Calculation of Mountain Bike Suspension Setup through Mobile 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

09/01/2017

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

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Contents

List of Tables

List of Figures

1 Introduction

1.1 Context

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

1.2 Background

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

Figure 1: Full suspension and hardtail mountain bikes

To ensure the fork and shock function correctly they must be set up for the rider's weight and intended use of the bike. As this is considered a specialist area, many entry and mid level riders will lack the knowledge of this process or be unsure of how the suspension should operate meaning the rider could use the bike without the suspension set up correctly.

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

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

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

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

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

1.3 Aims and Objectives

The aim of this project is to create a prototype mobile application for the Android operating system 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.
- Investigate current applications and products with similar aims
- Design the prototype application
- Implement the prototype application
- Evaluate the produced application against current products

1.4 Restrictions

1.5 Thesis Structure

Chapter 1 - Introduction - Outlines the context of the project and states the major aims and objectives

Chapter 2 - Literature Review -

Chapter 3 - Methodology -

Chapter 4 - Critical Evaluation -

Chapter 5 - Conclusion -

2 Literature Review

The purpose of this section is to introduce the concepts of mountain bike suspension setup and indicate why this may be difficult for a less knowledgeable individuals. It will then continue to identify and explain common image analysis techniques and how they may be applied to mountain bike suspension setup.

2.1 Mountain Bike Suspension Concepts

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

2.1.1 Travel and Stroke

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

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				

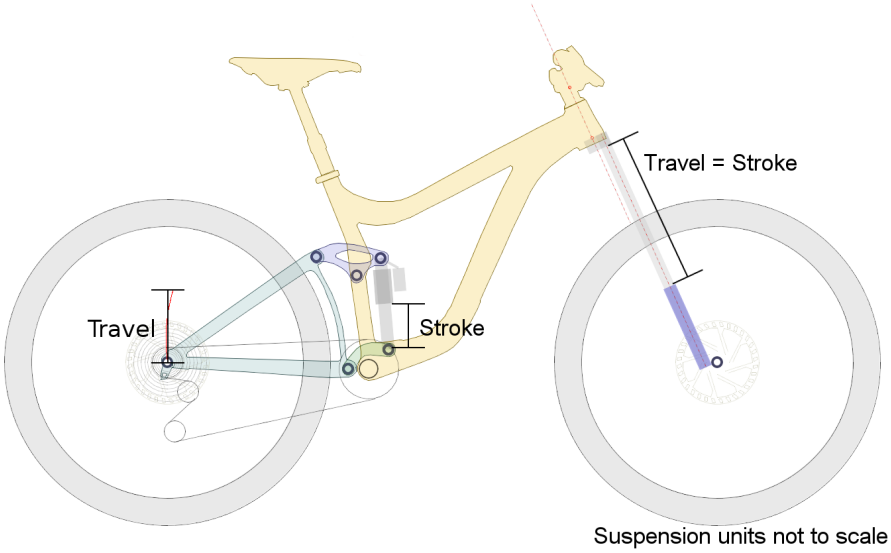

Figure 2: Diagram showing travel and stroke on a full suspension bike[1]

2.1.3 Rear Suspension

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

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

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

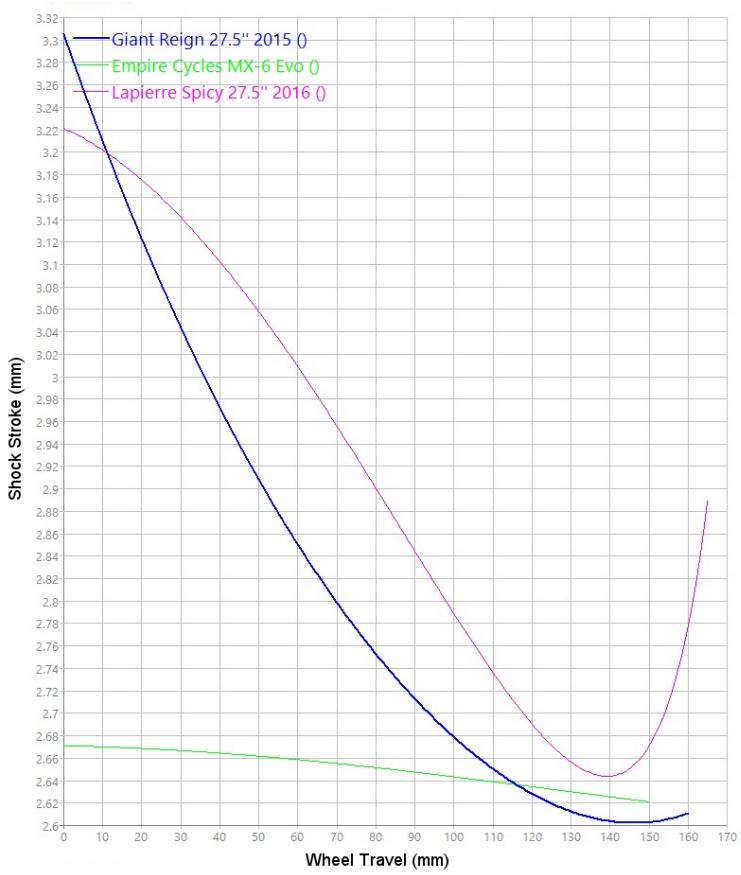

Figure 3: Leverage curves of three modern suspension designs

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

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

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

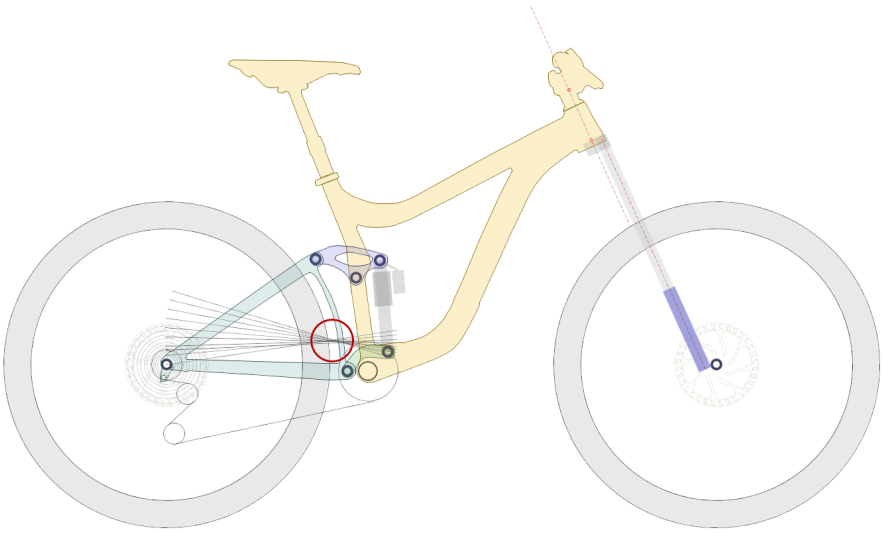

Figure 4: Maestro suspension

2.1.4 Sag

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

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

2.1.5 Damping

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

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

2.1.5.2 Rebound Damping This is used to control the speed at which the shock absorber extends once it has been compressed. An optimal setting will allow the suspension to track the ground, returning after a bump as well as dropping into any holes.

Too much rebound damping causes the suspension to return slowly and sometimes pack down meaning the absorber gradually runs out of travel. Too little can cause the suspension to buck the rider and lead to an accident.

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

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

2.1.6 Optimal Setup

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

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

2.2 Image Analysis

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

2.2.1 Digital Camera Operation

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

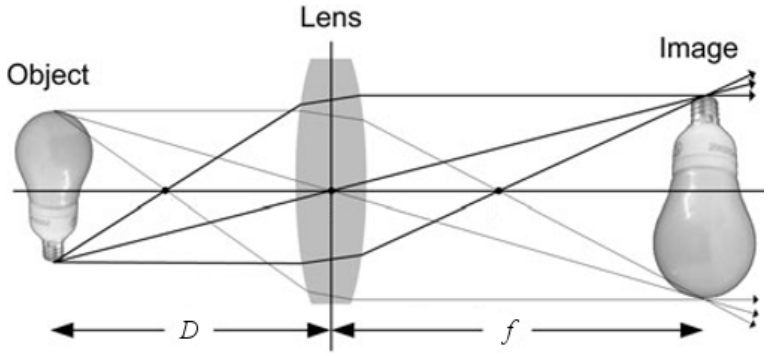

Figure 5: Diagram of camera and lens operation

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

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

These integer values in the image data can be manipulated to change the visual image or the values themselves analysed or compared to neighbouring cells in order to locate objects. The processes are known as image processing and image analysis.

2.2.2 Usages

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

2.2.2.1 Medical Arguably one of the most important uses of IA, advances in medical imaging have reduced costs, diagnosis time, recovery time, and improved the ability to localise and personalise treatments (European Science Foundation, 2007). Major uses of IA in medical applications are 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 6.

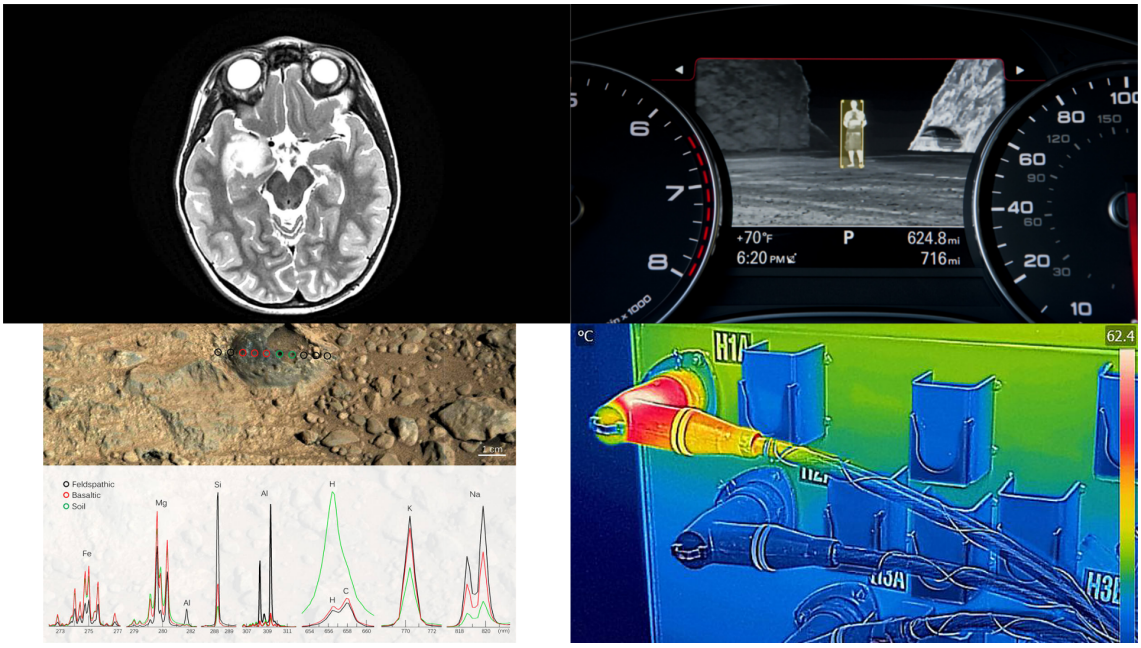

Figure 6: Uses of image analysis, from top left clockwise: An MRI brain scan, automotive night vision with pedestrian recognition, infrared image taken with a smartphone, chemical rock analysis from mars

2.2.2.2 Transport Image analysis has been included in the consumer automotive market on various models since 2004 when Honda introduced an thermographic night vision camera with pedestrian detection on the Legend (Honda Motor Co., 2004), Audi's implementation can be seen top right in figure 6. Since this initial introduction many vehicle manufacturers have included image analysis and recognition features as options such as speed limit sign recognition, lane departure warning systems, and automatic braking systems based on hazard recognition.

2.2.2.3 Engineering The use of image analysis in engineering has pushed to create more stable and efficient structures by looking at the materials used in their construction (Masad, Muhunthan, Shashidhar, & Harman, 1999) and monitoring their stresses and potential weak areas (Kim & Kim, 2013). Using image analysis by engineers on site has become more common with the advances in mobile computing and some manufacturers aiming their products at an engineering demographic with features like improved durability and built in infrared imaging (Liszewski, 2016), the output of which can be seen lower right in figure 6.

2.2.2.4 Space While some industries make use of satellite imagery to monitor changes on our own planet, agencies such as NASA and ESA make use of image analysis to look at other planets and celestial bodies. The Martian rover, Curiosity, uses multiple cameras for navigation, hazard avoidance, and scientific imaging the products of which are streamed back to Earth for analysis. Major uses for the various types of images returned include identification of geological formations and compositions (Blake et al., 2013; Garvin, Malin, & Minitti, 2014) and chemical location and identification using the "ChemCam" (Schröder et al., 2015). Chem cam analysis can be seen lower left in figure 6.

2.2.3 Image Analysis Techniques

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

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

Table 2: Table of pixel data showing an edge

5	7	6	4	152	148	149

Table 2 represents possible pixel values of an edge indicated by the large difference between 4 and 152. The applied algorithm will pick up on this discrepancy and in will be indicated on the resulting image. A common application for edge detection is text recognition such as in automatic number plate recognition (ANPR) (Ahmad, Boufama, Habashi, Anderson, & Elamsy, 2015) as the process can remove unwanted background data and highlight the block shapes of the number plate. Figure 7 shows an image which has had thresholding applied with the number plate of the car clearly visible.

Figure 7: Edge detection applied to an image for number plate recognition

2.2.3.2 Object Detection Much like edge detection, object detection is used to pick out features in images, the difference being that object is a more abstracted term than edge and could mean anything from faces to signs to company logos. A common technique is Binary Large OBject (BLOB) analysis which is two techniques combined into

one application (Moeslund, 2012), this includes BLOB extraction which is used to isolate large objects in an image, dismissing small objects as noise, which is then followed by BLOB classification, assigning objects a class based on predetermined parameters.

Regularly carried out after grayscaling and thresholding an image, BLOB extraction uses a grass-fire algorithm to locate pixels which display a significant difference to the image background and discover the full extent of this region. As all the pixels of each BLOB in a object are known, certain parameters are also known such as size, area, and boundaries. From these parameters, calculations can be carried out to discover the classification of each BLOB, for example with the boundary and area of each BLOB, the circularity can be calculated to locate all the circular objects in an image.

$$B_c = \frac{B_p}{2\sqrt{\pi \times B_a}} \tag{1}$$

where

B_c is the circularity of the BLOB
B_p is the perimeter of the BLOB's bounding box
B_a is the area of the BLOB

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

$$H_o = \left(\frac{f \times \left(\frac{H_i}{H_s}\right)}{D - f} \right) \times H_c \tag{2}$$

where

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

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

To circumvent the need for this data a reference object of known size can be included in the image, this allows a comparison between the object to be measured and the reference object. The previously mentioned mars rover, Curiosity, carries a United States penny and charts to calibrate its cameras against; this is shown in figure 8.

This process will be applied during this project in some manner, either using a reference object or measuring camera height, as knowing measurements is vital to suspension

Figure 8: Contact Instrument Calibration Targets on Mars Rover Curiosity (NASA JPL, 2012)

setup as previously stated. Implementation of equation 2 in an Android application will be relatively simple though collection of the relevant data with as little user interaction as possible will be difficult.

2.3 Image Analysis in Sports Science

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

A car, motorcycle, and mountain bike are all predictable and measurable as they have been designed and manufactured by engineers so can easily be fitted with sensors, analysed, and adjusted accordingly. Fitting sensors to humans while maintaining their full mobility is not so simple. To provide the best testing ground for fitness and technique, the participant should be unhindered and able to perform tasks without data capturing equipment getting in the way.

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

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

2.4 Using Image Analysis for Mountain Bike Suspension

Though image analysis has been used for calculations in many engineering 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.

3 Methodology

3.1 Introduction

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 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 Platform

As a proof of concept, the final product of this software project could take a variety of forms. Inline 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.

3.2.1 Experimentation

To further understand what may be required to produce both the Android and Python based approaches some basic experiments were carried out to compare the two and aid in deciding which method to use. 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 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 a stack trace of the errors 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 it 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 either.

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. 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.

3.3 Project Management

3.3.1 Agile Development

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 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. 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.

3.3.1.1 Requirements Analysis

3.3.1.2 MoSCoW First used in the DSDM agile framework, 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 requirements 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.

3.3.1.3 Sprint Cycles Used by Scrum development teams, 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.

3.3.2 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 free use of a cloud based 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.

4 Results

5 Critical Evaluation

6 Conclusion

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. (2017). *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

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

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)

Acronyms

FS Full Suspension.

HT Hard Tail.

IA Image Analysis.

VCS Version Control System.

VPP Virtual Pivot Point.

Glossary

Fork The front suspension unit on a mountain bike.

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

Hard Tail A mountain bike with only front suspension.

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

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

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

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

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

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

Travel The distance a wheel can move before bottoming out.

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

A Android Experiments

A.1 Table of Android Experiments

Experiment	Purpose	Functiona?l	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

```
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.

    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, mLoaderCallba
    }

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

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

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

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

    }

    @Override
    public void onCameraViewStopped() {

    }

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

A.3 15tile

```
package com.example.joe.a15tile;

import android.annotation.SuppressLint;
import android.app.ActionBar;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.WindowManager;
```

```
import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.JavaCameraView;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.Mat;

public class MainActivity extends Activity implements CameraBridgeViewBase.CvCameraV
    private static final boolean AUTO_HIDE = true;
    private static final int AUTO_HIDE_DELAY_MILLIS = 3000;
    private static final int UI_ANIMATION_DELAY = 300;
    private final Handler mHideHandler = new Handler();
    private View mContentView;
    private final Runnable mHidePart2Runnable = new Runnable() {
        @SuppressLint("InlinedApi")
        @Override
        public void run() {
            // Delayed removal of status and navigation bar

            // Note that some of these constants are new as of API 16 (Jelly Bean)
            // and API 19 (KitKat). It is safe to use them, as they are inlined
            // at compile-time and do nothing on earlier devices.
//              mContentView.setSystemUiVisibility(View.SYSTEM_UI_FLAG_LOW_PROFILE
//                      | View.SYSTEM_UI_FLAG_FULLSCREEN
//                      | View.SYSTEM_UI_FLAG_LAYOUT_STABLE
//                      | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY
//                      | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
//                      | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION);
        }
    };
    private View mControlsView;
    private final Runnable mShowPart2Runnable = new Runnable() {
        @Override
        public void run() {
            // Delayed display of UI elements
            ActionBar actionBar = getActionBar();
            if (actionBar != null) {
                actionBar.show();
            }
            mControlsView.setVisibility(View.VISIBLE);
        }
    };
    private boolean mVisible;
    private final Runnable mHideRunnable = new Runnable() {
        @Override
        public void run() {
            hide();
        }
```

```
};
private final View.OnTouchListener mDelayHideTouchListener = new View.OnTouchList
    @Override
    public boolean onTouch(View view, MotionEvent motionEvent) {
        if (AUTO_HIDE) {
            delayedHide(AUTO_HIDE_DELAY_MILLIS);
        }
        return false;
    }
};


private static final String TAG = "MainActivity";
private CameraBridgeViewBase mOpenCvCameraView;
private PuzzleProcessor mPuzzleProcessor;
private MenuItem mItemHideNumbers;
private MenuItem mItemStartNewGame;
private int mGameWidth;
private int mGameHeight;

private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
    @Override
    public void onManagerConnected(int status) {
        switch (status) {
            case LoaderCallbackInterface.SUCCESS: {
                Log.i(TAG, "OpenCV loaded successfully");
                mOpenCvCameraView.setOnTouchListener(MainActivity.this);
                mOpenCvCameraView.enableView();
            }
            break;
            default: {
                super.onManagerConnected(status);
            }
            break;
        }

    }
};

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
    Log.d(TAG, "onCreate: Creating and setting view");
    mOpenCvCameraView = new JavaCameraView(this, -1);
    setContentView(mOpenCvCameraView);
    mOpenCvCameraView.setVisibility(CameraBridgeViewBase.VISIBLE);
    mOpenCvCameraView.setCvCameraViewListener(this);
```

```
        mPuzzleProcessor = new PuzzleProcessor();
        mPuzzleProcessor.prepareNewGame();
        mVisible = true;
        //mControlsView = findViewById(R.id.fullscreen_content_controls);
        //mContentView = findViewById(R.id.fullscreen_content);
        // Set up the user interaction to manually show or hide the system UI.
        //mContentView.setOnClickListener(new View.OnClickListener() {
        //    @Override
        //    public void onClick(View view) {
        //        toggle();
        //    }
        //});
        // Upon interacting with UI controls, delay any scheduled hide()
        // operations to prevent the jarring behavior of controls going away
        // while interacting with the UI.
        //findViewById(R.id.dummy_button).setOnTouchListener(mDelayHideTouchListener
    }

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

    @Override
    public void onResume() {
        super.onResume();
        if (!OpenCVLoader.initDebug()) {
            Log.d(TAG, "onResume: Internal OpenCV library not found. Using OpenCV man
            OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0, this, mLoaderCa
        } else {
            Log.d(TAG, "onResume: OpenCV lib found inside package. Using it!");
            mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
        }
    }

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

    @Override
    protected void onPostCreate(Bundle savedInstanceState) {
        super.onPostCreate(savedInstanceState);

        // Trigger the initial hide() shortly after the activity has been
        // created, to briefly hint to the user that UI controls
```

```
        // are available.
        delayedHide(100);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        Log.i(TAG, "onCreateOptionsMenu: called");
        mItemHideNumbers = menu.add("Show/hide the tile numbers");
        mItemStartNewGame = menu.add("Start a new game");
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        Log.i(TAG, "onOptionsItemSelected: Menu item selected" + item);
        if (item == mItemStartNewGame) {
            mPuzzleProcessor.prepareNewGame();
        } else if (item == mItemHideNumbers) {
            mPuzzleProcessor.toggleTileNumbers();
        }
        return true;
    }

    public void onCameraViewStarted(int width, int height) {
        mGameWidth = width;
        mGameHeight = height;
        mPuzzleProcessor.prepareGameSize(width, height);
    }

    public void onCameraViewStopped() {
    }

    public boolean onTouch(View view, MotionEvent event) {
        int xpos, ypos;

        xpos = (view.getWidth() - mGameWidth) / 2;
        xpos = (int) event.getX() - xpos;

        ypos = (view.getHeight() - mGameHeight) / 2;
        ypos = (int) event.getY() - ypos;

        if (xpos >= 0 && xpos <= mGameWidth && ypos >= 0 && ypos <= mGameHeight)
            mPuzzleProcessor.deliverTouchEvent(xpos, ypos);

        return false;
    }

    public Mat onCameraFrame(Mat inputFrame) {
```

```
        return mPuzzleProcessor.puzzleFrame(inputFrame);
    }

    private void toggle() {
        if (mVisible) {
            hide();
        } else {
            show();
        }
    }

    private void hide() {
        // Hide UI first
        ActionBar actionBar = getActionBar();
        if (actionBar != null) {
            actionBar.hide();
        }
        //mControlsView.setVisibility(View.GONE);
        mVisible = false;

        // Schedule a runnable to remove the status and navigation bar after a delay
        mHideHandler.removeCallbacks(mShowPart2Runnable);
        mHideHandler.postDelayed(mHidePart2Runnable, UI_ANIMATION_DELAY);
    }

    @SuppressLint("InlinedApi")
    private void show() {
        // Show the system bar
        mContentView.setSystemUiVisibility(View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
                | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION);
        mVisible = true;

        // Schedule a runnable to display UI elements after a delay
        mHideHandler.removeCallbacks(mHidePart2Runnable);
        mHideHandler.postDelayed(mShowPart2Runnable, UI_ANIMATION_DELAY);
    }

    /**
     * Schedules a call to hide() in [delay] milliseconds, canceling any
     * previously scheduled calls.
     */
    private void delayedHide(int delayMillis) {
        mHideHandler.removeCallbacks(mHideRunnable);
        mHideHandler.postDelayed(mHideRunnable, delayMillis);
    }
}
```

```
package com.example.joe.a15tile;
```

```java
/**
 * 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;
        }
    }

    public synchronized Mat puzzleFrame(Mat inputPicture) {
        Mat[] cells = new Mat[GRID_AREA];
        int rows = inputPicture.rows();
        int cols = inputPicture.cols();

        rows = rows - rows % 4;
        cols = cols - cols % 4;

        for (int i = 0; i < GRID_SIZE; i++) {
            for (int j = 0; j < GRID_SIZE; j++) {
                int k = i * GRID_SIZE + j;
                cells[k] = inputPicture.submat(i * inputPicture.rows() / GRID_SIZE,
                        (i + 1) * inputPicture.rows() / GRID_SIZE,
                        j * inputPicture.cols() / GRID_SIZE,
                        (j + 1) * inputPicture.cols() / GRID_SIZE);
            }
        }

        rows = rows - rows % 4;
        cols = cols - cols % 4;

        for (int i = 0; i < GRID_AREA; i++) {
            int idx = mIndexes[i];
            if (idx == GRID_EMPTY_INDEX)
                mCells15[i].setTo(GRID_EMPTY_COLOR);
            else {
                cells[idx].copyTo(mCells15[i]);
                if (mShowTileNumbers) {
                    Imgproc.putText(mCells15[i],
                            Integer.toString(1 + idx),
                            new Point((cols / GRID_SIZE - mTextWidths[idx]) / 2,
                                    (rows / GRID_SIZE + mTextHeights[idx]) / 2),
                            3,
                            1,
                            new Scalar(255, 0, 0, 255),
                            2);
```

```
                }
            }
        }

        for (int i = 0; i < GRID_AREA; i++) cells[i].release();

        drawGrid(cols, rows, mRgba15);

        return mRgba15;
    }

    public void toggleTileNumbers() {
        mShowTileNumbers = !mShowTileNumbers;
    }

    public void deliverTouchEvent(int x, int y) {
        int rows = mRgba15.rows();
        int cols = mRgba15.cols();

        int row = (int) Math.floor(y * GRID_SIZE / rows);
        int col = (int) Math.floor(x * GRID_SIZE / cols);

        if (row < 0 || row >= GRID_SIZE || col < 0 || col >= GRID_SIZE) {
            Log.e(TAG, "deliverTouchEvent: Touch event ouside of image not expected")
            return;
        }

        int idx = row * GRID_SIZE + col;
        int idxToSwap = -1;

        if (idxToSwap < 0 && col > 0)
            if (mIndexes[idx - 1] == GRID_EMPTY_INDEX)
                idxToSwap = idx - 1;
        if (idxToSwap < 0 && col < GRID_SIZE - 1)
            if (mIndexes[idx + 1] == GRID_EMPTY_INDEX)
                idxToSwap = idx + 1;
        if (idxToSwap < 0 && row > 0)
            if (mIndexes[idx - GRID_SIZE] == GRID_EMPTY_INDEX)
                idxToSwap = idx - GRID_SIZE;
        if (idxToSwap < 0 && row < GRID_SIZE - 1)
            if (mIndexes[idx + GRID_SIZE] == GRID_EMPTY_INDEX)
                idxToSwap = idx + GRID_SIZE;

        if (idxToSwap >= 0) {
            synchronized (this) {
                int touched = mIndexes[idx];
                mIndexes[idx] = mIndexes[idxToSwap];
                mIndexes[idxToSwap] = touched;
```

```
            }
        }
    }

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

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

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

}
```

A.4 BLOB Analysis

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

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.MotionEvent;
import android.view.SurfaceView;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;

import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;

import java.util.List;

/**
 * Created by joe on 12/11/16.
 */

public class ColorBlobDetectionActivity extends Activity implements View.OnTouchList
    private static final String TAG = "Activity: ";

    private boolean mIsColorSelected = false;
    private Mat mRgba;
    private Scalar mBlobColorRgba;
    private Scalar mBlobColorHsv;
    private ColorBlobDetector mDetector;
    private Mat mSpectrum;
    private Size SPECTRUM_SIZE;
    private Scalar CONTOUR_COLOR;

    private CameraBridgeViewBase mOpenCvCameraView;

    private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
        @Override
```

```
        public void onManagerConnected(int status) {
            switch (status) {
                case LoaderCallbackInterface.SUCCESS: {
                    Log.i(TAG, "onManagerConnected: OpenCV Loaded");
                    mOpenCvCameraView.enableView();
                    mOpenCvCameraView.setOnTouchListener(ColorBlobDetectionActivity.t
                }
                break;
                default: {
                    super.onManagerConnected(status);
                }
                break;
            }
        }
    };

    public ColorBlobDetectionActivity() {
        Log.i(TAG, "ColorBlobDetectionActivity: Instantiated new");
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        Log.i(TAG, "onCreate: Called");
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
        setContentView(R.layout.color_blob_detection_surface_view);
        mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.color_blob_detec
        mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);
        mOpenCvCameraView.setCvCameraViewListener(this);
    }

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

    @Override
    public void onResume() {
        super.onResume();
        if (!OpenCVLoader.initDebug()) {
            Log.d(TAG, "onResume: OpenCV not found. Using Manager");
            OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0, this, mLoaderCa
        } else {
            Log.d(TAG, "onResume: OpenCV found in package.");
```

```
            mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
        }
    }

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

    public void onCameraViewStarted(int width, int height) {
        mRgba = new Mat(height, width, CvType.CV_8UC4);
        mDetector = new ColorBlobDetector();
        mSpectrum = new Mat();
        mBlobColorRgba = new Scalar(255);
        mBlobColorHsv = new Scalar(255);
        SPECTRUM_SIZE = new Size(200, 64);
        CONTOUR_COLOR = new Scalar(255, 0, 0, 255);
    }

    public void onCameraViewStopped() {
        mRgba.release();
    }

    public boolean onTouch(View v, MotionEvent event){
        int cols = mRgba.cols();
        int rows = mRgba.rows();

        int xOffset = (mOpenCvCameraView.getWidth() - cols) / 2;
        int yOffset = (mOpenCvCameraView.getHeight() - rows) / 2;

        int x = (int)event.getX() - xOffset;
        int y = (int) event.getY() - yOffset;

        Log.i(TAG, "onTouch: Touch coordinates: (" + x + ", " + y + ")");

        if ((x < 0) || (y < 0) || (x > cols) || (y > rows)) return false;

        Rect touchedRect = new Rect();

        touchedRect.x = (x > 4) ? x - 4 : 0;
        touchedRect.y = (y > 4) ? y - 4 : 0;

        touchedRect.width = (x + 4 < cols) ? x + 4 - touchedRect.x : cols - touchedRe
        touchedRect.height = (y + 4 < rows) ? y + 4 - touchedRect.y : rows - touchedR

        Mat touchedRegionRgba = mRgba.submat(touchedRect);
```

```
        Mat touchedRegionHsv = new Mat();
        Imgproc.cvtColor(touchedRegionRgba, touchedRegionHsv, Imgproc.COLOR_RGB2HSV_F

        mBlobColorHsv = Core.sumElems(touchedRegionHsv);
        int pointCount = touchedRect.width * touchedRect.height;
        for (int i = 0; i < mBlobColorHsv.val.length; i++) {
            mBlobColorHsv.val[i] /= pointCount;
        }

        mBlobColorRgba = convertScalarHsv2Rgba(mBlobColorHsv);

        Log.i(TAG, "onTouch: Touched rgba color: (" + mBlobColorRgba.val[0] + ", " +

        mDetector.setHsvColor(mBlobColorHsv);

        Imgproc.resize(mDetector.getSpectrum(), mSpectrum, SPECTRUM_SIZE);
        mIsColorSelected = true;

        touchedRegionRgba.release();
        touchedRegionHsv.release();

        return false;
    }

    public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
        mRgba = inputFrame.rgba();
        if (mIsColorSelected) {
            mDetector.process(mRgba);
            List<MatOfPoint> contours = mDetector.getContours();
            Log.e(TAG, "onCameraFrame: Contours count: " + contours.size());
            Imgproc.drawContours(mRgba, contours, -1, CONTOUR_COLOR);
            Mat colorLabel = mRgba.submat(4, 68, 4, 68);
            colorLabel.setTo(mBlobColorRgba);
            Mat spectrumLabel = mRgba.submat(4, 4 + mSpectrum.rows(), 70, 70 + mSpect
            mSpectrum.copyTo(spectrumLabel);
        }

        return mRgba;
    }

    private Scalar convertScalarHsv2Rgba(Scalar hsvColor) {
        Mat pointMatRgba = new Mat();
        Mat pointMatHsv = new Mat(1, 1, CvType.CV_8UC3, hsvColor);
        Imgproc.cvtColor(pointMatHsv, pointMatRgba, Imgproc.COLOR_HSV2RGB_FULL, 4);
        return new Scalar(pointMatRgba.get(0, 0));
    }
```

```
}
```

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

import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.Scalar;
import org.opencv.imgproc.Imgproc;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 * Created by joe on 12/11/16.
 */

public class ColorBlobDetector {

    private Scalar mLowerBound = new Scalar(0);
    private Scalar mUpperBound = new Scalar(0);

    private static double mMinContourArea = 0.1;

    private Scalar mColorRadius = new Scalar(25, 50, 50, 0);
    private Mat mSpectrum = new Mat();
    private List<MatOfPoint> mContours = new ArrayList<MatOfPoint>();

    Mat mPyrDownMat = new Mat();
    Mat mHsvMat = new Mat();
    Mat mMask = new Mat();
    Mat mDilatedMask = new Mat();
    Mat mHierarchy = new Mat();

    public void setColorRadius(Scalar radius) {
        mColorRadius = radius;
    }

    public void setHsvColor(Scalar hsvColor) {
        double minH = (hsvColor.val[0] >= mColorRadius.val[0]) ? hsvColor.val[0] - mC
        double maxH = (hsvColor.val[0] + mColorRadius.val[0] <= 255) ? hsvColor.val[0

        mLowerBound.val[0] = minH;
        mUpperBound.val[0] = maxH;

        mLowerBound.val[1] = hsvColor.val[1] - mColorRadius.val[1];
        mUpperBound.val[1] = hsvColor.val[1] + mColorRadius.val[1];
```

```
        mLowerBound.val[2] = hsvColor.val[2] - mColorRadius.val[2];
        mUpperBound.val[2] = hsvColor.val[2] + mColorRadius.val[2];

        mLowerBound.val[3] = 0;
        mUpperBound.val[3] = 255;

        Mat spectrumHsv = new Mat(1, (int)(maxH-minH), CvType.CV_8UC3);

        for (int j = 0; j < maxH-minH; j++) {
            byte[] tmp = {(byte)(minH+j), (byte)255, (byte)255};
            spectrumHsv.put(0, j, tmp);
        }

        Imgproc.cvtColor(spectrumHsv, mSpectrum, Imgproc.COLOR_HSV2RGB_FULL);
    }

    public Mat getSpectrum() {
        return mSpectrum;
    }

    public void setMinContourArea(double area){mMinContourArea = area;}

    public void process(Mat rgbaImage){
        Imgproc.pyrDown(rgbaImage, mPyrDownMat);
        Imgproc.pyrDown(mPyrDownMat, mPyrDownMat);

        Imgproc.cvtColor(mPyrDownMat, mHsvMat, Imgproc.COLOR_RGB2HSV_FULL);

        Core.inRange(mHsvMat, mLowerBound, mUpperBound, mMask);
        Imgproc.dilate(mMask, mDilatedMask, new Mat());

        List<MatOfPoint> contours = new ArrayList<MatOfPoint>();

        Imgproc.findContours(mDilatedMask, contours, mHierarchy, Imgproc.RETR_EXTERNA

        double maxArea = 0;
        Iterator<MatOfPoint> each = contours.iterator();
        while (each.hasNext()) {
            MatOfPoint wrapper = each.next();
            double area = Imgproc.contourArea(wrapper);
            if (area > maxArea) {
                maxArea = area;
            }
        }

        mContours.clear();
        each = contours.iterator();
```

```
        while (each.hasNext()) {
            MatOfPoint contour = each.next();
            if (Imgproc.contourArea(contour) > mMinContourArea*maxArea) {
                Core.multiply(contour, new Scalar(4,4), contour);
                mContours.add(contour);
            }
        }
    }

    public List<MatOfPoint> getContours(){return mContours;}

}
```

A.5 Face Detection

```
package com.example.joe.face_recognition;

import org.opencv.core.Mat;
import org.opencv.core.MatOfRect;

/**
 * Created by joe on 08/11/16.
 */

public class DetectionBasedTracker {
    public DetectionBasedTracker(String cascadeName, int minFaceSize) {
        mNativeObj = nativeCreateObject(cascadeName, minFaceSize);
    }

    public void start() {
        nativeStart(mNativeObj);
    }

    public void stop() {
        nativeStop(mNativeObj);
    }

    public void setMinFaceSize(int size) {
        nativeSetFaceSize(mNativeObj, size);
    }

    public void detect(Mat imageGray, MatOfRect faces) {
        nativeDetect(mNativeObj, imageGray.getNativeObjAddr(), faces.getNativeObjAddr
    }

    public void release() {
        nativeDestroyObject(mNativeObj);
        mNativeObj = 0;
```

```
    }

    private long mNativeObj = 0;

    private static native long nativeCreateObject(String cascadeName, int minFaceSize
    private static native void nativeDestroyObject(long thiz);
    private static native void nativeStart(long thiz);
    private static native void nativeStop(long thiz);
    private static native void nativeSetFaceSize(long thiz, int size);
    private static native void nativeDetect(long thiz, long inputImage, long faces);
}
```

```
package com.example.joe.face_recognition;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.WindowManager;

import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.Mat;
import org.opencv.core.MatOfRect;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;
import org.opencv.objdetect.CascadeClassifier;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;

/**
 * Created by joe on 08/11/16.
 */

public class FrActivity extends Activity implements CameraBridgeViewBase.CvCameraVie

    private static final String TAG              = "FrActivity";
    private static final Scalar FACE_RECT_COLOR = new Scalar(0,255,0,255);
    private static final int    JAVA_DETECTOR    = 0;
    private static final int    NATIVE_DETECTOR = 1;
```

```
    private MenuItem mItemFace50;
    private MenuItem mItemFace40;
    private MenuItem mItemFace30;
    private MenuItem mItemFace20;
    private MenuItem mItemType;

    private Mat                    mRgba;
    private Mat                    mGray;
    private File                   mCascadeFile;
    private CascadeClassifier      mJavaDetector;
    private DetectionBasedTracker  mNativeDetector;

    private int          mDetectorType = JAVA_DETECTOR;
    private String[]     mDetectorName;

    private float   mRelativeFaceSize = 0.2f;
    private int     mAbsoluteFaceSize = 0;

    private CameraBridgeViewBase mOpenCvCameraView;

    private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
        @Override
        public void onManagerConnected(int status) {
            switch (status){
                case LoaderCallbackInterface.SUCCESS:{
                    Log.i(TAG, "onManagerConnected: OpenCV loaded correctly");
                    System.loadLibrary("detection_based_tracker");
                    try{
                        InputStream is = getResources().openRawResource(R.raw.lbpcasc
                        File cascadeDir = getDir("cascade", Context.MODE_PRIVATE);
                        mCascadeFile = new File(cascadeDir, "lbpcascade_frontalface.x
                        FileOutputStream os = new FileOutputStream(mCascadeFile);

                        byte[] buffer = new byte[4096];
                        int bytesRead;
                        while ((bytesRead = is.read(buffer))!=-1){
                            os.write(buffer, 0, bytesRead);
                        }
                        is.close();
                        os.close();

                        mJavaDetector = new CascadeClassifier(mCascadeFile.getAbsolut
                        if (mJavaDetector.empty()){
                            Log.e(TAG, "onManagerConnected: Failed to load cascade cl
                            mJavaDetector = null;
                        }else{
                            Log.i(TAG, "onManagerConnected: Loaded cascade classifier
```

```
                    }
                    mNativeDetector = new DetectionBasedTracker(mCascadeFile.getA
                    cascadeDir.delete();
                }catch (IOException e){
                    e.printStackTrace();
                    Log.e(TAG, "onManagerConnected: Failed to load cascade. Excep
                }

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

public FrActivity(){
    mDetectorName = new String[2];
    mDetectorName[JAVA_DETECTOR]   = "Java";
    mDetectorName[NATIVE_DETECTOR] = "Native (tracking)";
    Log.i(TAG, "FrActivity: Instantiated new" + this.getClass());
}

@Override
public void onCreate(Bundle savedInstanceState){
    Log.i(TAG, "onCreate: Called");
    super.onCreate(savedInstanceState);
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
    setContentView(R.layout.face_detect_surface_view);
    mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.fd_activity_surf
    mOpenCvCameraView.setVisibility(CameraBridgeViewBase.VISIBLE);
    mOpenCvCameraView.setCvCameraViewListener(this);
}

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

@Override
public void onResume(){
    super.onResume();
    if (!OpenCVLoader.initDebug()){
        Log.d(TAG, "onResume: Internal OpenCV lib not found. Using OpenCV manager
        OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0,this,mLoaderCall
    }else{
```

```
            Log.d(TAG, "onResume: OpenCV lib found inside package.");
            mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
        }
    }

    public void onDestroy(){
        super.onDestroy();
        mOpenCvCameraView.disableView();
    }

    public void onCameraViewStarted(int width, int height){
        mGray = new Mat();
        mRgba = new Mat();
    }

    public void onCameraViewStopped(){
        mGray.release();
        mRgba.release();
    }

    public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame){
        mRgba = inputFrame.rgba();
        mGray = inputFrame.gray();

        if (mAbsoluteFaceSize == 0){
            int height = mGray.rows();
            if (Math.round(height * mRelativeFaceSize)>0){
                mAbsoluteFaceSize = Math.round(height * mRelativeFaceSize);
            }
            mNativeDetector.setMinFaceSize(mAbsoluteFaceSize);
        }

        MatOfRect faces = new MatOfRect();

        if (mDetectorType == JAVA_DETECTOR){
            if (mJavaDetector != null){
                mJavaDetector.detectMultiScale(mGray,
                        faces,
                        1.1,
                        2,
                        2,
                        new Size(mAbsoluteFaceSize, mAbsoluteFaceSize),
                        new Size());
            }
        }else if (mDetectorType == NATIVE_DETECTOR){
            if (mNativeDetector != null){
                mNativeDetector.detect(mGray, faces);
            }
```

```
        }else{
            Log.e(TAG, "onCameraFrame: Detection method is not selected");
        }

        Rect[] facesArray = faces.toArray();
        for (Rect aFacesArray : facesArray)
            Imgproc.rectangle(mRgba, aFacesArray.tl(), aFacesArray.br(), FACE_RECT_CO

        return mRgba;
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu){
        Log.i(TAG, "onCreateOptionsMenu: called");
        mItemFace50 = menu.add("Face size 50%");
        mItemFace40 = menu.add("Face size 40%");
        mItemFace30 = menu.add("Face size 30%");
        mItemFace20 = menu.add("Face size 20%");
        mItemType   = menu.add(mDetectorName[mDetectorType]);
        return true;
    }

    private void setMinFaceSize(float faceSize){
        mRelativeFaceSize = faceSize;
        mAbsoluteFaceSize = 0;
    }

    private void setDetectorType(int type){
        if (mDetectorType != type){
            mDetectorType = type;
            if (type == NATIVE_DETECTOR){
                Log.i(TAG, "setDetectorType: Detection based tracker enabled");
                mNativeDetector.start();
            }else{
                Log.i(TAG, "setDetectorType: Cascade detector enabled");
                mNativeDetector.stop();
            }
        }
    }
}
```

B Python Experiments

B.1 Table of Python Experiments

Experiment	Purpose	Functiona?l	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
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,
    cv2.imshow('image', image)
```

```
    cv2.waitKey(0)
```