**Image Processing using MATLAB**

**ECE107L/E01**

**Submitted by:**

Nionel Ibarra

Christian Pantoja

**Submitted to:**

Engr. Leonardo D. Valiente Jr.

**Introduction**

This project contains the fundamentals of image processing techniques. Image processing is a method to perform some operations on an image, to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too.

Image processing has a lot of applications. It is a subfield of Computer Vision where each image is prepared for further analysis. You can get useful information from the images and let the computer interpret what does it contain. There are two types of methods used for image processing namely, analog and digital image processing. Analogue image processing can be used for the hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. Digital image processing techniques help in manipulation of the digital images by using computers. The three general phases that all types of data must undergo while using digital technique are pre-processing, enhancement, and display, information extraction.

There are two popular programming languages that have a good library of image processing techniques; Python and MATLAB. For this project we are required to use MATLAB to analyze digital images with a graphical user interface.

**Acknowledgement**

We would like to acknowledge Engr. Leonardo D. Valiente Jr. for imparting his knowledge on the topics and concepts covered in the subject of Digital Signal Processing. Such impartations have made this project design a possibility as well as a success; and since without enough suffice understanding on the concepts of the course, the group wouldn’t be able to construct, improve, and develop the given project.

**Program Description**

This project has a graphical user interface (GUI) that allows user to have a nice user interface when using it. The overall process of the project is the program will ask the user to select an image, only .jpg and .png file are supported and the image should be a 3-channel image (ex: RGB). When the user has input a valid image, it will appear on the first axes. Then all the remaining button is used to show each process part of the image. Each button has each own process has required by the project: conversion of channel, opening, increase and decrease in contrast, threshold and background subtraction.

Each of the associated processing of the image in each button has a corresponding user-defined function in MATLAB. Each user-defined function accepts an input of the path of the images that was obtained from the first part then it will process it. The algorithm of the image processing is handled by the MATLAB Image Processing Toolbox. The user just must know which techniques to use in each step to provide the proper output.

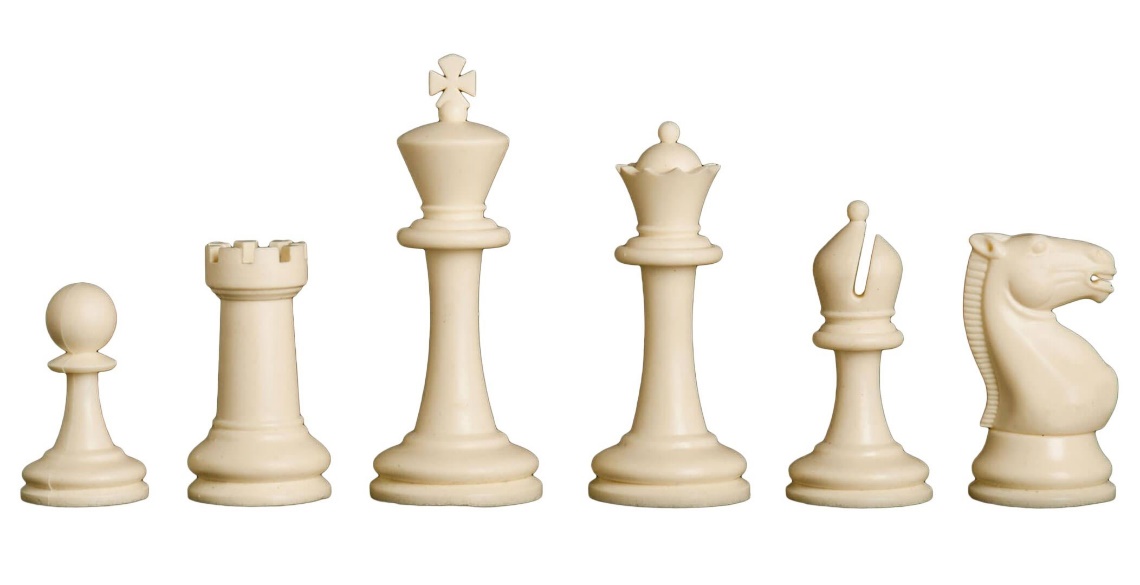
**Scope and Limitations**

* The input image must be a 3-channel image since all the steps in the image processing required a 3-channel image first before undergoing further analysis.
* The object and background must be highly emphasized so that it is properly distinguishable.
* The input image must have a uniform background. This is used in thresholding and background subtraction so that it can properly determine the required threshold.

**Screen Capture of The Program and Used Sample Images**

****

****

****

**M-files Codes**

dsp.m

function varargout = dsp(varargin)

% DSP MATLAB code for dsp.fig

% DSP, by itself, creates a new DSP or raises the existing

% singleton\*.

%

% H = DSP returns the handle to a new DSP or the handle to

% the existing singleton\*.

%

% DSP('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in DSP.M with the given input arguments.

%

% DSP('Property','Value',...) creates a new DSP or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before dsp\_OpeningFcn gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to dsp\_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 dsp

% Last Modified by GUIDE v2.5 07-Jan-2019 03:51:25

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @dsp\_OpeningFcn, ...

'gui\_OutputFcn', @dsp\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(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 dsp is made visible.

function dsp\_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 dsp (see VARARGIN)

% Choose default command line output for dsp

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes dsp wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = dsp\_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;

initialize(handles)

function initialize(handles)

movegui(gcf,'center');

axes(handles.axes1);

cla reset;

axes(handles.axes2);

cla reset;

set(handles.axes1,'XTick',[]);

set(handles.axes1,'YTick',[]);

set(handles.axes2,'XTick',[]);

set(handles.axes2,'YTick',[]);

global imageInput;

imageInput=0;

% --- 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)

global imageInput;

global imagePath;

[file,path]=uigetfile('\*.jpg;\*.png');

if ~isequal(file,0)

imagePath=fullfile(path,file);

axes(handles.axes1);

imshow(imagePath);

imageInput=1;

end

% --- Executes on button press in pushbutton2.

function pushbutton2\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global imageInput;

global imagePath;

if isequal(imageInput,1)

outputImage=func\_gray(imagePath);

axes(handles.axes2);

imshow(outputImage);

else

msgbox("No valid image input found","Error");

end

% --- Executes on button press in pushbutton3.

function pushbutton3\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton3 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global imageInput;

global imagePath;

if isequal(imageInput,1)

outputImage=func\_opening(imagePath);

axes(handles.axes2);

imshow(outputImage);

else

msgbox("No valid image input found","Error");

end

% --- Executes on button press in pushbutton4.

function pushbutton4\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global imageInput;

global imagePath;

if isequal(imageInput,1)

outputImage=func\_increase\_contrast(imagePath);

axes(handles.axes2);

imshow(outputImage);

else

msgbox("No valid image input found","Error");

end

% --- Executes on button press in pushbutton5.

function pushbutton5\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton5 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global imageInput;

global imagePath;

if isequal(imageInput,1)

outputImage=func\_decrease\_contrast(imagePath);

axes(handles.axes2);

imshow(outputImage);

else

msgbox("No valid image input found","Error");

end

% --- Executes on button press in pushbutton6.

function pushbutton6\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton6 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global imageInput;

global imagePath;

if isequal(imageInput,1)

outputImage=func\_threshold(imagePath);

axes(handles.axes2);

imshow(outputImage);

else

msgbox("No valid image input found","Error");

end

% --- Executes on button press in pushbutton7.

function pushbutton7\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton7 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global imageInput;

global imagePath;

if isequal(imageInput,1)

outputImage=func\_remove\_background(imagePath);

axes(handles.axes2);

imshow(outputImage);

else

msgbox("No valid image input found","Error");

end

% --- Executes on button press in inputCharButton.

function inputCharButton\_Callback(hObject, eventdata, handles)

% hObject handle to inputCharButton (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global imageInput;

global imagePath;

if isequal(imageInput,1)

imageinfo(imagePath);

else

msgbox("No valid image input found","Error");

end

% --- Executes on button press in outputCharButton.

function outputCharButton\_Callback(hObject, eventdata, handles)

% hObject handle to outputCharButton (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global imageInput;

if isequal(imageInput,1)

isEmptyAxes = isempty(get(handles.axes2, 'Children'));

if ~isEmptyAxes

currentImage=getimage(handles.axes2);

imwrite(currentImage,'currentview.jpg');

imageinfo('currentview.jpg');

else

msgbox("No processed image found.");

end

else

msgbox("No valid image input found","Error");

end

% --- Executes on button press in saveOutputImageButton.

function saveOutputImageButton\_Callback(hObject, eventdata, handles)

% hObject handle to saveOutputImageButton (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global imageInput;

if isequal(imageInput,1)

isEmptyAxes = isempty(get(handles.axes2, 'Children'));

if ~isEmptyAxes

[file,path]=uiputfile("./Saved Images/outputImage.jpg");

if ~isequal(file,0)

fullPath=fullfile(path,file);

imwrite(getimage(handles.axes2),fullPath);

msgbox("Image saved.");

end

else

msgbox("No processed image found.");

end

else

msgbox("No valid image input found","Error");

end

func\_gray

function grayImage=func\_gray(imagePath)

i=imread(imagePath);

if(size(i,3)==3)

grayImage=rgb2gray(i);

else

msgbox(["Input image must have 3 channels!","Ex: RGB"],"Error");

end

func\_decrease\_contrast.m

function decreasedContrast=func\_decrease\_contrast(imagePath)

i=imread(imagePath);

if(size(i,3)==3)

decreasedContrast=imadjust(i,[.5 .5 .5; .6 .6 .6],[]);

else

msgbox(["Input image must have 3 channels!","Ex: RGB"],"Error");

end

func\_increase\_contrast

function increasedContrast=func\_increase\_contrast(imagePath)

i=imread(imagePath);

if(size(i,3)==3)

increasedContrast=imadjust(i,[0 0 0; .6 .6 .6],[]);

else

msgbox(["Input image must have 3 channels!","Ex: RGB"],"Error");

end

func\_opening

function opening=func\_opening(imagePath)

i=imread(imagePath);

if(size(i,3)==3)

gray=rgb2gray(i);

filtered=imgaussfilt(gray,5);

se = strel('disk',10);

morp=imopen(filtered,se);

opening=morp;

else

msgbox(["Input image must have 3 channels!","Ex: RGB"],"Error");

end

func\_remove\_background

function removedBackground=func\_remove\_background(imagePath)

i=imread(imagePath);

if(size(i,3)==3)

gray=rgb2gray(i);

filtered=imgaussfilt(gray,5);

se = strel('disk',10);

morp=imopen(filtered,se);

level=graythresh(morp);

mask=imbinarize(morp,level);

mask=imcomplement(mask);

mask=imfill(mask,'holes');

masked=bsxfun(@times, i, cast(mask, 'like', i));

removedBackground=masked;

else

msgbox(["Input image must have 3 channels!","Ex: RGB"],"Error");

end

func\_threshold

function thresholded=func\_threshold(imagePath)

i=imread(imagePath);

if(size(i,3)==3)

gray=rgb2gray(i);

filtered=imgaussfilt(gray,5);

se = strel('disk',10);

morp=imopen(filtered,se);

level=graythresh(morp);

mask=imbinarize(morp,level);

mask=imcomplement(mask);

mask=imfill(mask,'holes');

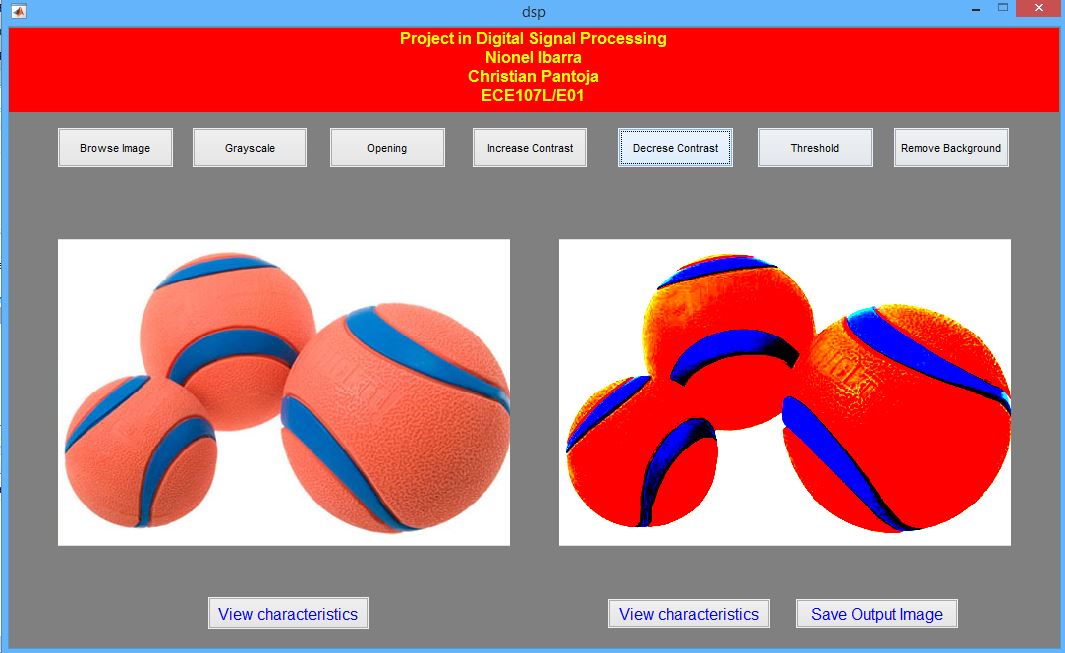
thresholded=mask;

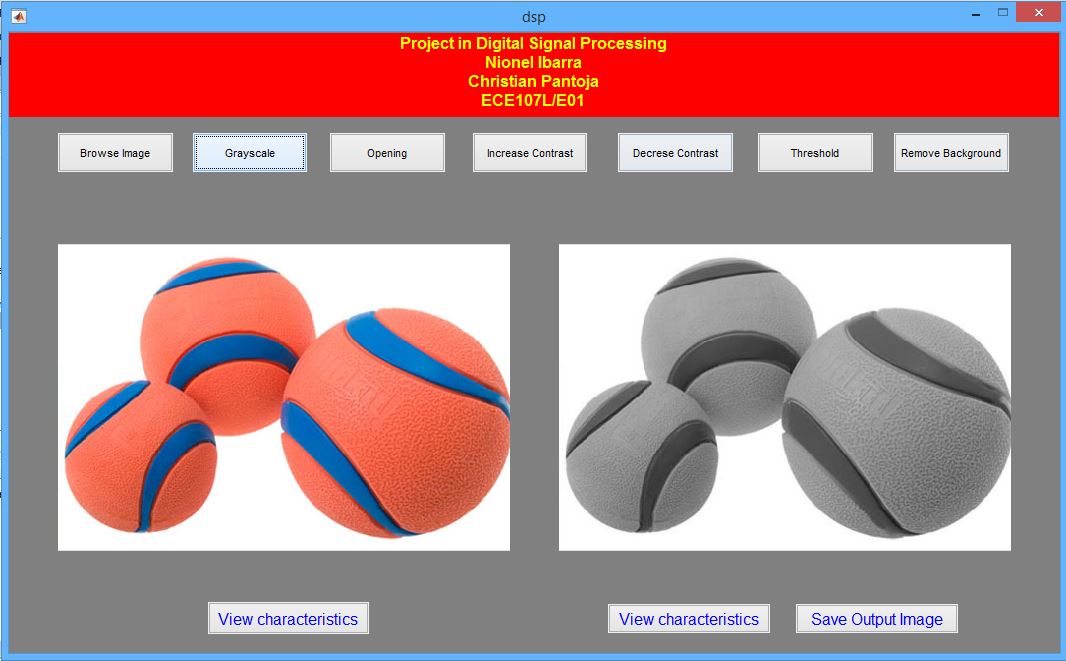
else

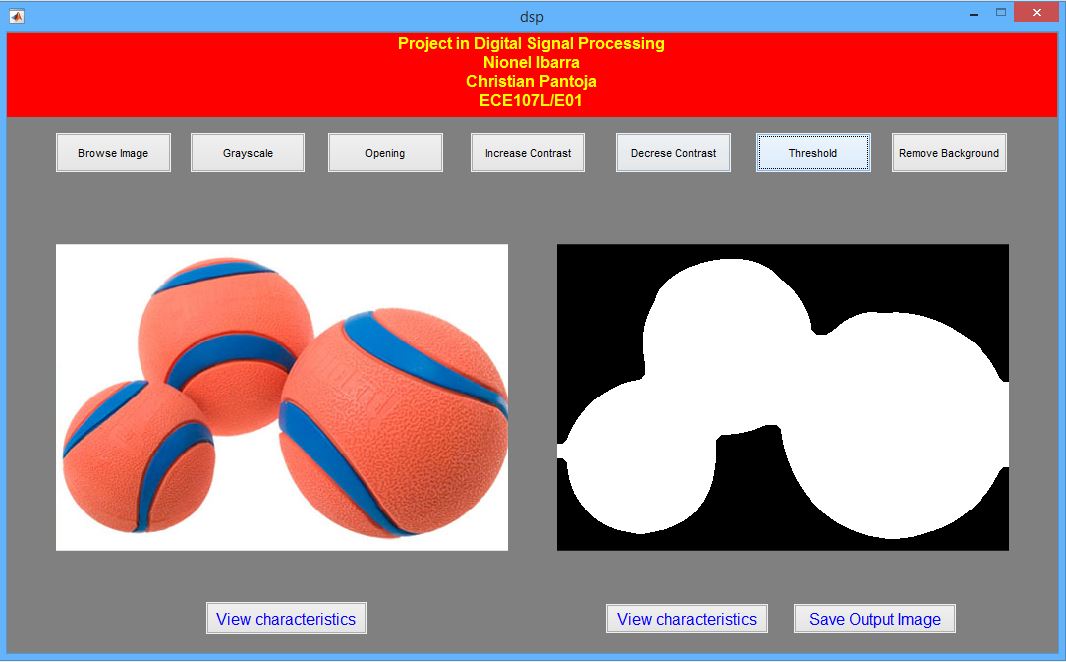
msgbox(["Input image must have 3 channels!","Ex: RGB"],"Error");

end

**Sample Outputs**

****

****

****

**References**

* Mathworks
* MATLAB Documentation
* MATLAB Image Processing Toolbox