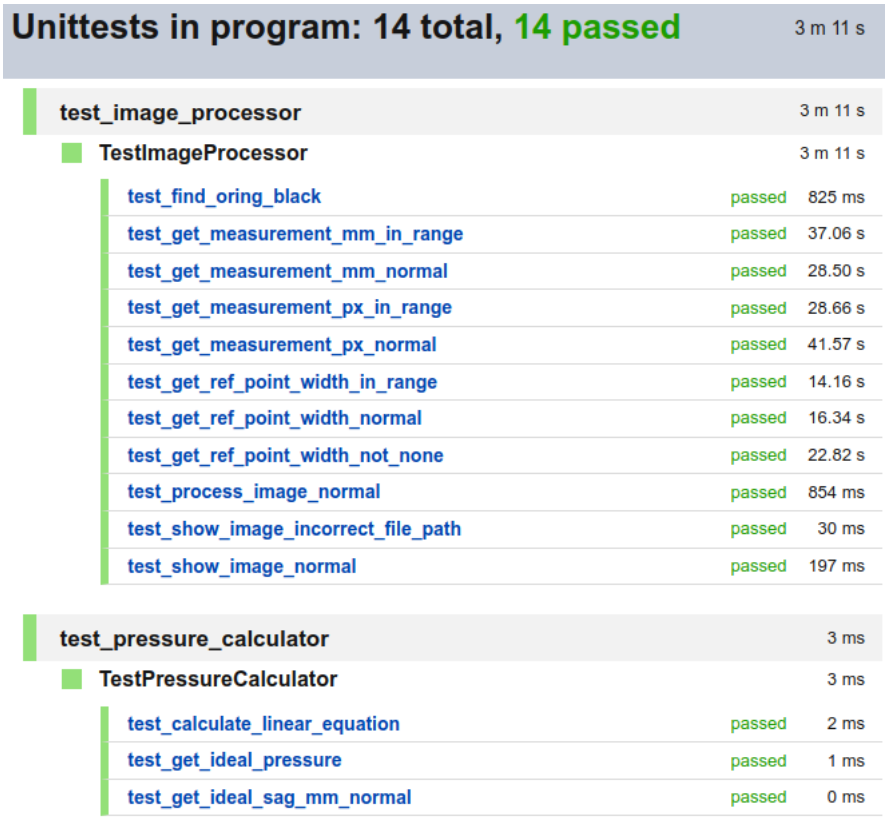


Figure 23: Unit test results

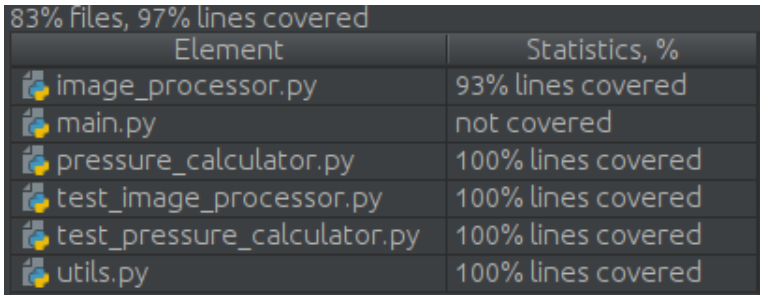


Figure 24: Test coverage results

4.3.2 Reliability and Accuracy

4.3.3 Comparison to Alternatives

4.3.4 Professional Opinion

Table 4: Table of uncertainty calculation results

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

5 Conclusions

5.1 Meeting Aims

5.2 Comparison to other Products

5.3 Future Work

5.4 Self Appraisal

References

Ahmad, I. S., Boufama, B., Habashi, P., Anderson, W., & Elamsy, T. (2015, Dec). Automatic license plate recognition: A comparative study. In *2015 ieee international symposium on signal processing and information technology (isspit)* (p. 635-640). doi: 10.1109/ISSPIT.2015.7394415

Aston, P. (2016). Sussmybike data acquisition - eurobike 2016. Retrieved 24/09/2016, from http://www.pinkbike.com/news/sussmybike-data-acquisition-eurobike-2016.html

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … others (2001). Manifesto for agile software development.

Benedict, T. (2012). Fox ird smartphone app sets ups shocks, forks for you. Retrieved 24/09/2016, from http://www.bikerumor.com/2012/08/29/fox-ird-smartphone-app-sets-ups-shocks-forks-for-you/

European Science Foundation. (2007). *Medical imaging for improved patient care* [Policy Briefing]. Retrieved 03/10/2016, from http://www.esf.org/fileadmin/links/EMRC/ESF_POLICY28_V09_HD.pdf

Giant Manufacturing Co. Ltd. (2017a). *Giant fathom* [Product Listing]. Retrieved from https://www.giant-bicycles.com/gb/fathom-1

Giant Manufacturing Co. Ltd. (2017b). *Giant reign* [Product Listing]. Retrieved from https://www.giant-bicycles.com/gb/reign-1

Giant Manufacturing Co. Ltd. (2017c). *Giant stance* [Product Listing]. Retrieved from https://www.giant-bicycles.com/en-gb/bikes/model/stance/28553/99239/

Honda Motor Co. (2004). Honda develops world's first intelligent night vision system able to detect pedestrians and provide driver cautions-available on legend model to be released in fall 2004 [Press Release]. Retrieved 03/10/2016, from http://world.honda.com/news/2004/4040824a-eng.html

IMBA Europe. (2015). *Imba european mountain bike survey* (Survey). International Mountain Bike Association. Retrieved 24/09/2015, from http://www.imba-europe.com/sites/default/files/IMBA_INFOGRAPHIC_final.pdf

Blake, D. F., Morris, R. V., Kocurek, G., Morrison, S. M., Downs, R. T., Bish, D., … Sarrazin, P. (2013). Curiosity at gale crater, mars: Characterization and analysis of the rocknest sand shadow. *Science*, *341*(6153). Retrieved from http://science.sciencemag.org/content/341/6153/1239505 doi: 10.1126/science.1239505

Callaham, J. (2015). Google says there are now 1.4 billion active android devices worldwide. Retrieved 25/09/2016, from http://www.androidcentral.com/google-says-there-are-now-14-billion-active-android-devices-worldwide

Cork, A., Justham, L., & West, A. (2012). Three-dimensional vision analysis to measure the release characteristics of elite bowlers in cricket. *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, 1754337112447264.

Diaz, K. M., Krupka, D. J., Chang, M. J., Peacock, J., Ma, Y., Goldsmith, J., … Davidson, K. W. (2015). Fitbit®: An accurate and reliable device for wireless physical activity tracking. *International Journal of Cardiology*, *185*, 138-140. Retrieved 24/09/2016, from http://dx.doi.org.ezproxy.napier.ac.uk/10.1016/j.ijcard.2015.03.038

Formosa, N. (2012). Social fitness apps storm cycling world. *Bicycle Retailer and Industry News*, *21*, 28-29. Retrieved 24/09/2016,

from http://search.proquest.com.ezproxy.napier.ac.uk/docview/1018189163?rfr_id=info%3Axri%2Fsid%3Aprimo

Garvin, J. B., Malin, M. C., & Minitti, M. E. (2014, March 17). *Sedimentology of 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

Harker, J. (2010). German brand rose hopes to bloom in britain. Retrieved 24/09/2016, from http://www.bikebiz.com/news/read/german-brand-rose-hopes-to-bloom-in-britain/08815

Hwang, J. C. B. (2016, 05 19). *Shockwiz* (Trademark Application No. 87042899). 1000 West Fulton Market, 4th Floor, Chicago, Illinois, 60607, USA. Retrieved 25/09/2016, from https://trademarks.justia.com/870/42/shockwiz-87042899.html

Kim, S.-W., & Kim, N.-S. (2013). Dynamic characteristics of suspension bridge hanger cables using digital image processing. *NDT & E International*, *59*, 25 - 33. Retrieved from http://www.sciencedirect.com/science/article/pii/S0963869513000807 doi: http://dx.doi.org/10.1016/j.ndteint.2013.05.002

Liszewski, A. (2016). Caterpillar's new s60 is the first smartphone with flir thermal imaging built right in. Retrieved 04/10/2016, from http://gizmodo.com/caterpillars-new-s60-is-the-first-smartphone-with-flir-1759685817

Martin, R. C. (2009). *Clean code: a handbook of agile software craftsmanship*. Pearson Education.

Masad, E., Muhunthan, B., Shashidhar, N., & Harman, T. (1999). Internal structure characterization of asphalt concrete using image analysis. *Journal of Computing in Civil Engineering*, *13*(2), 88-95. Retrieved from http://dx.doi.org/10.1061/(ASCE)0887-3801(1999)13:2(88) doi: 10.1061/(ASCE)0887-3801(1999)13:2(88)

Moeslund, T. B. (2012). *Introduction to video and image processing: Building real systems and applications*. Springer.

NASA JPL. (2012). *Contact instrument calibration targets on mars rover curiosity.* Retrieved from images-assets.nasa.gov/image/PIA15284/PIA15284~orig.jpg

Potter, C. (2013). Ten years of land cover change on the california coast detected using landsat satellite image analysis: part 1—marin and san francisco counties. *Journal of Coastal Conservation*, *17*(4), 697–707. Retrieved from http://dx.doi.org/10.1007/s11852-013-0255-2 doi: 10.1007/s11852-013-0255-2

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

Staff, B. (2015). Yt industries launches consumer-direct sales in north america, oceania. Retrieved 24/09/2015, from http://www.bicycleretailer.com/industry-news/2015/01/30/yt-industries-launches-consumer-direct-sales-north-america-oceania

Titlestad, J., Fairlie-Clarke, T., Davie, M., Whittaker, A., & Grant, S. (2003). Experimental evaluation of mountain bike suspension systems. *Acta Polytechnica*, *43*, 15-20.

VersionOne. (2015). *9th annual state of agile survey.*

Wejer, J., Lendo, I., & Lewczuk, D. (2013). The effect of training on the jumping 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)

A Android Experiments

A.1 Table of Android Experiments

Experiment	Purpose	Functional	Issues	Files	LOC
Hello CV	• Introduction to the android OpenCV library • Displays camera feed with fps	Yes	• Not fullscreen • Incorrect orientation	2	101
15Tile	• Sliding tile game • Uses camera feed as puzzle	Yes	• Not fullscreen	3	492
Blob Detection	• Demonstrates blob detection • Runs blob detection on tapped area from camera	No	• Crash on screen tap	3	311
Face Detection	• Detects faces in camera view • Puts boundary around detected faces	No	• Crash on load	3	279

A.2 Hello CV

```java
package com.example.joe.base_app;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.SurfaceView;
import android.widget.TextView;
import org.opencv.android.*;
import org.opencv.core.*;

public class MainActivity extends AppCompatActivity implements
    CameraBridgeViewBase.CvCameraViewListener2{

    private static String TAG = "Main Activity";
    private CameraBridgeViewBase mOpenCvCameraView;

    private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
        @Override
        public void onManagerConnected(int status) {
            switch (status){
                case LoaderCallbackInterface.SUCCESS:{
                    Log.i(TAG, "Open CV loaded successfully");
                    mOpenCvCameraView.enableView();
                }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();
        }
    }
```

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

A.3 15tile

```
1   package com.example.joe.a15tile;
2
3   import android.annotation.SuppressLint;
4   import android.app.ActionBar;
5   import android.app.Activity;
6   import android.os.Bundle;
7   import android.os.Handler;
8   import android.util.Log;
9   import android.view.Menu;
10  import android.view.MenuItem;
11  import android.view.MotionEvent;
12  import android.view.View;
13  import android.view.WindowManager;
14
15  import org.opencv.android.BaseLoaderCallback;
16  import org.opencv.android.CameraBridgeViewBase;
17  import org.opencv.android.JavaCameraView;
18  import org.opencv.android.LoaderCallbackInterface;
19  import org.opencv.android.OpenCVLoader;
20  import org.opencv.core.Mat;
21
22  public class MainActivity extends Activity implements
    ↪  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   /**
```

```
 * Created by joe on 07/11/16.
 */

import android.util.Log;

import org.opencv.core.*;
import org.opencv.imgproc.Imgproc;

public class PuzzleProcessor {

    private static final int GRID_SIZE = 4;
    private static final int GRID_AREA = GRID_SIZE * GRID_SIZE;
    private static final int GRID_EMPTY_INDEX = GRID_AREA - 1;
    private static final String TAG = "PuzzleProcessor";
    private static final Scalar GRID_EMPTY_COLOR = new Scalar(0x33, 0x33, 0x33, 0xFF);

    private int[] mIndexes;
    private int[] mTextWidths;
    private int[] mTextHeights;

    private Mat mRgba15;
    private Mat[] mCells15;
    private boolean mShowTileNumbers = true;

    public PuzzleProcessor() {
        mTextWidths = new int[GRID_AREA];
        mTextHeights = new int[GRID_AREA];
        mIndexes = new int[GRID_AREA];

        for (int i = 0; i < GRID_AREA; i++) mIndexes[i] = i;
    }

    public synchronized void prepareNewGame() {
        do {
            shuffle(mIndexes);
        } while (!isPuzzleSolvable());
    }

    public synchronized void prepareGameSize(int width, int height) {
        mRgba15 = new Mat(height, width, CvType.CV_8UC4);
        mCells15 = new Mat[GRID_AREA];

        for (int i = 0; i < GRID_SIZE; i++) {
            for (int j = 0; j < GRID_SIZE; j++) {
                int k = i * GRID_SIZE + j;
                mCells15[k] = mRgba15.submat(i * height / GRID_SIZE,
                        (i + 1) * height / GRID_SIZE,
                        j * width / GRID_SIZE,
                        (j + 1) * width / GRID_SIZE);
            }
        }

        for (int i = 0; i < GRID_AREA; i++) {
            Size s = Imgproc.getTextSize(Integer.toString(i + 1), 3, 1, 2, null);
            mTextHeights[i] = (int) s.height;
            mTextWidths[i] = (int) s.width;
        }
    }
```

```
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];
146                 mIndexes[idxToSwap] = touched;
147             }
148         }
149     }
150
151     private void drawGrid(int cols, int rows, Mat drawMat) {
152         for (int i = 1; i < GRID_SIZE; i++) {
153             Imgproc.line(drawMat,
154                     new Point(0, i * rows / GRID_SIZE),
155                     new Point(cols, i * rows / GRID_SIZE),
156                     new Scalar(0, 255, 0, 255),
157                     3);
158             Imgproc.line(drawMat,
159                     new Point(i * cols / GRID_SIZE, rows),
160                     new Point(i * cols / GRID_SIZE, rows),
161                     new Scalar(0, 255, 0, 255),
162                     3);
163         }
164     }
165
166     private static void shuffle(int[] array) {
167         for (int i = array.length; i > 1; i--) {
168             int temp = array[i - 1];
169             int randIx = (int) (Math.random() * i);
170             array[i - 1] = array[randIx];
171             array[randIx] = temp;
172         }
173     }
174
175     private boolean isPuzzleSolvable() {
176         int sum = 0;
177         for (int i = 0; i < GRID_AREA; i++) {
```

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

A.4 BLOB Analysis

```
1  package com.example.joe.blob_analysis;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.util.Log;
6  import android.view.MotionEvent;
7  import android.view.SurfaceView;
8  import android.view.View;
9  import android.view.Window;
10 import android.view.WindowManager;
11
12 import org.opencv.android.BaseLoaderCallback;
13 import org.opencv.android.CameraBridgeViewBase;
14 import org.opencv.android.LoaderCallbackInterface;
15 import org.opencv.android.OpenCVLoader;
16 import org.opencv.core.Core;
17 import org.opencv.core.CvType;
18 import org.opencv.core.Mat;
19 import org.opencv.core.MatOfPoint;
20 import org.opencv.core.Rect;
21 import org.opencv.core.Scalar;
22 import org.opencv.core.Size;
23 import org.opencv.imgproc.Imgproc;
24
25 import java.util.List;
26
27 /**
28  * Created by joe on 12/11/16.
29  */
30
31 public class ColorBlobDetectionActivity extends Activity implements
    ↪  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 }
```

A.5 Face Detection

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

```
14
15      public void start() {
16          nativeStart(mNativeObj);
17      }
18
19      public void stop() {
20          nativeStop(mNativeObj);
21      }
22
23      public void setMinFaceSize(int size) {
24          nativeSetFaceSize(mNativeObj, size);
25      }
26
27      public void detect(Mat imageGray, MatOfRect faces) {
28          nativeDetect(mNativeObj, imageGray.getNativeObjAddr(),
            ↪  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     }
```

B Python Experiments

B.1 Table of Python Experiments

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

B.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)
```

B.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()
```

B.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()
```

B.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']

image = cv2.imread(IMAGE_PATHS[0])
marker = find_marker(image)
focalLength = (marker[1][0] * KNOWN_DISTANCE) / KNOWN_WIDTH

for imagePath in IMAGE_PATHS:
    image = cv2.imread(imagePath)
    marker = find_marker(image)
    inches = distance_to_camera(KNOWN_WIDTH, focalLength, marker[1][0])

    box = np.int0(cv2.cv.BoxPoints(marker))
    cv2.drawContours(image, [box], -1, (0, 255, 0), 2)
    cv2.putText(image, '%.2fft' % (inches/12),
                (image.shape[1] - 200, image.shape[0] -20), cv2.FONT_HERSHEY_SIMPLEX,
                 ↪  2.0, (0, 255, 0), 3)
    cv2.imshow('image', image)
    cv2.waitKey(0)
```

C Development Log

D EXIF Extraction Code

```
36 def get_sensor_size(exif):
37     # (Resolution in pixels / Focal plane resolution in dpi) X 25.4(mm / in) = size in
       ↪  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
```

E Commit Log

F Photos

To do…

- ☐ 1 (p. i): Write abstract
- ☐ 2 (p. 2): Revise aims and objectives
- ☐ 3 (p. 2): Write project restrictions section
- ☐ 4 (p. 3): Finish structure section
- ☐ 5 (p. 5): Figure of 3 bikes overlayed
- ☐ 6 (p. 6): cite
- ☐ 7 (p. 6): cite
- ☐ 8 (p. 6): cite
- ☐ 9 (p. 7): review section adding detail on setup method
- ☐ 10 (p. 9): Make this paragraph not introduce IA and IP again, talk about integer values are manipulated in these
- ☐ 11 (p. 12): cite
- ☐ 12 (p. 12): cite
- ☐ 13 (p. 12): is it just brightness?
- ☐ 14 (p. 12): explain thresholding
- ☐ 15 (p. 13): complete hough transform section
- ☐ 16 (p. 13): explain
- ☐ 17 (p. 13): What is going on here?
- ☐ 18 (p. 14): Add section or briefly explain
- ☐ 19 (p. 14): cite
- ☐ 20 (p. 15): cite
- ☐ 21 (p. 16): revise
- ☐ 22 (p. 17): change to suit
- ☐ 23 (p. 17): cite
- ☐ 24 (p. 17): cite

- ☐ 25 (p. 18): cite
- ☐ 26 (p. 18): cite
- ☐ 27 (p. 18): cite?
- ☐ 28 (p. 19): cite
- ☐ 29 (p. 22): Remove statements saying this will be used
- ☐ 30 (p. 22): cite
- ☐ 31 (p. 22): cite
- ☐ 32 (p. 22): cite
- ☐ 33 (p. 23): cite
- ☐ 34 (p. 23): cite
- ☐ 35 (p. 23): revise in light of dev methods
- ☐ 36 (p. 24): Weird
- ☐ 37 (p. 24): segway
- ☐ 38 (p. 31): cite