

Real Time Colour Perception Enhancement

Author: Zubair Masood

Supervisor: Dr Fergal Shevlin

M.A.I Electronic and Computer Engineering

Submitted to the University of Dublin, Trinity College, May, 2015

Declaration

I, Zubair Masood declare that the following dissertation, except where otherwise stated, is entirely my own work; that it has not previously been submitted as an exercise for a degree, either in Trinity College Dublin, or in any other University; and that the library may lend or copy it or any part thereof on request.

Signature:

Date:

Summary:

Colour blindness is the failure to distinguish colour differences under normal lighting conditions. It is a real issue which affects a significant portion of the population. It affects 1 in 12 men and 1 in 200 women or around 4.5 % of the entire world population. If the world population is estimated at 7 billion, around 315 million people suffer from some form of colour blindness. This is roughly 80 times the population of Ireland. Due to the significant number of people affected, a viable solution for real time colour perception enhancement is needed.

The proposed solution consists of a mobile application which performs colour recognition in real time using the phone's camera in conjunction with a hand held projector which displays the colour information. . The advantage of the projector is to not limit the users view to the phone's small screen size; however the application should still work without the projector.

A user driven design approach was taken to develop the user interface of the application. After each change the design was tested from the user's perspective to ensure that the change was beneficial. Initial colour recognition was done by calculating the average colour in a region of interest. Then the average value was normalised to take into account the lighting conditions. Finally the colour was labelled using one of the built in algorithms.

Through rigorous testing it was found that the developed application can recognise a host of different colours in different lighting conditions making the application practical. The application was developed on the Android OS which powers 79 % of the mobile devices worldwide. This insures that the application will be accessible to a large number of users.

This report discusses the design, development and evaluation of the proposed solution.

Acknowledgements:

I would like to thank my supervisor Dr. Fergal Shevlin for his invaluable guidance and support throughout the project.

Furthermore I would like to thank my family and friends, who provided instrumental feedback and support throughout the project.

TABLE OF CONTENTS

INTRODUCTION.....	6
APPLICATION MOTIVATION	6
PROPOSED SOLUTION.....	8
BACKGROUND	9
HUMAN VISION.....	9
RETINA	10
COLOUR PERCEPTION.....	11
CAUSES	12
DIAGNOSIS.....	14
LITERATURE REVIEW	16
COMPENSATION VISION SYSTEM, JAPAN.....	16
AUGMENTED REALITY SYSTEM, INDONESIA	18
DYNAMIC COLOR TRANSFORMATION, SOUTH KOREA.....	20
WEBSITE COLOR TRANSFORMATION, MALAYSIA	22
SPEECH RECOGNITION IMPLEMENTATION, INDONESIA.....	24
DESIGN.....	26
IMPORTANCE OF DESIGN.....	26
EARLY DRAFTS.....	27
ALTERNATIVE DESIGN	34
FINAL DESIGN.....	35
IMPLEMENTATION.....	36
WHY ANDROID.....	36
DEVELOPMENT TOOLS.....	37
APPLICATION OVERVIEW.....	39
UI CREATION	40
COLOUR DETECTION.....	42
CALIBRATION	43
COLOUR CLASSIFICATION.....	45
EVALUATION	46
LOGCAT	46
COLOUR SPACE TESTS.....	47
CHANGING LIGHTING CONDITIONS.....	49
REPLICATION TESTS.....	50
SIMPLE CLASSIFICATION TESTS.....	51
ADVANCED CLASSIFICATION TESTS	56
USER RESPONSE	58
DISCUSSION OF RESULTS.....	58
CONCLUSION.....	59
SIMILAR PRODUCTS.....	59
APPRAISAL	64
FUTURE WORK	65
LIST OF FIGURES.....	67
REFERENCES.....	69

Introduction

Application Motivation:

Colour blindness affects a significant portion of the population. It affects 1 in 12 men and 1 in 200 women or around 4.5 % of the entire world population¹. If the world population is estimated at 7 billion, around 315 million people suffer from some form of colour blindness. This is roughly 80 times the population of Ireland.

Normal colour vision uses all three types of light cones correctly and is known as trichromacy¹. People with normal colour vision are known as trichromats. People who have one of the cones missing are known as Dichromats: Protanopia (red cones absent), Deuteranopia (green cones absent) Tritanopia (blue cones absent). People who have no cones are known to have Rod Monochromacy. The table below shows the division of different types of colour blindness²:

Classification	Incidence (%)	
	Males	Females
Anomalous Trichromacy	6.3	0.37
Protanomaly (L-cone defect)	1.3	0.02
Deuteranomaly (M-cone defect)	5.0	0.35
Tritanomaly (S-cone defect)	0.0001	0.0001
Dichromacy	2.4	0.03
Protanopia (L-cone absent)	1.3	0.02
Deuteranopia (M-cone absent)	1.2	0.01
Tritanopia (S-cone absent)	0.001	0.03
Rod Monochromacy (no cones)	0.00001	0.00001

Figure 1: Classification of colour blindness.

People with colour blindness have difficulties in many aspects of life. Problems range from occupations where colour identification is an integral part (assembling electronics) to driving (traffic lights) ³. Some countries have even refused to grant driving licences to colour blind people (Romania). Colour is a vital tool for processing information. Therefore colour blind people also have difficulty while cooking (unable to see when the food is “done”), understanding graphs, playing sports etc.

Due to the significant number of people affected and the lack of feasible existing solutions I feel research to develop an accurate colour identification system needs to be undertaken. This system could help identify the parts when building a circuit; help differentiate colours to make reading a map easier etc. The system could be adapted to use with augmented reality systems such as the Google glass. The applications of this system are many therefore research in this field is vital.



Figure 2: Problems faced by the colour blind.

Proposed Solution:

The primary purpose of this project is to build a real time colour identification system which will enhance the users colour perception. The system will aim to improve the colour perception of colour blind people, aiding them in tasks in which colour identification is vital. The diagram below illustrates how the system will be used. The user will point their phone camera at the object of interest and a mounted pico projector will identify the colour in real time. The advantage of the projector is to not limit the users view to the phone's small screen size. The system needs to be robust and take into account changes in the environment and a feedback system needs to verify that the colour identifier is indeed correct.

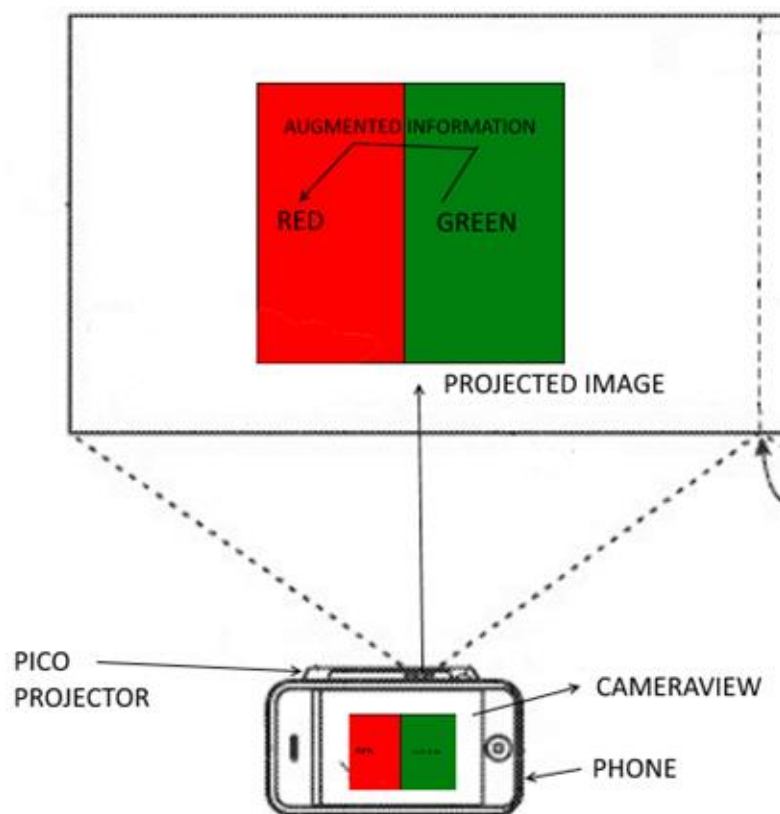


Figure 3: Proposed solution.

Background:

Human Vision:

The human vision system is complex; however a basic understanding is necessary to ensure that the final application solves the root cause of the problem.

The diagram below shows the anatomical structure of the eye. The clear front surface of the eye is called the cornea and its primary function is to focus the incoming light. Like the diaphragm of a camera the iris controls the amount of light that reaches the back of the eye by adjusting the size of the pupil⁴. To focus on objects of interest (like the autofocus of a camera) the eyes lens uses a process known as accommodation⁴. After the incoming light is focused by the cornea along with the lens and limited by the iris and the pupil it reaches the light sensitive inner part of the eye known as the retina. The retina processes the image and converts it into electronic signals. These electronic signals are sent via the optic nerve to the visual cortex which is responsible for interpreting visual information⁴.

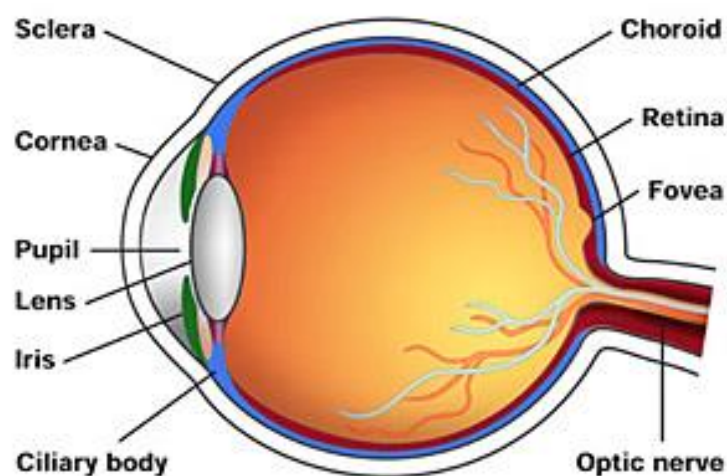


Figure 4: Anatomy of the human eye.

The Retina:

The retina covers about 65 percent of the interior surface of the eye and is located at the back. The retina contains two types of photoreceptors rods and cones⁵. There are 120 million rods and 7 million cones. Rods are more sensitive than cones however are not sensitive to colour. Rods are responsible for scopic (dark adapted) vision. The cones are concentrated in one region known as the fovea centrails⁵. The cones can be further divided based on the colours they are sensitive to. Red cones are the most predominant (64%), followed by the green cones (32%); however the numbers of blue cones are few (2%). Cones are responsible for all high resolution vision and as stated before the problem in development of these is the predominant cause of colour blindness⁵.

When light strikes a cone it interacts with a visual pigment which consists of a protein called opsin⁶. Three different kinds of opsins respond to short, medium and long wavelengths which result in the response curves below. When a person sees in colour at least two of these responses are triggered and the perceived colour is based on the relative level of excitation of the different cones⁶.

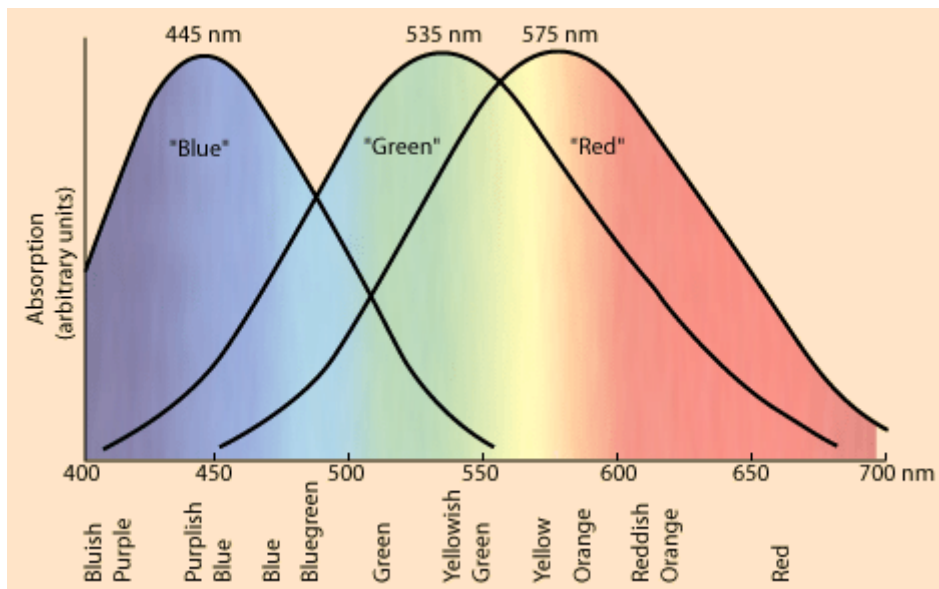


Figure 5: Response curves of cones.

Colour Perception:

The three primary properties of colour are hue, saturation and brightness. Hue is the dominant wavelength, saturation defines the intensity of a colour and brightness refers to the reflectivity of the colour. Even though each colour has a unique wavelength many combinations produce the same perception of colour⁷. This phenomenon can be described by the CIE chromaticity diagram.

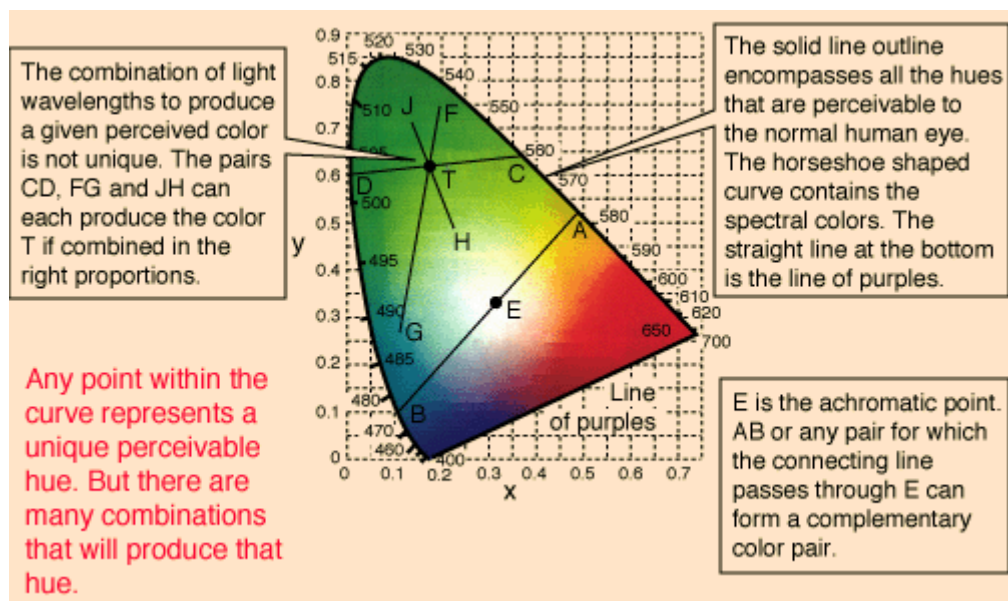


Figure 6: CIE chromaticity diagram.

Other factors also affect colour perception. Most people have a memory for grass colour and can produce this stimulus even if it is not present in a photograph. This phenomenon is known as memory colour⁸. Similarly even if significant changes occur in the illumination of an object, the overall colour perception of the object remains unchanged. This phenomenon is known as colour constancy⁹. Object recognition is generally driven by the spatial, temporal and light dark properties rather than purely by chromatic properties. Thus once the objects are recognized, the mechanisms of memory color and discounting the illuminant can fill in the appropriate colour. Unfortunately these mechanisms are not available (unless implemented) to any mobile application trying to process localised colour in an object.

Causes:

The most common cause of colour blindness is the fault in the development of one or more sets of cones due to genetics. However there are other non-hereditary causes¹⁰:

Kallman's syndrome: This is an inherited condition where the primary symptom is a failure to start puberty or to complete it. It is prevalent in both genders and has many additional symptoms such as an altered sense of smell. Color blindness is sometimes also a symptom of this condition.

Parkinson's disease: This is a degenerative disorder of the central nervous system caused by the death of dopamine generative cells. Early symptoms include shaking, rigidity, slowness of movement, behavioural problems etc. Because this is a neurological disorder, light sensitive nerve cells in the retina may be damaged leading to colour blindness.

Shaken Baby Syndrome: Any physical damage at a young age to the brain or the retina can cause permanent damage to an infant. One effect of such damage can be permanent color blindness.

UV damage: Over exposure to ultra-violet rays causes degeneration of the retina. This is one of the most common causes of colour blindness however only presents itself later in the patient's life.

Cataracts: A cataract is a clouding of the lens of the eye. The most common cause of cataracts is ageing. By age 80, more than half of all Americans either have had a cataract or have had cataract surgery. The cloudiness caused makes colours less bright making it harder to distinguish them.

Medications: Ample prescribed medications have a side effect which causes colour blindness. For example Tiagabine which is an antiepileptic drug which has been shown to reduce the user's colour vision.

There are also a number of non-inherited diseases such as Alzheimer's, Diabetes, Glaucoma, Leukaemia, Liver disease that cause colour blindness.

Hereditary Causes:

The most usual cause of color blindness is the inheritance of a genetically mutated gene on the X chromosome¹¹. Research has also shown that mutations that can lead to colour blindness originate from in excess of 19 different chromosomes.

Females have two X chromosomes where males only have one. Therefore in females, if one X chromosome is defective, the normal one will prevent the woman from becoming colour blind. This explains why 1 in 12 men but only 1 in 200 women suffer from any type of colour blindness.

Women who have a flawed gene and are carriers, their sons have a 50% chance of being colour blind, and their daughters have a 50% chance of becoming a carrier. Women who suffer from color blindness as a result of mutated X chromosomes will typically have a carrier mother, and a colour blind father, getting two mutated X chromosomes.

There are also inherited diseases¹⁰ such as Cone Dystrophy, Achromatopsia, Blue cone monochratism, Retinitis pigmentosa, Retinoblastoma etc. that can lead to colour blindness.

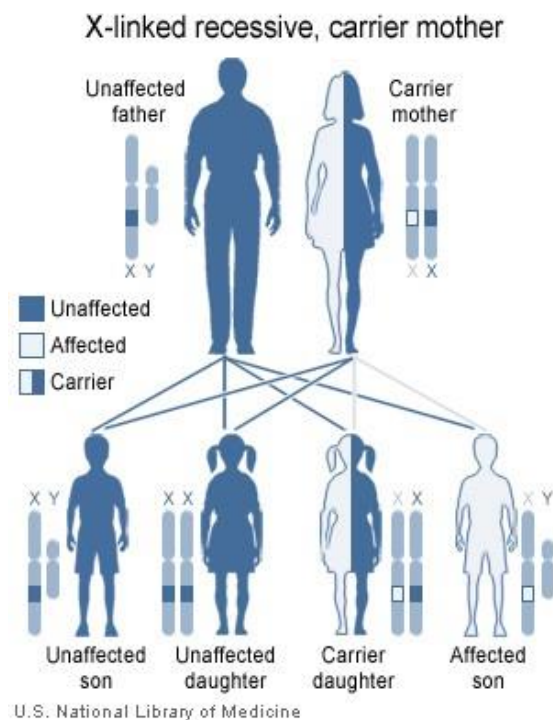


Figure 7: Genetic Tree showing possible outcomes.

Diagnosis:

Colour blindness can be hard to identify, predominantly in children with inherited colour vision deficiency as they may be uninformed that they have any problems with their colour vision. A child with a condition such as Deuteranopia may seemingly be able to accurately identify colours which they can't see (e.g. red) because they have been taught the colour of objects from an early age and will know for example that the sky is blue and cucumbers are green even if they cannot actually see those colours.

The most common test for colour blindness is the **Ishihara test**¹². This test is useful for red/green colour blindness but does not work for blue colour blindness. The test consists of 38 plates of circles which contains numbers imbedded with an irregular pattern. The number is of a different colour therefore those with normal vision should be able to identify them. However colour blind people cannot accurately distinguish colours therefore cannot identify the number. Special tests have been developed for young infants as they cannot correctly identify numbers. The plates can identify the type and the severity of colour blindness present.

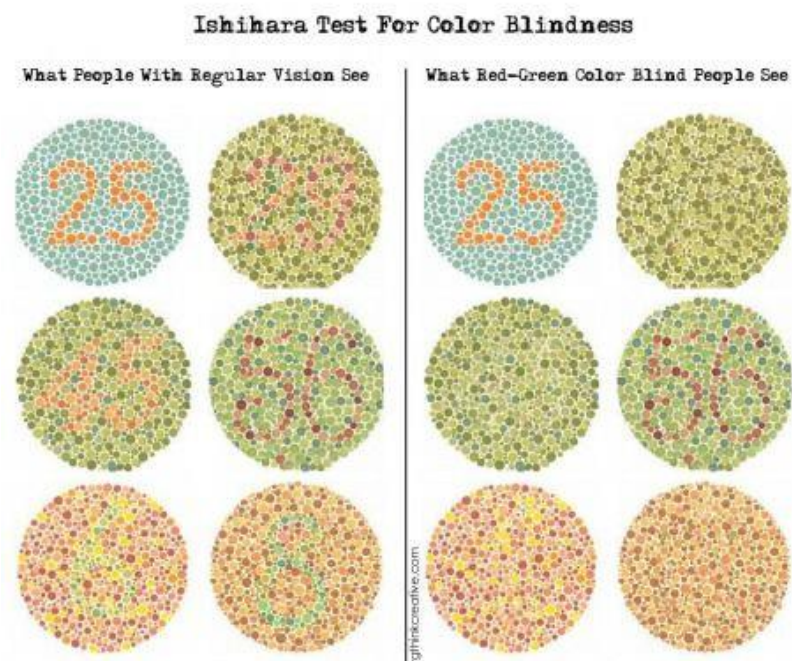
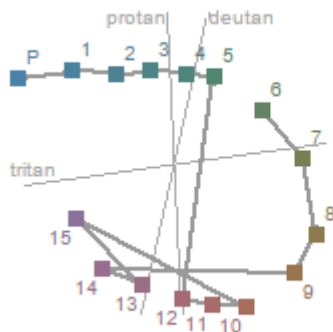


Figure 8: Ishihara Test.

Colour Arrangement Test:

This is an Alternative test. This test differs from the Ishihara test style as it utilizes vector mathematics to plot and formulate results, offering a formulated method of defining the user's condition¹³.

To take the test the user is asked to arrange the colours in an ordered list. After the user has completed this task a graph is generated outlining the user's performance. A detailed table outlining the metrics of the user's performance along with the severity of the user's condition is also displayed.



The thick line describes your order of the test plates. People with normal color vision order them in a circle (P, 1, 2, ..., 15). Crossings indicate some form of color blindness. Parallellism of crossings to a confusion line (protan, deutan, tritan) is a clue for the type of your color blindness.

ANGLE	MAJOR	MINOR	TES	S-INDEX	C-INDEX
56.7	18.6	14.8	23.8	1.26	2.01

Find an explanation of the resultset and sample values to compare in the description below.

According to this test result you have a **protan color vision defect**.



Figure 9: Arrangement Test.

Literature Review

Due to the increasing processing power of mobile devices, researchers across the globe have attempted to use real time processing to improve human accessibility. Below some of the existing research and solutions to enhance colour perception are discussed.

In their paper titled “A Color Compensation Vision System for Color-blind People” Tomoyuki Ohkubo and Kazuyuki Kobayashi¹⁴ propose an “eye glass conversion display system to enhance color difference”. The proposed system contains a small camera, image processing unit and a wearable display to form a self-contained system for colour perception enhancement.



Photo 2 Appearance of traffic signal to color-blind people

Figure 10: Proposed Solution. The image on the left shows the original image. The two preceding images show the result of colour transformation.

The camera captures the image in the RGB colour space; this is then converted to the HLS colour space. In this space a transformation is done which replaces the colours which are difficult to identify with colours which are easily distinguishable by the user. This transformed image is then displayed on the head mounted display in real time.

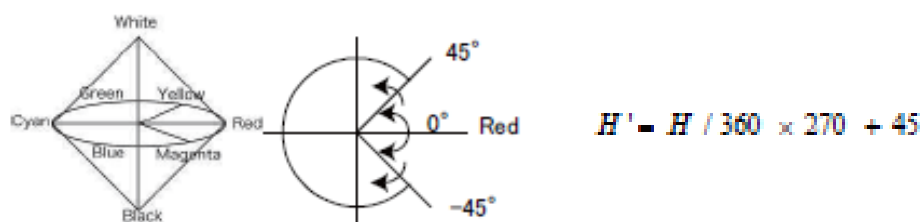


Figure 11: Transformation Algorithm.

To evaluate their solution 10 subjects were asked to test the product on a set of traffic lights. To keep the results fair the same subjects who initially had normal vision were asked to wear glasses which simulate colour blindness. After the subjects had used the product they were asked to describe their experience by choosing from a set of adjectives provided. This test was done again but this time the subjects had no aid to help them perceive colour better.

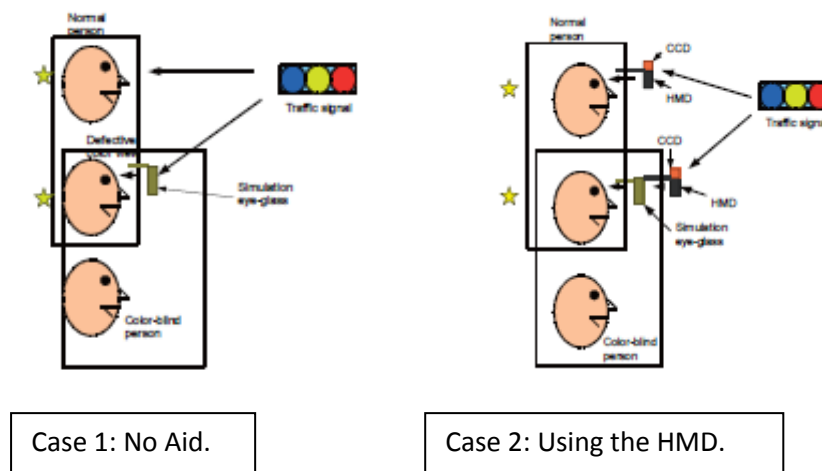


Figure 12: Testing Methodology.

The results in the case when the users were wearing the head mounted display were much better. Their implementation is commendable, however is not sufficient. Their implementation makes the colour visible, however does not identify it. For example in case of a traffic light it will show the red traffic light as blue instead of identifying the colour itself. This makes this application unsuitable in cases where correct colour recognition is important (such as assembling electronic circuits). Furthermore their testing was done only on a set of traffic lights using a small data set (male users aged 20+), therefore it's impossible to gauge the performance of the application in different circumstances.

Nevertheless this study was done 8 years ago. Today mobile devices are both more powerful and accessible. This means that if their application can be ported to a mobile application, using and evaluating it becomes more feasible.

Augmented Reality System: University of Indonesia

In their paper titled “Color Transformation for Color Blind Compensation on Augmented Reality System” Bayu Sri Ananto, Riri Fitiri Sari and Ruki Harwahu¹⁵ propose their own system similar to the one proposed in this thesis. Their system consists of a mobile application or a dedicated embedded device working in conjunction with a head mounted display for better colour perception.

Like many others, their implementation uses Daltonization. Daltonization is a process in which the ranges of colours that are harder to distinguish are replaced by colours which are easier to distinguish.

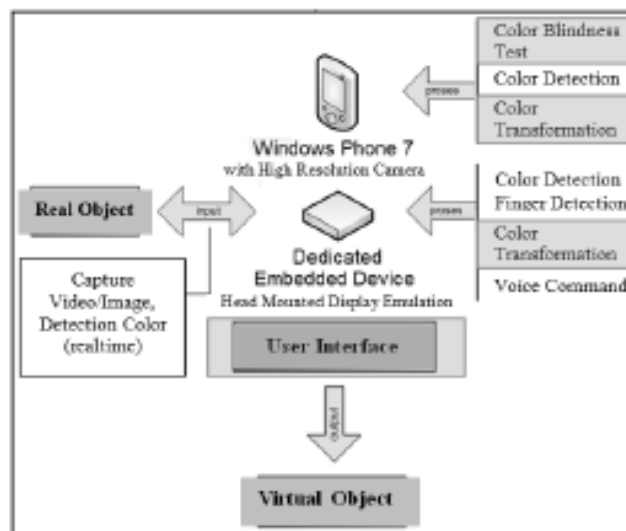


Figure 13: Proposed Solution.

The design of the mobile application’s user interface is impressive. Their interface is clean, intuitive practical and based on the barrier free principle. The barrier free principle is a concept that aims to design the software according to the operational habits of the user in order to reduce fatigue and make it easier for the user to use the application. This dedication to the user interface was demonstrated when the user responses to the interface were examined. On average the users gave the interface a score of 4.4 out of a maximum of 5.

As stated before the Ishihara test is one of the most popular tests for diagnosing colour blindness. It consists of numbers embedded in a noisy pattern. The number is of a different colour to the pattern, therefore people with normal vision can identify the number. To test their application they asked 10 users to undertake the Ishihara test with and without the application. 2 out of these users suffered from colour blindness. After using the application the users were able to identify the numbers more clearly.

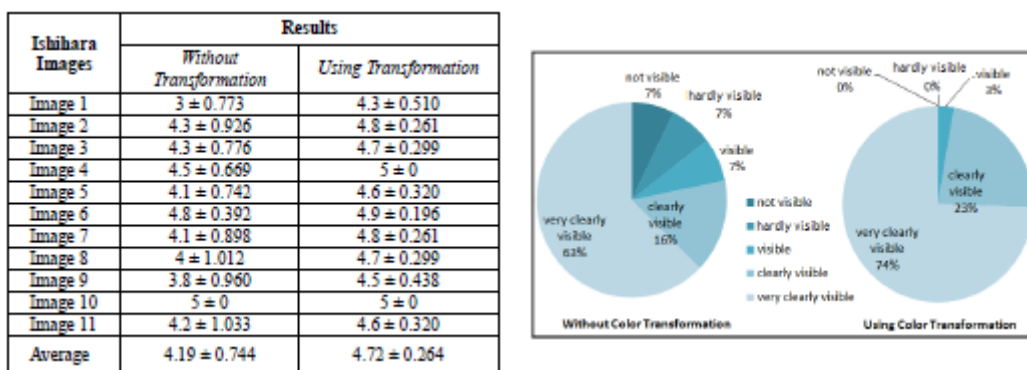


Figure 14: Results.

Their implementation is admirable, however is not adequate. Their implementation makes the colour visible, however does not identify it. For example in case of a traffic light it will show the red traffic light as blue instead of identifying the colour itself. This makes this application unsuitable in cases where correct colour recognition is important (such as assembling electronic circuits).

Their testing methodology is also slightly flawed. The Ishihara test is used to diagnose colour blindness and an application that passes this test does not prove that it's viable in any practical use case. By their own admission the clarity of the tests when using the mobile application was only improved slightly. Furthermore only two of the users had colour blindness therefore other effects such as the placebo effect may have altered the results.

Nevertheless the user centric approach of the application is commendable and the fact that it works on a mobile device makes it commendable.

Dynamic Colour Transformation, University of South Korea:

Many colour transformation systems use traditional Daltonization to enhance colour perception. Conventional Daltonization algorithms divide the pixels of the image into two sets of pixels: PU (protanope unperceptible) and PP (protanope perceptible). For the PU pixels a transformation is applied until the modified PU colours are not confused with the PP colours. This approach is complex and comes at a high computational cost. In their paper titled “Color Modification for Color-blind Viewers Using the Dynamic Color Transformation Hyun-Ji Kim, Jae Yun Jeong and colleagues¹⁶ propose a much more efficient clustering Daltonization algorithm which reduces the processing time without impacting accuracy.

The proposed process begins by separating the pixels into two categories, Icorrect containing PP colours and Iincorrect containing PU colours. To do this firstly the RGB image is converted to an LMS image. Linear transformation is done on this matrix to reduce the colour to the protanope domain using the formula:

$$\begin{bmatrix} L_p \\ M_p \\ S_p \end{bmatrix} = \begin{bmatrix} 0 & 2.02344 & -2.53581 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix}.$$

The resultant matrix is converted back to the RGB to obtain the protanope image. From this matrix the error image can be obtained using the formula:

$$R_E = |R - R_p|, \quad G_E = |G - G_p|, \quad \text{and} \quad B_E = |B - B_p|.$$

Using a low threshold image binarization of the error image is done. Moreover, an additional red pixel mask is considered to generate a final binary image. This mask contains 1s only when R is less than G and B. The purpose of this mask is to indicate whether Daltonization is required or not.

The final part of the algorithm involves colour checking and Daltonization. Daltonization is done by the application of the error modification matrix M onto Iincorrect. The original colours are converted into new colours that protanopes can discriminate. The protanope simulation (part1) is done repeatedly to see if further Daltonization is required.

COMPARISON OF THE PROCESS TIME			
Algorithm	Conventional algorithm [1]	Conventional algorithm [2]	Proposed algorithm
Time (ms)	16814625	1375	1234

Image size: 400x300

Figure 15: Performance Analysis 1.

We can see from figure 15 that the proposed solution is much quicker than conventional Daltonization. It is only slightly quicker than the conventional Daltonization algorithm which uses clustering however the proposed algorithm is more accurate as shown below in figure 16.



Examples in digitized paintings. First row: conventional daltonization algorithm [1], Second row: conventional daltonization algorithm [2], Third row: proposed algorithm, First column: Image to daltonization, Second column: Protanope perception after running algorithm

Figure 16: Performance Analysis 2.

In conclusion the proposed solution is impressive as it significantly cuts down computational time without affecting performance. This property is vital in real time applications.

Website Image Colour Transformation, Malaysia:

Distinguishing content on the web is difficult for colour blind people. Some websites such as Facebook design their websites to make sure that it is accessible, however many smaller websites do not. In the paper titled “Website Image Colour Transformation for the Colour Blind” Siew-Li Ching and Maziami Sabudin¹⁷ propose a retrieval and transformation system to tackle this problem.

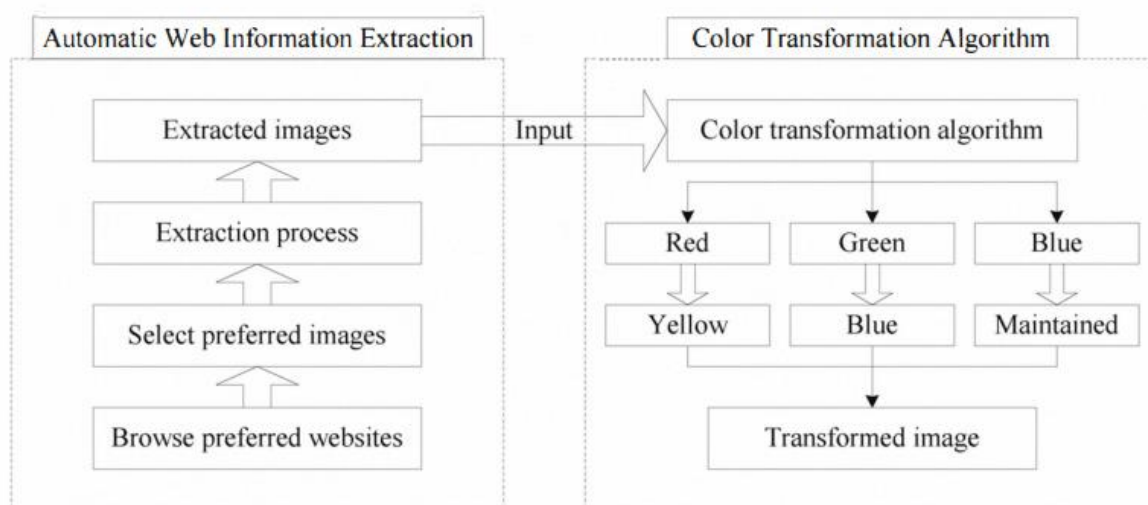


Figure 17: Proposed Framework.

The first part of the process is image retrieval. The user uses the web content extractor to select the images of interest. The images are then saved for processing. The purpose of the transformation algorithm is to transform the red and green colours to yellow and blue respectively. The blue colours are maintained while the red colours are transformed in the RGB colour space and the green colours are transformed in the HSV colour space.

To turn red into yellow, pixels which have R as the greatest component are chosen, then the G value is made equal to the R value. Equal amounts of red and green produce a yellow result. To turn green into blue, pixels with the greatest G value in the RGB colour space are chosen. This is done in the HSV colour space to ensure accuracy of conversion. Firstly the ratio of the original green hue value and the green hue range is used to determine the position of the green colour in the hue range. This is used to calculate the relative hue value then

finally 180 is added to obtain the position of the transformed colour in the hue domain with respect to the 0 degrees vertical line. Finally a HSV to RGB conversion is done.

To evaluate the performance of the algorithm a questionnaire was given to colour blind people and they were asked to recognize the original colours in a transformed image. They were given 4 transformed images. Two images were transformed using the proposed algorithm while 2 were transformed using the traditional “self-organising” method. 30 out of 35 participants correctly recognised the colours when using the proposed algorithm. 5 participants were unsure. However when the traditional algorithm was used only 2/35 participants recognised the colours appropriately. 28 were incorrect and 5 were unsure.

Question	Incorrect	Correct	Not Sure
1(a)	28	2	5
1(b)	0	30	5
1(c)	28	2	5
1(d)	0	30	5

Figure 18: Results. In questions “a” and “c” the traditional algorithm was used where as in questions “b” and “d” the proposed algorithm was used.

The proposed algorithm is excellent judging purely from the results. Furthermore it is less complex than traditional self-organising algorithms therefore much more efficient. The evaluation methodology is excellent as it is not subjective and double blind. Unfortunately this solution only works for those who suffer from red-green colour blindness. Furthermore I feel that the application would be better as an extension on a browser instead as a standalone application. Also the user needs to know how the image is being transformed to determine the original colours. In conclusion the proposed application is robust and practical however severely limited.

Speech Recognition for Embedded Systems, Indonesia:

Increasing user accessibility should be a primary goal when designing any application especially if the intended use is long term. If the application is difficult to use, it will fail to acquire a large user base regardless of how impressive its capabilities are. In their paper “Implementing Speech Feature For Embedded System to Support Color Blind People” Riri Fitri Sari and Ruku Harwahyu¹⁸ propose a speech synthesis and recognition system which can be used to enhance the user experience while using an embedded system for colour recognition.



Figure 19: Proposed System. The user can control the application through voice commands or through pointing their finger at the object of interest.

The system uses two speech recognition algorithms. The first (system “A”) uses the Microsoft Speech Application Programming Interface or SAPI. This interface is provided by Microsoft and provides functions for speech synthesis and recognition. A grammar list or a list of 21 colours was used as the dictionary to limit misclassification. This system is robust however only works with windows applications. The second algorithm (system “B”) uses dynamic time warping and phoneme concatenation for speech recognition and Google’s text to speech function provided with Google translate for speech synthesis.

To test the performance of the speech synthesis 10 Indonesian participants were used. First they listened to a native speaker pronouncing all 21 colours to familiarise themselves with the dictionary. Then they were asked to recognise the words synthesised randomly by systems A and B. The overall performance for system A was 88.33 % and system B was 62.9 %.

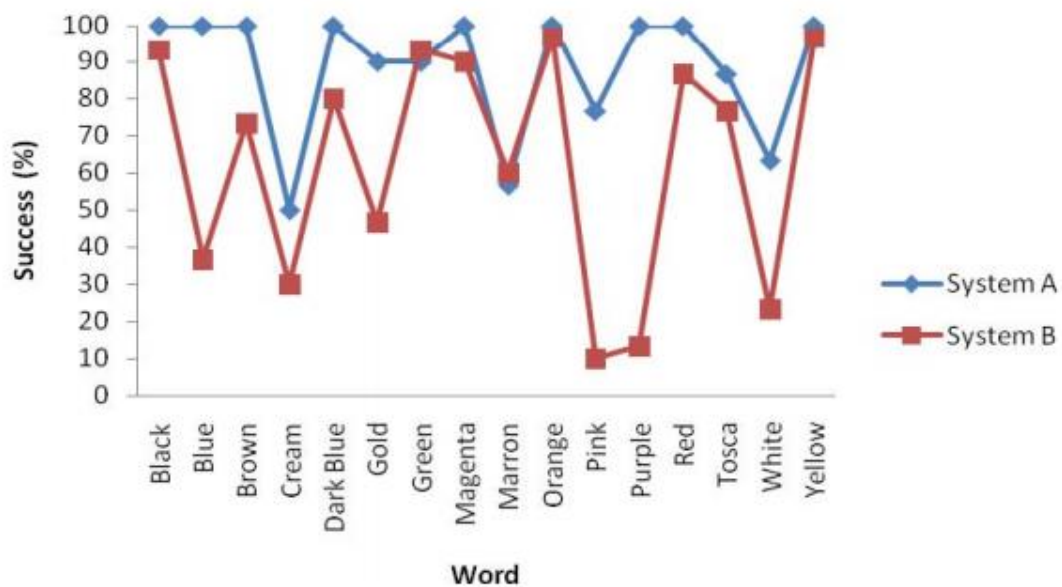


Figure 20: Synthesis Evaluation.

To test the performance of the recognition system, the same respondents were asked to pronounce the name of the colours. 76% of the words when no noise was present were recognised; however even with the presence of noise 75% of the words were recognised. At the end of this test the participants were asked about their opinion of the statement “Commanding using voice is more fun”; 70 % of the participants chose strongly agree. These days there are many speech syntheses and recognition libraries available across all platforms. I believe speech recognition and synthesis greatly increases the accessibility of this application and has the ability to increase the accessibility of many more applications.

Due to the large numbers of people affected, considerable research is being done to enhance the perception of colour blind people. Unfortunately due to the variety of different colour deficiencies no “perfect” solution has been found yet and research needs to continue.

Design:

“Design is the fundamental soul of a man-made creation that ends up expressing itself in successive outer layers of the project, or service.”

-Steve Jobs

Importance of design:

Good software design must combine functionality, informativeness, and simplicity. John D. Gould and Clayton Lewis¹⁹ outline three key design principles in their paper designing for usability: Focus on user and tasks, empirical measurement and Iterative design. These principles were used as a foundation when designing the user interface of the application.

The first principle states that design should be user driven. Each design decision taken was to ensure that this principle was upheld. Each design modification was done to increase user accessibility. After each design modification the application was tested from the user’s perspective, and if the end result did not produce a more intuitive UI (User Interface), the modification was discarded.

The second principle states that early in the development process, intended users should use simulations and prototypes to carry out real work. Even before the implementation of the detection system was complete, a prototype UI was used to analyse the user experience with the application.

The third principle states that problems found through user testing must be fixed. Every time a problem was noticed it was immediately recorded and addressed before continuing to ensure a fluid UI.

The following pages discuss the design process from the first draft to the final implementation. The rationale for each change is also outlined. The difference between the first draft and the final implementation is a testament to the importance of following the guidelines aforementioned.

Early Drafts:

Draft 1:

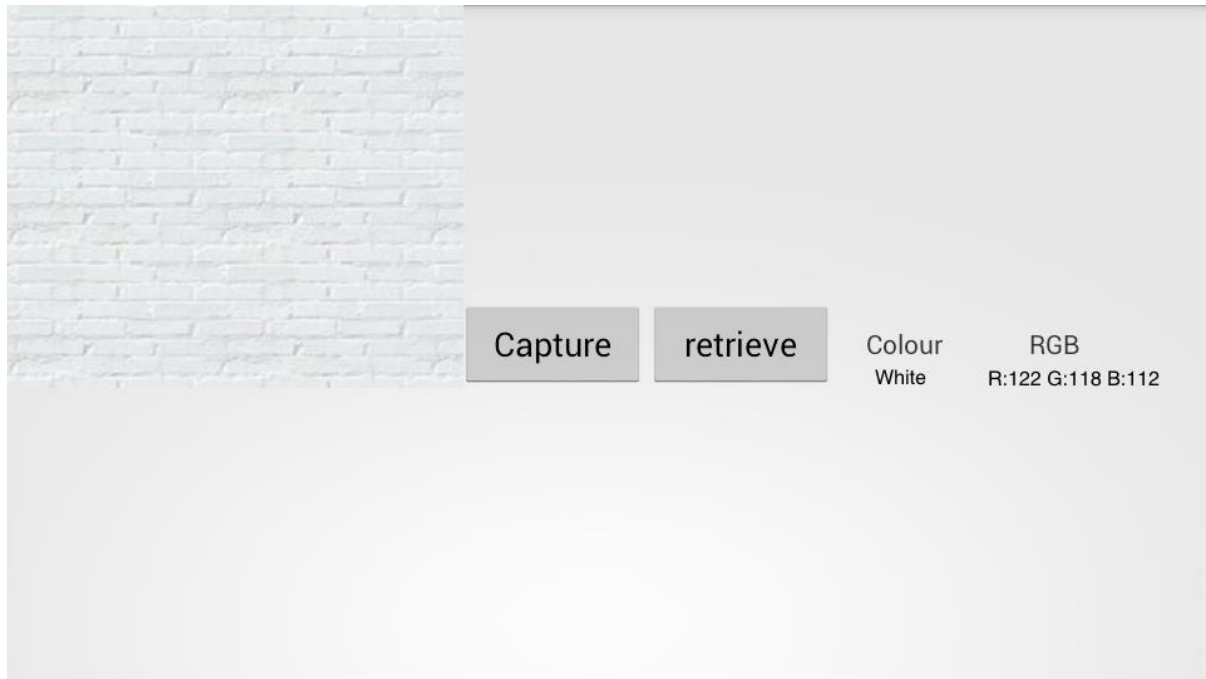


Figure 21: Draft 1.

This was the first implemented draft. As soon as the application started the above screen was shown. The rectangle starting at the top of left is the camera view. The user placed the object of interest in the camera view and pressed the capture button. Once the capture button was pressed the camera view was replaced by the captured image and the image was saved on the phones storage. When the user pressed the retrieve button the average colour of the photo was calculated and displayed. The RGB value of the retrieved colour was also displayed.

Even though the design was intuitive it was flawed. As can be seen above, a lot of screen space was wasted. The colour tag was also hard to read. The user needed to press two buttons to retrieve one colour which was found cumbersome. Due to these reasons this design was dropped.

Draft 2:

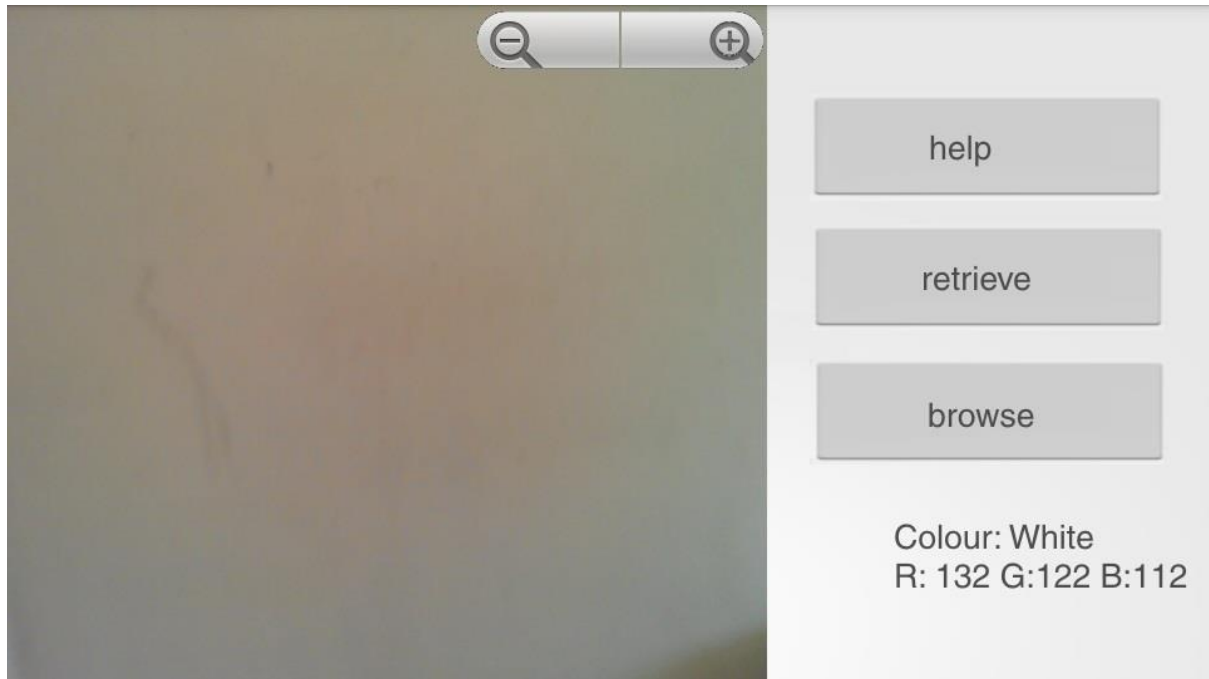


Figure 22: Draft 2.

This was the second implemented draft. The reason the camera view was so small in the first draft was to allow the user to focus on smaller objects. Zoom controls were introduced in this draft. An additional help button was introduced, which when pressed displayed a guide on how to use the application.

The font was also increased to make reading the colour much easier. The capture and retrieve buttons were also merged into one button to save the user time. A browse button was added. This allowed the user to retrieve the average colour from any photo. This design was an improvement but still had room for improvement.

Draft 3:

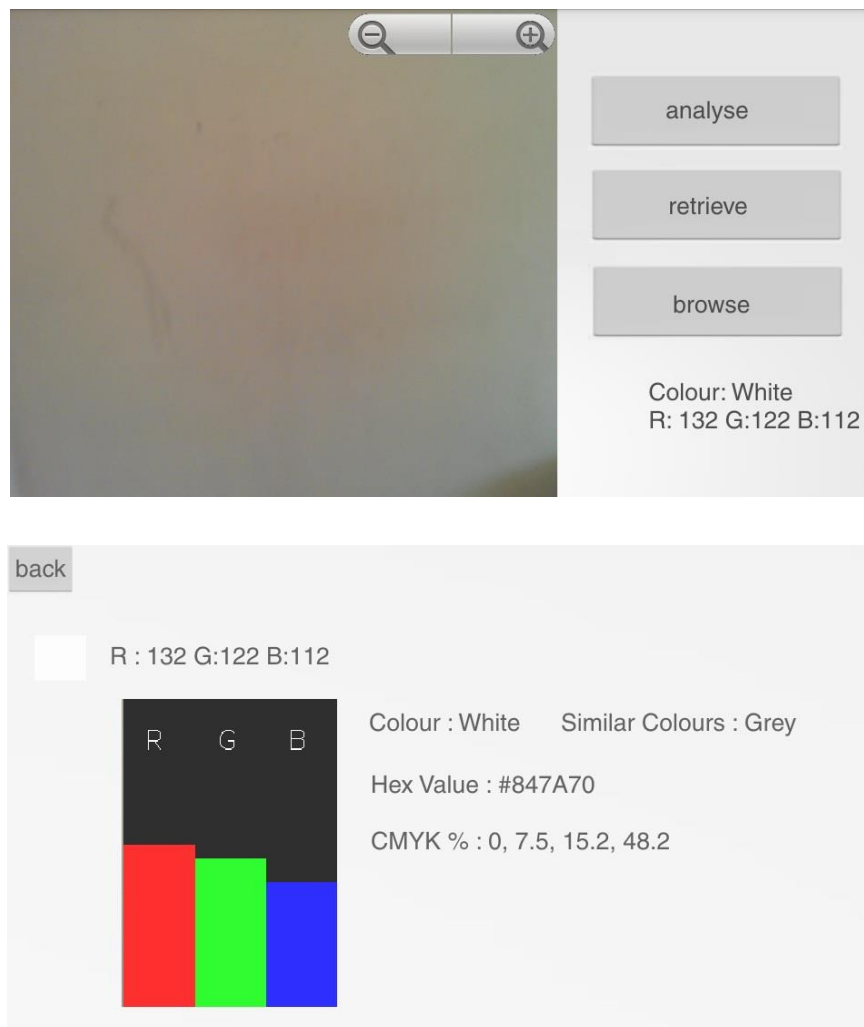


Figure 23: Draft 3.

It was found through testing that the user would only use the help button once or twice. Therefore the help button was moved into the menu list and was replaced by the analyse button. This allowed the user to retrieve more information about the colour such as composition, similar colours etc. Converting the retrieved colour into other colour spaces was trivial, yet gave the application a professional feel. Unfortunately due to issues with the projector (discussed later) this approach was abandoned, a new camera implementation was used and the application was redesigned.

Draft 4:

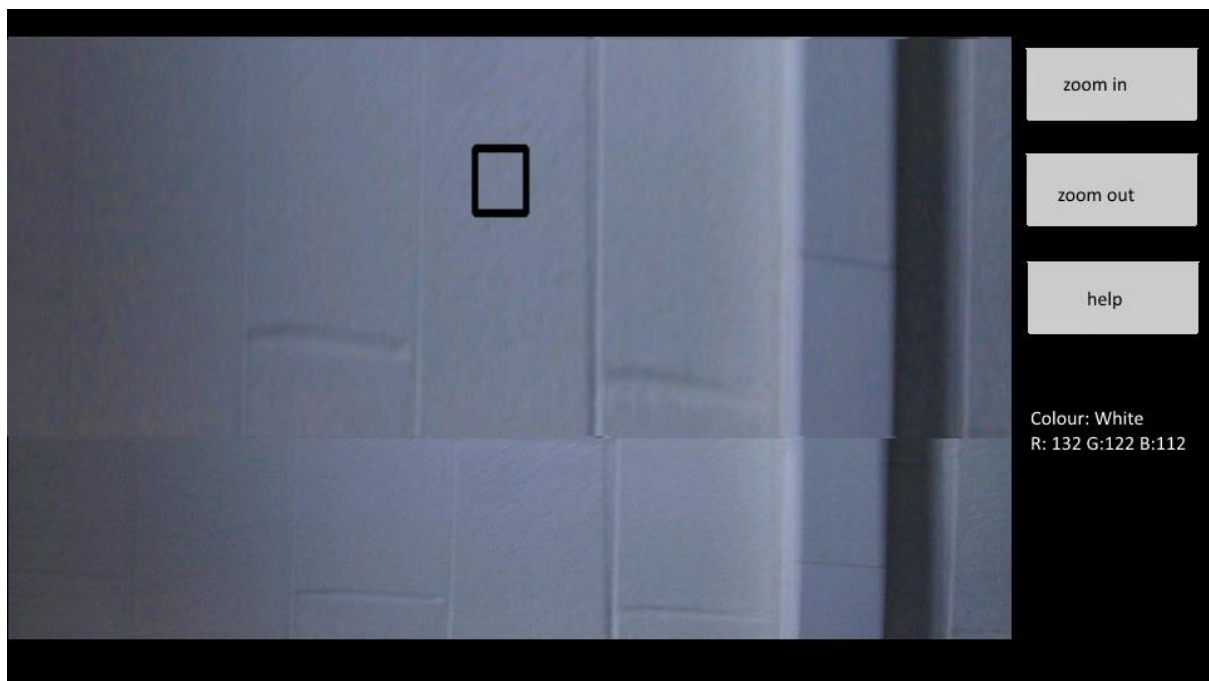


Figure 24: Draft 4.

The previous designs were created on the assumption that when the projector is connected the display would be mirrored. Unfortunately this was not the case. Android only shows the camera preview on the projected screen (on the assumption that if media is being shown on a secondary display, controls should only be displayed on the primary screen as not to ruin the viewing experience).

Now the openCV implementation of the camera was being used. OpenCV is designed for real time processing so the implementation is designed for capturing video frames and camera features such as zoom are not implemented by default. To combat this manual zoom buttons were included. The top $\frac{3}{4}$ of the screen is the bottom $\frac{1}{4}$ zoomed. When the user touched the screen the average colour at inside the onscreen cross hair (black rectangle) was calculated and displayed. This made the user interface more intuitive as the user did not have to press any button and the update was quick. As this was the first openCV draft there were a few issues that needed to be addressed.

Draft 5:

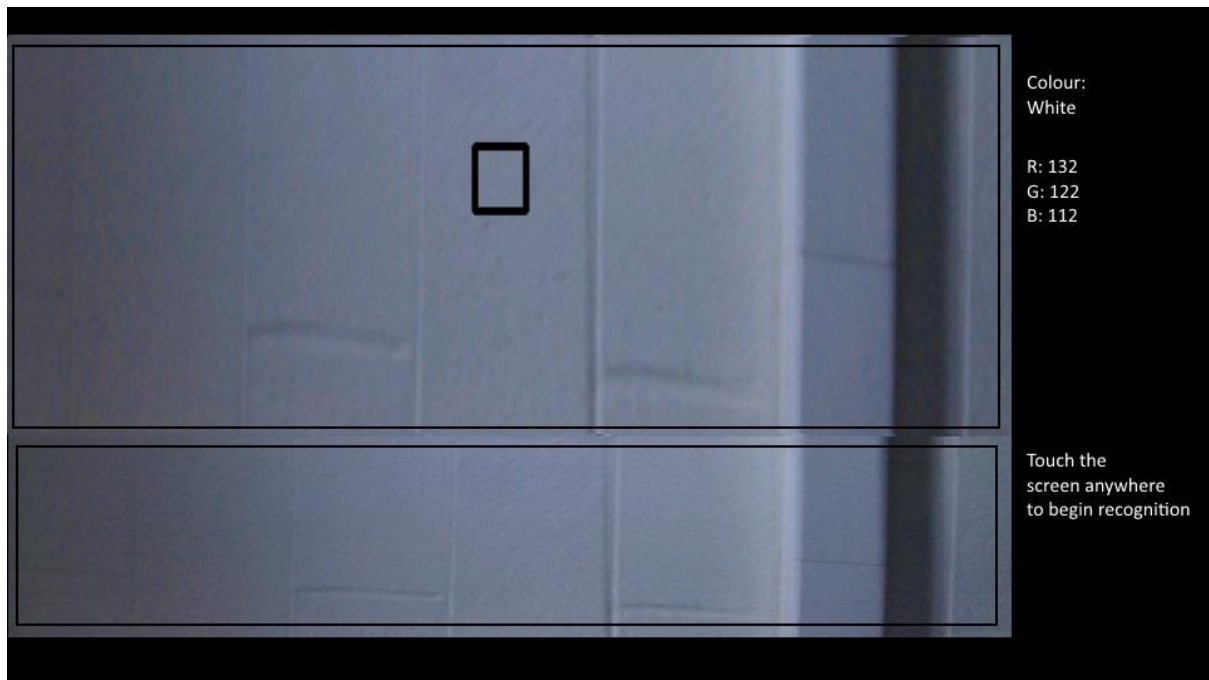


Figure 25: Draft 5.

After using the application it was found that the zooming buttons were redundant due to the size of the cross hair marker. One major issue with previous draft was that there was no separation between the zoomed and the un-zoomed parts of the frame giving the illusion of a broken video feed. To solve this issue rectangles highlighting the outlines of these two views drawn. The size of the colour label was increased and usage instructions were also displayed at the bottom right corner of the screen. The UI without any buttons looked much cleaner and was quicker (no need for separate on click listeners).

Draft 6:

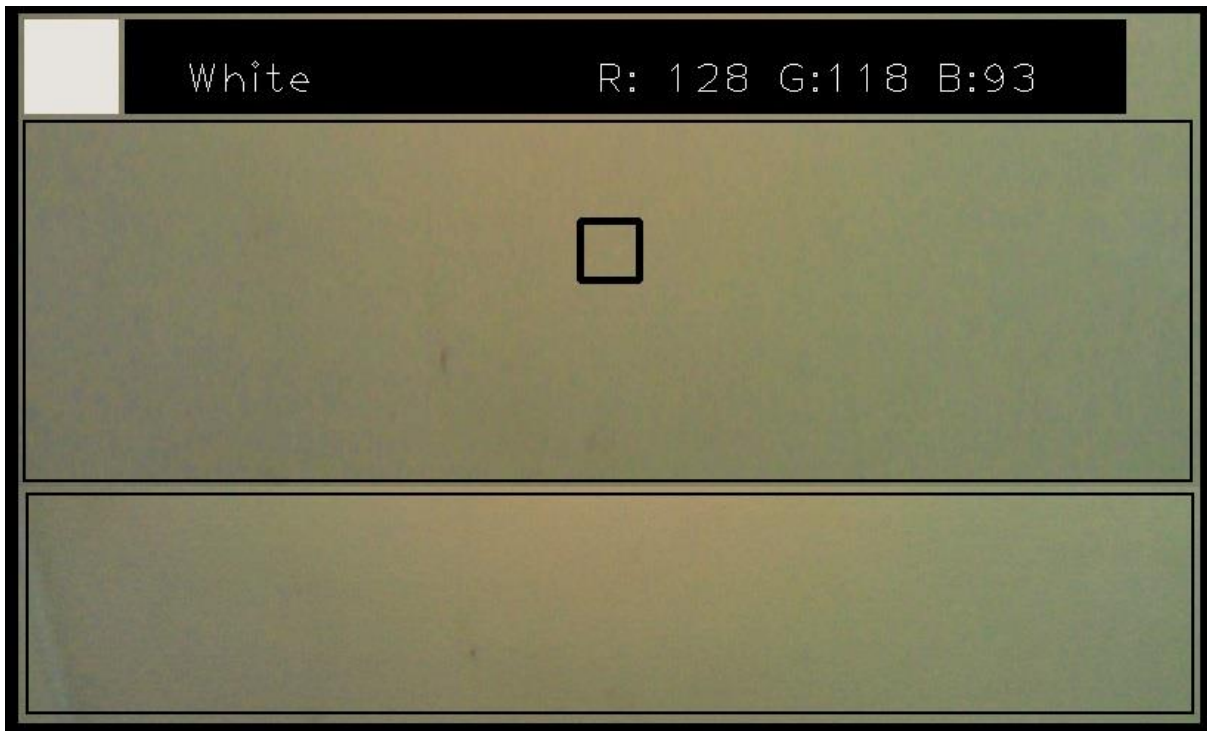


Figure 26: Draft 6.

Camera view space at the side was being underutilised, therefore the whole width of the screen was filled with the camera view. Another problem was that there was a noticeable delay between the user touching the screen and the text views being updated. To solve this problem the text was drawn on every new frame. Even though this was more computationally intensive it felt quicker. A palette of the average colour was also drawn. The size of the colour label was further increased.

Draft 7:

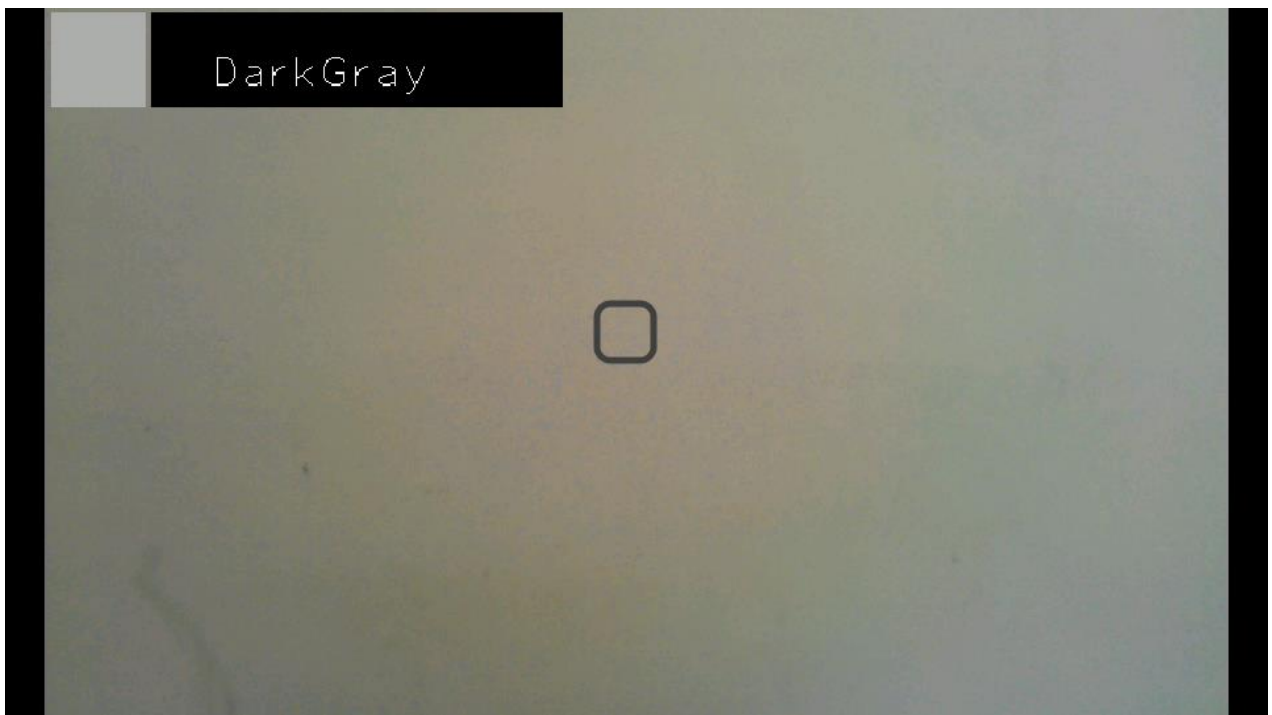


Figure 27: Draft 7.

Many users felt that the zoomed window from the previous designs was confusing. So a design decision was chosen to take it out. Some users also felt that the RGB values were not useful. So it was decided to take them out. Instead of zooming parts of the image the whole image is slightly zoomed. This also had the bonus effect of improving the stability of the camera preview. The size of the crosshair marker (rounded rectangle) was also changed to account for smaller objects. This was the most minimalistic design by far.

Alternative Design:

The final design is designed to work on a secondary display such as a mounted projector. It is minimal therefore the projected image should not distort the end user's field of view. When a projector is connected the application will mirror the display from the phone screen onto the projected area. Newer Android API's allow a custom presentation on any secondary display. This feature is new and only works on modern devices only. To make the application accessible it was designed to work on most devices, never the less an alternate design was created for newer phones (however ultimately not implemented).



Figure 28: Concept 1.

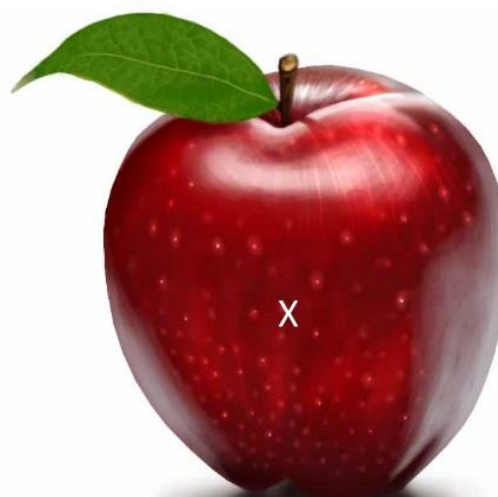


Figure 29: Concept 2.

RED
R: 200
G: 70
B: 110

The designs are shown in concepts 1 and 2 respectively. In concept 1 the colour is projected directly on the object of interest. Even though this seems intuitive, for it to be accurate a calibration procedure needs to ensure that the projector and the camera are in sync. In concept 2 an x is projected on to the object of interest, the mobile application uses this to locate the image and then colour information is displayed on a text box with a contrasting background from the text. Adding new features also increases the possibility of error. Therefore this design was not implemented as it was not tested enough.

Final Design:

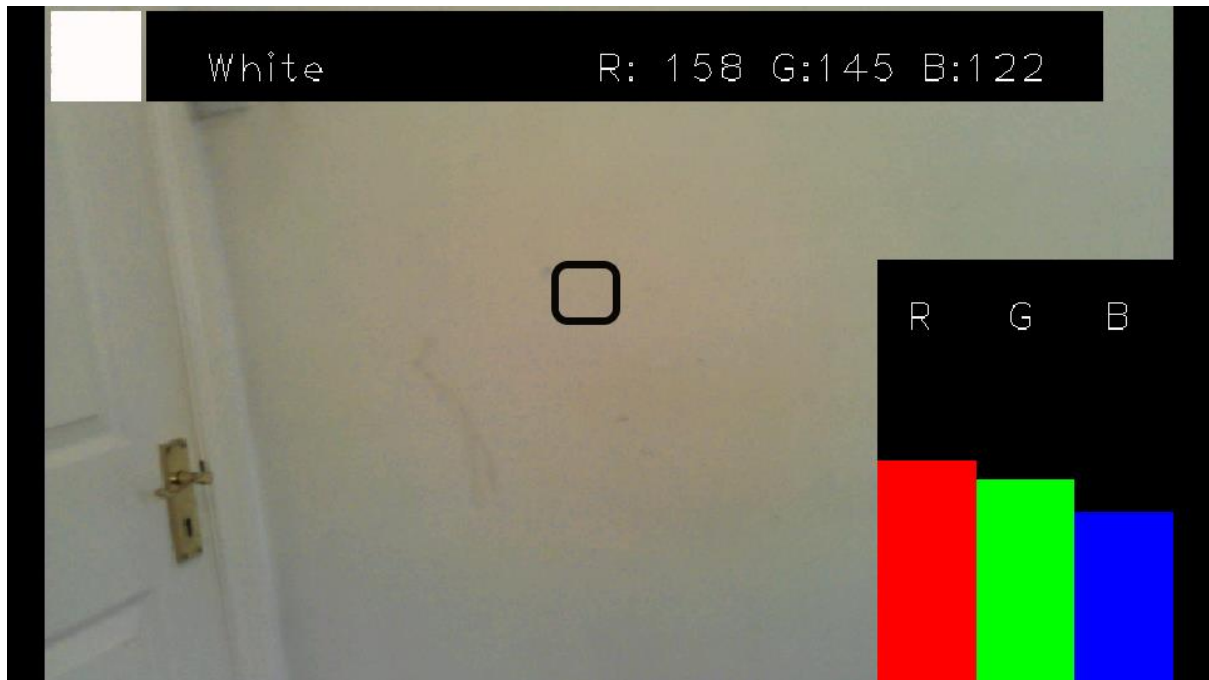


Figure 30: Final Design.

The picture above is an actual screenshot from the final application. Even though some users complained about RGB values, many competing applications had this feature therefore it was reintroduced. The original rationale for removing these values was that to an average person these values do not provide much information. To solve this problem a graph was also included. This graph shows the proportion of each of the primary colours present in the region of interest.

The final design is vastly superior to the first draft. In one touch the user gets as much information as needed. The interface is faster, lighter, cleaner and more importantly much more practical than the first one. Each design was an improvement from the previous one and the gradual improvement can be clearly seen.

Implementation:

Why Android?

In October 2003 Andy Rubin along with 3 other technology executives established Android Inc. to develop in Rubin's words "smarter mobile devices". On August 17 2005, Google acquired Android Inc²⁰.

Today Android is the world's most popular mobile operating system powering 79 % of mobile devices worldwide²¹. Androids popularity transcends mobile phones. It has been used in televisions, game consoles, car dashboards etc.

Android is based on the Linux kernel, is open source and released under the Apache license²². Because of its open source nature and excellent development tools by Google, Android has a large collection of applications which extend the capabilities of Android devices. Due to the large market share, App development is a lucrative industry. As of May 2013 1 million apps have been downloaded 48 billion times²³.

Despite at least 11000 distinct android devices²⁴ most of the market share belongs to approximately 20 high end devices manufactured by Samsung, HTC, and ASUS. These devices have extraordinary processing power, giving developers opportunities to develop power intensive apps.

Due to its market dominance and open source nature Android has a large development community and a host of development tutorials, tools etc. Many third party libraries are also available which makes development easier.

Due to the large market size, excellent development tools and my previous development experience, Android was chosen as the development platform for this application.

Development Tools:

HTC Sensation XE was the primary testing device for this application. There are many features which made it the ideal candidate. The phone has a high quality 4.3 inch screen (540 * 960 pixels). The phone contains a 1.5 gigahertz processor with 768 Mb ram²⁵. The phone also has built in development tools which provide additional debugging information such as memory usage, touch locations etc. The phone has a high resolution 8 megapixels camera which makes it ideal for image processing.



Figure 31: Primary device.

Samsung Galaxy Note 8.0 was the secondary testing device. This was a rooted device and was primarily used to record the application for testing and demonstration purposes. There are many features of this device which made it ideal for testing. It has a large 8.0 inches 800 *1280 pixels screen so the UI could be clearly seen. It also has a high resolution 5 MP camera making it suitable for image processing²⁶. The added advantage of using two test devices was to ensure that the results were not entirely device dependent.



Figure 32: Secondary device.

Microvision show wx plus was used as the pico projector. This is a lightweight (100g) projector with a small form factor and excellent output resolution. It has an autofocus mode which means that the image is always in focus, even on curved surfaces making it ideal for this application. More importantly it works on any android device which supports hdmi output. It also has an excellent battery life that lasts in excess of 4 hours, and due to the small form figure can rest easily on the phones surface²⁷.



Figure 33: Pico Projector

Tegra Android Development Pack (TADP) 2.0:

This suite provided by NVIDIA²⁸ contained most of the development tools needed for this project. Even though this suite is primarily designed for Tegra devices it worked perfectly on both devices. It contained the latest tools Android development tools along with OpenCV 2.4.5 and a variety of NVIDIA's tools. All the libraries were pre linked and all the paths were correctly configured. This unproblematic installation was intuitive compared to the steps required to configure each library and path independently.

Application Overview:

The application principally consists of one activity. Activities are essentially blankets which focus on numerous features of an application. Because there is essentially only one view, one main activity is enough to encapsulate the whole application.

Android applications incline to have 4 states which describe the state of affairs of the application. They are onCreate, onPause, onResume and onDestroy²⁹. These states create the basic guidelines outlining how the application works in each state.

OnCreate is called when the application starts. It is mainly used to initialize the user interface. In this app the onCreate function just starts the camera view and initializes any toasts (notifications) which may be needed later. It also requests the android OS to provide this application with the whole screen (no android status bar).

OnPause describes what happens when the application is paused (for example the user receives a call). This is typically used to commit unsaved changes to persistent data, stop animations and other things that may be consuming CPU, etc. In this application the camera view is disabled.

OnResume is called when the activity starts interacting with the user after onPause. In this application onResume asynchronously reloads the openCV library which resumes the application.

OnDestroy is the final call received before the activity is destroyed. In this application the camera view is disabled.

The above functions are staple functions of most applications therefore nothing time consuming or complex is done in them. The two main functions are onTouch and onCameraFrame. The colour detection is done onTouch (when the user touches the screen) and the UI is updated on every frame (onCameraFrame). The next few pages outline how these functions were implemented.

UI Creation:

The rationale behind the UI is described in the design section. As soon as the application is started the application requests a window feature called no title and then adds a flag called keep screen on. These couple of steps that the application has the full screen (no window or android title) and that the screen does not turn off while the user is using the application.

```
requestWindowFeature(Window.FEATURE_NO_TITLE);
getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
```

This is followed by setting the layout. The layout initially is just a surface view which displays the camera preview. As soon as the first frame is captured by the camera it is sent to the onCameraFrame function. Firstly a black rectangle is drawn approximately in the middle of the screen. This black rectangle is the region of interest where colour detection takes place.

```
Core.rectangle(mRgba, new Point(390,170), new Point (450,230), new Scalar(0, 0, 0, 255), 3);
```

Then the frame counter is updated. If a threshold value is reached the calibration toasts (discussed later) are displayed. Then a host of drawing functions are used to complete the UI. Firstly a rectangle is created then filled with the colour detected. This palette visually describes the colour.

```
Mat colorLabel = mRgba.submat(4, 68, 4, 68);
colorLabel.setTo(detectedRGB);
```

Then a black rectangle is drawn beside the palette. Inside this rectangle the matched name of the identified colour is printed in white.

```
Mat color_name_label = mRgba.submat(4, 68, 72, 350);
color_name_label.setTo(black);
Core.putText(color_name_label, identified_name, new Point (40,50), fontFace, fontScale, new Scalar (255,255,255));
```

The same procedure is repeated to display the RGB values of the colour at the top of the screen. The white text against a black background makes reading the text easier.

```
Mat color_rgb_label = mRgba.submat(4, 68, 350, 750);
color_rgb_label.setTo(black);
Core.putText(color_rgb_label, rgb_values, new Point (40,50), fontFace, fontScale, new Scalar (255,255,255));
```

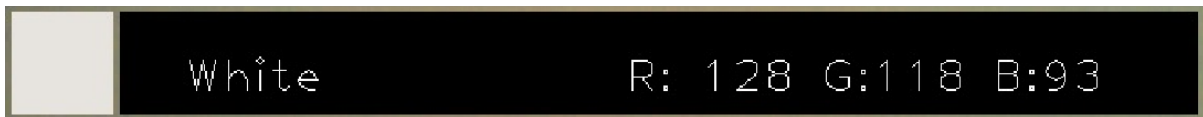



Figure 34: Information bar.

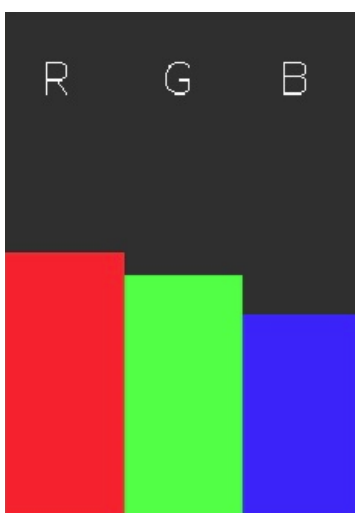
The next step involves drawing the RGB graph at the bottom right of the screen. This procedure begins by allocation a 200*310 rectangle at the bottom of the screen and colouring it black.

```
Mat rgbalabel = mRgba.submat(mRgba.height() - 300, mRgba.height(), mRgba.width()-210, mRgba.width());
rgbalabel.setTo(black);
```

The next step is putting the text “R”, “G” and “B”. To do this accurately a sub rectangle is formed for each colour and using precise coordinates the text is drawn.

```
Core.putText(rtext, "R", new Point (20,50), fontFace, fontScale, new Scalar (255,255,255));
Core.putText(gtext, "G", new Point (20,50), fontFace, fontScale, new Scalar (255,255,255));
Core.putText(btext, "B", new Point (20,50), fontFace, fontScale, new Scalar (255,255,255));
Mat rtext = rgbalabel.submat(0, 100, 0, 70);
Mat gtext = rgbalabel.submat(0, 100, 70, 140);
Mat btext = rgbalabel.submat(0, 100, 140, 210);
```

The final part is drawing the bars representing the proportions of each colour present. To calculate the height first an offset is calculated by subtracting the component value (for example R = 150 in RGB) from 200. This offset is then added to 100 to give a height between 0 and 200. The width is a constant so finally the bar is drawn and filled with the component colour.



```
Mat rlabel = rgbalabel.submat(100+rset, 300, 0, 70);
Mat glabel = rgbalabel.submat(100+gset, 300, 70, 140);
Mat blabel = rgbalabel.submat(100+bset, 300, 140, 210);
rlabel.setTo(red);
glabel.setTo(green);
blabel.setTo (blue);
```

Figure 35: RGB bar graph.

Colour Detection:

The colour detection algorithm is the basis of this application. To save resources and have the ability to target smaller objects colour detection is only done on a small region within the camera preview. A new rectangle known as roiRect (region of interest rectangle) is defined; this rectangle has the same specifications to the one drawn in the UI creation section.

```
Rect roiRect = new Rect();
roiRect.width=50;
roiRect.height =50;
roiRect.x = 390;
roiRect.y = 170;
```

This region is then converted into the HSV colour space (HSV is more robust for colour differentiation). Then the HSV value for each pixel is calculated; this value is then summed and then the average HSV value for the region is calculated. Finally this value is converted back into the RGBA colour space.

```
Mat roiRgba = mRgba.submat(roiRect);
Mat roiRectHSV = new Mat();
Imgproc.cvtColor(roiRgba, roiRectHSV, Imgproc.COLOR_RGB2HSV_FULL);
// Calculate average color of rect
mColorHSV = Core.sumElems(roiRectHSV);
int pointCount = roiRect.width*roiRect.height;
for (int i = 0; i < mColorHSV.val.length; i++)
    mColorHSV.val[i] /= pointCount;

detectedRGB = converScalarHsv2Rgba(mColorHSV);
```

The retrieved RGBA value is stored in an array. The individual values can be accessed by selecting the appropriate values in the array. These individual values are then used for colour classification.

```
red = (int) detectedRGB.val[0] ;
green = (int) detectedRGB.val[1];
blue = (int) detectedRGB.val[2] ;
```

Calibration:

The earlier versions of the implementation did not contain this feature. It was found however that in certain lighting conditions colour classification was inaccurate; this was due to lighting increasing the prominence of one or more colours.

As soon as the application is created two toasts (notifications) are initialized. The purpose of these is to inform the user when calibration starts and when it is completed.

```
start = Toast.makeText(context, "Point at white and touch screen to begin calibration", duration);
cal_complete = Toast.makeText(context, "Calibration Complete", duration);
```

As soon as the first camera frame is captured a counter is updated. When the counter hits 40 (approximately 5 s after start) the user is prompted to point the camera at a white surface and touch the screen.

```
fcount++;

if (fcount ==40)
{
    start.show();
}
```

When the user touches the screen RGB values are recorded and sent to the normalise function.

```
// normalise once
if (tcount == 0)
{
    normalise(r,g,b);
    ncounter =1;
}

tcount++;
```

When the colour is white the three values (R,G and B) should be roughly the same. The normalise function compares the three values and performs either addition or subtraction to make the largest and the smallest value equal to the middle value. The values added or subtracted are known as offsets and are initially 0.

```

if (rlargest == true)
{
    diff = r -second;
    rts = diff* -1;
    r = r -diff;
}
if (third == r)
{
    diff = second -r;
    rts = diff;
    r = r+diff;
}
}

if (glargest == true)
{
    diff = g -second;
    g = g -diff;
    gts = diff* -1;
}
if (third == g)
{
    diff = second -g;
    g = g+diff;
    gts = diff;
}
}

if (blargest == true)
{
    diff = b -second;
    b = b -diff;
    bts = diff* -1;
}
if (third == b)
{
    diff = second -b;
    b = b+diff;
    bts = diff;
}
}

```

After these values are calculated the toast informing the user that calibration is completed is shown.

```

if (ncounter == 1 && fcount > 80 )
{
    cal_complete.show();
    ncounter = 3;
}

```

From this point on these offset values are added to each value before classification to take into account illumination.

```

r = (int) detectedRGB.val[0] +rts ;
g = (int) detectedRGB.val[1] +gts;
b = (int) detectedRGB.val[2] +bts ;

```

After calibration was implemented the applications accuracy increased greatly in different lighting conditions. This calibration procedure is one of the greatest strengths of this application. Androids camera has a built in white balance parameter for colour correction, it however has a very limited number of lighting settings to choose from.

Colour Classification:

Colour classification is the most important aspect of this application. From the onset, it was known that the strength of the classification algorithm will determine the quality of the final application. Without a robust colour classification algorithm, other aspects of the application are of little value.

There are two algorithms used for classification, SimpleRGB and LutRGB. SimpleRGB uses the ratios of the colour components present to determine the colour present. It works better with absolute colours but struggles with shades. LutRGB compares the retrieved RGB value with a list of known values and finds the closest match to determine the colour. Due to the size of the list this algorithm has a higher chance of misclassification.

The default setting of the application is to use SimpleRGB due to its accuracy but LutRGB is more practical as it can name more colours. Even with the same RGB values the algorithms can produce different results.

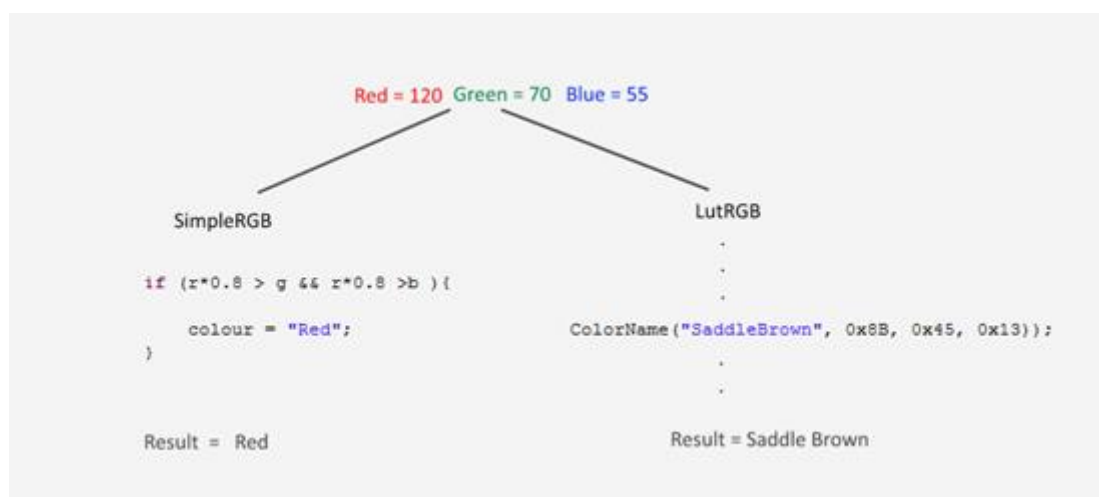


Figure 36: same values, different classification

Evaluation:

LogCat:

The purpose of software testing is to determine the overall quality of the product. It should be done in a methodological manner to appraise the quality – related information of the product.

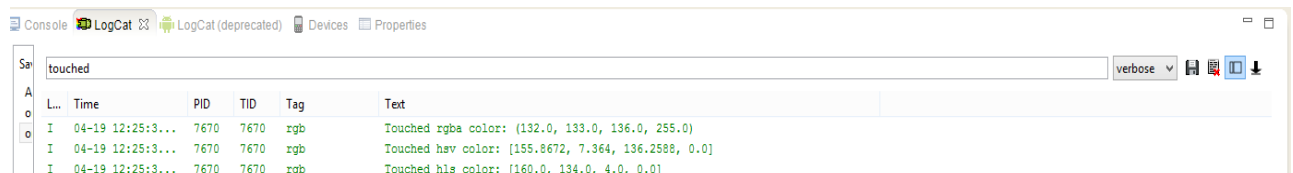
The Android logging system provides a tool for gathering and viewing system debug output. Logs from numerous applications and portions of the system are collected in a series of circular buffers, which can then be viewed and filtered by the LogCat command³⁰.

LogCat was used extensively when testing different aspects of the project. Whenever the application was started directly from eclipse, the LogCat window started displaying a constant stream of messages about the state of the application such as “OpenCV loaded successfully”. This is primarily due to the admirably written openCV library.

OpenCV library facilitated easier debugging. When the program crashed it logged possible exceptions. For example if the dimensions of the drawn items exceeded the screen size, the application crashed, however LogCat displayed that the size of matrix was greater than the screen size.

To handle the large number of messages, LogCat provides an excellent filtering system. By using filter words such as colour, LogCat can be requested to display only the messages which contained the tag colour.

```
Log.i(TAG2, "Touched rgba color: (" + detectedRGB.val[0] + ", " + detectedRGB.val[1] +
", " + detectedRGB.val[2] + ", " + detectedRGB.val[3] + ")");
Log.i(TAG2, "Touched hsv color: "+mColorHSV);
Log.i(TAG2, "Touched hls color: "+cspace1);
```



The screenshot shows the Eclipse IDE's LogCat window. The search filter is set to 'touched'. The window displays three log entries, each with a timestamp, PID, TID, Tag, and Text. The text of the logs matches the code shown in the image above.

Time	PID	TID	Tag	Text
04-19 12:25:3...	7670	7670	rgb	Touched rgba color: (132.0, 133.0, 136.0, 255.0)
04-19 12:25:3...	7670	7670	rgb	Touched hsv color: [155.8672, 7.364, 136.2588, 0.0]
04-19 12:25:3...	7670	7670	rgb	Touched hls color: [160.0, 134.0, 4.0, 0.0]

Figure 37: LogCat output.

Colour Space Tests:

To build an accurate colour classifier first an understanding of how colours differ numerically was needed. To perform this test a phone with the colour detection algorithm running was pointed at a printed paper with three sections: red green and blue. The response at each colour was recorded 10 times using three different colour spaces HLS, HSV and RGB. The conversions were done inside the application using the openCV functions and the values were recorded through logCat outputs. The table below shows the average value of the parameters for the three colours.

	Red	Green	Blue
R	144	107	53
G	62	123	53
B	53	74	95
H	3.5	55	138
S	159	99	112
V	144	123	95
H	4	55	139
L	99	99	72
S	115	62	0

It is important to note that **these tests were done before calibration** was implemented and the values above are the raw unaltered values. We can see that each colour space has its advantages and disadvantages. In the RGB colour space red and green are difficult to distinguish but red and blue are easy to distinguish. In this case hue might seem sufficient to differentiate the colours, but it when white and black are introduced, using hue alone leads to many misclassifications.

To test the consistency of these results a simple classifier was built using the table above. Without varying the environment greatly (same time, different day), the application was run again but this time the classifier was added. Instead of numerical values the colour name was printed and at the logCat output. To make the test slightly more difficult two more colours black and white were added.

The results of these tests are summarised below.

	Accuracy %	Accuracy %
	RGB	HLS
Red	90	100
Green	20	100
Blue	80	70
White	90	70
Black	90	80

Even though the tests were done in natural lighting, it was found (by looking at the rgb values of a white page) that the red component of the colour was enhanced by the environment. This explains why red had such a high accuracy and why green's detection rate was far less (In RGB space even green had a high red component (see previous table)). In the RGB colour space recognising black and white was easier compared to the HLS colour space.

In reality colours are not absolute. They are usually shades of different colours. Similarly natural lighting is not prevalent in most indoor situations. Therefore even if this alpha classifier had a 100 % accuracy rate it would not be sufficient for real life applications. Nevertheless it showed that differentiation of colour was possible and **calibration** was needed.

After calibration was implemented (as described in the implementation section) these results were achieved for the simple classifier in natural lighting.

	Accuracy %
	RGB
Red	100
Green	100
Blue	100
White	100
Black	100

As calibration removed the excess red, the simple classifier worked as expected. Now the classifier could be improved to work on real world objects in real time applications. However this would be a gradual progress.

Changing Lighting Conditions:

The purpose of these tests was to see how changes in illumination affect the detected R, G, B values. The tests were done in three different lighting conditions: natural lighting, incandescent lighting and low levels of lighting. These tests were done twice, with and without calibration to see how effective the calibration mechanism was. The testing methodology was the same as before the only thing that was altered was the lighting conditions. The results are shown below:

Natural Lighting							
	With Calibration				Without Calibration		
	R	G	B		R	G	B
Red	201	100	75		162	127	92
Green	65	156	93		42	96	85
Blue	46	72	122		127	135	161
Black	0	0	2		0	0	10
White	186	191	192		154	151	137
Incandescent Lighting							
	With Calibration				Without Calibration		
	R	G	B		R	G	B
Red	161	73	77		169	141	103
Green	61	107	67		124	113	67
Blue	41	53	109		154	150	122
Black	5	1	6		55	34	18
White	150	150	149		55	34	18
Low levels of Lighting							
	With Calibration				Without Calibration		
	R	G	B		R	G	B
Red	191	78	53		140	101	128
Green	83	139	84		83	115	99
Blue	47	31	91		1	24	55
Black	10	11	10		140	132	151
White	139	137	141		0	0	10

We can see that in incandescent lighting calibration is needed to reach any acceptable form of recognition. This is due to the fact that incandescent lighting adds a significant value to the

red component. In natural lighting and low lighting calibration makes the differences larger decreasing the possibility of misclassification.

Replication Tests:

The purpose of these tests was to see how accurately the application was detecting the RGB values. This was an important test as without validating proper detection validating colour classification would be of little value.

To run these tests a web application was used to generate a colour palette from known R, G, B values, and then the applications evaluation of the values of the palette was recorded, finally these values were compared.

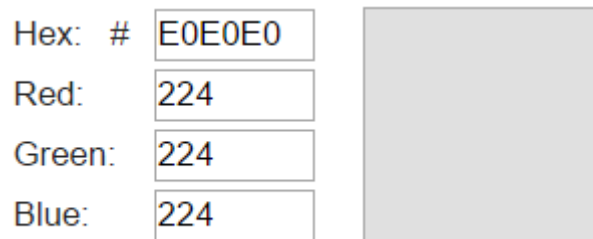


Figure 38: Colour Picker.

To minimise the effect of the environment calibration was done before running these tests. The table below shows some of the results.

Original			Detected		
R	G	B	R	G	B
200	200	200	165	167	167
0	0	0	0	4	7
255	0	0	250	74	58
0	255	0	30	220	50
0	0	255	27	62	217
150	150	150	132	137	133

It is important to note that the reproduction of colours is highly hardware dependent i.e. two different monitors may render two colours with same RGB values differently. It is also important to note that some of the colour intensity is lost as there are reflections on the

screen, the phone's optical sensor etc. We can see that even though some of the intensity was lost, the ratios of each of the colour components roughly remained the same.

Simple Classification Test:

On Screen Test:

Now that calibration had been implemented it was time to test the simple classifier (simple because only 5 states red, green, blue, white and black) on a different data set than the training data set.

The colour names were put into the Google image search engine and the application was tested on the search results. The purpose was to see were the thresholds robust enough to work on a wide variety of objects. The results were very accurate. Unfortunately in very few situations where there was a shadow cast on the object the results were inaccurate. This was due to the reason that calibration was done in an environment with sufficient natural lighting. The images below are actual screen captures from the application. For this test an earlier implementation was running where the region of interest wasn't fixed but was taken by the location of the users touch.

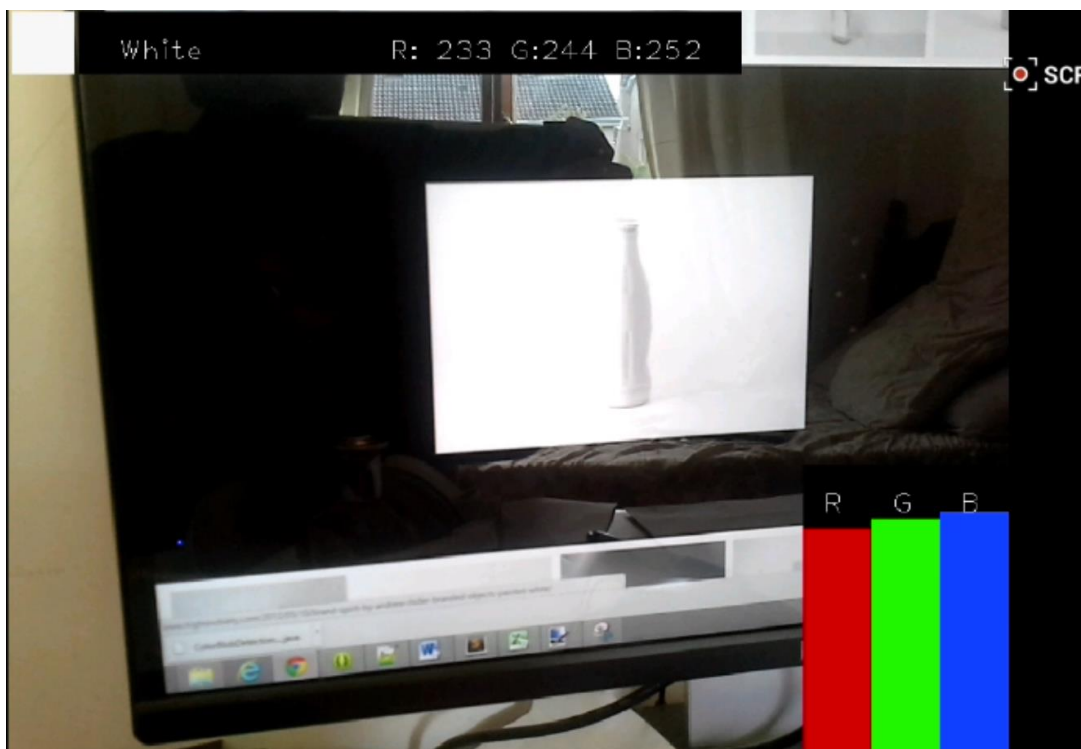


Figure 39: White Bottle.

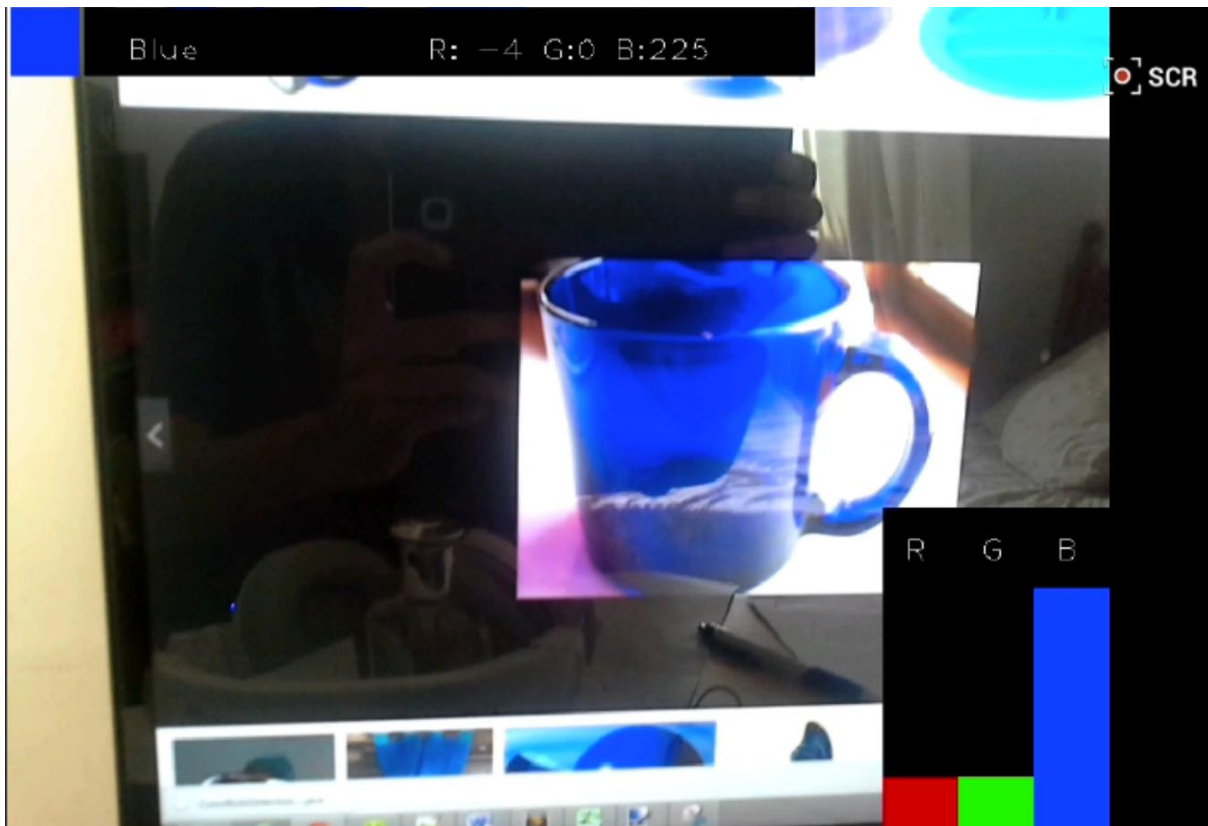


Figure 40: Blue Cup.

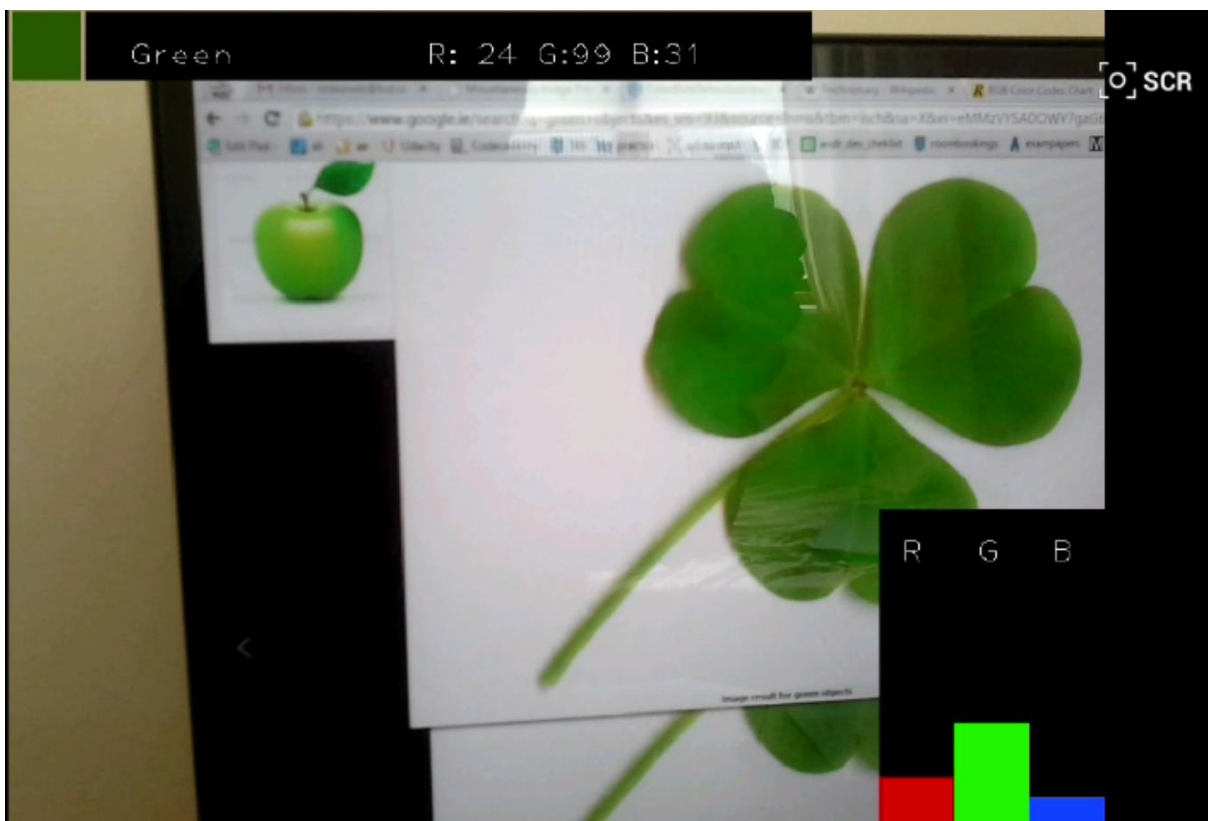


Figure 41: Green Clover.

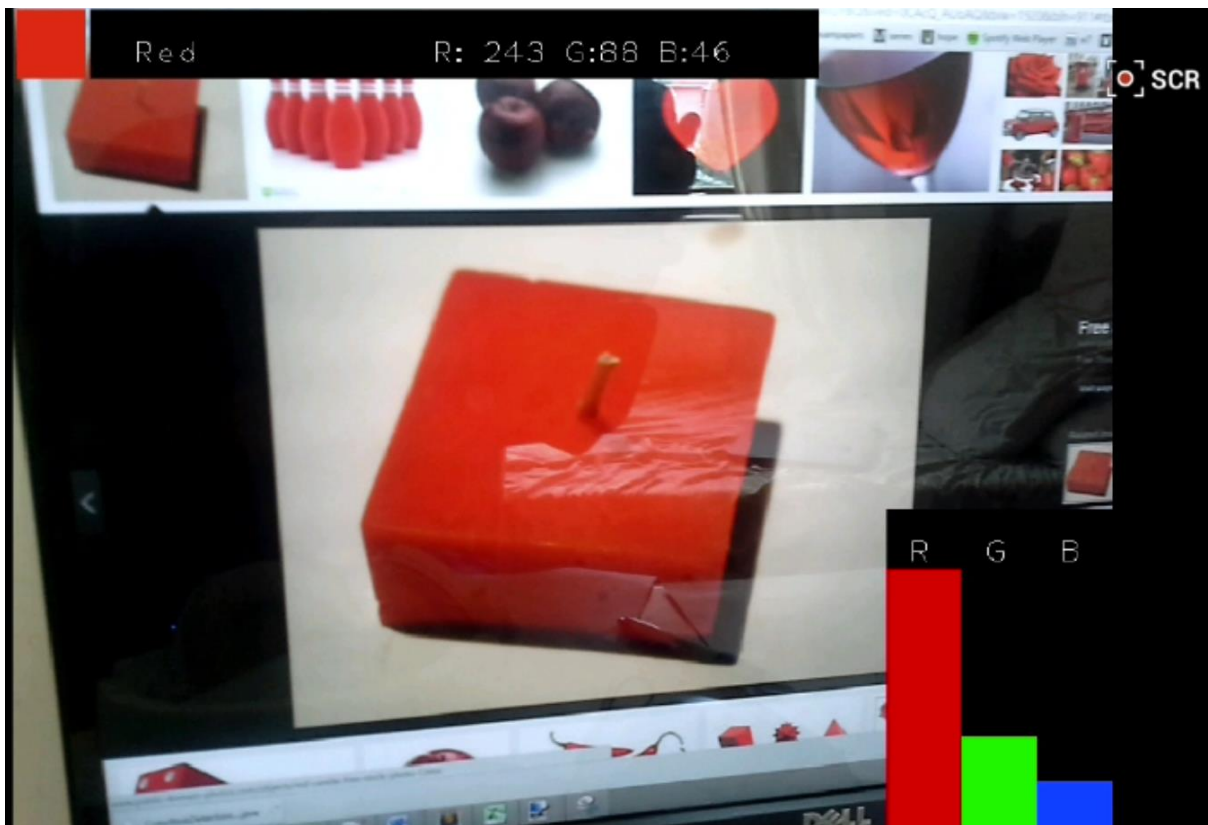


Figure 42: Red Candle.

Real World Objects:

The on screen performance from the last section was admirable, however in most cases real world objects are not as well illuminated as the onscreen ones. Therefore another test was run to test the performance of this classifier on real world objects. The images in the next couple of pages are actual screen captures from the application. For this test an earlier implementation was running where the region of interest wasn't fixed but was taken by the location of the users touch. The lack of illumination should have been compensated by the fact that calibration was done in the same environment.

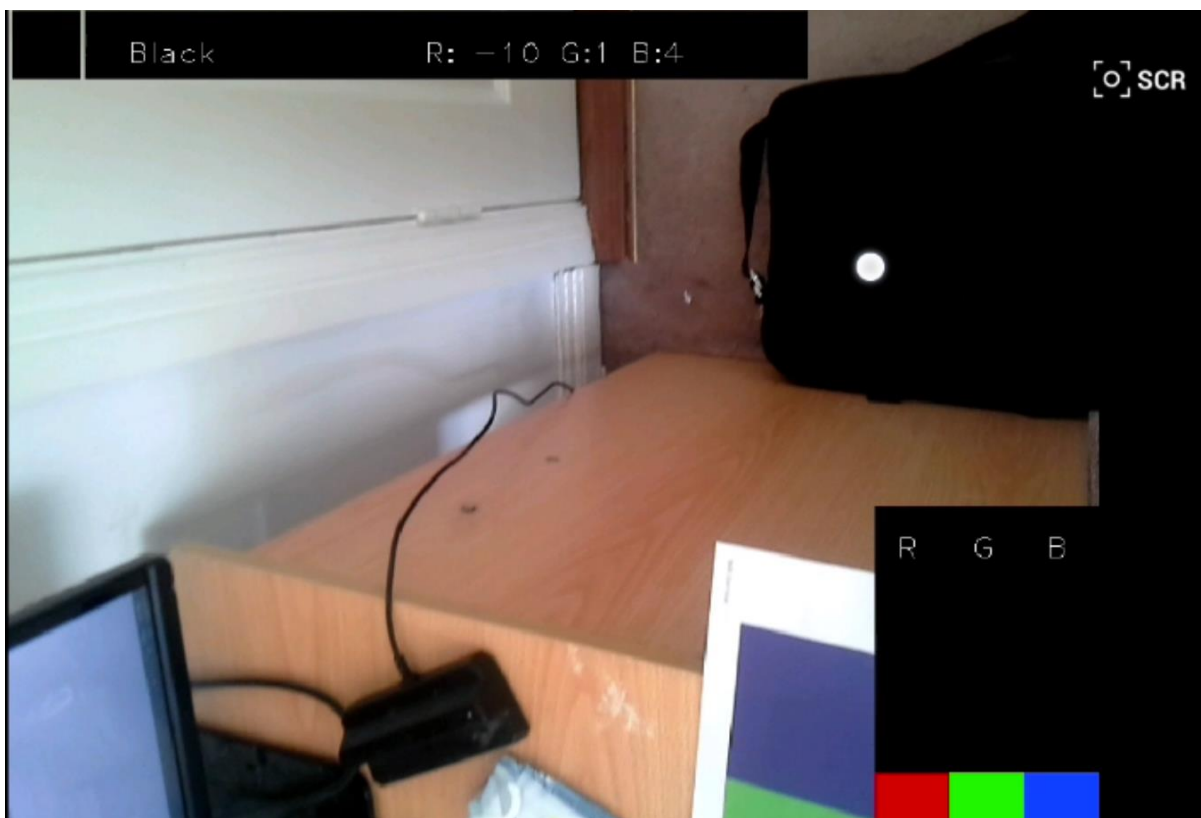


Figure 43: Black Bag.



Figure 44: White Door.



Figure 45: Blue Jeans.

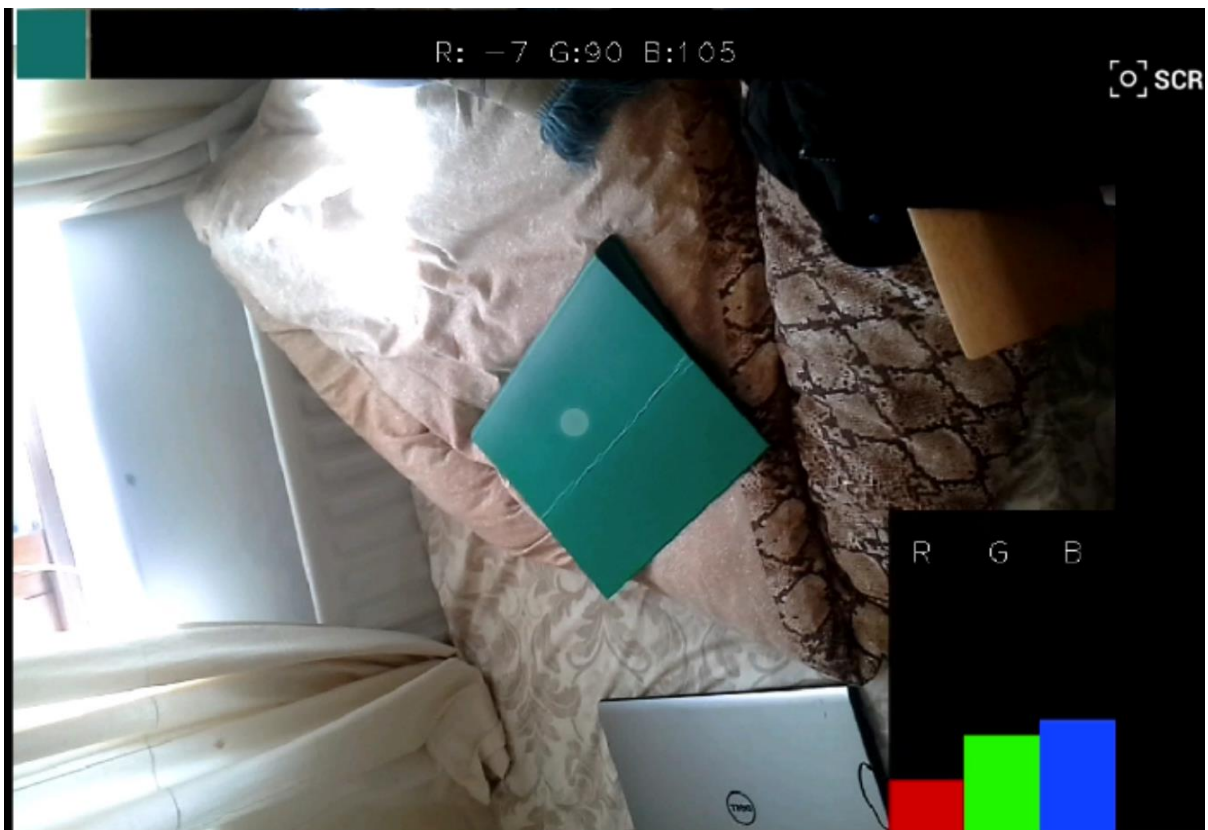


Figure 46: Not Classified (blue =green).

Advanced Classification Test:

We can see that the blueish green file was not classified by the simple classifier. The simple classifier is robust at what it does however is limited by the number of possible states. Therefore an alternative algorithm (lutRGB) was also implemented with this application. Even though disabled by default, this algorithm uses a look up table of more than 50 colours and finds the closest match to determine the colour.

To test the robustness of this algorithm it was tested on real world objects. The images in the next couple of pages are actual screen captures from the application. For this test an earlier implementation was running where the region of interest wasn't fixed but was taken by the location of the users touch.

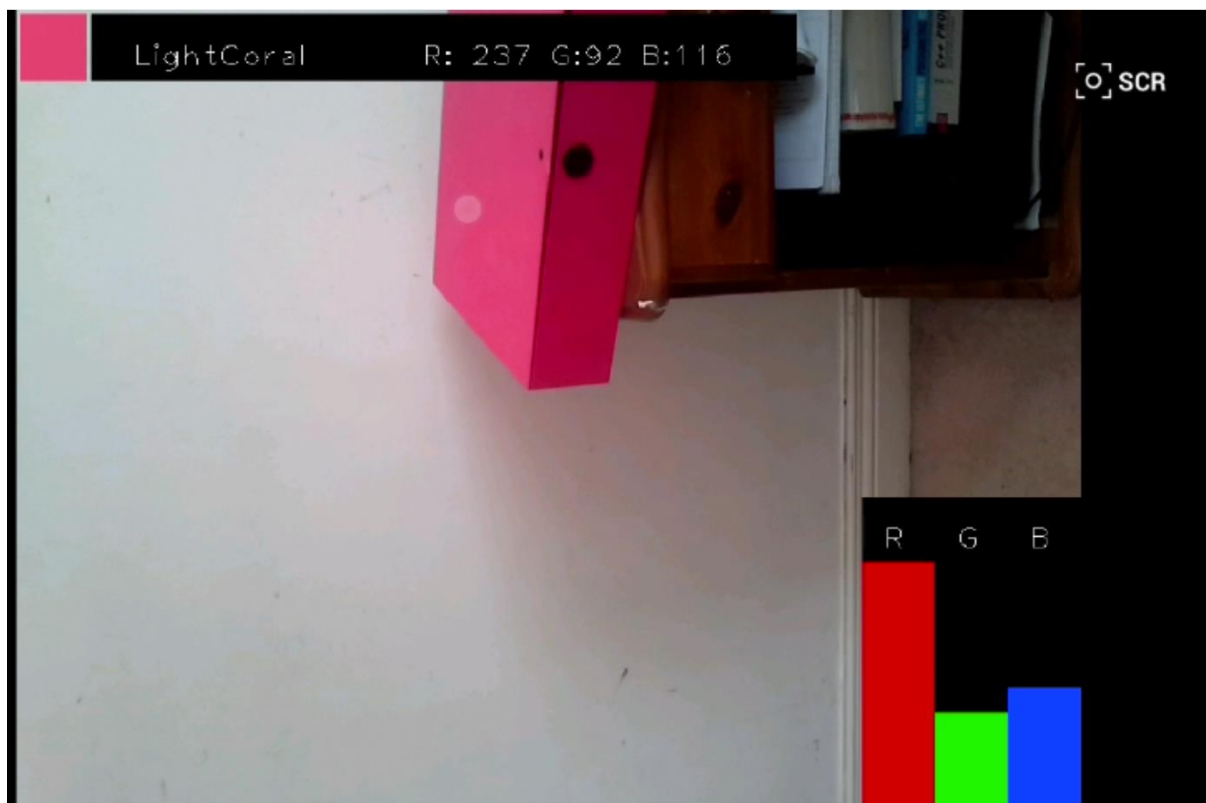


Figure 47: Light Coral Folder.

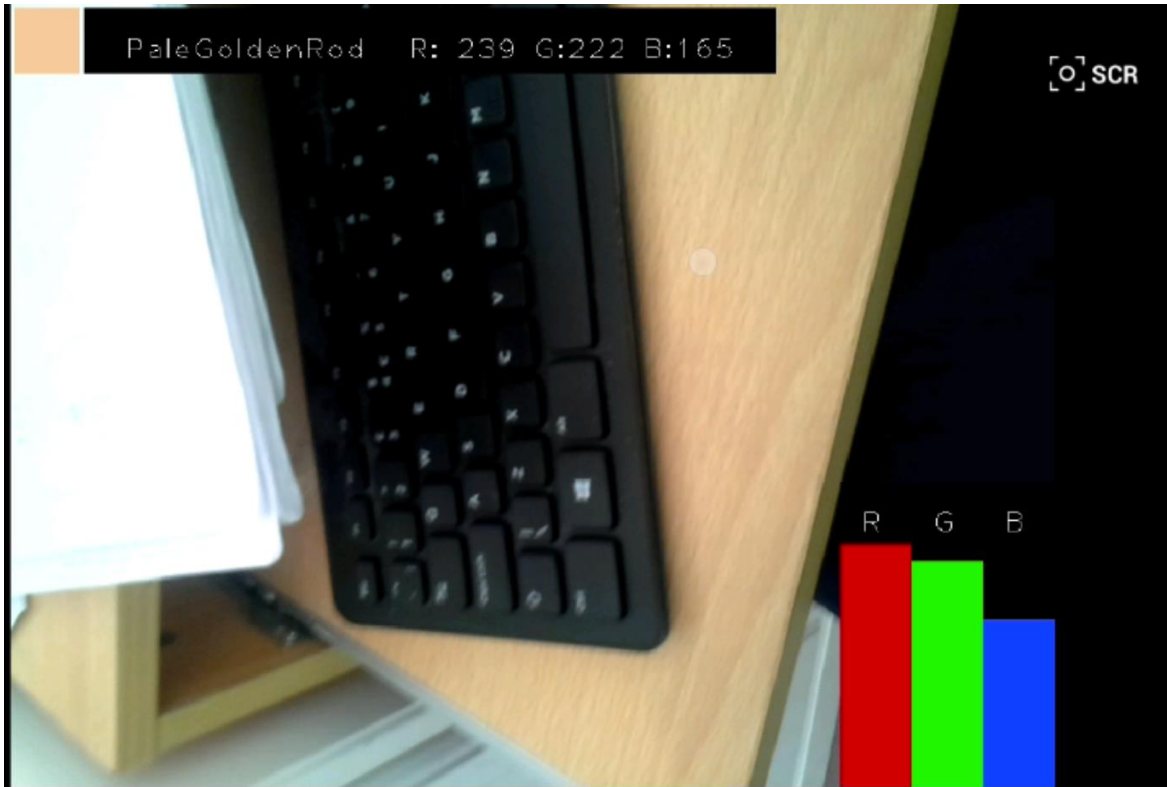


Figure 48: Pale Golden Rod Desk.



Figure 49: Dark Grey Laptop.

User Response:

Even though user input influenced the final design greatly, unfortunately due to time constraints, no formal testing was done by the intended users (colour blind) on the final application. Nevertheless informal sessions were held where subjects were asked to express their opinion about the application. In these sessions first users were given a short presentation to explain the motivation behind the application and then they were asked to test the application themselves. The general response was positive; the users found the user face intuitive and were impressed by the accuracy of recognition. However some users complained that the calibration mechanism was cumbersome.

Discussion of Results:

These are just some of the tests that were done. For example the thresholds used by the simple classifier were modified more than 100 times. Similarly the advanced classifier's entries were modified many times to reflect the values that would be expected in the real world.

If we look at figure 43 we see that the R value is negative. We know that no colour can have a negative red component, so what is the reason behind this negative value? This is due to the fact that illumination where calibration was done had a slightly higher red component which is subtracted from every value. Future versions of the application will take this into account and only show positive values.

The simple classifier is very robust. The numbers of misclassifications were extremely rare when using this algorithm. Even though the numbers of states are extremely low it can be easily modified to increase the number of states to most commonly prevalent colours. The advanced classifier can recognise most colours, however is not as accurate. Another problem with the advanced classifier is that most people cannot visualise some colours it names (such as Light Coral, Golden pale rod etc.). Due to this reason the simple classifier is the default algorithm used by this application.

The accuracy of the application can be mainly attributed to the calibration function. It ensures that the results are not affected by illumination. Before calibration was implemented even the simple classifier didn't function properly. The tests were done on two devices to make sure that they were device independent. Unfortunately the camera implementation used by

OpenCV prevents it from working on emulated devices. This severely limited the type of devices the application could be tested on.

Conclusion:

Similar Products:

Similar products confirm the fact that a solution is required. Similar products might also offer insight on how to improve this application. In this section some of the existing solutions are evaluated. There are many applications which are either designed to aid the colour blind, or simulate colour blindness. However most of these are substandard, and this is reflected in their user reviews. Therefore only applications with a large user base are critiqued. It is important to note that most of the applications discussed use the java implementation for the camera view therefore will not work as intended with a projector on most devices.

What colour is it?

This application by Nico Troia has between 10000 and 50000 installs³¹ and has an average rating of 3.5/5 based on 168 reviews. To use the application the user either takes or retrieves an image, then selects a region of interest, then presses the analyse button. A new activity starts which displays the possible list of hex values. When the user selects one of these values a matching colour name is retrieved from a database.

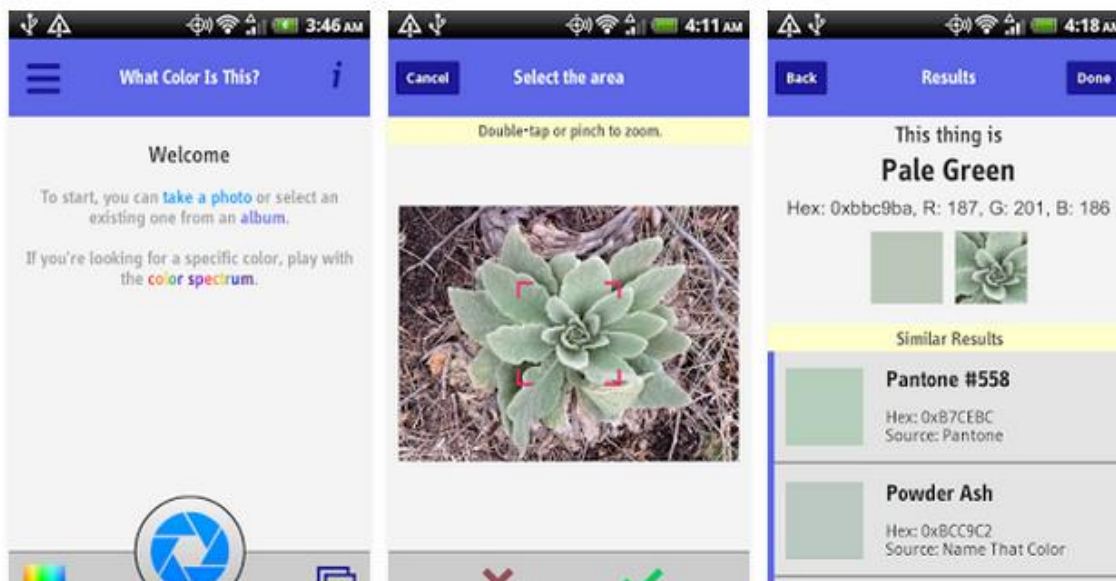


Figure 50: What colour is it?

The user interface of the application is very intuitive. As soon as the application starts instructions are displayed on how to use the application and they are displayed during the various stages of the colour retrieval process.

This approach however takes a long time to retrieve the colour. On average it took more than 10 seconds to retrieve a single colour. Furthermore even though the hex values are generated on the phone, the matching name is retrieved from a web server; this means the user will need access to the internet to fully use this application. The application also struggles in different lighting situations.

Color Id:

This application by Gordon Hempton has between 10000 and 50000 installs³² and has an average rating of 3.5/5 based on 135 reviews. As soon as the camera view is started the colour detection procedure begins. An on screen circle displays the region of interest and the colour name along with the hex value is displayed on the bottom of the screen.

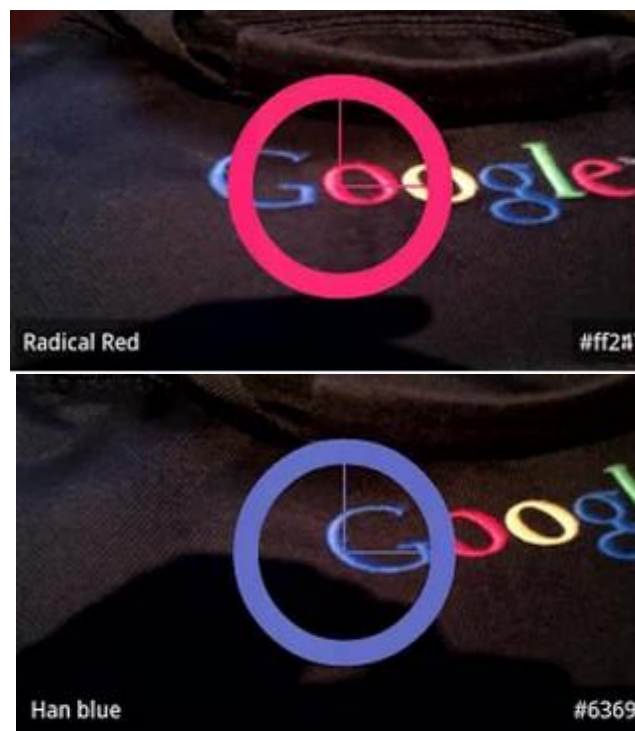


Figure 51: Color id.

The user interface of the application is intuitive and fluid. It is very quick to retrieve the colour as no user input is required. Furthermore the cross-hair at the centre makes it really easy to retrieve colours with smaller areas.

Many of the users complained that the list of colours is very large and they would just like the nearest primary colour. Since the colour name is updated every frame and fluctuates a lot some times its difficult to read the colour. Some users requested a hold feature to rectify this problem. The application also struggles with very bright and very dark colours and does not produce consistent results in different lighting situations.

Colorblind Vision:

This application by Bradley C. Grimm has between 50000 and 100000 installs³³ and has an average rating of 3.8/5 based on 518 reviews. The application has 4 different modes vision, test, learn and extract. The vision section simulates different types of colour deficiencies. The test section allows the user to determine the state of their vision by taking Ishihara tests. The learn section allows the user to learn about the causes, cures, benefits etc. of colour blindness and finally the extract section allows colour blind people to extract specific colours from an image.



Figure 52: Colorblind vision.

The user interface of the application is excellent. The host of features make it an excellent suite for anyone who wants to understand colour blindness. The extract feature is excellent as it makes the background black and white. Unfortunately this feature is only useful if the user knows what colour to extract. Also it only simulates a number of colour deficiencies. There is no tool for naming colours but many colour blind users rated the application highly.

Color Detector:

This application by mobialia.com has between 50000 and 100000 installs³⁴ and has a 3.6/5 rating based on 326 views. Once the application starts the camera view is shown with a circle at centre to show the region of interest. As soon as the user touches the screen, the camera view pauses briefly and the colour information is displayed at the bottom of the screen.



Figure 53: Color detector.

The application has a host of features which make this application praiseworthy. The application has speech synthesis so the user doesn't have to read the colour from the screen. Even though the database has 1600 colours, the application also displays a simple colour to make understanding the colour easier. The application also allows the user to choose from a host of white balance settings to take into account illumination. However the most impressive feature of this application is its small code base. At 68 kilobytes this application is magnitudes smaller than most other applications discussed (5Mb). This increases the applications potential user base as it is downloaded instantaneously and occupies less storage space on the phone.

Unfortunately despite its amazing list of features, the colour recognition algorithm is very inaccurate. It only recognises the primary colours consistently and struggles with simple colours such as white and black. Many users complained about its poor detection rate.

Color Blindness Simulate Correct:

This application by Seewald Solutions has between 50000 and 100000 installs³⁵ and has a 3.8/5 rating based on 429 views. This application allows the user to simulate colour blindness and also shows a corrected view. Once the application is opened, a camera view is started. Using the menu button the user can ask the application to simulate different kinds of colour blindness. The user can also ask the application to show a corrected view which transforms the unperceptible colours to perceptible ones.

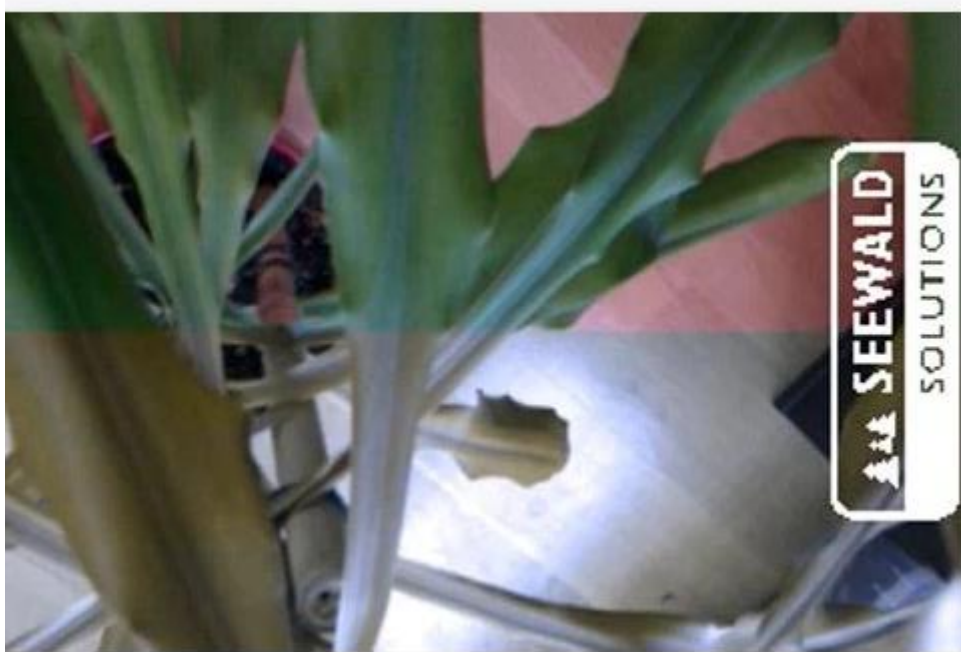


Figure 54: Color Blindness Simulate Correct.

Despite the variety of features, the user interface is unintuitive. There is no option to select a particular state so the user has to cycle through many states before they can reach the state they want. As the application uses Daltonization for colour transformation, the actual colour information is not made available to the user, so it is unsuitable for applications where the user needs to know the colour. Some users also complained that the application did not run properly on low end devices. Nevertheless this application is the highest rated application compared to all the other applications discussed in this section.

In this section 5 of the most popular detection applications were critiqued. One interesting observation was that most of the users were not colour blind people themselves (this could be due to the fact that a small percentage of the population is colour blind) but people wanting to understand colour blindness or perform colour detection for other purposes. Each application had its advantages and disadvantages as shown by the user scores. These applications

between them were installed approximately 200,000 times and these are a small sub section of the applicable applications out there. This shows that there is a need for such an application.

Appraisal:

There are certain aspects of this solution, which were implemented exactly as the initial design, while others could have been implemented better.

The user interface implementation is praiseworthy. The final implementation looks exactly like the design. It is minimalistic yet intuitive. This is mainly due to the attention to detail that was given and the strict compliance to the design guidelines as proposed by Clayton and Gould.

Another aspect that is admirable is the rapid response rate of the application. The application does not lag at all on the devices listed and no random crashes occurred while using the application. Credit here is due to the OpenCV library as it handles the camera functions. The code base is relatively small and any features that slowed down the application were removed.

The calibration algorithm is excellent. It allows the application to work in different lighting conditions and ensures that the results are accurate. This feature was not found in any other similar application and is one of the two unique selling points (other being projector compatibility) of the application.

The accuracy of the simple classifier is also noteworthy. Even though basic in its implementation, thanks to the calibration mechanism, the simple classifier has a very high accuracy rate. This is due to the many tests that were done to ensure that the thresholds used were robust.

Unfortunately the performance of the advanced classifier is not as good. Even though it has a large library of colours and uses calibration as well, its misclassification rate is intolerable. This is due to the fact that this list was mainly designed for web applications and does not take into account how colours appear in real life.

In the literature review section, many similar applications were criticised for their lack of end user testing. Unfortunately due to constraints of resources, this application could not be tested by colour blind users themselves. Lack of user testing has been the biggest disappointment in this project.

Nevertheless now that the application is developed there is still time to test it extensively and make other improvements. These are discussed in the next section.

Future Work:

The implementation discussed in this project has sufficient functionality but there is still room for improvement.

The current application only deals with a live video feed. This has two major drawbacks. Firstly the resolution is much lower when compared to a picture. However more importantly the object of interest could be time sensitive, where the user would like to do recognition later. This feature is still in development and hasn't been fully integrated with the application yet.

Even though the user interface is fluid it can still be polished. A menu section needs to be implemented which allows the user to save preferences, access advanced help etc. Adding a menu section also presents an opportunity to add additional modes such as simulating different types of colour blindness. Adding too many features could however slow down the application so needs to be done carefully.

The current calibration procedure is another area where improvements can be made. Even though the procedure itself is relatively quick it can become cumbersome if the user needs to do it on every use. Therefore a mechanism needs to be developed where the user can save and retrieve calibration settings. The application should also allow the user to save the label and the image for later retrieval.

Another aspect that needs improvement is user accessibility. Currently the labels are only in English and need to be translated to different languages. The intended user may have reading difficulties so the application should also have a text to speech feature where each label and the instructions are spoken. The application currently only works on Android devices. Most

of the application is built using the OpenCV libraries. OpenCV is cross platform and has libraries for iOS, Windows etc. Therefore extending it to other platforms should be feasible.

Even though the application is catered towards colour blind people, colour detection has many other applications. It can be used on an assembly line to filter products by colour; self-driving cars can use it to see the status of the traffic lights, in sports for automatic decision making (goal line technology), in cooking appliances to stop once a particular colour is reached etc.

Ultimately the application needs to be used by the intended user in real situations to see where further improvements can be made. It needs to be tested on a wide range of users in a wide variety of situations.

List of Figures Used:

Figure 1: <http://www.webexhibits.org/causesofcolor/2C.html>

Figure 2: <http://www.colourblindawareness.org/colour-blindness/living-with-colour-vision-deficiency/>

Figure 3: edited form of <http://www.picoprojector-info.com/apple-shared-workspace-patent>

Figure 4: <http://webvision.med.utah.edu/book/part-i-foundations/gross-anatomy-of-the-eye/>

Figure 5: <http://hyperphysics.phy-astr.gsu.edu/hbase/vision/colcon.html#c1>

Figure 6: <http://hyperphysics.phy-astr.gsu.edu/hbase/vision/cie.html>

Figure 7: <http://www.colour-blindness.com/general/causes/>

Figure 8: <http://www.ideastry.com/truth-or-myth-fun-facts-about-eyes/511>

Figure 9: <http://www.color-blindness.com/color-arrangement-test/>

Figures 10, 11, 12: Ohkubo, T.; Kobayashi, K., "A color compensation vision system for color-blind people," SICE Annual Conference, 2008 , vol., no., pp.1286,1289, 20-22 Aug. 2008, doi: 10.1109/SICE.2008.4654855

Figures 13, 14: Ananto, B.S.; Sari, R.F.; Harwahu, R., "Color transformation for color blind compensation on augmented reality system," User Science and Engineering (i-USER), 2011 International Conference on , vol., no., pp.129,134, Nov. 29 2011-Dec. 1 2011 doi: 10.1109/iUSER.2011.6150551

Figures 15,16: Hyun-Ji Kim; Jae-Yun Jeong; Yeo-Jin Yoon; Young-Hyun Kim; Sung-Jea Ko, "Color modification for color-blind viewers using the dynamic color transformation," Consumer Electronics (ICCE), 2012 IEEE International Conference on , vol., no., pp.602,603, 13-16 Jan. 2012 doi: 10.1109/ICCE.2012.6162036

Figure 17,18 Siew-Li Ching; Sabudin, M., "Website image colour transformation for the colour blind," Computer Technology and Development (ICCTD), 2010 2nd International Conference on , vol., no., pp.255,259, 2-4 Nov. 2010 doi: 10.1109/ICCTD.2010.5645874

Figure 19,20: Harwahu, R.; Sari, R.F., "Implementing Speech Feature for Embedded System to Support Color Blind People," Informatics and Computational Intelligence (ICI), 2011 First International Conference on , vol., no., pp.256,261, 12-14 Dec. 2011 doi: 10.1109/ICI.2011.49

Figure 28, 29: edited from <http://pixshark.com/red-apple-photos.htm>

Figure 31: edited from <http://www.pickmore.com/review-htc-sensation-xe-with-dr-dre-beats-audio-3780>

Figure 32: edited from <http://phandroid.com/samsung-galaxy-note-8/>

Figure 33: edited from <http://www.gizmag.com/microvision-show-pico-projector/8573/>

Figure 38: http://www.rapidtables.com/web/color/RGB_Color.htm

Figure 50:

<https://play.google.com/store/apps/details?id=air.com.nicotroia.whatcoloristhis&hl=en>

Figure 51: <https://play.google.com/store/apps/details?id=com.hempton.colorid&hl=en>

Figure 52:

https://play.google.com/store/apps/details?id=com.givewaygames.colorblind_ads&hl=en

Figure 53: <https://play.google.com/store/apps/details?id=com.mobialia.colordetector&hl=en>

Figure 54:

<https://play.google.com/store/apps/details?id=com.SeewaldSolutions.ColorBlindnessSimulator&hl=en>

All other figures are either screen shots or diagrams created specifically for this project.

References:

1. *Colour Blindness*, Available at: <http://www.colourblindawareness.org/colour-blindness/> (Accessed: 27th December 2014).
2. *Causes & incidence of Colour Blindness*, Available at: <http://www.webexhibits.org/causesofcolor/2C.html> (Accessed: 21st December 2014).
3. *Colour Blindness*, Available at: <http://www.allaboutvision.com/conditions/colordeficiency.htm> (Accessed: 20th December 2014).
4. *How does the human eye work?*, Available at: <https://www.nkcf.org/how-the-human-eye-works/> (Accessed: 11th February 2015).
5. *The Retina*, Available at: <http://hyperphysics.phy-astr.gsu.edu/hbase/vision/retina.html> (Accessed: 12th February 2015).
6. *Rods and cones*, Available at: <http://hyperphysics.phy-astr.gsu.edu/hbase/vision/rodcone.html#c3> (Accessed: 12th February 2015).
7. *The C.I.E color space*, Available at: <http://hyperphysics.phy-astr.gsu.edu/hbase/vision/cie.html> (Accessed: 12th February 2015).
8. Dzul kifli MA, Mustafar MF. The Influence of Colour on Memory Performance: A Review. *The Malaysian Journal of Medical Sciences : MJMS*. 2013;20(2):3-9.
9. *Color constancy*, Available at: http://colorconstancy.com/?page_id=9 (Accessed: 12th February 2015).
10. *Color blindness causes*, Available at: <http://www.colour-blindness.com/general/causes/> (Accessed: 13th February 2015).
11. D.M. Tait and J. Carroll, Color Blindness: Acquired, In *Encyclopedia of the Eye*, edited by Darlene A. Dartt, Academic Press, Oxford, 2010, Pages 312-317, ISBN 9780123742032, <http://dx.doi.org/10.1016/B978-0-12-374203-2.00216-5>.
12. Jennifer Birch, Efficiency of the Ishihara test for identifying red-green colour deficiency, *Ophthalmic and Physiological Optics*, Volume 17, Issue 5, September 1997, Pages 403-408, ISSN 0275-5408, [http://dx.doi.org/10.1016/S0275-5408\(97\)00022-7](http://dx.doi.org/10.1016/S0275-5408(97)00022-7).
13. *Color Arrangement Test*, Available at: <http://www.colour-blindness.com/colour-blindness-tests/colour-arrangement-test/> (Accessed: 15th February 2015).

14. Ohkubo, T.; Kobayashi, K., "A color compensation vision system for color-blind people," SICE Annual Conference, 2008 , vol., no., pp.1286,1289, 20-22 Aug. 2008, doi: 10.1109/SICE.2008.4654855
15. Ananto, B.S.; Sari, R.F.; Harwahu, R., "Color transformation for color blind compensation on augmented reality system," User Science and Engineering (i-USEr), 2011 International Conference on , vol., no., pp.129,134, Nov. 29 2011-Dec. 1 2011 doi: 10.1109/iUSEr.2011.6150551
16. : Hyun-Ji Kim; Jae-Yun Jeong; Yeo-Jin Yoon; Young-Hyun Kim; Sung-Jea Ko, "Color modification for color-blind viewers using the dynamic color transformation," Consumer Electronics (ICCE), 2012 IEEE International Conference on , vol., no., pp.602,603, 13-16 Jan. 2012 doi: 10.1109/ICCE.2012.6162036
17. Siew-Li Ching; Sabudin, M., "Website image colour transformation for the colour blind," Computer Technology and Development (ICCTD), 2010 2nd International Conference on , vol., no., pp.255,259, 2-4 Nov. 2010 doi: 10.1109/ICCTD.2010.5645874
18. Harwahu, R.; Sari, R.F., "Implementing Speech Feature for Embedded System to Support Color Blind People," Informatics and Computational Intelligence (ICI), 2011 First International Conference on , vol., no., pp.256,261, 12-14 Dec. 2011 doi: 10.1109/ICI.2011.49
19. John D. Gould, Clayton Lewis (1985) 'Designing for usability: key principles and what designers think', *Communications of the ACM*, 28(13), pp. 300-311.
20. The Verge. 2011. *Android history*. [online] Available at: <http://www.theverge.com/2011/12/7/2585779/android-history> [Accessed: 31 Mar 2014].
21. Rating, A. 2014. *Android climbed to 79 percent of smartphone market share in 2013, but its growth has slowed*. [online] Available at: <http://www.engadget.com/2014/01/29/strategy-analytics-2013-smartphone-share/> [Accessed: 31 Mar 2014].
22. Project!, W. 2014. *Android Source*. [online] Available at: <https://source.android.com/> [Accessed: 31 Mar 2014].
23. Appleinsider.com. 2014. *Google announces 900 million Android activations, 48 billion app installs*. [online] Available at: <http://appleinsider.com/articles/13/05/15/google-announces-900-million-android-activations-48-billion-app-installs> [Accessed: 31 Mar 2014].

24. OpenSignal. 2014. *Fragmentation Report*. [online] Available at: <http://opensignal.com/reports/fragmentation-2013/fragmentation-2013.pdf> [Accessed: : 9th November 2014].
25. *HTC Sensation XE*, Available at: http://www.gsmarena.com/htc_sensation_xe-4164.php (Accessed: 9th November 2014).
26. *Samsung Galaxy Note 8.0*, Available at:http://www.gsmarena.com/samsung_galaxy_note_8_0-5252.php (Accessed: : 9th November 2014).
27. *MicroVision ShowWX +* , Available at:<http://www.pcmag.com/article2/0%2c2817%2c2387839%2c00.asp?tab=Specs>(Accessed: : 9th November 2014).
28. *Tegra Android Development Pack*, Available at: <https://developer.nvidia.com/tegra-android-development-pack> (Accessed: : 9th November 2014).
29. *Activity*, Available at:<http://developer.android.com/reference/android/app/Activity.html> (Accessed: 19th November 2014).
30. *Reading and Writing Logs*, Available at:<http://developer.android.com/tools/debugging/debugging-log.html> (Accessed: 19th November 2014).
31. *What color is it?*, Available at: <https://play.google.com/store/apps/details?id=air.com.nicotroia.whatcoloristhis&hl=en> (Accessed: 12 April 2015).
32. *ColorId*, Available at: <https://play.google.com/store/apps/details?id=com.hempton.colorid&hl=en> (Accessed: 12 April 2015).
33. *Color Blind Vision*, Available at: https://play.google.com/store/apps/details?id=com.givewaygames.colorblind_ads (Accessed: 12 April 2015).
34. *Color Detector*, Available at: <https://play.google.com/store/apps/details?id=com.mobialia.colordetector> (Accessed: 12th April 2015).
35. *Color Blindness Simulate Correct*, Available at:<https://play.google.com/store/apps/details?id=com.SeewaldSolutions.ColorBlindnessSimulator> (Accessed: 12th April 2015).