Matlab Based Eye Color Visibility

Banothu Thavu
Assistant Prof.,
Kavitha Engg. College, Khammam, India.
Email: tej.abhinaik@gmail.com

Abstract: Color is a particular sensation created in the brain, caused when light radiation of a certain wavelength reaches our eyes. The colors we perceive in an object are determined by the nature of the light reflected from the object .This definition is built up from two parts, which are quite different. The first is of a psychological nature. This portion deals with the way the sensation of color is processed by the mind. The second one is merely the eye's detection of physical radiant energy. Therefore, color is in fact a psychophysical phenomenon, inter-relating both psychological and physical processes. In this paper explained and writing a matlab code for Eye coloring.

Keywords: Eye Color, Brain Process, Matlab, Image processing.

I. INTRODUCTION

Color is determined by an interaction among three photo pigments; the perceived color is a mixture of the relative responses of the red, green, and blue photo pigments, in much the same way as a television camera creates color. Given a dramatic imbalance among the percentages of cells containing red (approximately 64%), green (approximately 32%), and blue (approximately 2%) photo pigments, it is clear that the perception of color is both highly specialized and physiologically biased. Due to the structure of human eye, all colors are seen as variable combinations of the three so-called Primary colors Red, Green and Blue (RGB).

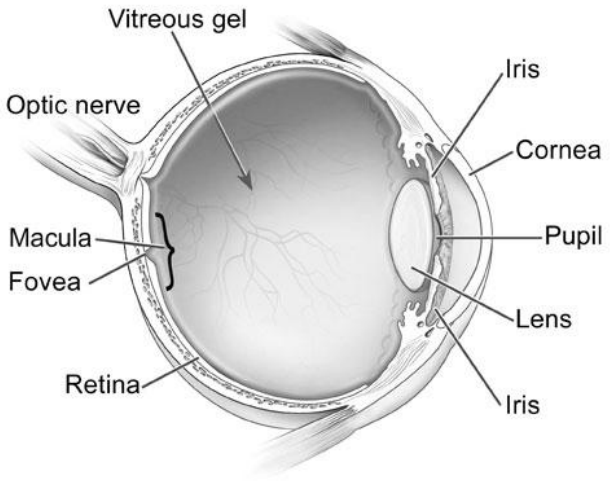


Fig. 1: Eye Structure.

Color results from the interaction of light with the nervous system. There are several components that affect color perception, including the eye lens, the retina, and a color processing unit along the optic nerve. These areas are discussed in the following sections.

II. EYE LENS

The function of the lens is to focus the incoming light on the retina, which contains the photoreceptors. Different wavelengths of light have different focal lengths. Therefore, for pure hues, the lens must change its shape so that the light is focused correctly.

For a given lens curvature, longer wavelengths have a longer focal length, i.e., red is the longest focal length and blue is the shortest. To have an image focused on the retina, the lens curvature must change with wavelength with red light requiring the greatest curvature and blue light the least curvature. This means that if pure blue and pure red hues are intermixed, the lens is constantly changing shape and the eye becomes tired. A related effect is called chromostereopsis, here pure colors located at the same distance from the eye appear to be at different distances, e.g. reds appear closer and blues more distant. Sometimes pure blues focus in front of the retina and so appear unfocused. At night, a deep blue sign may appear fuzzy while other colors appear sharp. The lens also absorbs light about twice as much in the blue region as in the red region. As people age the lens yellows, which means it absorbs more in the shorter wavelengths. Therefore, the result is that people are more sensitive to longer wavelengths (yellows and oranges) than they are to shorter wavelengths (cyan to blue) and these increases with age. The fluid between the lens and the retina also absorb light and this effect increases as people age, so the older people get the less sensitive they are to light in general (the apparent brightness level decreases) and especially the sensitivity to blue decreases.

III. RETINA

The retina contains the photoreceptors that absorb photons and transmit chemical signals to the brain. There are two types: rods, which are night-vision receptors and have no color dependency, and cones, which have color sensitivity and require a higher level of light intensity than the rods.

There are three types of photo-pigments in the cones; "blue" with a maximum sensitivity at 430 nm, "green" with a maximum sensitivity at 530 nm, and "red" at 560 nm. (This wavelength actually corresponds to yellow). Light at a single

wavelength will partially activate all three types of cones, e.g. at a wavelength of 470 nm, blue is strongest plus some red and green components.

The percentage of cones is not equal but is as follows: blue (4%), green (32%), and red (64%). In addition, the cones are differentially distributed in the retina. The center of the retina has a dense concentration of cones but no rods while the periphery has many rods but few cones. The color distribution is also asymmetrical. The center of the retina is primarily green cones, surrounded by red-yellow cones, with the blue cones being mainly on the periphery. The center of the retina has no blue cones.

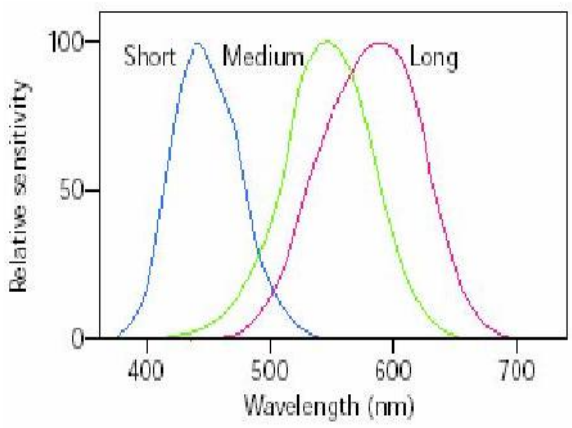


Fig. 2. Spectral sensitivities of the three classes of photoreceptors in the retina

Objects are seen by edge detection, where an edge can be created by a difference in color or brightness or both. Edges formed by color differences alone, with no brightness differences, appear fuzzy and unfocused, so changes in brightness should be added to get sharp edges. Photoreceptor adjusts their sensitivity to the overall light level, e.g. going into or out of a dark room requires some adjustment time. There is also a required minimum intensity level for the photoreceptors to respond. This minimum varies with wavelength with the highest sensitivity in the center of the spectrum. Therefore, blues and reds must have a higher intensity than greens or yellows in order to be perceived.

IV. BRAIN

From the retina, the optic nerve (actually a collection of nerves) connects to the brain but before it reaches the brain, there is a color-processing unit, called the lateral geniculate body. This recombines the RGB color information into three new channels as follows:

1) R-G gives red or green color perception
2) R+G gives the perception of brightness and yields yellow (Y)
3) Y-B gives yellow or blue color perception

Thus, blue plays no part in brightness so that colors differing only in amount of blue don't produce sharp edges. Also, note that since blue and yellow and red & green are linked together it is impossible to experience combinations such as reddish green or bluish yellow.

About nine percent of the population has some kind of color perception problem. The most common is red-green deficiency, which can arise from a deficiency of either the red or the green photo-pigments. These people have difficulty distinguishing any color that is dependent upon the red: green ratio.

As well understood as the physiology of color is, this factor provides little explanation for our opinions of color and color combinations. At the very least, opinions of color are learned and highly associative. For example, as children, we often had a “favorite color” and we liked everything: clothes, toys, books that matched our preference. Over time, we learned a variety of color schemes and in most cases, our tastes become more refined. But even as adults, we are influenced by fashion, and may still associate our more sophisticated sense of color with increasingly more sophisticated emotions, desires, or impressions. For example, even a cursory examination of changes in interior design from the 1950s to the present reveals a dramatic evolution of what was considered warm or even comfortable color combinations. A lively debate still rages about the psychology of color, and various claims are made for using color in the environment to stimulate, calm, or enhance the performance of individuals.

Color psychology is a vast field in which color-constancy, simultaneous contrast, the effects of various backgrounds on color perceptions, and so on, are examined, and competing explanations are debated. Colors tend to look darker and smaller against white, and lighter and larger against black. The apparent changes in size arise, at least in part, from flare in the eye’s optics, which causes light from the bright areas to be scattered into the neighboring dark areas. Colored surroundings can cause a colored region to appear.

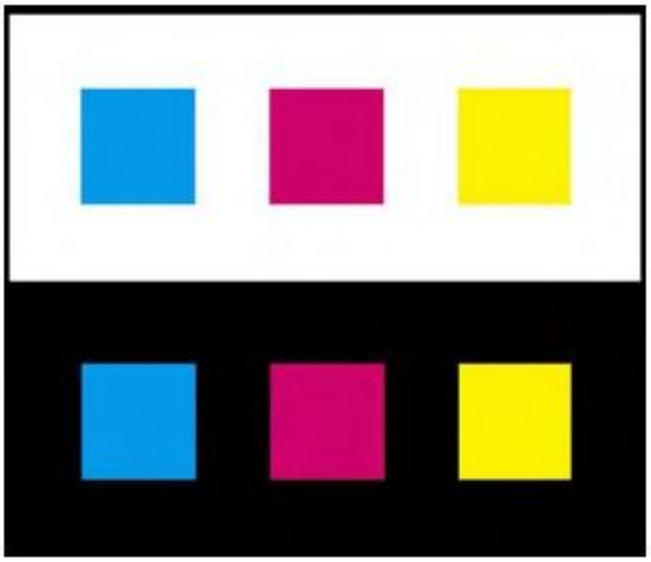


Fig3. Colors look darker and smaller against a white background and lighter and larger against black.

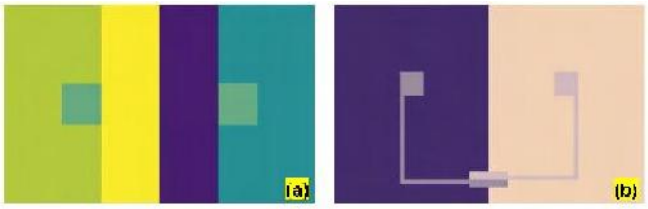


Fig3.4. (a) Simultaneous contrast can make the same colors look different.(b) Simultaneous contrast can make different colors look the same.

Tinged with the complementary hue of the surround, an effect known as chromatic induction. This relativity of color can be used to enrich a display, and it has many applications in art and design. Nevertheless, it can also cause the viewer to see colors differently from the way the designer intended them. Chromatic induction can make the same colors look different or different colors look the same. With recognizable objects, judgments about the color are made in the present image according to memories that have been amassed from our experience of looking at similar objects. Studies have revealed a discrepancy between memory colors and the colors of actual objects and significant changes in saturation may occur in some cases.

Two objects of the same color may appear markedly different in color depending on background color. The ineffective use of colors can cause vibrations and shadows, images that distract the user and may cause eye strain.

V.OPTIMUM COLOR ASSIGNMENT

The proper use of color communicates facts and ideas more quickly and aesthetically to the user. Color can also help develop workable, efficient mental models if simplicity, consistency, clarity, and language of color are followed.

There is an inherent simplicity in color, which should be used when developing the design. The four physiologically primary colors are red, green, yellow and blue. These colors are easy to learn and remember.

A color scheme should be simple; according to Miller the magic number for short term memory is seven plus or minus two (7 ± 2). When using color in interface, the number of colors should not exceed five plus or minus two (5 ± 2). Limit the palette per screen to what the eye can actually keep track of at one glance, usually about six colors depending on the complexity of the screen design. The background color affects the effectiveness of the other colors. Pick an effective background color and use only about five colors. The number of colors on the screen should be limited accordingly. If the user is overwhelmed or confused by too many colors vying for his attention, he is unlikely to develop an effective mental model

4.2 CONSISTENCY

Consistency is vital when assigning meanings to colors. The intuitive ordering of colors can help establish intuitive consistency in the design. The spectral and perceptual order red, green, yellow, blue can guide the order of the concepts attached to colors. Red is first in the spectral order and focuses in the foreground, green and yellow focus in the middle, while blue focuses in the background.

Color can be used to encode or chunk information items. This helps increase the number of items a user can retain in short-term memory. Avoid changing the meaning of colors for different screens in the interface.

There are physiological aspects that hinder consistency in the use of color. Various shades of the same color should be avoided for different concepts and ideas. This is especially true for the blues. Different shades of blue are very difficult to distinguish and may not be recognized as different by the user. If the concept is different, use a different color. Avoid using colors that appear differently due to variation in background color. Clarity is also an important guideline for using color. Experiments have shown that the search time for finding an item is decreased if the color of the item is known ahead of time, and if the color only applies to that item. Standardized interface colors should be established and used across the development. The clear, concise use of color can help users find items more quickly and efficiently.

In conclusion, a few general key issues to consider in finalizing a color scheme are: 1.Discernibility — how easy it is to distinguish an item from its background 2.Conspicuity — how obvious is the item relative to other objects in the scene. 3.Salience — how well the item “pops out” from the display as a whole.

We list out some guidelines based on the above said conditions. A perfectly natural color system can be easily visualized if we adhere to the below said guidelines.

VI. COLOR SELECTION GUIDELINES BASED ON HUMAN COLOR VISION

1. Avoid adjacent areas of strong blue and strong red in a display to prevent unwanted depth effects (colors appearing to lie in different planes).
2. Never use the blue channel alone for fine detail such as text or graphics. Do not use, for example, blue text on a black background or yellow text on a white background.
3. Areas of strong color and high contrast can produce afterimages when the viewer looks away from the screen, resulting in visual stress from prolonged viewing. Do not use hue alone to encode information in applications where serious consequences might ensue if a color-deficient user were to make an incorrect selection.
4. Avoid the simultaneous display of highly saturated, spectrally extreme colors. This causes the lens to rapidly change shape and thus tires the eyes. De-saturate the colors or else use colors that are close together in the spectrum.
5. Pure blue should be avoided for text, thin lines, and small shapes. Since there are no blue cones in the center of the retina, these are difficult to see. But

blue makes an excellent background color, e.g. for a computer display it tends to blur the raster lines.

6. Avoid adjacent colors that differ only in the amount of blue. Since blue does not contribute to brightness, this creates fuzzy edges.
7. Older operators need higher brightness levels to distinguish colors.
8. Colors change in appearance as the ambient light level changes.
9. The magnitude of a detectable change in color varies across the spectrum.
10. It is difficult to focus upon edges created by color alone.
11. Avoid red and green in the periphery of large displays.
12. Opponent colors go well together.
13. For color-deficient observers, avoid single color distinctions.
14. Surrounding colors, field size, and viewing conditions can all change the appearance of colors.
15. Where accurate visual judgment of a color is necessary, the surrounding should be a neutral mid-gray to avoid unwanted perceptual color changes.

Color can described more meaningfully in terms of the perceptual dimensions of lightness, hue, and colorfulness than in terms of device signals.

The most common display-based models are RGB and YIQ/YUV.

The RGB Color Model:

The RGB color model is normally used on monitors, namely for display. The RGB color model is composed of the primary colors Red, Green, and Blue. This system defines the color model that is used in most color CRT monitors and color raster graphics. They are considered the "additive primaries" since the colors are added together to produce the desired color. The RGB model uses the Cartesian coordinate system. This color model is device-dependent and easy to implement, but is non intuitive for interpretation.

The International Commission on Illumination is a worldwide organization that developed the first version of the spectrally measured model known as CIE in 1931. CIE is a precise spectral measurement used to pinpoint colors and remove color ambiguity.

The CIE model is physically based; therefore, it does not fit well into either the perceptually based or the display-based categories.

All colors displayed on a computer must be translated into the RGB color space. Unfortunately, there is not a one-to-one mapping from the perceptually based models to the display-based. This fact can explain some of the difficulties encountered when recreating the right color for an interface screen. The exact shade is not always obtained. The CIE model allows translations from HSV to RGB.

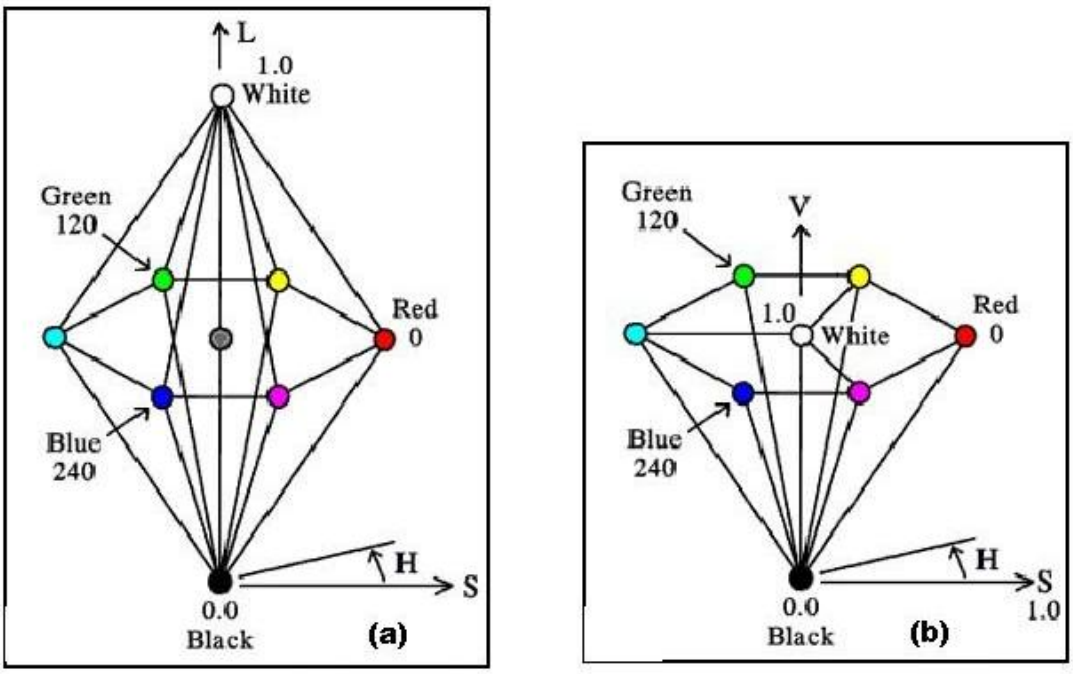
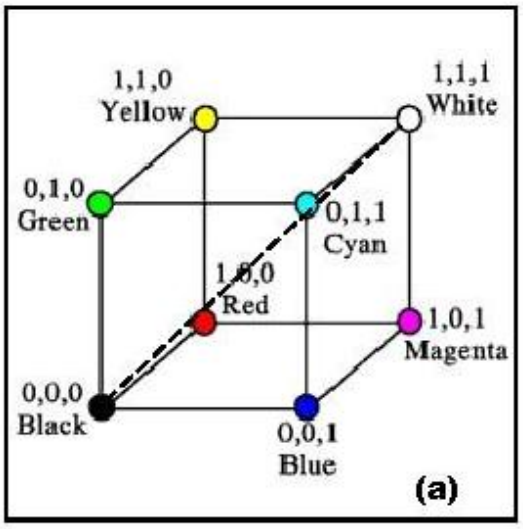


Fig5.1. (a) HLS – (Hue, Lightness, Saturation) developed by Gerald Murch at Tektronix, late 1970s, perhaps a “truer” representation than HSV, compared to RGB cube rotated (b) HSV - Hue, Saturation, Value



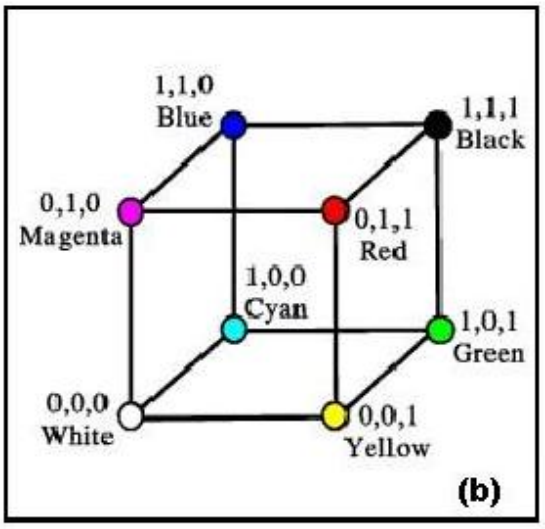


Fig5.2. (a) RGB color space **(b) CMYK color space**

VII. MATLAB CODE

```
function varargout = pseudogui(varargin)
% PSEUDOGUI M-file for pseudogui.fig
%      PSEUDOGUI, by itself, creates a new PSEUDOGUI or raises the existing
%      singleton*.
%
%      H = PSEUDOGUI returns the handle to a new PSEUDOGUI or the handle to
%      the existing singleton*.
%
%      PSEUDOGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PSEUDOGUI.M with the given input arguments.
%
%      PSEUDOGUI('Property','Value',...) creates a new PSEUDOGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before pseudogui_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to pseudogui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help pseudogui
% Last Modified by GUIDE v2.5 08-Feb-2008 15:44:56
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
   'gui_Singleton',  gui_Singleton, ...
   'gui_OpeningFcn', @pseudogui_OpeningFcn, ...
   'gui_OutputFcn',  @pseudogui_OutputFcn, ...
   'gui_LayoutFcn',  [] , ...
   'gui_Callback',   []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before pseudogui is made visible.
function pseudogui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pseudogui (see VARARGIN)
axes(handles.axes1); axis off;
axes(handles.axes2); axis off;
axes(handles.axes3); axis off;
axes(handles.axes4); axis off;
axes(handles.axes5); axis off;
% HSV Method Initialization
handles.hue1 = 0.7;
handles.hue2 = 0.4179;
handles.hue3 = 1;
handles.hue4 = 0.0850;
%Cosine Method Initialization
handles.freq = 0.9;
handles.ph = 0.2;
% Choose default command line output for pseudogui
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes pseudogui wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = pseudogui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[handles.fname, handles.pname]=uigetfile('*.jpg','Browse to....');
handles.img=imread([handles.pname handles.fname]);
handles.img = rgb2gray(handles.img);
mmax=max(max(handles.img));
mmax=double(mmax)/256;
handles.img = imadjust(handles.img,[0 mmax],[0 1]);
axes(handles.axes1);
imshow(handles.img); title('Input Image');
axes(handles.axes2);
imshow(imcomplement(handles.img)); title('Image Negative');
guidata(hObject, handles);
% --- Executes on button press in hsv.
function hsv_Callback(hObject, eventdata, handles)
% hObject    handle to hsv (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% size of the image
[m,n] = size(handles.img);
newImg = zeros(m,n,3); %rgb format
% Setting the threshold
threshold = [0 1000 10000 35000 65535]/256;
% Fix the selected Hue
%hue = [0.7, 0.4179, 1, 0.0850];
hue = [handles.hue1, handles.hue2, handles.hue3, handles.hue4];
% ex % orange, green, blue (0.0850, 0.4179, 0.58170)
for i=1:length(threshold)-1
   grayImg = imadjust(handles.img,[threshold(i) threshold(i+1)]/256, []);
   h = zeros(m,n);
   s = h;
```

```
    v = h;
    [I,J] = find((handles.img>=threshold(i)) & (handles.img<threshold(i+1)));
    ind = sub2ind([m,n],I,J);
    h(ind) = hue(i);
    s = 1;
    v(ind) = ((double(grayImg(ind))/65535)/2) + 0.1; % 0.1 is added as an offset
    vmax=max(max(v));
    v = imadjust(v,[0 vmax],[0 1]);
    hsvImg = zeros(m,n,3);
    hsvImg(:,:,1) = h;
    hsvImg(:,:,2) = s;
    hsvImg(:,:,3) = v;
    rgbImg = hsv2rgb(hsvImg); % convert HSV to RGB
    newImg = imadd(newImg,rgbImg);
end
axes(handles.axes3); imshow(newImg); title('Output Image');
% --- Executes on button press in cosine.
function cosine_Callback(hObject, eventdata, handles)
% hObject    handle to cosine (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%figure, imshow(img)
% Intensity adjustment
mmax=max(max(handles.img));
mmax=double(mmax)/256 ;
img1 = imadjust(handles.img,[0 mmax],[0 1]);
f=handles.freq; % Frequency
p=handles.ph; % Phase
[r,c]=size(img1);
z=double(img1(1:r,1:c))/256;
img2(1:r,1:c,1) = (abs(cos(2*pi*f*z)));
img2(1:r,1:c,2) = abs(cos(2*pi*f*z+.5*p));
img2(1:r,1:c,3) = abs(cos(2*pi*f*z+p));
axes(handles.axes4); imshow(img2); title('Output Image');
% --- Executes on button press in rainbow.
function rainbow_Callback(hObject, eventdata, handles)
% hObject    handle to rainbow (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% histogram of the image
% figure, imhist(img1,64)
[r,c]=size(handles.img);
% h = waitbar(0,'Processing...');
%level=input('Enter the number of COLOR levels:');
%level = 200;
level = str2double(inputdlg('Enter the Number of Color Levels(100-255)'));
for i=1:level
    map(i,1)=(1+cos((4*pi*double(i))/(3*255)))/2;
    map(i,2)=(1+cos((4*pi*double(i)/(3*255))-(2*pi/3)))/2;
    map(i,3)=(1+cos((4*pi*double(i)/(3*255))-(4*pi/3)))/2;
end
temp = ind2rgb(handles.img, map);
axes(handles.axes5); imshow(temp); title('Output Image');
% --- Executes during object creation, after setting all properties.
function hue1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to hue1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on slider movement.
function hue1_Callback(hObject, eventdata, handles)
% hObject    handle to hue1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
handles.hue1 = get(hObject,'Value');
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function hue2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to hue2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on slider movement.
function hue2_Callback(hObject, eventdata, handles)
% hObject    handle to hue2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
handles.hue2 = get(hObject,'Value');
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function hue3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to hue3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on slider movement.
function hue3_Callback(hObject, eventdata, handles)
% hObject    handle to hue3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
handles.hue3 = get(hObject,'Value');
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function hue4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to hue4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
% --- Executes on slider movement.
function hue4_Callback(hObject, eventdata, handles)
% hObject    handle to hue4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
handles.hue4 = get(hObject,'Value');
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function freq_CreateFcn(hObject, eventdata, handles)
% hObject    handle to freq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
   set(hObject,'BackgroundColor',[.9 .9 .9]);
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on slider movement.
function freq_Callback(hObject, eventdata, handles)
% hObject    handle to freq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
handles.freq = get(hObject,'Value');
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function ph_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
   set(hObject,'BackgroundColor',[.9 .9 .9]);
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on slider movement.
function ph_Callback(hObject, eventdata, handles)
% hObject    handle to ph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
handles.ph = get(hObject,'Value');
guidata(hObject, handles);
```

REFERENCES

[1] www.adobe.com/support/techguides/color/colormodels
[2] Heimann Systems: www.heimannsystems.com
[3] hyperphysics.phy-astr.gsu.edu/hbase/vision/cie.html
[4] wavelet/wavelet.html
[5] http://www.fas.org/irp/imint/niirs.htm
[6] Mathworks: www.mathworks.com