AALBORG UNIVERSITY COPENHAGEN

Camera Companion

Semester Project, **MED9**

Hristo Dimitrov

hdimit11@student.aau.dk

Supervisor: **Paolo Burelli**

Table of Contents

1. Introduction.

In our present digital age, “media innovation” is an abstract term which can vary greatly. With the rapid development of smartphones, SSD hard drives and powerful yet energy efficient processors and graphic chips, demand for portability grows higher by the day. The contemporary mobile phones are far more sophisticated than their analogues from a few years back and are able to perform many different operations. One of them is the ability to take pictures. There are many different mobile applications which address the camera capability by either applying some filters and effects to the already taken picture or entirely augment the surrounding environment through the use of 3D objects or any other means. Such programmes are for instance: Cartoon Camera, Zoom Camera, After Focus, Sensor Camera, Lomo Camera, Camera Pro, etc. Handling their mobile phone as a camera “on-the-go”, the use of the *Rules of Photography* (DPReview, 2012) is usually not something that the casual user considers when taking a picture. There are many different rules by applying of which more aesthetically pleasant picture can be achieved. Such rules as mentioned are *Rule of Thirds* which virtually divides the image plane into 9 squares by intersecting 2 horizontal and 2 vertical lines. The points which are formed by the cross-sections are called “point of interest” and serve as guidelines in composing the prominent objects in the picture around them. Another rule of photography called *Visual Balance* states that there should be an equal distribution of salient objects around the image plane. The use of *Complementary Colours* is also a rule to achieve better looking picture. It states that the use of contrast colours within a picture contributes to its overall aesthetical value. Contrast colours are colours which “face” each other on the colour wheel (Color, 2012). The present paper will explain the beginning of the development of an application whose final goal is **through the use of the rules of photography to aid people in taking better pictures with their Android smartphones in terms of composition and visual aesthetics**. One of the rules of photography states that complementary colours in a picture contribute to its overall aesthetics. But what if you were a colourblind person? With this thought in mind, in its present state, the application addresses people who are suffering from colour vision deficiency by allowing them to select an object of their choice and read its colour.

2. Related Work.

The Android is an open source operating system for mobile hardware such as smartphones, tablets, and most recently – digital cameras (Samsung, 2012). Android core was designed to be portable and to be able to run on different devices. Although it is a relatively new operating system on the market, it is a “game changer” (Gargenta, 2011) and enjoys the “higher percentage sales growth rate in the mobile industry” (Jordan & Greyling, 2011). There are a lot of programmes for Android which deal with the phone’s camera:

Cartoon Camera for instance is using techniques to manipulate pictures in real time through various effects.
Hachune Camera positions a Hachune anime character in the frame by using augmented reality techniques and taking into account the phone’s sensors.

Zoom Camera is also a programme which works real-time. It gives one the freedom to change the scene mode (portrait, landscape, party, etc.) and this way, to make a visually pleasant picture. *Sensor Camera* is probably the best application on the market which addresses frame composition by considering some of the rules of photography. Sensor Camera applies the Rule of Thirds to automatically adjust the horizon line.
After Focus is a quite interesting application which allows one to manually select the focus plane in a picture which is has already been taken.
Camera Pro gives one the freedom of a real dSLR camera by letting one change the ISO settings of the phone's camera as well as showing the on-screen grid. *Pudding Camera* is an application equipped with 9 high quality virtual filters which can be applied after the picture has been taken.
Lomo Camera is quite similar, but has 10 different cameras and 12 filter effects in total. *CameraMX* is a real-time picture manipulation tool where one can use photo effects and different filters.
Camera FV5 is a whole dSLR simulator which gives one the ability to choose between ISO, EV, Flash, Focus, bracketing and continuous shooting.

There are different methods in photography to accomplish a successful picture. According to the highly rated and acclaimed website DPReview (DPReview, 2012), *The Rule of Thirds* is probably the most popular technique and "the idea is that significant compositional elements be placed along imaginary lines that break the image into thirds, both horizontally and vertically" (thomaspark, 2012). Another technique is the *Visual Balance* which aims at accomplishing a sense of balance throughout the whole picture. The *Golden Ratio* comes from Greek art, where the ancient Greeks considered the proportions achieved through that rule to add aesthetical value and be pleasant to the eye. The rule is somehow similar to the *Rule of Thirds* in a sense that it also serves as a guideline for dividing the picture into segments. *Leading lines* is another rule which helps making a simple picture more dynamic by leading the eye away from one element to another and by creating a sense of movement. Through the use of complementary colours (pairs of colours that are of "opposite" hue in some colour model (Wikipedia)), we can achieve contrast and communicate different moods in a picture (Clark, 2011).

One of the most popular techniques, *Rule of Thirds*, is used to create a method for unsupervised automation in a dSLR (Banerjee & Evans, 2004). In their paper, they present low-complexity methods for dSLR to segment the main subject and apply the rule of thirds. The segmentation is made by placing the main object in focus and blurring the rest of the picture where the second method moves the centroid of the main subject to the closest of the four rule of thirds locations.

Another computational approach was taken by (Datta, Joshi, Li, & Wang, 2006) in which the team focuses on aesthetics in photography and builds a classifier that can distinguish between pictures with high and low aesthetical value from a website called Photo.net and can also build a regression model that can predict the aesthetics score. They argue that the classification model is considered to be more appropriate than regression by stating that the measurements are "highly subjective” and there are no rating standards. So an absolute score can be meaningless. They consider using a regression model to help achieving “ideal” case in which the machine can give an image a score in a range between 1.0 and 7.0.

Somehow similar methods are proposed by (Gooch, Reinhard, Moulding, & Shirley, 2001) and (Liu, Chen, Wolf, & Cohen-Or, 2010). The first method is a proof of concept and uses a 3D space while in the second they not only evaluate the aesthetics of a picture but also try to optimise its appeal by taking into account RT (rule of thirds), DA (diagonal), VB (visual balance) and SZ (region size) and applying a “crop-and-retarget” operator which basically modifies the composition by cropping the image. They have presented some very interesting algorithms to calculate the different rules they employ as well as a combined aesthetics score function.

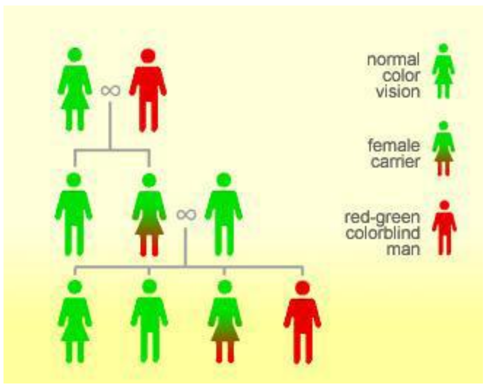
In another recent work which deals with visual aesthetics (Bhattacharya, Sakthankar, & Shah, 2010), the team presented an interactive application for photo assessment which also serves as a learning tool providing the user with feedback on how to compose her picture. The user selects a foreground and based on that selection the programme suggests a possible aesthetically pleasant frame composition.

An autonomous robot photographer (Byers, Dixon, Smart, & Grimm, 2004) was developed in order to take “well composed” photographs of people at a social event. The robot, called Lewis, is able to navigate through the environment and locate subjects by using methods such as face recognition and edge detection. They have applied the strategy of isolating a skin colour tone in order to map specific regions as potential faces. With different rules provided, the robot knows that a photograph can be taken if the distance between the camera and the subject is no less than 7 feet. When there is a picture opportunity, the robot navigates through the environment calculating the best possible route and angle to take the shot. If there are some obstacles in the way, it chooses the closest possible route. The team found out that a random navigation mode that allowed the robot to wander around taking pictures was better for a crowded environment where by calculating the best possible route, the robot spent too much time trying to avoid people and takes fewer photos. By specifying different rules (such as rule of thirds, empty space, no middle and edge) the team provided the robot with guidelines of what kind of picture to take.

By briefly summarising the different computational approaches from the world of photography, it is apparent that 1) the majority of mobile applications use methods to alter the appearance of an already taken image and 2) most people use the *Rule of Thirds* as a major guideline in creating their algorithms. *Rule of Thirds* along with a *Visual Balance* are used both for real-time compositional methods and as picture post-processing tools. For example, the approach (Datta, Joshi, Li, & Wang, 2006) have proposed can be successfully implemented in a digital camera (or smartphone) and can be quite beneficial for photographers (especially amateur photographers) to rate their own photos “on the go”. Although *Camera Companion* is not meant to be implemented on a dSLR, the algorithms and segmentation methods proposed (Banerjee & Evans, 2004) as well as the methods to calculate the “RT, DA and VB” (Liu, Chen, Wolf, & Cohen-Or, 2010) can still be of great help. Unfortunately none of these approaches considers colour as a valid guideline not only for achieving an aesthetically pleasant photograph, but also as a way to compose a picture.

For the majority of people, the ability to recognise colours is considered quite normal simply because they do not suffer from any colour vision deficiencies.

Colour blindness (also referred as colour vision deficiency) however is “a condition in which certain colours cannot be distinguished, and is most commonly due to an inherited condition” (Cooper, Demchak, & Burton, 2007). In their review, they explain that red-green colour blindness is the most common form (99%) where 75% of the people having this kind of deficiency are unable to see the green and 24% - the red. Blue/yellow colour blindness as well as total colour blindness are quite rare. The condition itself is due to the lack of one (or more) of the colour cone cells inside the human eye. A person with such underlying condition has difficulties distinguishing between certain hues simply because *“if you are suffering from a color vision deficiency you perceive a narrower color spectrum compared to somebody with normal color vision”* (Flueck, Color Blind Essentials, 2010).
In his book, *Color Blind Essentials,* Flueck explains the origin of colour blindness which dates back to 1793 when John Dalton (who himself had red/green colour blindness) wrote a scientific paper about colour blindness claiming that there was a colour liquid inside the human eye responsible for the perception of different colours. That was proven wrong after his death and later Thomas Young and Hermann von Helmholtz were the first people to describe the trichromatic colour vision (RGB). Flueck explains that 8% of all men and 0.5% of all women are colourblind which leads to the conclusion that women are mostly carriers. He also explains that the eye’s colour perception depends on three different types of cones where “each cone is sensitive to a certain wavelength of light (red, green, blue)”. A very interesting colour blindness inheritance pattern is also presented in his book:



Red-green colour blindness inheritance pattern

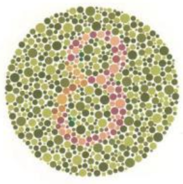
By presenting a list of different type of colour blindness he summarises:

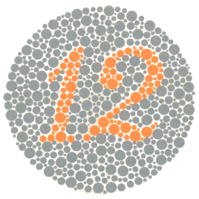
Monochromatism: Either no cones available or just one type of them.
Dichromatism: Only two different cone types. Third one is missing completely.
Anomalous trichromatism: All three types but with shifted peaks of sensitivity for one of them, which results in a smaller colour spectrum.

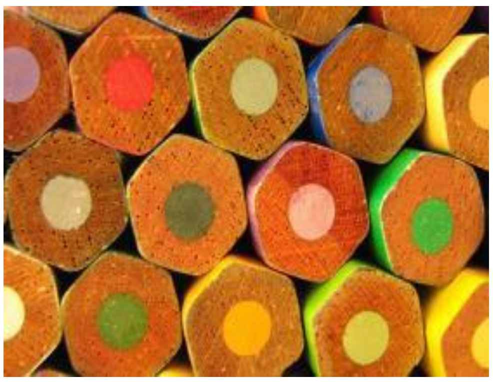
Where dichromats and anomalous trichromats exist again in three different types according to the missing cone:

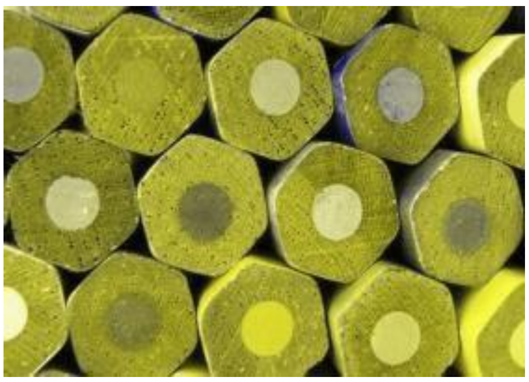
Tritanopia: Missing short wavelength light S-cone (blue)
Deuteranopia: Missing medium wavelength light M-cone (green)
Protanopia: Missing long wavelength light L-cone (red)

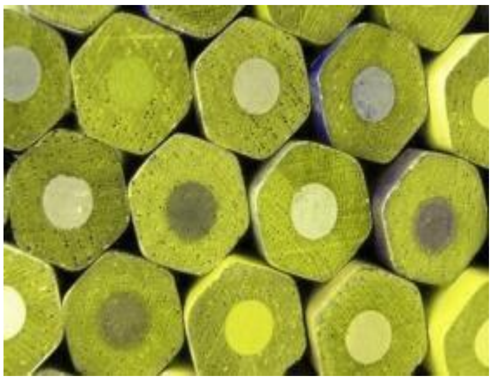
Flueck claims that people consider that if one suffers from red-green or blue-yellow colour blindness, these are the only colours which one cannot see and explains that this is wrong because "Colour blindness doesn't relate to just two colour hues you can't distinguish, it is the whole colour spectrum which is affected".
Later in his book he presents the different tests available for detecting and measuring colour blindness such as the *Ishihara plate test* which according to him is "not the best and definitely not the most current one", but still a tool widely used (Birch, 1997) as well as a picture comparison of how exactly colourblind people see the surrounding world.

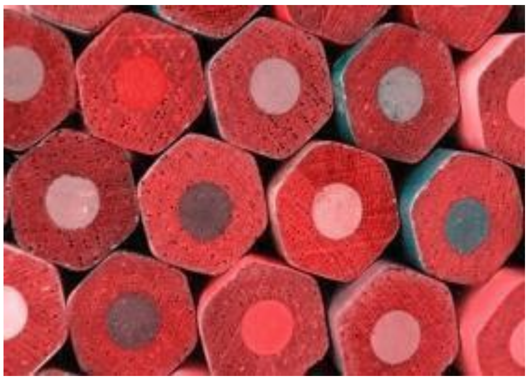

A person having red-green CVD should see "3"


This is recognised by everyone


Normal colour vision


Protanopia


Deuteranopia


Tritanopia

We can clearly see that if a person suffers from red-green colour blindness (Protanopia/Deuteranopia) not only the specific colours are not recognised, but also the whole spectrum that includes reds or greens is altered.

In a study (Douglas & Kirkpatrick, 1996), the different colour models (RGB / HSV) have been closely investigated. The team concluded that RGB is a faster model to use in hardware but there was not much difference between these two models.

In another study a mobile solution for aiding colourblind people is described (Petzel). Driven by their hypothesis that "a mobile application will decrease, if not completely eliminate, misinterpretations of red and green colours", the team has developed a mobile application for Android mobile platform which processes regions of colour using the RGB model. For their test, they have used a paper full of red and green stripes and have asked red-green colourblind people (9 male and 1 female) to try and differentiate between the colours on the paper using their Android smartphones. The results showed that their application is working well with their current setup and people can successfully classify the different colours.

In direct comparison, (Tian & Yuan, 2010) developed a slightly different method which takes into consideration major colour variables such as an object's illumination, rotation and texture. Their goal is by using a computer vision-based technology to develop a method which can propose different clothes matching options to blind or colourblind people. In their study as well as in another quite prominent paper (Huang, Wu, & Chen, 2008), they outlined one of the biggest concerns when it comes to colour recognition in computer-vision – the fact that although people perceive an "object to be of the same colour across a wide range of illumination conditions.....the actual pixel of an object has values that range across the colour spectrum depending on the light conditions" (Tian & Yuan, 2010). They have also presented an interesting way to compute saturation and luminance:

- if the luminance *I* of a pixel is larger than 0.75, and saturation *S<0.25+3(0.25-(1-I)),* then the colour of the pixel is defined as "***white***".
- if the luminance *I* of a pixel is less than 0.25 and saturation *S<0.25+3*(0.25-I),* the colour is "**black"**.
- for the colour "*gray*", the saturation *S* and luminance *I* should meet following conditions: 0.25 ≤ *I* ≤ 0.75 and *S* < *1/5*I*.

Where for other colours, such as *red, orange, yellow, green, cyan, blue, purple and pink,* hue information is employed. They have defined the colour "*red*" between 345°--360° and 0--9°, "*orange*" in the range of 10°--37°, "*yellow*" between 38°--75°, "*green*" between 76°--160°, "*cyan*" between 161°--200°, "*blue*" in the range of 201°--280°, "*purple*" between 281°--315°, and "*pink*" between 316°--344°.
To detect if an image has texture or not, they have firstly converted the picture into gray scale, performed a Gaussian smoothing for noise reduction and then applied a Canny edge detection algorithm.

By exploring the HSV plane, (Huang, Wu, & Chen, 2008) have proposed a fast re-colouring algorithm to improve the perception of people who suffer from colour vision impairment. They review the previous methods for addressing the colourblind people by classifying them into two categories – "1) tools that provide guidelines for designers to avoid ambiguous color combinations, and 2) methods that (semi-)automatically reproduce colors that are suitable for CVD viewers". Their algorithm aims at being fast, able to maintain the luminance and saturation of an object so that it keeps its current contrast and to stretch or compress the distance between the hue values of a colour without altering the hue itself.

In a similar manner, (Kuhn, Oliveira, & Fernandes, 2008) have proposed another type of re-colouring technique for dichromats that "highlights important visual details that would otherwise be unnoticed by these individuals". With their re-colouring algorithm, they have achieved tremendous results for the processed picture by not only altering the hue to an extent which doesn't seem odd, but also by managing to preserve the other colour's gamma spectrum. They claim that their technique can be "efficiently implemented on modern GPUs and both its CPU and GPU versions are significantly faster than previous approaches".

Now let's take a look at some mobile applications which address colourblind people. On his website (Flueck, Colblindor, 2010) we can find some of the top iPhone applications for colourblind people:

HueVue Colorblind Tools: helps you identify, match and coordinate colours by displaying the value of the colour in question.
ColorBlinds Easy: this app has methods to improve the contrast between red and green and to show to others how the world of a colourblind person looks like.
Chromatic Vision Simulator: this is a simulation tool which shows you in real time how people having some sort of colour vision deficiency see the world.
Colorblind Vision: this app simulates the most severe colour vision deficiencies in real-time. *Chromatic Glass:* it divides the colour spectrum into segments so that such colours do not overlap depending on the type of colour deficiency a user suffers from in real-time. *Color Blind Aid:* this app enables people with red-green colour blindness to detect red and green in their environment and pass colour blindness tests in real-time using augmented reality technology.
Colorblind: this is a game that challenges your ability to memorise and create colours. The premise of the game is simple; after being shown a colour, you must reproduce it. *Kolorami:* it analyses colours of a picture and can also use the camera to analyse the colours of in its visual field.
iSpectrum Color Blind Assistant: quickly and easily identifies any colour by name. Simply touch your photo wherever you need to know the colour and shake your device to reveal the RGB (Red/Green/Blue) components of any colour.
Color Reader: allows the user to "touch" camera live video to know the colour of an object. *Say Color:* speaks the name of the nearest colour in the centre of a camera preview pane using speech synthesis.
Colorin: point your camera and take a picture and Colorin will tell you what the most dominant colour for that part of your image is.
Colorblind Avenger: touch anywhere on the already taken image to see the colour of that area. Displays colour names and shades as well as rgb decimal and hex values. *ColorBlind Suite:* finds colours in real-time (augmented-reality) from your iPhone's camera or photos in your library.

Some of these programmes are compatible with Android as well and by reviewing the different applications, we can virtually divide the types of programmes into two different categories - programmes which use the camera in real-time to either show how a colourblind person sees the world, enable the user to click on an object and read its colour, use an augmented reality in a

way that enhances human's colour perception and recognition or programmes which process an image after being taken in order to extract a colour value. The present paper tries to address the first category by letting one manually choose a colour by clicking on the screen in a real-time video image. I consider (Tian & Yuan, 2010)'s approach as the most comprehensive so far because even though (Petzel)'s application is proven to work well with people having red-green colour vision deficiency, it is designed for and tested on a piece of paper in a controlled environment, meaning that no field test has been taken where colour hue, saturation and brightness can vary greatly according to light conditions, object's texture and view angle. Unfortunately the algorithm (Tian & Yuan, 2010) have proposed for computing luminance and saturation didn't work for my application well. I have used a somewhat similar, yet simplified version of their approach giving the "white" and "black" HSV range information only and in a few occasions this method manages to produce better results for the purpose and scope of my programme.
Through the development of Camera Companion I have used both RGB and HSV models in order to try and distinguish which one is more accurate. As such, I can argue with the conclusion (Douglas & Kirkpatrick, 1996) have made simply because throughout my analysis using a HSV model turned out to be much more accurate in real-time colour recognition than the RGB. I also have to agree with the fact that the RGB model gives faster results and uses less computational power than the HSV model.

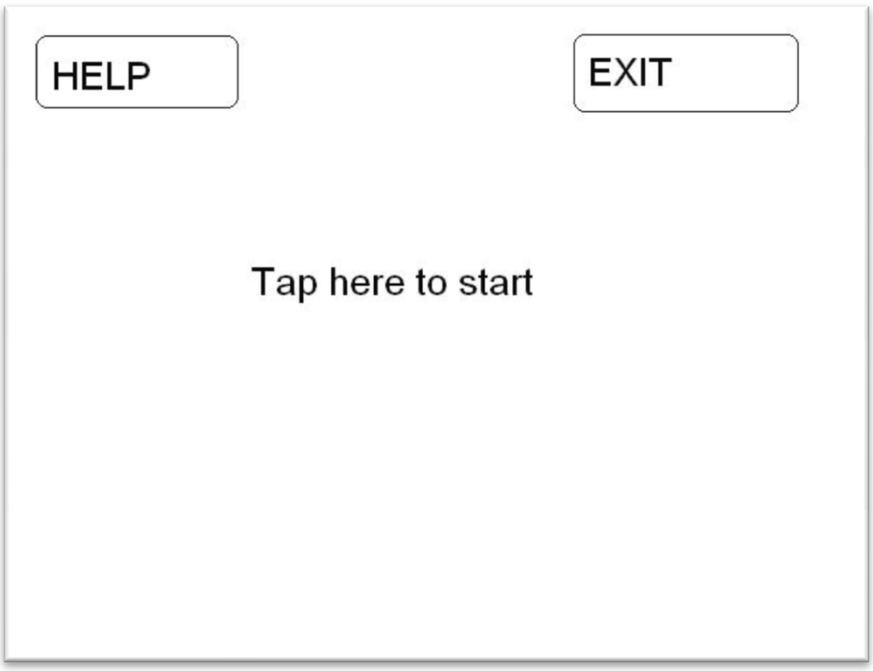
To address the programme's premise, a computer vision technology had to be implemented. With the rapid development of more powerful or portable hardware and the programmes that take advantage of this hardware, the computer vision technology can be vastly used. For example computer vision technology serves as a basis for "Augmented Reality" (AR) which despite still being a new and emerging field, presents many programmes meant to be used by the common user. They can be roughly classified into three types – "games, world browsers and navigation apps. They are usually using the accelerometer and the GPS to obtain location and the physical state of the device" (Sood, 2012). A vivid example is the Android Developer Challenge 2 winner, the game *SpecTrek.* In his book, Soods explains how this technology is used in the *military and law enforcement* (where AR is used in training simulators as well as unmanned vehicles), *vehicles* (windscreens replaced by HD displays, multiple small cameras outside the vehicle which project on onboard displays important information such as compass, road arrows, etc.), *medical* (AR-enabled surgeries), *trial rooms* (user can select different clothing options and sees how they would fit), *tourism* (a head-mounted AR system that displays information about the current site and buildings), *architecture* (machines that displays a virtual structure from a blueprint), *entertainment* (in an amusement park for more realistic experience or KINECT), *education* (act as an add-on to a textbook), *art* (to try out a design before putting it down in ink), *translation* (translate text from multiple languages), *weather forecasting* (the map on which the weather is forecast), *astronomy* (display the location of stars during day).
Another use of computer vision technology is for recognising "regions of similar pixels in an image" (Demaagd, Oliver, Oostendorp, & Scott, 2012) using the "vision-based tracking techniques" (Zhou, Duh, & Billinghurst, 2008) which are essentially face, colour, blob and edge detection.
One way to accomplish that is through the use of *Python* (Solem, 2012) in a combination with *SimpleCV* (Demaagd, Oliver, Oostendorp, & Scott, 2012). In their respective books, the authors

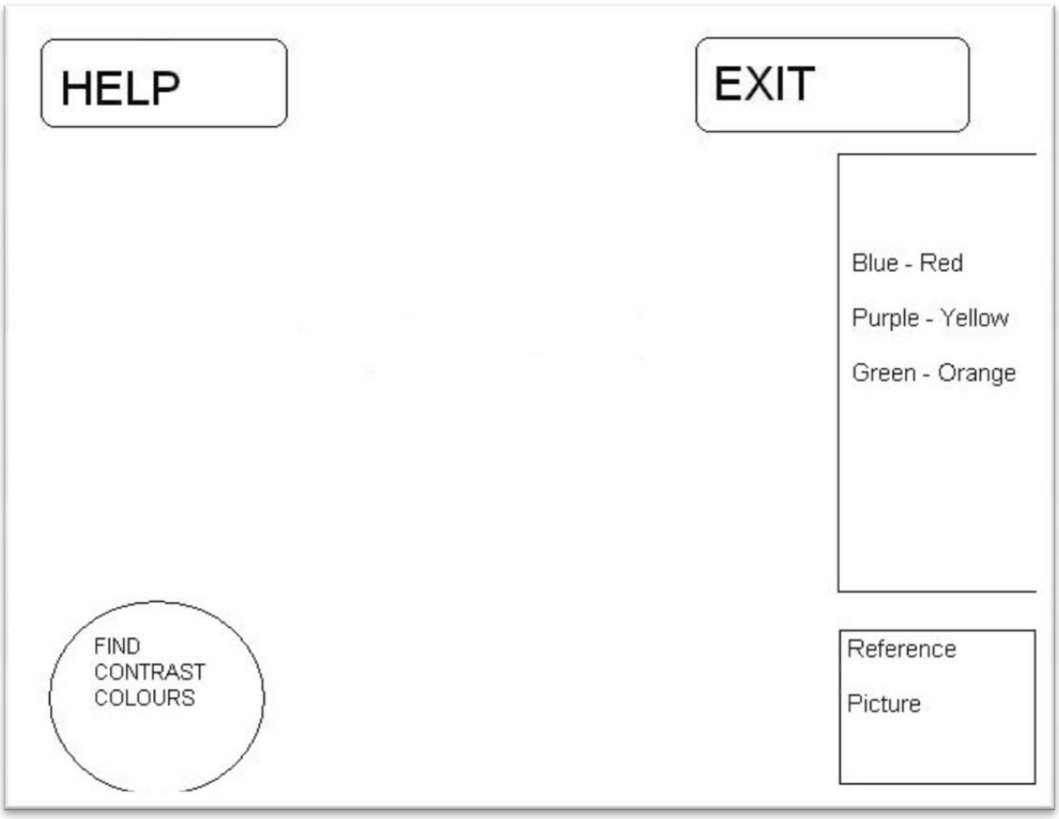
have explained in detail how to install and work with these tools. The other way, which the present paper adopted, is through *Processing* (Reas & Fry, 2007) in a combination with the Android SDK (Android, 2012).

3. Design.

3.1. Project Intention:

A photographic mobile application for Android-based platform was set to be developed using Java, the Android SDK and computer vision technology. The application should be able to suggest possible frame compositional methods taking into account one (or more) of the rules of photography (such as *Rule of Thirds, Visual Balance, Leading lines or Complementary Colours*). The interface of such application should be able to give the user the opportunity to select a colour of her choice and read its name. Also there should be a function allowing the user to quickly find a pair of complementary colours such as red-blue, green-orange, purple-yellow, etc. There should also be a HELP function which will display a HELP message by tapping on the appropriate part of the screen as well as an EXIT function which will quit the application and stop the camera.





Such programme should also be able to track the colour of choice by using colour blob detection or some other type of displaying the colour areas or the whole object. If possible, an "edge detection" function should be implemented in order to find objects' edge which is hypothesised to help the people who are having colour vision deficiency to better distinguish between two objects sharing the same colour or hue range.

3.2. Deciding on the Approach:

For programming such application, several of the major programming languages have taken into consideration – Java, C++, Python. *Python* is a relatively easy to study language and with a combination with SimpleCV makes for a powerful tool to use for some recognition method. SimpleCV has some great algorithms for colour and blob recognition in a picture. One can even specify the desired colour to track by simply inputting its name (for example Color.BLUE). After a few trials and errors, however, it was understood that *Python* won't be the most appropriate tool to use when developing an Android application, so that option was abandoned. Java language was selected instead, because it is a well established language as well as being the language used by programmers to write applications for Android.

3.3. Colour Blindness:

During the research and development of the application it became clear that for the programme to have all the desired functions implemented (such as a working algorithm for applying the *Rule of Thirds* or *Visual Balance*) and eventually, to successfully meet the requirements, it needed more time and better programming skills than initially anticipated. Given that, it was clear that in view of the time limit, the application would not be able to address any particular group of people or help anybody if it were to be carried out in the same manner.

As a visual person, having a keen interest in photography, every time when I go outside to take pictures, one of the first things I pay attention to are the surrounding colours. I always look for contrast colours which will contribute to the future picture either by giving it a specific feeling and some interesting point of view or by having an influence on my decision of how to compose the shot.

The final application should be able to aid people in making a better image composition with their Android smartphones by the use of different rules of photography. In order for the programme to be meaningful at every stage of its development however, I decided to stick with colours and try and target the people who are unable to recognise specific colours. By this means I have set the basis as well as the goal of the featured study.

4. Implementation.

4.1. The Programming Environment:

As already mentioned, Java was used as a programming language with *Eclipse* (Foundation, 2012) and *Processing* (Reas & Fry, 2007) as programming environments. *Eclipse* has some helpful methods which are put to use when one writes a programme such as line-by-line correction and suggestions. *Eclipse* is also preferred by the programming communities. Choosing the correct version of Eclipse turned out to be very important simply because some versions (such as Helios) are not supported by Android (Felker & Dobbs, 2011). *Eclipse Juno* was installed along with the Android SKD.

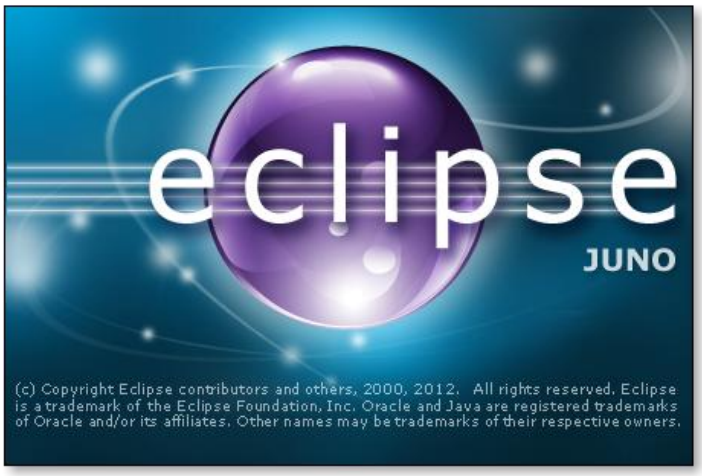

eclipse
JUNO
(c) Copyright Eclipse contributors and others, 2000, 2012. All rights reserved. Eclipse is a trademark of the Eclipse Foundation, Inc. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Fig.1: *Eclipse loading screen*

Processing was also simultaneously installed mainly due to its user-friendly interface, some easy to use programming methods and a few recommendations from different friends. During the development process I have caught myself "reaching" for *Processing* more often than for *Eclipse.*

This was mainly due to the fact that as a person with no previous programming knowledge, *Eclipse* as well as the Java language can turn out to be quite complex after a certain point. *Processing* on the other hand had offered me a simplified version of the Java language as well as an opportunity to write an Android programme that I can instantly try on my phone by simply plugging in the phone to my computer and selecting "Run on a Device". With its overall ease of use and the ability to switch between Java mode and Android mode, *Processing* became my final choice.

Processing
An open project initiated by Ben Fry and Casey Reas
© 2004-2011 Ben Fry and Casey Reas
© 2001-2004 Massachusetts Institute of Technology

Fig. 2: *Processing loading screen*

4.1.1. Processing and Coding.

The structure of the programming code in *Processing* is quite simple – in the very beginning we import a specific library which we are going to use. This can be either some of the build-in *Processing* libraries or an external one. Depending on the operating system and the version of *Processing* (last stable release – 1.5.1), the external libraries should be put in the *Processing* folder "libraries" in *My Documents* or inside the *Processing* folder where the programme is installed. Next, we define our global variables. In "*setup()*" function we define the whole setup such as (for video) the video window size, the capture device, the text font etc. In the "*draw()*" function we write our code which is going to be "drawn" on-screen.

In general, the present code uses a HSB mode to calculate, recognise and track colours:

colorMode(HSB, 360, 100, 100);

After some extensive experimenting, the HSB mode was chosen due to its ability to provide better results when it comes to ambient light variations which have impact on the perception of colours.

Learning Processing (Shiffman, 2008) is a comprehensive book with a lot of great examples. For my code I have used the colour recognition method and first started tracking for the red colour:

```
trackColor = color(360, 100, 100);
```

Afterwards I implemented a revised version of the proposed looping function which goes through each pixel and using Euclidean distance, it compares the current colour with the tracked colour:

```
int closestX = 0;
 int closestY = 0;

 for (int x = 0; x < video.width; x ++ ) {
   for (int y = 0; y < video.height; y ++ ) {
     int loc = x + y*video.width;
     color currentColor = video.pixels[loc];
     float h1 = hue(currentColor);
     float s1 = saturation(currentColor);
     float v1 = brightness(currentColor);
     float h2 = hue(trackColor);
     float s2 = saturation(trackColor);
     float v2 = brightness(trackColor);

     float d = dist(h1,s1,v1,h2,s2,v2);
```

So far, the programme uses the web camera and gives the user the ability to track a selected colour within the frame. The colour value is saved and tracked in-between different frames so the user knows all the time if the selected colour is present within the current frame. For displaying the colour value as a name, I have used "*text*" where in a similar to (Tian & Yuan, 2010)'s manner, a hue, saturation and brightness information for each colour was employed. This way I have specified the following ranges:

Black*: h < 20 && s < .25 && v < .20*
White*: h < 20 && s < .25 && v > .90*
Gray*: h < 20 && s < .25 && v > .50 && v < .80*
Red*: h < 20 && s > .90 && v > .25*
Yellow*: h > 40 && h < 80 && s > .90 && v > .25*
Green*: h > 100 && h < 140 && s > .90 && v > .25*
Cyan*: h > 160 && h < 200 && s > .90 && v > .25*
Blue*: h > 220 && h < 260 && s > .90 && v > .25*
Magenta*: h > 280 && h < 320 && s > .90 && v > .25*

Where *h* = ***hue***, *s* = ***saturation*** and *v* = ***brightness***

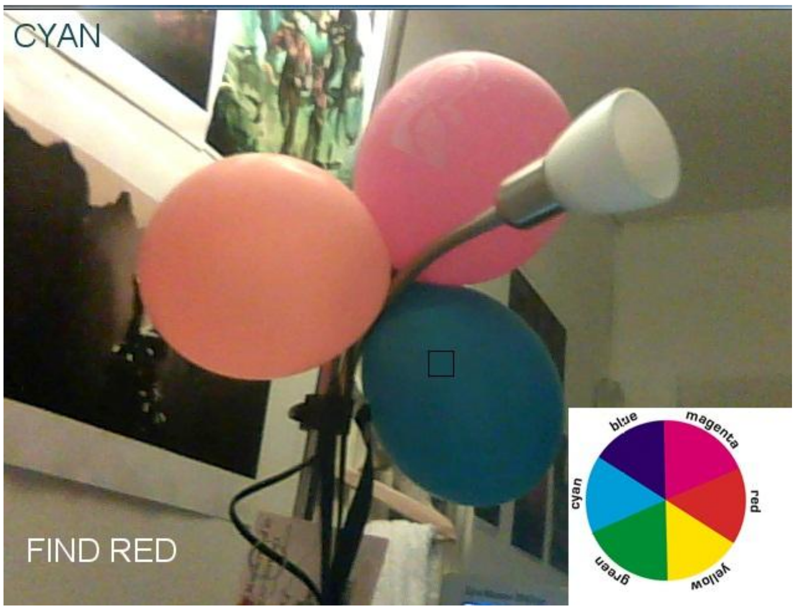


Fig. 3: *Tracking a specific colour*

The colour name remains on the screen for 5 seconds specified by:

```
final int DISPLAY_DURATION = 5000;
```

These colour value ranges were taken from a table which consists of the RGB hexadecimal as well as percentage code, HSB and CMYK codes for each of the sixteen colours (December, 2012).

Since the programme is targeting colourblind people, an option which instantly finds a specific colour had to be implemented. Red colour was chosen due to the fact that (1) the most common form of colour vision deficiency is the red-green CVD (Cooper, Demchak, & Burton, 2007), (Flueck, Color Blind Essentials, 2010) and (2) another function which quickly finds red-blue pair as contrast colours was into consideration. Using the guidelines from the *Processing*'s website, I have incorporated a function which tracks for the red colour only, taking into account the pixel's hue value:

```
int redX = 0;
    int redY = 0;
    float redHue = 0;
    video.loadPixels();
    int index = 0;
    for (int y = 0; y < video.height; y++) {
     for (int x = 0; x < video.width; x++) {
       int pixelValue = video.pixels[index];
             float pixelHue = hue(pixelValue);
       if (pixelHue > redHue) {
         redHue = pixelHue;
         redY = y;
         redX = x;
       }
       index++;
```

Unfortunately no function which quickly finds the red-blue pair was able to be carried out in time. A button was also programmed by the click of which the function began tracking for the red colour and a circle around the object containing red was formed.

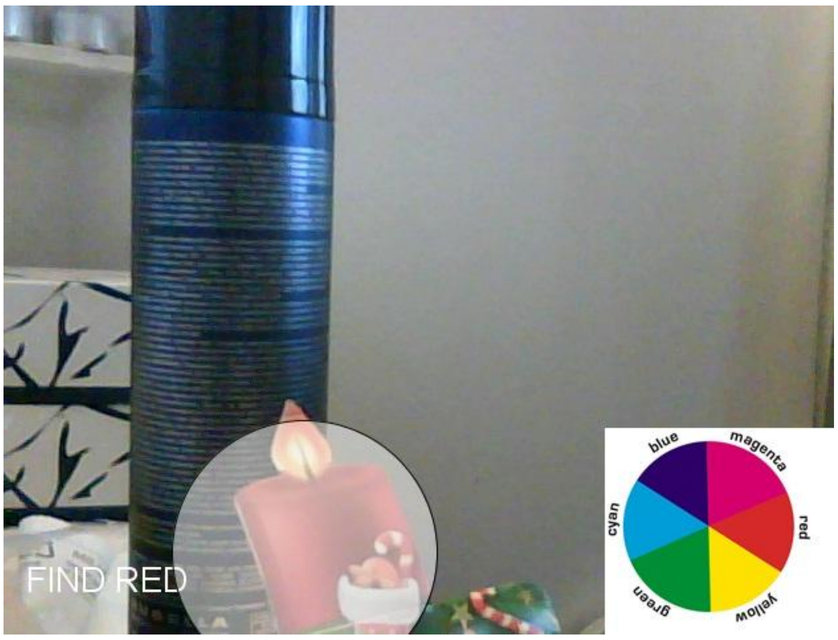


Fig. 4: *Tracking only for red colour*

4.1.2. User Interface

As previously mentioned, *Camera Companion* is an application developed both for PC and Android. The PC interface consists of a reference picture of complementary colours on the right bottom corner of the window as well as a “FIND RED” button which triggers the corresponding function (seen above). Although the code used for the Android version is essentially the same, a few changes had to be made in order to work on a mobile phone. A new user interface was implemented with one extra title screen, so that it has a better accessibility and more comprehensive visual feedback.

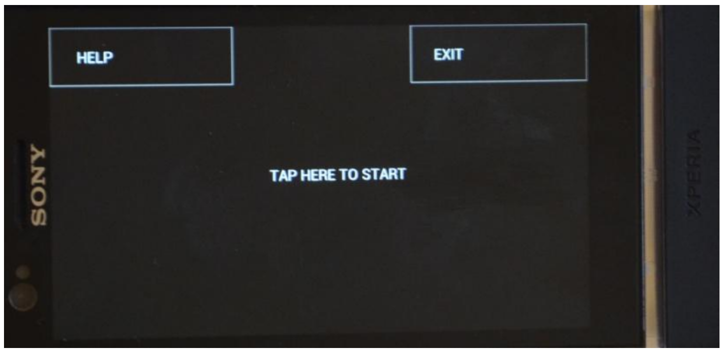
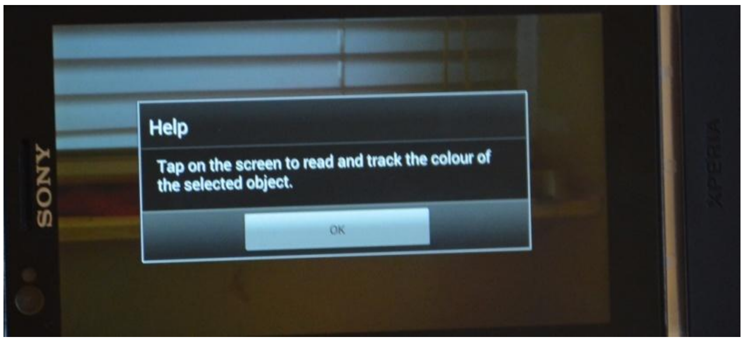


Fig. 5: *User interface for Android*

By selecting "HELP" the user is introduced with how the programme works and what is he/she expected to do whereas "EXIT" quits the application.



Tap on the screen to read and track the colour of the selected object.

Fig. 6: *User interface for Android*

For the implementation of a title screen, an external library was used (Sauter, 2012) and the code has to be modified to include the new library's camera and UI functions:

```
import ketai.camera.*;
import ketai.ui.*;
...
cam = new KetaiCamera(this, width, height, 24);
void draw(){
...
drawUI();
...
void mouseClicked(){
 if(mouseY < 100){
 if (mouseX > width/3)
   exit();
   else if (mouseX < width/3)
   KetaiAlertDialog.popup(this, "Help", "Tap on the screen to read
and track the colour of the selected object.");
 }

.....

void drawUI(){
 pushStyle();
 textAlign(LEFT);
 fill(0);
 stroke(255);
 rect(0, 0, width/3, 100);
 rect((width/3)*2, 0, width/3, 100);
 fill(255);
```

```
  text("     HELP", 5, 60);
  text("     EXIT", width/3*2, 60);
  popStyle();
}
```

The library has a lot of different versions so after several trials and errors it became clear that the current version of use (version 8) renders the camera not working and for Android 4.0. version 6 had to be used.

5. Testing Procedure.

Since this was meant to be a mobile application, the initial idea for the test procedure was for people to be able to test it on an Android mobile phone. With the code written on a desktop PC, no lagging, stuttering or any other abnormality was observed. This fact was taken for granted and with some minor tweaks the code was successfully translated to work on an Android platform. That however turned out to be quite a problem because the mobile phone (SONY Xperia P) simply couldn't calculate the code fast enough rendering the whole application working obnoxiously slow. While this was an unfortunate turn of events, a few different optimisation methods were tried in order to speed up the application. The only method which made a minor difference was to use RGB instead of HSB mode. It was finally decided to test on a PC equipped with a web camera instead. The testing PC was a DELL D420 notebook running on a single core Intel Centrino 1.2Ghz CPU, with a 2GB DDR2 RAM and Logitech C210 SD Webcam. Two persons however were instructed about the cons the Android version has and were sent the .apk for personal use on their devices (SAMSUNG GALAXY SIII and SAMSUNG GALAXY Tab). The Android version differs from the PC by using an RGB colour mode instead of HSB. It is also a bit more simplified in order to try and boost up the application's speed.

The current main goal of this application is to address colourblind people. In order to do that it was essential to find such people who can test it and collect their comments and general feedback. Unfortunately from a tested sample of 20 people only 5 were colourblind. All 5 were red-green CVD males with an average age of 28,6. 3 of them tried the application on the test PC and only 2 – on Android ready devices.
It was decided to inspect how the programme performed in different light conditions. This way the testing procedure was virtually divided into two general setups – 12 of the participants tested the programme in-door where there was a variety of light conditions (such as incandescent light, luminescent light and even night lamp). The rest of the participants tested the application outdoor in order to see how it performs in daylight. A brief tutorial of how the programme works, what is its premise and a questionnaire was given to everyone. The questionnaire can be found here.

Fig. 7: *Testing*

Before the review of the results a few remarks and comments have to be made.
Logitech C201 is a standard definition camera with no possibility of manual control. As such, after each start, it automatically decides on the environmental light and corrects its exposure accordingly in order to give the user as much as normally exposed picture as it can. It also has a low quality noise filtration (if any), which presents a lot of noise and picture distortion especially in a low light environment. At one point in development the opportunity to use dSLR camera instead of a web camera was into consideration mainly because of its high quality and the possibility to manually set the exposure for a current setup which will remain a constant throughout the whole test. This idea was dropped however because even though smartphone cameras are able to shoot in high definition, they still can't compete with dSLR cameras' sensor abilities and overall picture quality. In that regard, a web camera is somewhat more similar to the cameras used in the mobile devices.

6. Results.

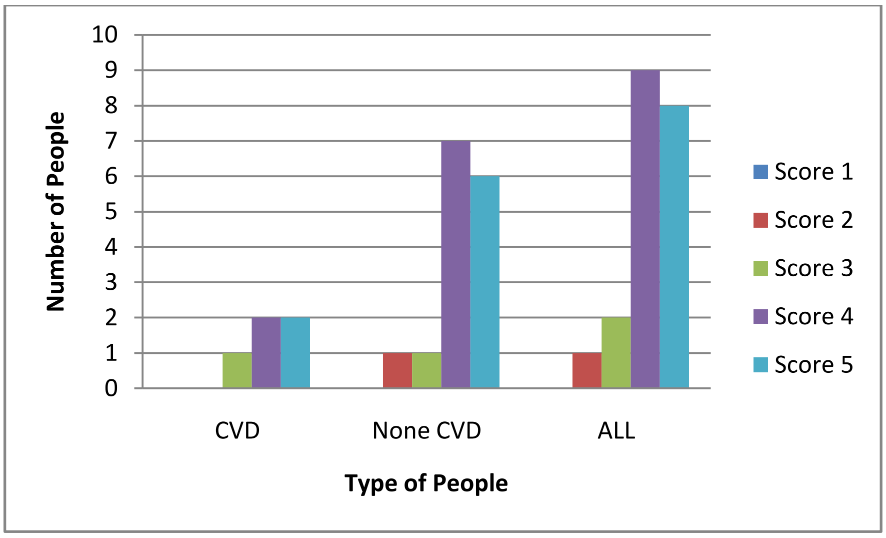
The testing procedure was virtually divided into indoor and outdoor environment, with the idea to inspect how the recognition algorithm performs in different light setups. While performing the test outside however, the weather was rather bad with lots of clouds and rain rendering the whole environment relatively dark, so I couldn't actually inspect what would be the difference between indoor and outdoor colour recognition. Indoor, the programme was tested throughout the university environment where a variety of different light setups could be observed (for instance in the workshop the light is brighter than the one on each floor). The programme was also tested at home in order to have even darker environment illuminated only by a reading light. It was in this study's scope to expect if (and how) the colour would be perceived by the camera's sensor when it is not only illuminated by various lights and general light setups, but also by different types of light. During the testing, it was observed that regardless of the type of light, the programme performed best with a rather scarce amount of light and the overall environmental exposure ranging from -1 to 0 where –

2 is “underexposure”, 0 is “correct exposure” and +2 is “overexposure” (Setzler Jr., 2004). Depending on the extent to which an object was overexposed, the programme had difficulties recognising the correct colour with a few occasions of mapping the “very bright red” as “white”. Since the colour recognition method performed equally the same in the two different environments, a future testing and analysis are in order when there is clear weather and sunshine. A curious thing to note is the response which more than 40% of the none-CVD people gave while performing the test. They either had no knowledge of what “colour blindness” was, thought that the colourblind people could not see any colour or simply hadn’t thought about that particular group of people.

The results from the questionnaire are as follows:

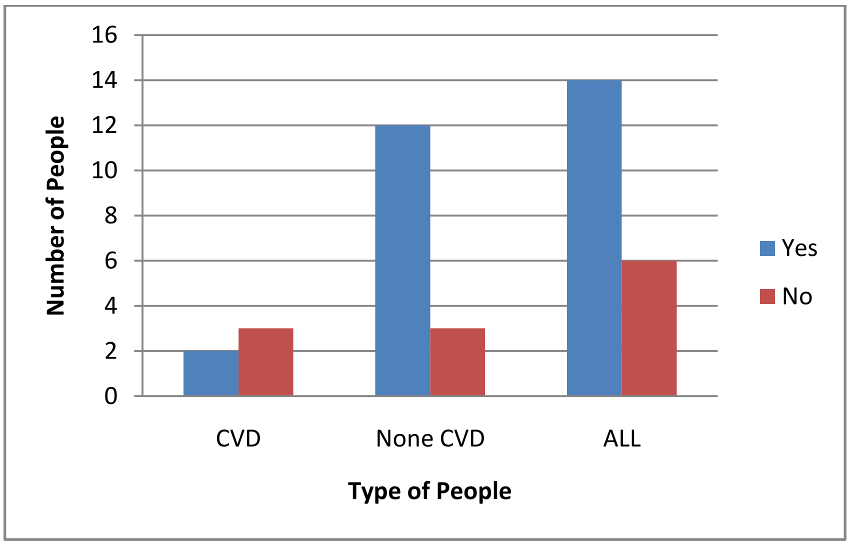
Question: *Would you say that the programme serves its intended purpose?*

On the scale from 1 to 5 (1 being “*Not at All*” and 5 being “*Very much so”)*, the average answer was 4,25. 45% of the participants [9/20] answered the question with a score of 5 and 40% [8/20] with a score of 4.



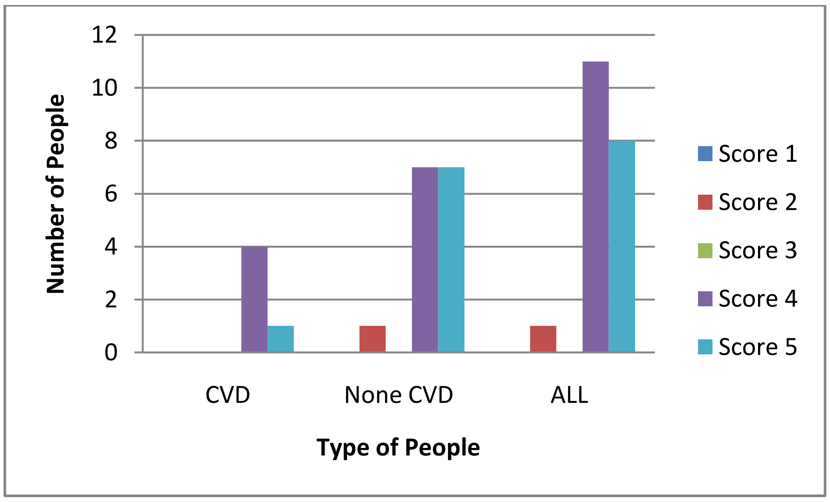
Question: *Do you think the interface gives you enough information?*

35% [7/20] of the participants answered positively and the rest gave a negative answer. Among the CVD people 3 out of 5 found that the interface didn’t give enough information and the other 2 answered that it did.



Question: *Does this app help you pin-point the colours the different objects have?*

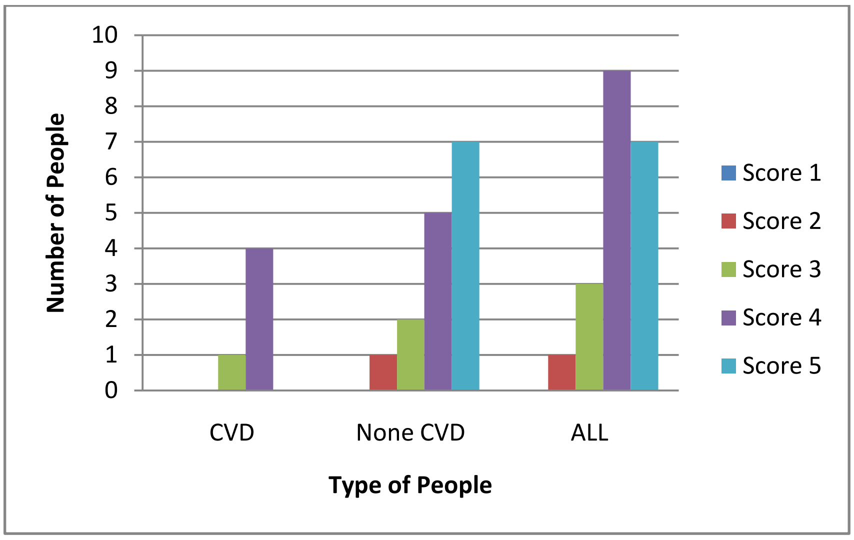
The average answer was 4,3. 55% [11/20] of the participants answered with a score of 4 and 40% - with a score of 5. 1 participant answered with a score of 2. He also commented that certain colours should be distinguished better and that the use of a ring to identify certain colours might not be enough. As a proof of concept though he stated that the programme did a “relatively good job at determining colours”. Out of the people with CVD, 4 answered with a score of 4 and 1 – with a score of 5. One of them commented that it helped him “to recognise the hue of the chosen object in the frame.”



Question: *Using these colour recognition tools, do you think you can build a "colour scheme" for the current frame in your mind?*

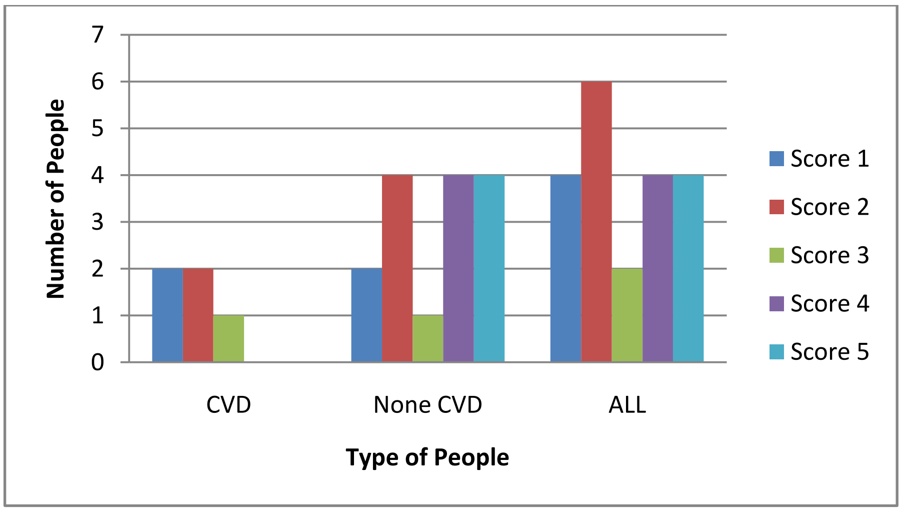
The average answer was 4,1. 45% [9/20] of the participants answered with a score of 4. 35% [7/20] - with a score of 5, 15% [3/20] - with a score of 3 and 1 person answered with a score of 2. Out of the colourblind people, 4 gave a score of 4 and 1 gave a score of 3 stating the fact that some colours were not recognised and also that sometimes he got “two different colours for the same object”.

Another one commented that the "light is central for the colour perception of a surface" so he was a bit "careful about being certain about the colour definition in general".



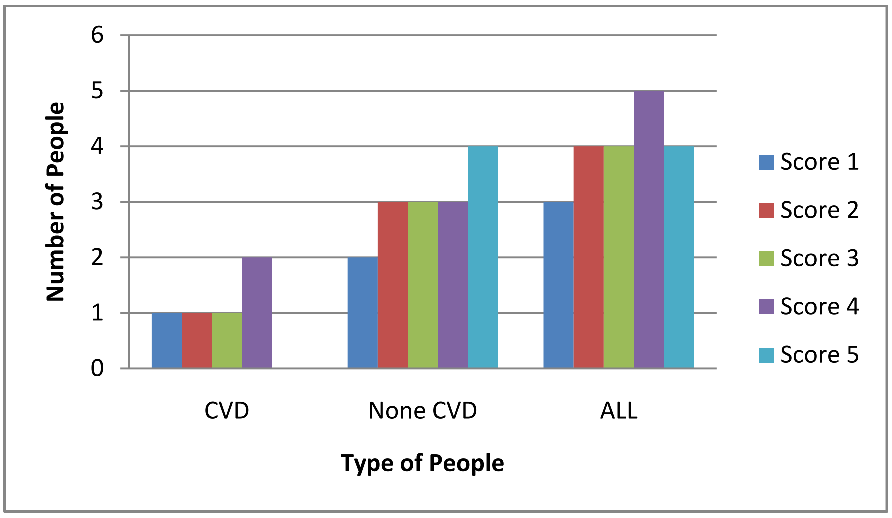
Question: *Do you think you can find complementary colours (such as red - blue) fast enough?*

For this question the average answer was 3, 85 where 30% [6/20] of the participants answered rather negatively giving a score of 2. 20% [4/20] – a score of 1 and another 20% [4/20] – a score of 5. The rest are equally distributed. This feedback however was expected due to the fact that the intended function which would quickly show the red-blue pair was not programmed in time.



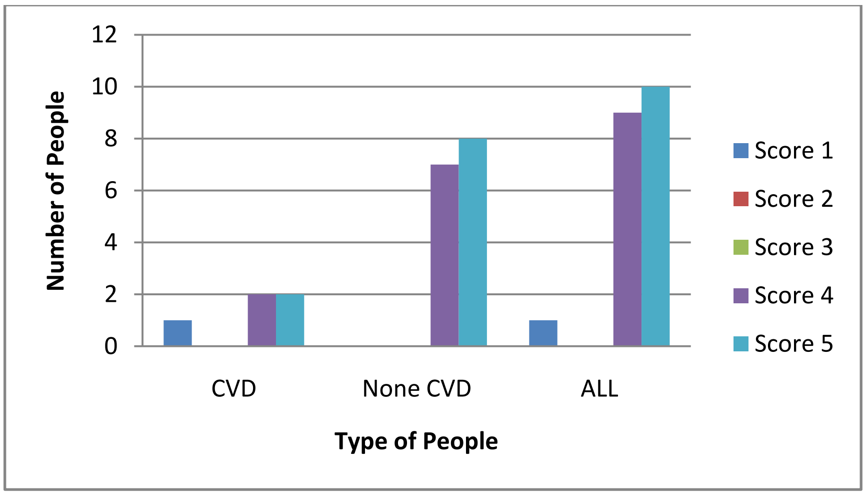
Question: *Do you think such programme can help you in deciding what kind of picture you would like to take?*

This question gave mixed results. The average answer was 3,3. 25% [5/20] of the participants gave a score of 4. Scores of 2, 3 and 5 were each given by 20% [4/20]. 3 participants gave a score of 1. Out of the colourblind people, 2 answered with a score of 4 and the rest with 1, 2 and 3 respectively. In the “comments” section, both participants who rated this question with a 4 commented that if a method to show contrast colours were to be implemented, they could decide which objects to include in the picture.



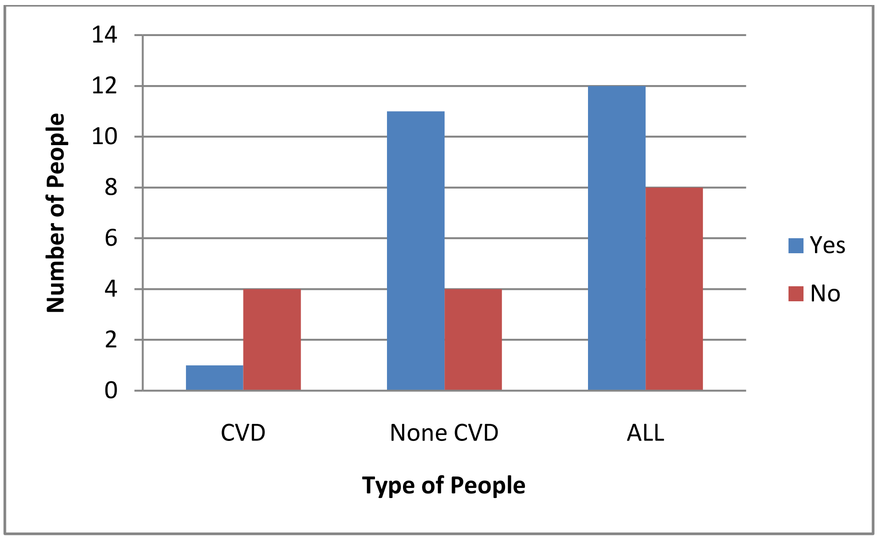
Question: *Is it easier for you to locate the RED colour?*

This question received a rather straightforward answer with an average of 4,4. 55% [11/20] of the participants gave a score of 5 where 40% [8/20] gave a score of 4. Two of the CVD people gave a score of 4, another two – 5 and one answered with a 1. The reason behind that, as seen in the comments, is that the red colour is not shown throughout the whole frame, but only for a given object that has red colour.



Question: *Do you think that "edge detection" can be a useful tool for future frame composition?*

60% [12/20] of the participants found edge detection to be of certain help when it comes to frame composition. Among the comments backing up this answer is that “edge detection can outline object's borders. This way we might see something we've missed before and compose our picture accordingly", it will help in distinguishing between two objects of a similar colour especially if they are overlapping and also can "show a hidden or missed detail". Some of the participants who answered negatively also commented that they had no previous knowledge of *edge detection*. From the CVD people, 1 answered with "Yes" commenting that it would be great if there were a visual reference of where the edge was between two overlapping objects of the same (or similar) colour. The rest answered with "No".



6.1. Results Based Summary.

As brief conclusion from the results we can observe that the majority of participants consider the programme to serve its intended purpose but the people having colour vision deficiency didn't find the interface explicit enough. A generally positive response was recorded from everyone with regards to pin-pointing the colours of the different objects as well as to the opportunity such programme can present to the CVD people to build a "colour scheme" of the existing colours within the current frame with 4 out of the 5 CVD participants giving a positive feedback. The programme however was considered unable to serve as an assistant when it came to finding a pair of complementary colours fast enough. This is definitely to be addressed in the future development. The question which targeted the sphere of photography managed to collect a rather diverse response. Analysing the data, it is unclear if the people had some previous knowledge of photographic composition and do not consider using complementary colours as a valid part of it, or if they have no such knowledge. Had there been another question addressing their photographic knowledge, it could have been easily mapped to their answer. From the comments it became clear that one of the CVD participants was a freelance amateur photographer, but other than that, no other subject claimed to possess such skills. Generally everyone evaluated the red colour as easy to find due to the programme's ability to instantly point to the red object in the frame. Some of the comments however suggested a rather different take on the representation of red colour using either blobs outlining the whole object or general representation of all the objects consisting of red colour within the image frame. That is worth considering and will definitely serve a better purpose than the method featured in the programme. The question about "edge detection" registered quite a controversial response. It was hypothesised that by the use of an edge detection algorithm, the CVD people would be aided in recognising either two different objects of the same colour (or at least the same hue range) which are somehow overlapping or be able to find a detail they have previously missed. The recognition of two objects of similar colour would have been made possible through the clear contour which an edge detection function would draw around the object's edges. This would eventually contribute to the frame composition by giving people more information about the surroundings than they had before. On the contrary, 4 out of 5 CVD participants didn't consider such function of any help. One of them didn't know if edge detection would help and the rest simply didn't consider it helpful. The none-CVD persons on the other hand thought that such function would turn out to be helpful.

7. Conclusion.

An application which aims at aiding people in making more aesthetically pleasant pictures through their Android smartphones was set to be developed. In its current stage of development, the application targets colourblind people by helping them register the different colours in the image plane.Although this is a work-in-progress product, it appears that the majority of none-CVD people along with the participants having colour vision deficiency considered such programme to be able to successfully relay information about surrounding colours. They had however a rather negative attitude towards its inability to quickly show complementary colours within the image's frame due to the fact that although such method was considered, it was never fully programmed. Had such function been implemented, a more comprehensive statistics could have been drawn, mapping the participants' response to the question targeting frame composition in photography. Based on their comments, a list of key areas of future development was composed and will be examined in the next section of the report.

8. Discussion and Future Work.

During the creation of this programme, a lot of obstacles were met and a lot of compromises were made. There were some major difficulties with the *adb* server being "down" all the time, the processing libraries were not recognised correctly, the phone Camera needed a manual set of permissions, etc. I should note that there is significant room for future development of the existing code as well as implementing new functions. First and foremost, a function which instantly shows pairs of complementary colour is set to be developed. This decision is backed up by the participants' ratings and comments as well as by the fact that this is part of the programme's premise – to aid colourblind people in finding complementary colours. A function that quickly finds the blue colour could not be carried out in time, but one way to do so is in a similar to "FIND RED" function manner, this time by specifying the hue range of the blue:

```
float redHue = 0;
if (pixelHue > redHue){
redHue = pixelHue;
}
```

Could be altered to:

```
float blueHue = 0;
if (pixelHue > blueHue && pixelHue < 180){
blueHue = pixelHue;
}
```

And afterwards implement a function which shows both these values at the same time. Secondly, the code has to be optimised to the extent to which it smoothly operates on an Android mobile phone. This can be done by optimising the loop functions, removing "println" if any, use "short" instead of "int" or "float" when possible, define the size of an array instead of "arraylist", etc. Another module which could benefit from certain modifications is the interface. It needs to be more graphical, more comprehensive and of course, it needs to incorporate all the featured functions. Edge detection algorithm is also into consideration although the given results were not quite positive. The colour recognition method has to be revised by making the colour ranges more precise and the issue of recognising the same colour as two different values has to be resolved. To do that, a high definition web camera would be a better choice to use for the PC version as well as applying an image noise filter. A different method for pin-pointing the selected colour rather than jumping squares has to be developed. This could be done by blob(s) which outline the object(s) having the selected colour within the current frame.
Since some of the other rules of photography address *Rule of Thirds* and *Visual Balance*, an algorithm which is going to calculate the relative distance between an object's centre and one of the four "points of interest" will also be implemented. This can also help with the proper distribution of salient object around the image plane. With the existence of such function, the programme can suggest a possible frame composition by shifting a virtual frame accordingly.

This conceptual application was build in order to outline the framework of the future study. In its current state the programme targets the people having a colour vision deficiency. It tries to aid them in recognising different colours. By this means the programme tries to address one of the rules of photography. Considering the general feedback and outcome of the project, some of its current functions have to be revised and altered and few new functions are set to be developed.

Bibliography

Android. (2012). *Android SDK*. Retrieved from http://developer.android.com/sdk/index.html

AppFoundry. (n.d.). *HueVue*. Retrieved from https://itunes.apple.com/app/huevue-colorblind-tools/id318177578?mt=8

Banerjee, S., & Evans, B. L. (2004). *Unsupervised Automation of Photographic Composition Rules in Digital Still Cameras.*

Bhattacharya, S., Sakthankar, R., & Shah, M. (2010). *A Framework for Photo-Quality Assessment and Enhancement based on Visual Aesthetics.*

Birch, J. (1997). *Efficiency of the Ishihara test for identifying red-green colour deficiency.* Ophthalmic and.

Byers, Z., Dixon, M., Smart, W. D., & Grimm, C. M. (2004). *Say Cheese! Experiences with a Robot Photographer.*

Clark, T. (2011). *Digital Photography Composition.* Wiley.

Color, T. (2012). Retrieved from http://www.tigercolor.com/color-lab/color-theory/color-theory-intro.htm#Color_Wheel

Cooper, E., Demchak, M. A., & Burton, A. (2007). *Facts About Colour Blindness.* Retrieved from www.healthscout.com/ency/1/001002.html

Datta, R., Joshi, D., Li, J., & Wang, J. Z. (2006). *Studying Aesthetics in Photographic Images Using a Computational Approach.*

December, J. (2012). Retrieved from http://www.december.com/html/spec/color16codes.html

Demaagd, K., Oliver, A., Oostendorp, N., & Scott, K. (2012). *Practical Computer Vision with SimpleCV.* O'REILLY.

Douglas, S., & Kirkpatrick, T. (1996). *Do Color Models Really Make Difference?* Proceedings of the SIGCHI.

DPReview. (2012). Retrieved from http://www.dpreview.com/

Felker, D., & Dobbs, J. (2011). *Android Application Development.* WILEY.

Flueck, D. (2010). Retrieved from Colblindor: http://www.colblindor.com/2010/12/13/20-iphone-apps-for-the-color-blind/

Flueck, D. (2010). *Color Blind Essentials.*

Foundation, T. E. (2012). Retrieved from http://www.eclipse.org/

Gargenta, M. (2011). *Learning Android.* O'REILLY.

Gooch, B., Reinhard, E., Moulding, C., & Shirley, P. (2001). *Artistic Composition for Image Creation.*

Huang, J.-B., Wu, S.-Y., & Chen, C.-S. (2008). *Enhancing Color Representation for the Color Vision Impaired.*

Jordan, L., & Greyling, P. (2011). *Practical Android Projects.* APRESS.

Kuhn, G. R., Oliveira, M. M., & Fernandes, L. A. (2008). *An Efficient Naturalness-Preserving Image-Recoloring Method for Dichromats.*

Liu, L., Chen, R., Wolf, L., & Cohen-Or, D. (2010). *Optimizing Photo Composition.*

Petzel, E. *A Mobile Solution to Color Deficiency.*

Reas, C., & Fry, B. (2007). *Processing - A Programming Handbook for Visual Designers and Artists.* The MIT Press.

Samsung. (2012). *Samsung GALAXY Camera*. Retrieved from http://www.samsung.com/in/promotions/galaxycamera/

Sauter, D. (2012). *Rapid Android Development.* Pragmatic Bookshelf.

Setzler Jr., J. M. (2004). *Exposure.*

Shiffman, D. (2008). *Learning Processing - A Beginner's Guide to Programming Images, Animation and Interaction.* Morgan Kaufmann.

Solem, J. E. (2012). *Programming Computer Vision with Python.* O'REILLY.

Sood, R. (2012). *Pro Android Augmented Reality.* APRESS.

thomaspark. (2012). *Digital Photography Review*. Retrieved from http://www.dpreview.com/articles/6426089447/compositional-rules

Tian, Y., & Yuan, S. (2010). *Clothes Matching for Blind and Color Blind People.* New York.

Wikipedia. (n.d.). Retrieved from http://en.wikipedia.org/wiki/Complementary_colors

Zhou, F., Duh, H. B.-L., & Billinghurst, M. (2008). *Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR.*

9. APPENDIX

Questionnaire as presented to the participants:

Age *

Gender *

- [✓ Male]
- Female

Are you a colourblind person?

- [✓ Yes]
- No

If so, what type of colour vision defficiency do you have?

- [✓ Red - Green]
- Blue - Yellow
- Complete
- None

Would you say that the programme serves its intended purpose?

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much so

Do you think the interface gives you enough information?

- Yes
- No

Does this app help you pin-point the colours the different objects have?

1	2	3	4	5

Not at all	○	○	○	○	○	Very much so

Using these colour recognition tools, do you think you can build a "colour scheme" for the current frame in your mind?

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much so

Do you think you can find complementary colours (such as red - blue) fast enough?

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much so

Do you think such programme can help you in deciding what kind of picture you would like to take?

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much so

Is it easier for you to locate the RED colour?

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much so

Do you think that "edge detection" can be a useful tool for future frame composition?

- ○ Yes
- ○ No

If so, can you briefly elaborate on your answer?

Any comments would be most welcome!

Source code (PC):

```
import processing.video.*;
Capture video;
PImage img, maskImg;
color trackColor;

boolean cText;
boolean scanText;
int startTime;
int startTime2;
final int DISPLAY_DURATION = 5000;
final int DISPLAY_DURATION2 = 8000;

int inside = -1;
int bx=50;
int by=400;


void setup() {
  //define video frame size and the capture device
  size(640,480);
  video = new Capture(this,width,height,30);
  //project and overlay the image on the video
  maskImg = loadImage("abc.jpg");
  video.mask(maskImg);
  //specify the color mode
  colorMode(HSB, 360, 100, 100);

  //start off tracking for red
  trackColor = color(360, 100, 100);
  smooth();
  //specify the font
  PFont f = createFont("Arial", 26);
  textFont(f);
}
```

```
void draw() {
 // Capture and display the video
 if (video.available()) {

  video.read();

  image(video,0,0);
  image(maskImg,width/1.4,height/1.5);
  String s = "FIND RED";
  fill(360);
  //display the FIND RED text at the bottom of the screen
  text(s, 20, 420, 160, 160);

   if(inside == 1){
   int redX = 0; // X-coordinate of the red video pixel
   int redY = 0; // Y-coordinate of the red video pixel
   float redHue = 0; // hue of the red video pixel
   video.loadPixels();
   int index = 0;
   for (int y = 0; y < video.height; y++) {
    for (int x = 0; x < video.width; x++) {
     // Get the color stored in the pixel
     int pixelValue = video.pixels[index];
     // Determine the hue of the pixel
     float pixelHue = hue(pixelValue);

     if (pixelHue > redHue) {
      redHue = pixelHue;
      redY = y;
      redX = x;
     }
     index++;
    }
  }

  // Draw a large, white circle at the brightest pixel
  fill(0, 0, 100, 128);
  ellipse(redX, redY, 200, 200);


  if (scanText){
   fill(360);
   text("                  ...", 20, 420, 190, 190);
   //make the text appear on the screen no more than 8 seconds
   if (millis() - startTime2 > DISPLAY_DURATION2){
    scanText = false;
    }

   }
  }
 }

 // Before we begin searching, the "world record" for closest color is set to a high number that is easy for the first pixel to beat.
 float worldRecord = 400;

 // XY coordinate of closest color
 int closestX = 0;
 int closestY = 0;

 // Begin loop to walk through every pixel
 for (int x = 0; x < video.width; x ++ ) {
  for (int y = 0; y < video.height; y ++ ) {
   int loc = x + y*video.width;
   // What is current color
   color currentColor = video.pixels[loc];
   float h1 = hue(currentColor);
   float s1 = saturation(currentColor);
   float v1 = brightness(currentColor);
   float h2 = hue(trackColor);
   float s2 = saturation(trackColor);
   float v2 = brightness(trackColor);

   // Using euclidean distance to compare colors
   float d = dist(h1,s1,v1,h2,s2,v2);

   // If current color is more similar to tracked color than
   // closest color, save current location and current difference
   if (d < worldRecord) {
    worldRecord = d;
    closestX = x;
    closestY = y;
   }
  }
 }

 if (worldRecord < 10) {
  // Draw a square at the tracked pixel
  fill(trackColor);
  strokeWeight(1.0);
  stroke(10);
  rect(closestX,closestY,20,20);
 }


if(cText){
 //display the text according the the colour HSB values
 float h = (hue(trackColor));
float s = (saturation(trackColor));
float v = (brightness(trackColor));

if(h < 20 && s < .25 && v < .20){
  text("BLACK", 10, 30);
}
if(h < 20 && s < .25 && v > .90){
  text("WHITE", 10, 30);
}
if(h < 20 && s < .25 && v > .50 && v < .80){
  text("GRAY", 10, 30);
}
if(h < 20 && s > .90 && v > .25 ){
 text("RED", 10, 30);
}
if(h > 40 && h < 80 && s > .90 && v > .25){
  text("YELLOW", 10, 30);
}
if(h > 100 && h < 140 && s > .90 && v > .25){
  text("GREEN", 10, 30);
}
if(h > 160 && h < 200 && s > .90 && v > .25){
  text("CYAN", 10, 30);
}
if(h > 220 && h < 260 && s > .90 && v > .25){
  text("BLUE", 10, 30);
}
if(h > 280 && h < 320 && s > .90 && v > .25){
  text("MAGENTA", 10, 30);

}

 if (millis() - startTime > DISPLAY_DURATION){
   cText = false;
          }

 }
// println(hex(trackColor, 6));
}
void mousePressed() {
 // Save color where the mouse is clicked in trackColor variable
 int loc = mouseX + mouseY*video.width;
 trackColor = video.pixels[loc];
 cText = true;
 scanText = true;
 startTime2 = millis();
 startTime = millis();
//set a rule to check if we have clicked inside the button
if(!(((mouseX > ( bx+80))||(mouseY > (by+80)))||((mouseX < bx)||(mouseY < by))))
inside=inside*-1;
}
```

Source code (Android):

```
import ketai.camera.*;
import ketai.ui.*;

color backgroundcolor = color(0,0,0);
KetaiCamera cam;

color trackColor;
boolean cText;
int startTime;
final int DISPLAY_DURATION = 5000;

void setup() {
 orientation(LANDSCAPE);
 imageMode(CENTER);
 cam = new KetaiCamera(this, width, height, 24);
 // Start off tracking for red
 trackColor = color(255,0,0);
 smooth();
 PFont f = createFont("Arial", 26);
 textFont(f);
 }

void draw() {
 // Capture and display the video
 background(backgroundcolor);
 drawUI();
 text("TAP HERE TO START", width/2.5, height/2);
 cam.loadPixels();
 image(cam, width/2, height/2);
}
```

```
void mouseClicked(){
 if(mouseY < 100){
 if (mouseX > width/3)
   exit();
   else if (mouseX < width/3)
   KetaiAlertDialog.popup(this, "Help", "Tap on the screen to read
and track the colour of the selected object.");
 }
}

public void onCameraPreviewEvent()
{
 cam.read();
 // Before we begin searching, the "world record" for closest color
is set to a high number that is easy for the first pixel to beat.
 float worldRecord = 100;

 // XY coordinate of closest color
 int closestX = 0;
 int closestY = 0;

 // Begin loop to walk through every pixel
 for (int x = 0; x < cam.width; x ++ ) {
   for (int y = 0; y < cam.height; y ++ ) {
     int loc = x + y*cam.width;
     // What is current color
     color currentColor = cam.pixels[loc];
     float r1 = red(currentColor);
     float g1 = green(currentColor);
     float b1 = blue(currentColor);
     float r2 = red(trackColor);
     float g2 = green(trackColor);
     float b2 = blue(trackColor);

     // Using euclidean distance to compare colors
     float d = dist(r1,g1,b1,r2,g2,b2);

     // If current color is more similar to tracked color than
     // closest color, save current location and current difference
     if (d < worldRecord) {
       worldRecord = d;
       closestX = x;
       closestY = y;
     }
   }
 }

 // We only consider the color found if its color distance is less
than 10.
 if (worldRecord < 10) {
   // Draw a circle at the tracked pixel
   fill(trackColor);
   strokeWeight(1.0);
   stroke(10);
   rect(closestX,closestY,10,10);
 }

if(cText){

 int r = (trackColor >> 16) & 0xFF;  // Faster way of getting
red(argb)
 int g = (trackColor >> 8) & 0xFF;   // Faster way of getting
green(argb)
 int b = trackColor & 0xFF;
 println(r + " " + g + " " + b);
 if(r < 20 && g < 20 && b < 20){

 text("BLACK", 10, 30);
 }
 if(r < 50 && g < 255 && g > 190 && b < 255 && b >180){
  text("CYAN", 10, 30);
 }
 if(r> 100 && r < 255 && g < 30 && b < 30){
  text("RED", 10, 30);
 }
 if (millis() - startTime > DISPLAY_DURATION){
    cText = false;
  }
 }
}


void mousePressed() {
 // Save color where the mouse is clicked in trackColor variable
 if(cam.isStarted()){
 int loc = mouseX + mouseY*cam.width;
 trackColor = cam.pixels[loc];
 cText = true;
 startTime = millis();
 } else {
 cam.start();
 }
}

void drawUI(){
 pushStyle();
 textAlign(LEFT);
 fill(0);
 stroke(255);
 rect(0, 0, width/3, 100);
 rect((width/3)*2, 0, width/3, 100);
 fill(255);
 text("     HELP", 5, 60);
 text("     EXIT", width/3*2, 60);
 popStyle();
}

void exit() {
 cam.stop();
}
```