Journal of Computer Science 9 (6): 794-810, 2013
ISSN: 1549-3636
© 2013 Science Publications
doi:10.3844/jcssp.2013.794.810 Published Online 9 (6) 2013 (http://www.thescipub.com/jcs.toc)

Implementation of Color-Blind Aid System

Ruki Harwahyu, Alfa Sheffildi Manaf,
Bayu Sri Ananto, Burhan Adi Wicaksana and Riri Fitri Sari

Department of Electrical Engineering, Faculty of Engineering, Universitas Indonesia, Depok, Indonesia

Received 2012-11-27, Revised 2012-11-28; Accepted 2013-06-08

ABSTRACT

Color-blind is a physical defect causing patient loses the ability to recognize colors either particular or the whole of them. This disability is problematic in daily life, moreover in some specific areas that require carefulness eyesight. We propose a vision aid kit with improved user experience, such as finger pointer and voice command-and-response. Our work proofs the design and implementation of color-blind aid system for embedded and mobile device. This study used Windows Embedded Standard 2009, Windows Phone 7, Speech API library, Open CV library and EmguCV wrapper. The performance of each functionality is evaluated. From various testing that are conducted, the system can best detect color samples with precision level of 90.67% on embedded device implementation and 95.33% on mobile device implementation. The best fingertip detection rate attained is 89.6% with normal lighting condition. The voice command works with detection rate of 75.87%, meanwhile for the synthesized speech response, 88, 33% respondents can understood the words well.

Keywords: Color-Blind, Color Detection, Speech Recognition and Synthesis, Fingertip Detection, Color Transformation, Embedded System, Augmented Reality

1. INTRODUCTION

Despite of how important color in our daily life is, a generative deficiency called color-blind exists among several people. This physical defect reduces the eye's ability to distinguish colors, either partial or all of them. Based on the previous research that has been conducted by Ohkubo *et al.* (2010), there are 8-12% out of all men and 1% out of all women in the world are born with colorblind. Combined together with UN's statistic compiled at the end of 2011 (UN, 2011), it means that there are at most 420,000 men and 34,000 women among us who cannot enjoy this colorful world utterly.

Color-blind is a deficiency where the patient loses the ability to recognize color. Color-blind is caused by an anomaly in retina. Retina in human eyes uses cone cells to detect colors. Each specific light wavelength is detected using specific cone cells. In partial color-blind, one or some specific cone cells do not work perfectly, causing disability to distinguish respective color. In full color-blind, there is no cone cell at all. People with full color-blind can only have monochromatic vision.

There have been several works both providing improved methods and applying existing techniques that are aimed to help color-blind people. Ohkubo *et al.* (2010) have provided a model to accurately compensate color-blind vision in several commonly-used color format. Poret *et al*. (2009) have proposed an image processing technique for color correction. Leec and dos Santos (2010) have presented a simulation to help color-blind people that is based on fuzzy algorithm. Muttaqin and Suwandi (2011) have conducted a specialized work to build a simulation for color-blind aid glass. All of these previous researches are usefull enough to deal with the problem. However, there is no research that incorporate several methods and bound them together to provide a helpful, robust and easy-to-use color-blind aid system.

We propose this study to help color-blind people to have the equal chances with normal people in daily life. This study designs a prototype of color-blind aid system

Corresponding Author: Ruki Harwahyu, Department of Electrical Engineering, Faculty of Engineering, Universitas Indonesia, Depok, Indonesia

to help color-blind people distinguishing colors in form of a vision aid kit. This aid kit is implemented in both dedicated embedded system and mobile phone application. The system is developed using .NET platform for Windows Embedded Standard 2009 and Windows Phone 7 operating system. The system implements some color enhancement algorithm, augmented reality concept for information display, fingertip pointer for interactivity and voice command and response to provide enhanced user interface.

1.1. Color Model, Speech Processing And Augmented Reality

1.1.1. Color Model

Computer uses a certain model to represent color. There are 3 color models that is widely used, namely RGB, HSV and HSL. RGB defines each color by combination of red, green and blue element. Each of them ranging from 0 to 255. RGB model is easyer for human to understand.

HSV defines each color in hue, saturation and value element. HSL model is simpler for image processing (Wicaksana and Sari, 2011). Hue express color variation, meanwhile saturation express the intensity and value express the brightness.

HSL defines color in hue, saturation and lightness. Here, color intensity is determined by lightness. Conceptually, HSL represents a double cone with a white top, black on the bottom and gray at its center

1.2. Speech Processing

Sounds are basically a mechanical wave propagating through any medium that is sensible in ear. Most of human's ear can sense the wave within the frequency range of 20-20 kHz. Human can classify sounds into speech of human's communication voice, any sound that can be generated by instrument, or just other unimportant sound that we call noise.

In term of computation, computer 'sense' the sounds as continuous wave. Although it is a sound of human speech, that sound has no certain separation, meanwhile human can understand and detect any separation between the words mentioned in that speech. Computer sense no separation in the wave, between words or between the meaning in it.

Speech is audible wave that convey semantic information. To be able to understand this semantic data, computer needs speech recognition system. Alternatively, computer needs speech synthesis system to deliver understandable audio stream based on semantic data.

Human can interact with computer using voice, in form of command and response. One of speech interaction that is pretty wide used is Audio CAPTCHA. CAPTCHA is a method to ensure a response is given by human, not machine. Audio CAPTCHA uses sound in-stead of picture. The generated sound is usually a speech that said a phrase, with some background noise. This noise is just to hinder a machine detect the phrase, but is tolerable for human to do so. Knowing the level of tolerable noise is important in developing robust speech processing system.

Simple speech synthesis system can be made by arranging phonemes according to the letter in a word. Phoneme is the smallest unit of an utterance. It is an audio that express how letter or group of letters should be pronounced. One letter can have more than one phoneme. Letter 'a' in 'real' and 'data' has different phoneme, since it pronounced differently. Letter 'n' in 'number' and 'orange' also has different phoneme.

For speech recognition, there are some algorithms that are widely used, including Hidden Markov Model, Artificial Neural Network and Dynamic Time Wrapping (DTW). DTW is the simplest algorithm suitable for simple recognition and simple system. DTW expands or compresses input stream and then seeks the resemblance toward some pattern of the reference signals (Harwahyu and Sari, 2011). The most resemble reference signal is then choosed as the correct answer.

1.3. Augmented Reality

Augmented Reality (AR) is a concept where the representation of the computer-made object is arranged atop of the real object. This is made possible by captur-ing or maping the physical real condition of user's surrounding, embedding computer-made objects into itf and served them together to the user. AR is the improvement of the Virtual Reality (VR). The idea of virtual reality is to make the virtual version of the real world meanwhile the AR's idea is to combine virtual objects created by computer into the real world (Manaf and Sari, 2011).

The development of AR is mainly drived by the need to create more intuitive system. In the near future, AR is expected to bring the experience of the seamless computing, bringing digital content-computerized objects that is previously only exist in the computer's screen- into the real world together with any real objects around us.

2. MATERIALS AND METHODS

The color-blind aid system proposed in this study is presented in 2 form factor. The first is as a dedicated embedded system and the second is an application for mobile device (smartphone).

2.1. Dedicated Embedded System

The color-blind aid system on a dedicated embedded device proposed in this study serves as a prototype for further development on a Head-Mounted Display (HMD). To create a system that is similar to HMD, we use an eBox-3310 mini-computer as the central processing unit, a camera, microphone, earphone and a button on the keyboard that acts as a trigger button. The functionalities included in the embedded system are color detection with AR, color enhancement, fingertip pointer and speech feature. Speech feature that incorporates speech recognition and speech synthesis, together with fingertip pointer functionality, are the main interface for the user to control the device, since this device is designed with only a trigger button.

For the software system, our proposed system is implemented using NET 3.5 framework. The former attempt using. NET Compact Framework (CF) met the limitation for the speech processing functionality. Image processing works is done using EmguCV library, which is a wrapper for the widely-used OpenCV library for NET framework. Speech processing is done using Speech API (SAPI) 5.1 speech engine. The rest programming is done using C# language. For the underlying operating system, we use Windows Embedded Standard 2009 since the former attempt to build the functionality on Windows Compact Embedded 5 and 6 does not yields the expected outcomes due to the support of the newer. NET library.

2.2. Application for Mobile Device

Our proposed system that is implemented on mobile device is mainly developed for smart phone running Windows Phone 7 operat-ing system. The software is build atop of NET 3.5 framework with the support of Windows Presentation Foundataion (WPF) and Silverlight. The image processing utility is taken from EmguCV library. The functionalities that are included in this application are color detection with AR, color enhancement and color-blind test.

2.3. Color Detection with Augmented Reality

We implement color detection functionality into the system together with augmented reality to display the result.

Fig. 1. Color detection in mobile device (left) and embedded device (right)

Table 1. HSV value table for each colour names defined

Color names	Hue	Saturation	Value
White	0-255	0-170	200-255
Gray	0-255	0-170	110-200
Black	0-255	0-255	0-500
Pink	0-3/175-182	70-120	230-255
Dark red	0-3/170-182	230-255	145-175
Red	0-10/165-190	0-255	0-255
Brown	15-17/190-197	79-255	155-190
Orange	10-23/191-200	0-255	0-255
Yellow	24-34/201-210	0-255	0-255
Light green	35-50/211-230	0-255	0-255
Green	51-65/231-250	0-255	0-255
Tosca	66-83/251-255	0-255	0-255
Light blue	84-107	0-255	0-255
Blue	108-136	91-255	0-255
Purple	108-140	0-900	0-255
Magenta	141-165	0-255	0-255

Table 2. HSL value table for each color names defined

Color names	Hue	Saturation	Lightness
White	0-255	0-255	223-255
Gray	0-255	0-640	0-255
Black	0-255	0-255	0-320
Pink	0-3/175-182	0-255	191-223
Dark red	0-3/170-182	0-255	95-140
Red	0-10/165-190	0-255	141-179
Brown	15-17/190-197	0-255	63-950
Orange	10-23/191-200	0-255	127-160
Yellow	24-34/201-210	0-255	127-178
Light green	35-50/211-230	0-255	127-191
Green	51-65/231-250	0-255	63-127
Tosca	66-83/251-255	0-255	127-204
Light blue	84-107	0-255	127-191
Blue	108-136	0-255	51-153
Purple	108-140	0-255	95-191
Magenta	141-165	0-255	95-140

This functionality categorizes all visible colors into some easy-to-understand color names. The AR delivers overlay text information atop of real object that is captured

on the display screen. To use this functionality, user must select a color first. System then marks the real object that has the specified color. It is marked with rectangle and text. **Figure 1** shows the implementation on mo-bile device (left) and embedded device (right).

The processing is done in HSL color format, adjusted with the color perception which is recognized by normal-vision people. The color names used in this study for classification is limited to color that is oftenly used in everyday live and is also limited to its English name. **Table 1 and 2** contain each color name and its values in the HSV and HSL color format respectively.

2.4. Color Enhancement

This functionality intends to enhance the color of captured object to make it easier to differentiate by colorblind people. Color enhancement is made by performing color channel shifting. The shifting is done in HSL color format rather than RGB color format. This method changes the color channel values (by changing its hue and lightness) to obtain more contrast image. By doing this, the weak or ambiguous color can be minimized, so that color-blind people can know that an object has a different color. The vision weakness on some certain colors would be helped by displaying a sharper color that can be tolerated and accepted by color-blind eyes. This principle is based on the well-known theory of Dalton, a British scientist who became the first to examine the color-blind.

Conversion of color at the early stages of color transformation will not automatically make the same color in different formats. Colors that seem to contribute to change in accordance with a shifting is represented in the following Equation 1-3 (Shuhua and Gaizhi, 2010):

$$h = \begin{cases} 0^0 & \text{if } \max = \min \\ 60^0 \times \dfrac{g-b}{\max-\min} 0^0, + & \text{if } \max = \text{rand } g \geq b \\ 60^0 \times \dfrac{g-b}{\max-\min} + 360^0, & \text{if } \max = \text{rand } g \geq b \\ 60^0 \times \dfrac{b-r}{\max-\min} + 120^0, & \text{if } \max = g \\ 60^0 \times \dfrac{r-g}{\max-\min} + 240^0, & \text{if } \max = b \end{cases} \quad (1)$$

$$s = \begin{cases} 0, \text{if } \max = 0 \\ 0, \\ \dfrac{\max-\min}{\max} = 1 - \dfrac{\min}{\max} \end{cases} \quad (2)$$

$$v = \max \quad (3)$$

Hue (h) generally is perceived by the boundaries between 0 to 360, whereas saturation (s) and value (v) between 0 and 1. Analysis results for definition of the above equation, if the main color is red (max = r), h will be in the range (0, 60) and (300, 360). If the main color is green or blue, in other words, when conditions max = g or max = b, then the variations of each color range is 60 to 120 and 180 to 300, with a maximum value of each range is 120 and 240.

Figure 2 shows the detailed flow chart of this functionality. **Figure 3** shows the implementation on mobile device (left) and embedded device (right).

2.5. Fingertip Pointer

This functionality is only implemented on embedded device, which is to be implemented on HMD and has few button to control it. This functionality enables user to 'ask' what color is that things? by pointing his finger-tip to the object in question. It determine user's fingertip 2D coordinate in real-time frame captured by camera. This coordinate is further used by color detection functionality to extract color information of the pixels of the frame at the same coordinate. Finally, speech processing functionality (explained in section Error! Reference source not found) will 'say' the answer to the user using synthesized voice.

There are several steps to detect fingertip explained as follows.

2.5.1. Skin Classification

Skin classification aims to detect which object is user's hand, based on pixel's color. It uses explicit threshold on HSV and YCbCr color model. In HSV, skin color is when H≤50 and 0.23≤ S ≤0.68. In YCbCr, skin color is when 77≤ Cb ≤127 and 133≤ Cr ≤173. Image is then converted to biner (black and white) and skin area is colored white, as shown in **Fig. 4**.

2.5.2. Hand's Contour Extraction

System then extract the contour the biner image. Extracted contour will be saved as list of 2D coordinates. Contour is obtained based on the edges of objects which is explicitly separated between white and black region. If there is more than one object suspected as hand (because of their color), system will only takes the biggest object, assuming hand contour's size is bigger than noise's contour. On **Fig. 4**, noises are marked in red circle.

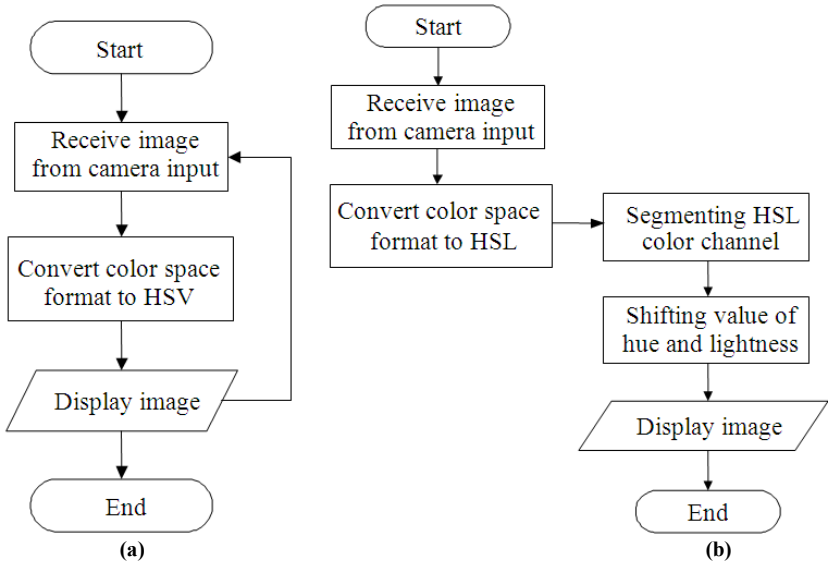


Fig. 2. Algorithm of color transformation on (a) embedded device and (b) mobile device

Fig. 3. Color enhancement in mobile device (left) and mbedded device (right)

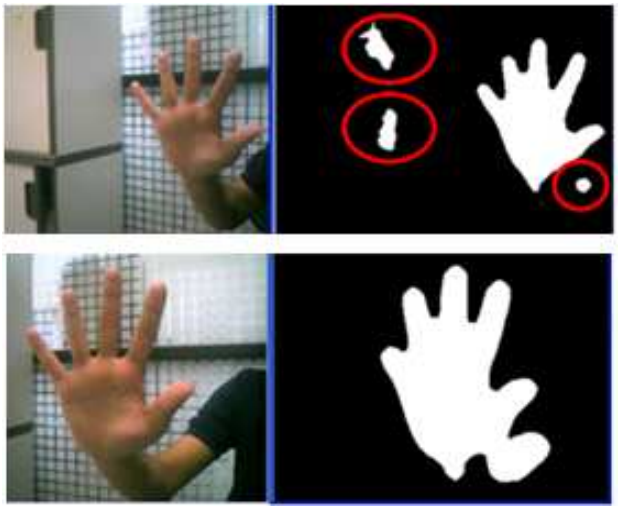


Fig. 4. Result of skin classification in HSV (upper row) and YCbCr (lower row)

2.5.3. Hand's Contour Analysis

To determine fingertip position from hand's contour extracted from previous process, hand's contour analysis is required. The hand's contour analysis is done based on contour's morphology (shape) which is gathered from hand contour extraction process.

Hand's contour analysis is done by examining convexity and concavity of the contour. To get the convex contour from hand's contour extracted and fingertip position, cvConvexHull and cvConvexityDefects function can be used in OpenCV. cvConvexHull function is used to generate convex polygon from extracted hand's contour.

The polygon will be comprised of vertices which are all nodes comprising polygon and vertexes which are lines that connect all vertices. In next process, the vertices of convex polygon gathered will be considered as fingertips. **Figure 5** depicts hand's contour which is processed into a polygon form before the polygon processed into convex polygon which its vertices are representation of fingertips.

In **Fig. 5**, the left-most hand side window shows blue line which is represents hand's contour. Whereas the centre window shows hand's contour which has been processed to polygon form that comprised of some vertices. The right-most hand side window shows polygon of hand's contour which has been processed to convex polygon. In this convex polygon, the concave which is formed from previous polygon's vertexes was eliminated, so the convex polygon as seen in right-most hand side window formed.

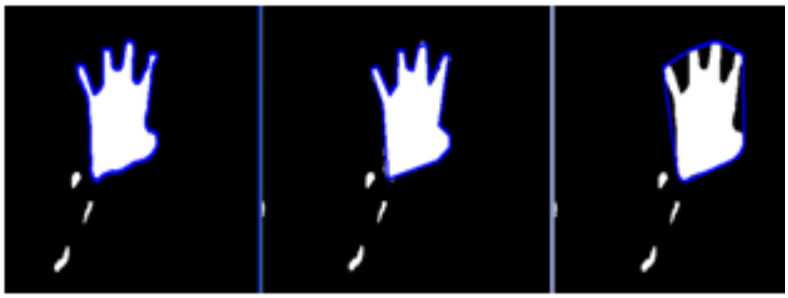
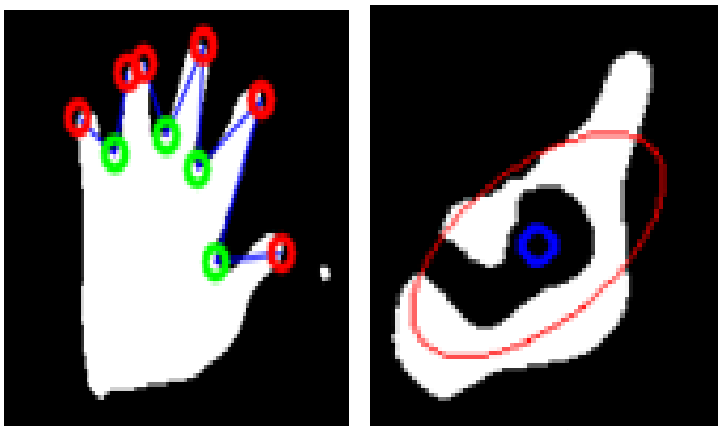


Fig. 5. Hand's contour, polygon of hand's contour and convex polygon



(a) (b)

Fig. 6. (a) Five fingertips (red) and points between finger (green) detected on hand's contour. (b) Ellipse (red line) on hand's contour and centre coordinate of hand's contour detected (blue circle)

Fig. 7. Pointing fingertip position detected by the system (red circle)

The convex polygon generated by checking each vertex, if the angle resulted from two vertexes less than 1800, it means the vertex is a part of con-vex polygon. If the vertices not satisfy the condition, so the vertex ignored. With convex polygon obtained, the vertex of the polygon is fingertips representation in hand's contour.

In the later process, the function cvConvexityDefects is used to obtain the position of the fingertip. This function employs hand's contour and convex polygon that has been obtained from the previous step. Algorithm used in this function based convexity and concavity nodes calculation technique used by Lee *et al.* (2011). **Figure 6a** shows the result of this step.

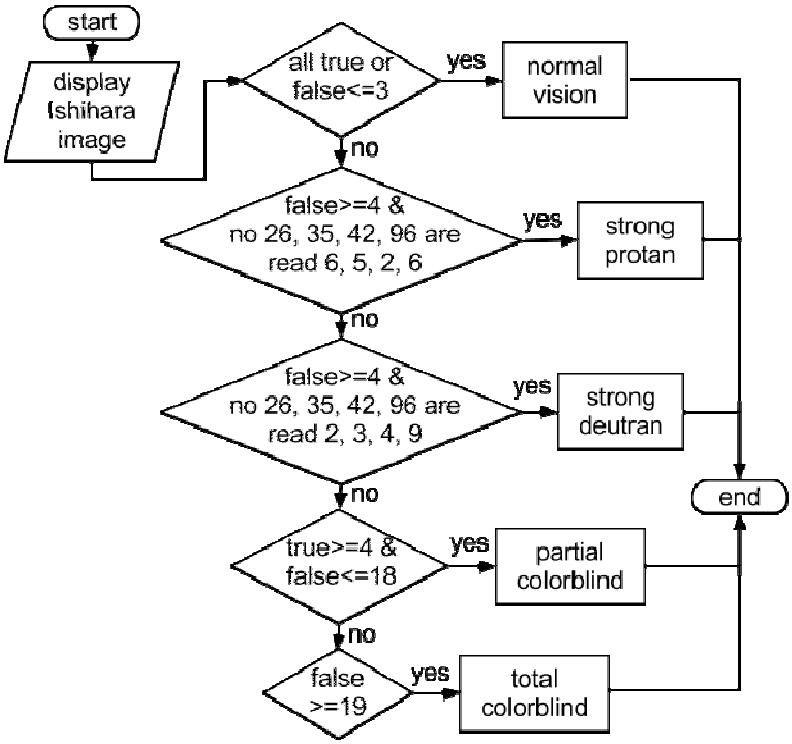


Fig. 8. Algorithm of color-blind test system

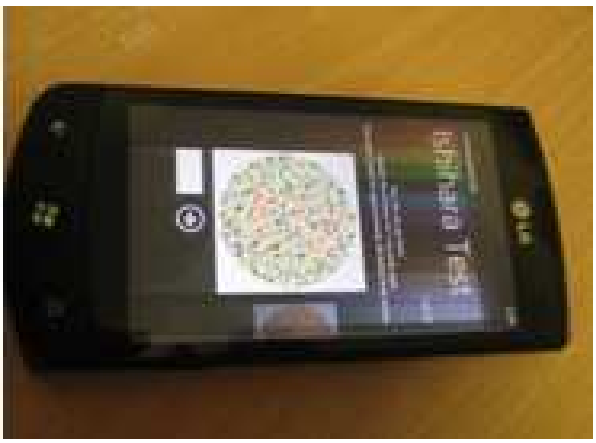


Fig. 9. Implementation result of color-blind test on Windows Phone 7

2.5.4. Determining Fingertip Position

This step uses cvConvexityDefects function to obtain 2 arrays. The first array gives 2D coordinate position of the convexity maximum point of the contour. The second array gives 2D coordinate position of the concavity minimum point of the contour. A center coordinate of the hand's contour is then calculated based on all coordinates in the arrays in detail. It is conducted by doing ellipse fitting technique and obtains the centre of ellipse's coordinate position. Ellipse fitting technique basically is an approximation with least square method within distribution of points obtained which are representation hand object in binary image. **Figure 6b** shows how an ellipse is fitted with the hand contour witht the center.

The fingertip coordinate is then determined as the maximum point that has the maximum distance with the center coordinate of the hand's contour. This method is used based on the assumption that most of the hand-pointing contour that is captured by the camera will have the pointing finger stands out much farther to the center of the hand compared to other convex points. The end results of the process is shown in **Fig. 7**.

2.6. Color-Blind Test

Color-blind test is an additional feature that is only implemented on mobile devices. This test can determine the type of user's vision, whether it is strong protan type (weakness of red color), strong deutan (weakness of

green color), total color-blind, normal vision, or partial color-blind (it could be weak protan type, weak deutan, or tritan) (Ananto *et al*., 2011).

This test utilizes 12 Ishihara images that each of them has a certain condition which serves as the parameter to determine the test result. In addition, to reduce the error rate, is made a tolerance value, such as when someone answers wrong, there is still a possibility that the users is normal vision in color-blind test results, but on condition that not more than 4 incorrect answers. This is done to minimize mistakes made inadvertently in the use of applications by users. The algorithm of this test functionality is depicted in **Fig. 8** and the result of this implementation on the mobile device is shown in **Fig. 9**.

2.7. Speech Feature

Speech feature is only implemented in embedded device. The speech feature includes speech recognition to interpret user's word and speech synthesis to give a response to user. User operate the system by saying words or commands, such as mentioning number for selecting options, saying "yes", "no", "okay" or "cancel" for do or cancel an action and saying "mode #" for changin between functionalities indicated by number. When the user choose to activate color detection functionality, the system will say the name of the color of the object pointed by user using word synthesized by the speech synthesizer. Thus, for the embedded device, speech is the main way to interact between the user and the system.

We implement speech recognition and synthesis using Speech API (SAPI) that will only work with .NET 3.5 environments. For the speech recognition functionality, we use limited grammar to increse the robustness of recognizing available words. We set this limited grammar to only load 24 words, which is consisting of 24 color names and several words as mentioned before. These words are stored in a Grammar class to be loaded when the application is started. Limited grammar is suitable for controling the system, since not all words in english dictionary are necessary to command the system. As for the speech synthesis functionality, we use asynchronous mode, thus the application can process an input text to the audio stream when another text is being pronounced.

3. RESULTS AND DISCUSSION

We conduct some tests to assess the functionalities, performance and user experience for both embedded device and mobile device application.

Table 3. Test result on mobile device and embedded device

	Success (%)	
Reference color	Mobile device	Embedded device
Red	90.00	90.00
Orange	100.00	100.00
Yellow	100.00	100.00
Light green	100.00	100.00
Green	80.00	80.00
Tosca	100.00	100.00
Light blue	50.00	50.00
Blue	90.00	90.00
Purple	100.00	100.00
Magenta	100.00	100.00
Dark green	80.00	80.00
Dark orange	100.00	100.00
Yellow-green	90.00	90.00
Dark gray	80.00	80.00
Dark red	100.00	100.00
Average	95.33	90.67

3.1. Color Detection

Table 3 shows the detection performance of the system in both platform, embedded device and mobile device. Testing is conducted with 5 MP USB-attached webcam on embedded device and 1.3 MP built-in camera on mobile device. In this test, the system is used to detect several printed color samples and each sample is displayed 10 times in random. The color samples are printed in HLS color format, so the printed color is consistence with the color displayed in computer's screen. The table shown that the detection performance is better on the mobile device application.

In addition to the system performance test, we also conduct user testing. User testing is done on 5 normal vision user and 5 partial color-blind user. In this test, user is asked to use both system to towards printed sample colors, repeated 10 times for each in random. We then ask the user whether they agree or disagree with the color name that is mentioned by the system for each printed color sample. Their answer is quantisized using the numer from 1 to 5, where 1 means "strongly disagree" and 5 means "strongly agree". The main purpose of this test is to know how much our system matches user's perception about the colors. The result of this test are displayed in 95% confidence interval in **Fig. 10 and 11** for the embedded device and the mobile devie application respectively. The figures yields that on embedded device the average value is 2.86 and on mobile device application the average value is 3. It means that on average user agree with what our system said about the name of the color.

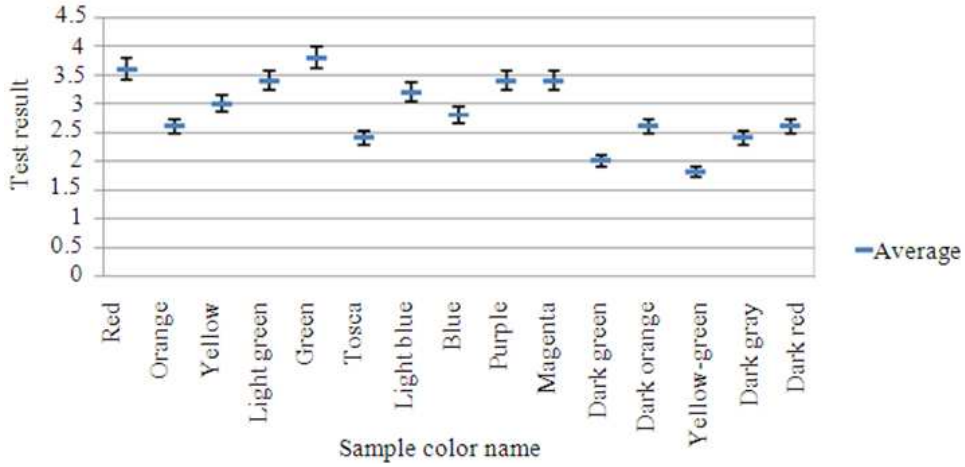


Fig. 10. Mathcing between user's and system's perspective for the color on embedded device

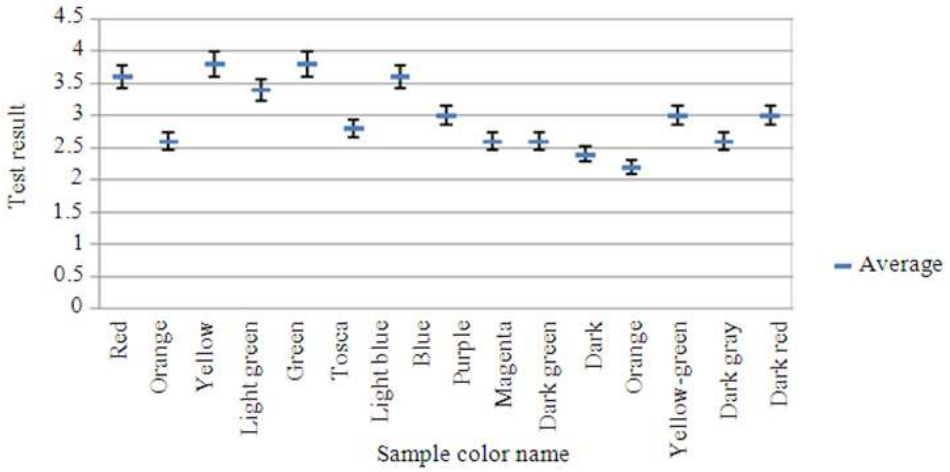


Fig. 11. Mathcing between user's and system's perspective for the color on mobile device application

3.2. Color Enhancement

The next test is to assess the performance of our color enhancement functionality. This functionality is applied in both embedded and mobile device and this test is conducted toward both of them. In this test, we use 10 users to see images and ask their vision clarity for the images, with and without using our color enhancement functionality. User's should represent their vision clarity using number 1 for "not visible", 2 for "hard visible", 3 for "visible", 4 for "clearly visible", 5 for "very clearly visible". The images used in this test are 11 Ishihara color-blind test images. We want to know whether our system can help color-blind peope to recognize the number obscured on each image. **Figure 12** shows the images that are used.

Figure 13 and 14 both shows the result of the test when the user is using our system on the mobile device application. In this test, before using our system there are 7% occurence of "not visible". After the use of our system, their vision clarity is increased resulting 0% of "not visible".

Comparison of average value of the confidence interval testing showed that using the color transformation system, users can more clearly see the numbers on Ishihara image than without the transformation color system. That means, the implementation of color transformation method on embedded devices help users, both for normal vision and partial color-blind, to recognize objects by color ambiguous.

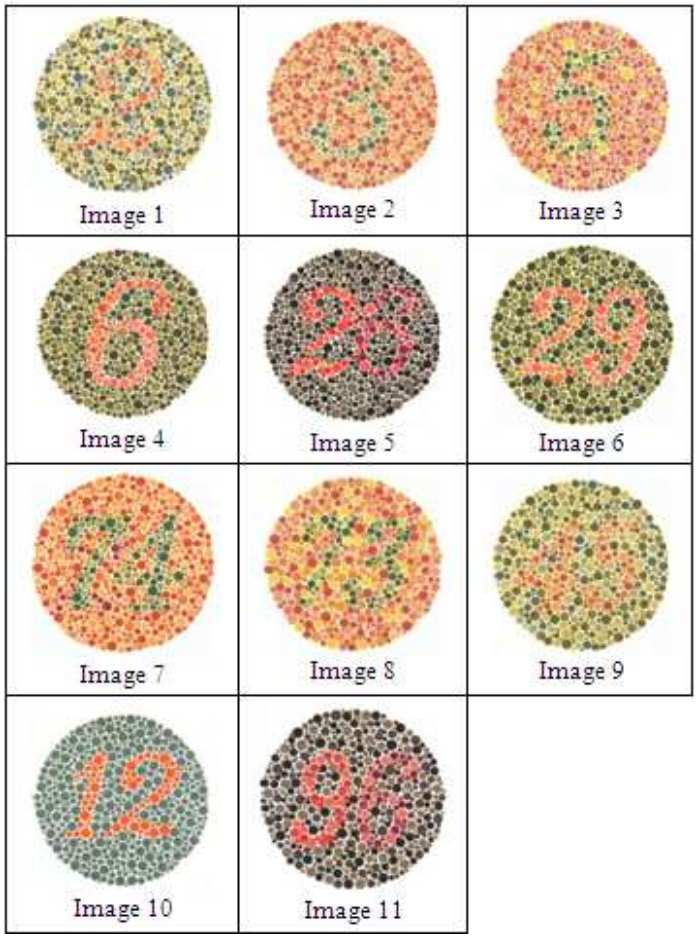


Fig. 12. Types of Ishihara image that is used in the testing method

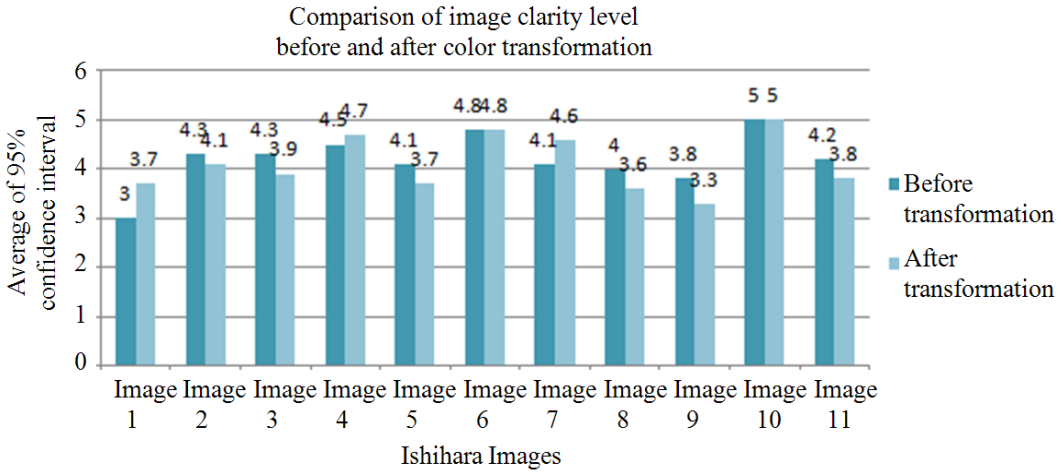


Fig. 13. The comparison graph of image clarity level

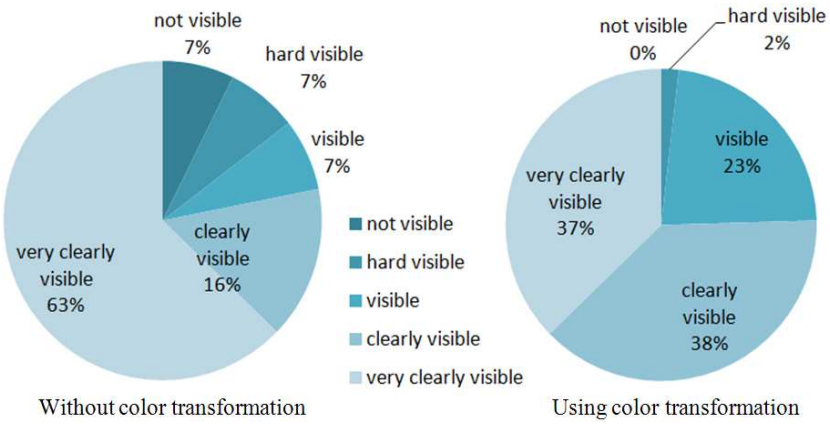


Fig. 14. The percentages graph of vision condition of images

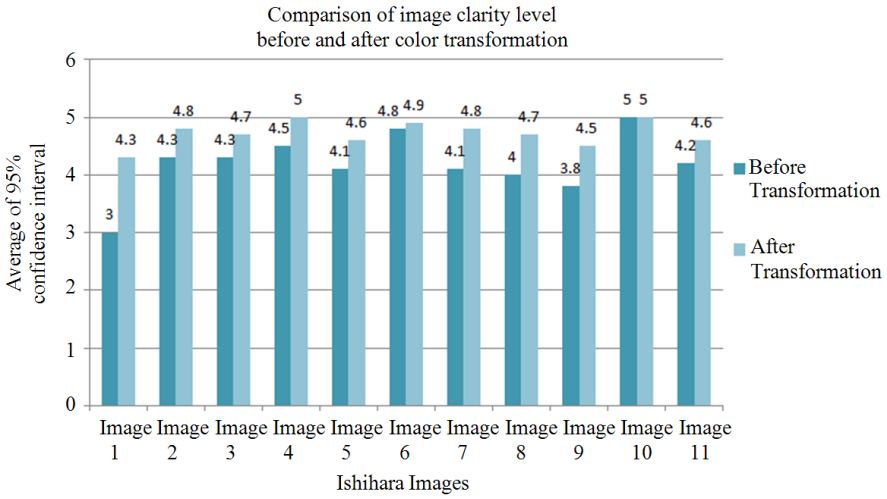


Fig. 15. The comparison graph of image clarity level

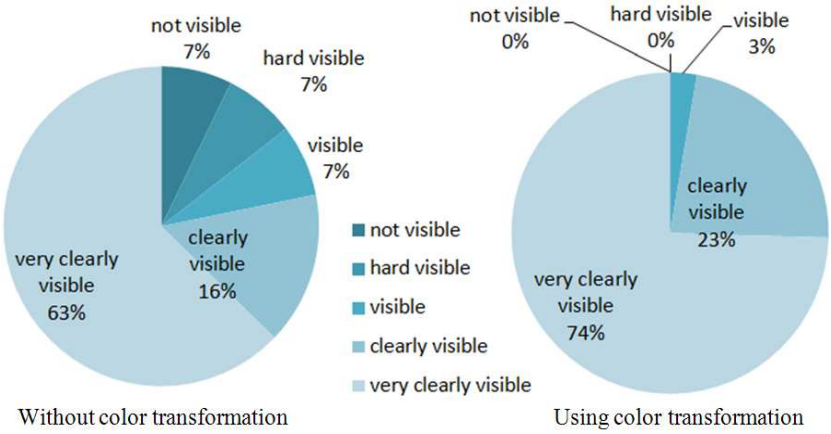


Fig. 16. The percentages graph of vision condition of images

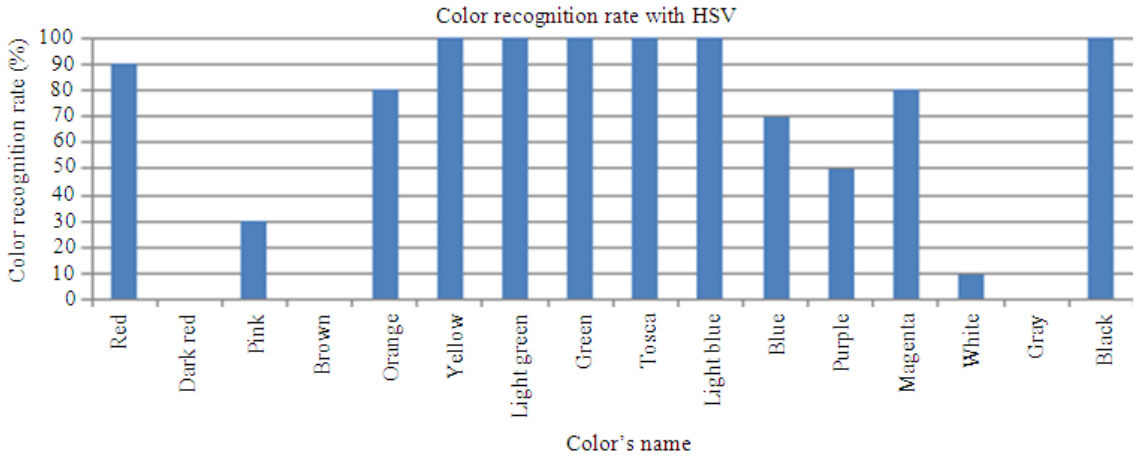


Fig. 17. Testing result of color Recognition rate in HSV color format

This is evidenced by the average value of 95% confidence interval for the conditions without color transformation system of 4.19 ± 0.744, while for the condition using a color transformation system is 4.72±0.264 of the maximum interval value scale 5 points.

Figure 15 and 16 both shows the result of the test when the user is using our system on the embedded device. In this test, after the use of our system, their vision clarity is increased resulting 0% of "not visible" and 0% of "hard visible", most of them are "clearly visible" and "very clearly visible".

3.3. Fingertip Pointer for Color Detection

The next functionality to test is the fingertip pointer. In this test, we use the embedded system to detect the color of an object that is pointed out (using finger). We asssess the detection performance of the system in 2 color format, HSV and HSL. For this test, we also use the printed sample colors that is used in the previous test. The test is repeated 10 times for each sample color in random. The result is depicted in **Fig. 17** for HSV color format and **Fig. 18** for HSL color format.

From the figures, it is clear that there are some color with poor recognition rate. All colors which is only depend on Hue component in both HSV and HSL color formats have high detection rate meanwhile the colors which have dependencies to Saturation in HSV and Lightness in HSL have low recognition rate. According to the environment when the test is conducted, this result is due to some reasons: the color classification is done by observing the color at computer's sceen, not on real object; there could be slight inaccuracies when printing the sample colors; the threshold defined as skin color has some similar values with colors in the object that is to be detected.

3.4. Color-Blind Test

The next test is to assess whether our color-blind test can accurately detect user's vision condition. In this assessment, we ask 10 users to perform the color-blind test on the mobile device application and compare the result from this test with their vision report from the test that is carried out manually by a medical doctor or expert. In this assessment, two out of ten users are having partial color-blind. From **Fig. 19** that is showing the result of this assessment, it is clear that our implementation of color-blind test is highly accurate to detect user's vision condition. It further indicates that the use of 11 images in our system to represent 24 Ishihara images is enough to fit the result from the manual color-blind test.

3.5. Speech Functionalities

Since there is performance limitation of the hardware, this tests is done on a normal computer. This alternative is taken to accelerate the process and since the speech feature's work is based on voice samples and database which are unchanged wherever it being implemented.

For the speech synthesis, the test is conducted using 10 Indonesian students as the user. Firstly, they have to listen the native pronunciations of the tested words. Then they guess every word said by the system with 3 opportunities for each word. We apply a simple weighting mechanism for each occurrence.

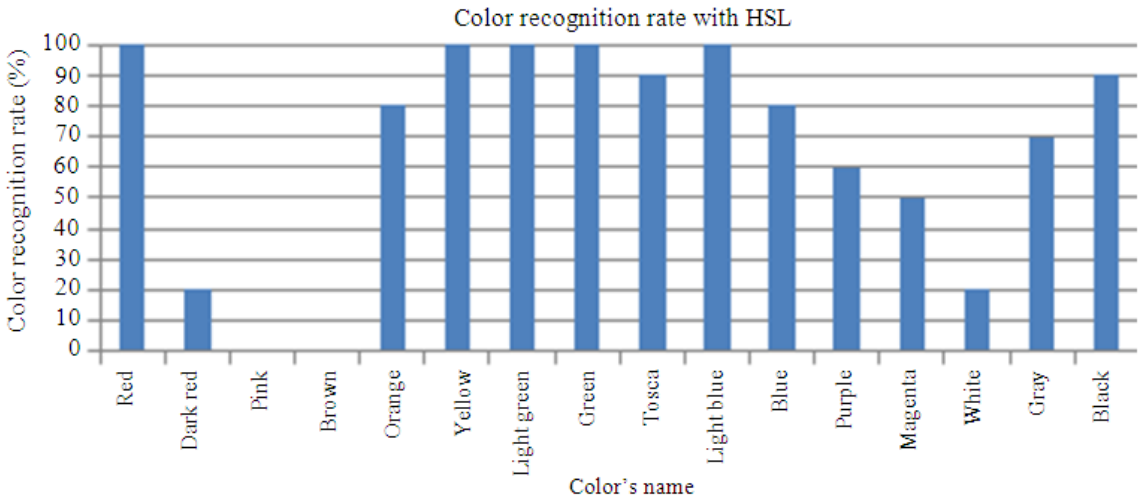


Fig. 18. Testing result of color recognition rate in HSL color format

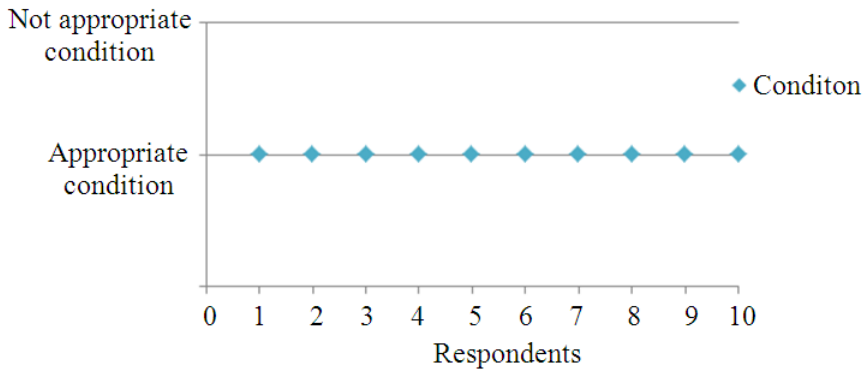


Fig. 19. Matching between the result of our proposed test and manual test carried out by medical expert

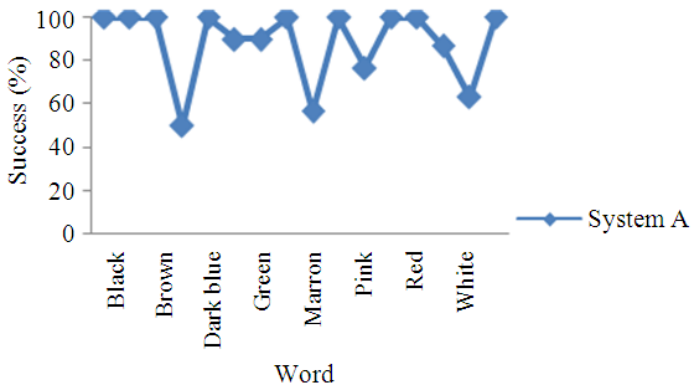


Fig. 20. Success percentage for each word in each system

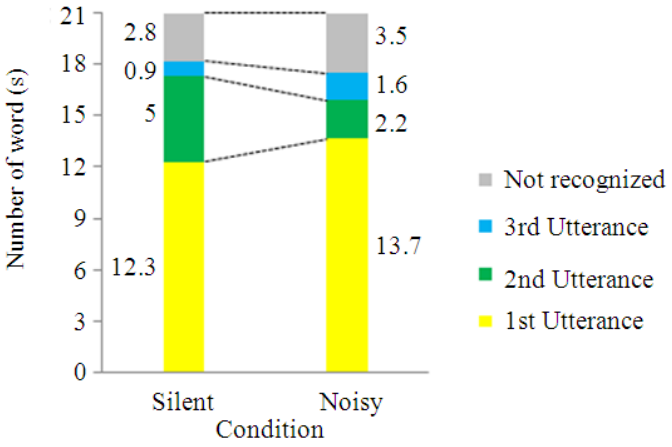


Fig. 21. Cumulative distribution of words for each occurrence

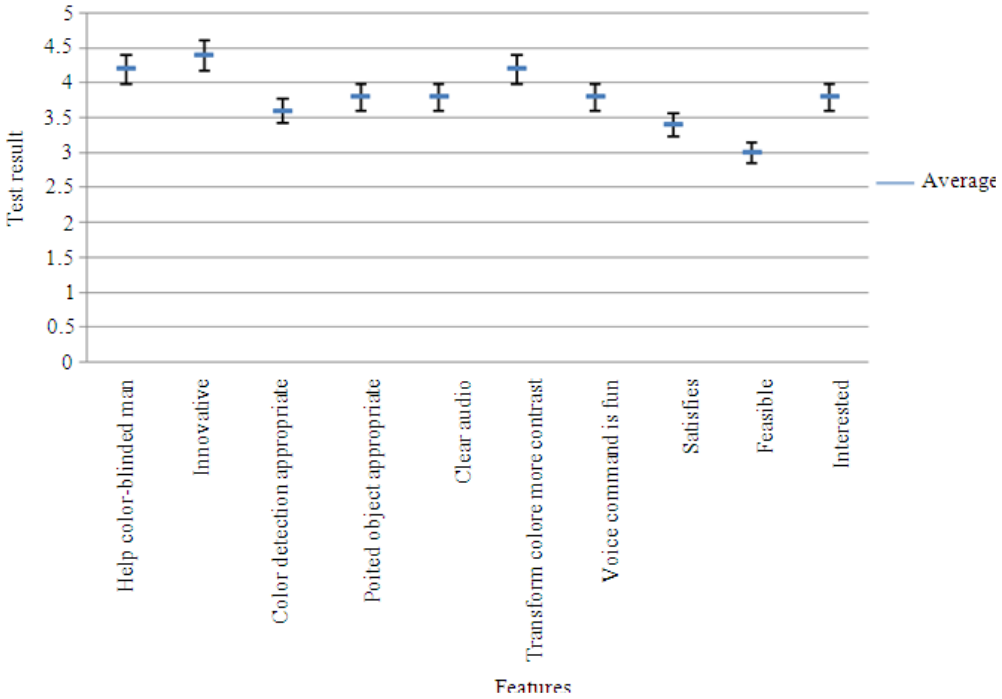


Fig. 22. User experience on embedded device for normal user

The occurence where user can recognize the pronounciation in first utterance has 3 points; 2 points for recognition in second utterance; 1 point for recognition in the last utterance; and 0 point for if user cannot recognize the pronounciation until the last utterance. This test yields that overall performance of system A is 88,33% and system B is 62,92%. The rest result are shown in **Fig. 20**.

For the speech recognition, the test is conducted in 2 condition varied by its background noise. The first condition is a silent and the second condition is a noisy condition with about 20% SNR. Background noise audio is taken randomly from average audio CAPTCHA in popular sites such as captcha. biz, recaptcha. NET and facebook.com.

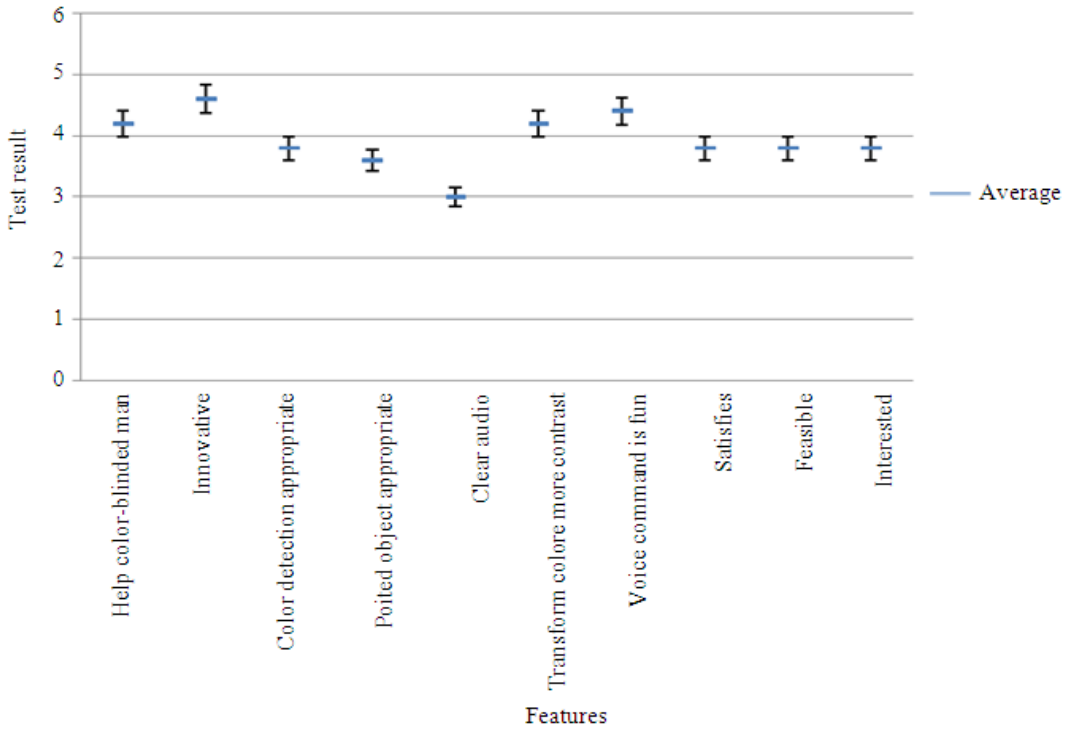


Fig. 23. User experience on embedded device for partial color-blind user

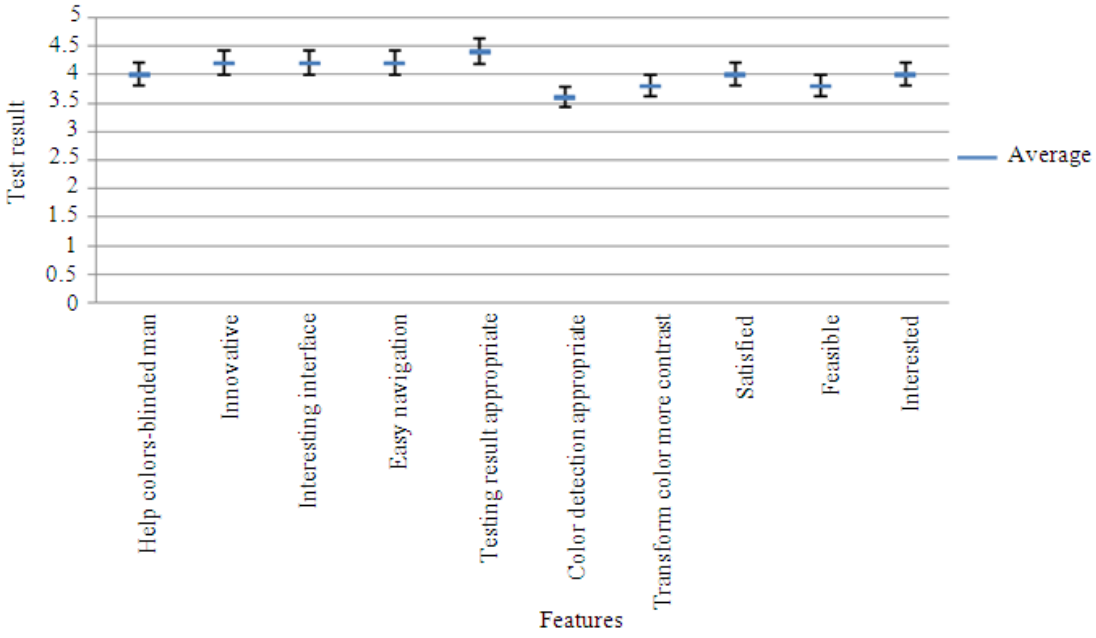


Fig. 24. User experience on mobile device application for on normal user

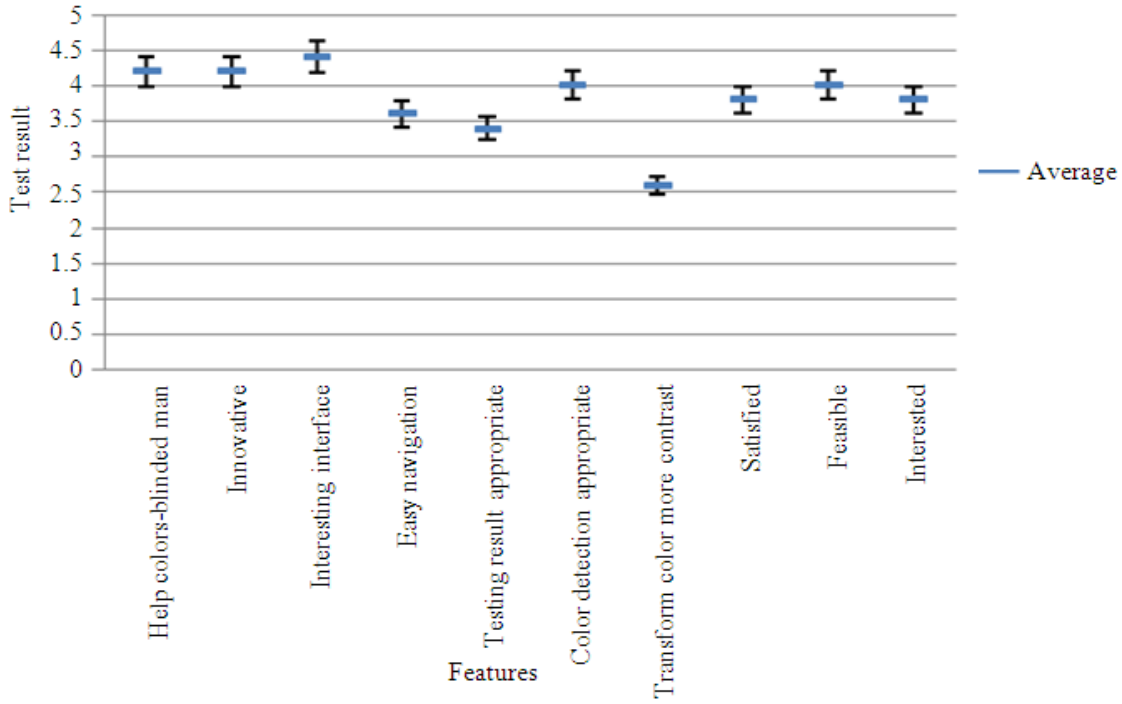


Fig. 25. User experience on mobile device application for partial color-blind user

The choice to use 20% SNR is taken from this average CAPTCHA as the best practice used nowadays.

This test uses 10 Indonesian students as the user. They listened to native pronunciation of tested words and they pronounced every word for the system to recognize.

At every condition, every word can be repeated until 3 times. The overall performance is then measured by implementing the same weighting mechanism as mentioned in previous test. As the result, performance of this system is 75,87% in silent condition and 74,76% in noisy condition. The result of this test is depicted in **Fig. 21**.

In addition, users are asked to try our system and questioned whether the pronunciation created by the system is clear for them. As the result, 10% of them choose "strongly agree", 70% of them choose "agree", 10% of them choose "fairly agree", 10% choose "disagree" and 0% of them choose "strongly opposite".

3.6. User Experience

The last test that is conducted in this work is to assess user experience in using our system, both for the embedded device and the application fo mobile device. In this test, we ask several user consisting normal-visioned user and partial color-blind us. We later ask their opinion using some question about their satisfaction in using the system for helping their vision. In answering the question, they must choose a number from 1 to 5, where 1 means "strongly disagree" and 5 means "strongly agree".

Figure 22 shows the test result for experience in using our system as an embedded device for the normal user. **Figure 23** shows the test result for experience in using our system as an embedded device for the partial color-blind user. **Figure 24** shows the test result for experience in using our system as an mobile device application for the nor-mal user. **Figure 25** shows the test result for experience in using our system as an mobile device application for the partial color-blind user. All are displayed in 95% of con-fidence interval.

From all the figure provided, the answer for all question from all user always lies above 3. It means that our propsed implementation of color-blind aid system is sufficient to meet the user expectation, both for color-blind people and normal-visioned user that is likely to assist color-blind people.

4. CONCLUSION

We have done several technical implementation to create a prototype of color-blind aid system in two form factors, as an embedded device and application for mobile device. In this study, we have implemented some

main imaging functionalities as well as additional functionalities to make the system user-friendly and fun to be used. The preliminary results that are provided in this study shows that all the functionalities on both form factors are useful to help color-blind people to have more clear vision toward some ambiguous colors. The other test result also indicates that our prototype system provides good user experience.

5. REFERENCES

Ananto, B.S., R.F. Sari and R. Harwahyu, 2011. Color transformation for color blind compensation on augmented reality system. Proceeding of the International Conference on User Science and Engineering (i-USEr), Nov. 29-Dec. 1, IEEE Xplore Press, Shah Alam, Selangor, pp: 129-134. DOI: 10.1109/iUSEr.2011.6150551

Harwahyu, R. and R.F. Sari, 2011. Implementing speech feature for embedded system to support color blind people. Proceedings of 1st International Conference on Informatics and Computational Intelligence (ICI), Dec. 12-14, IEEE Xplore Press, Bandung, Indonesia, pp: 256-261. DOI: 10.1109/ICI.2011.49

Lee, D.H., S.G. Lee and J.H. Choi, 2011. Region-based corner detection by radial projection. J. Opt. Soc. Korea, 15: 152-154.

Leec, J. and W.P. dos Santos, 2010. Fuzzy-based simulation of real color blindness. Proceedings of the Annual International Conference of the IEEE on Engineering in Medicine and Biology Society (EMBC), Aug. 31-Sep. 4, IEEE Xplore Press, Buenos Aires, Argentina, pp: 6607-6610. DOI: 10.1109/IEMBS.2010.5627128

Manaf, A.S. and R.F. Sari, 2011. Color recognition system with augmented reality concept and finger interaction: Case study for color blind aid system. Proceedings of the 9th International Conference on ICT and Knowledge Engineering, Jan. 12-13, Bangkok, Thailand, pp: 118-123. DOI: 10.1109/ICTKE.2012.6152389

Muttaqin, G.F. and I.S. Suwandi, 2011. Simulation system of color blind glasses by image processing. Proceedings of the International Conference on Electrical Engineering and Informatics, Jul. 17-19, IEEE Xplore Press, Bandung, Indonesia, pp: 1-4. DOI: 10.1109/ICEEI.2011.6021616

Ohkubo, T., K. Kobayashi, K. Watanabe and Y. Kurihara, 2010. Development of a time-sharing-based color-assisted vision system for persons with color-vision deficiency. Proceeding of SICE Annual Conference, Aug. 81-21, IEEE Xplore Press, Taipei, pp: 2499-2503.

Poret, S., R.D. Jony and S. Gregori, 2009. Image processing for colour blindness correction. Proceedings of the IEEE Toronto International Conference on Science and Technology for Humanity, Sep. 26-27, IEEE Xplore Press, Toronto, Canada, pp: 539-544. DOI: 10.1109/TIC-STH.2009.5444442

Shuhua, L. and G. Gaizhi, 2010. The application of improved HSV color space model in image processing. Proceedings of the 2nd International Conference on Future Computer and Communication, May 21-24, IEEE Xplore Press, Wuhan, China, pp: V210-V213. DOI: 10.1109/ICFCC.2010.5497299

UN, 2011. Statistics and indicators on women and men.

Wicaksana, B.A. and R.F. Sari, 2011. Implementing text information display of detected color for partially color blinded person using .NET platform and EmguCV library. Proceedings of the International Conference on Information Technology and Multimedia (ICIM), Nov. 14-16, IEEE Xplore Press, Kuala Lumpur, Malaysia. DOI: 10.1109/ICIMU.2011.6122760