Digital Signal Processing Laboratory

EE 384-02

Photo Editor Using MATLAB GUIDE

Nolan Anderson

7 December 2021

1. Introduction

This project implements a photo editor using MATLAB's GUIDE tool. As a GUI, you can change the properties, or the "look" of an imported photo. You are then able to export this edited image. This application implements several image processing features such as contrast, noise, sharpness, grayscale, inverting colors and graphs for the RGB channels. These features will be covered in the theory section below.

2. Theory

2.1 Image Processing

Image processing is the action of manipulating digital images using a computer. Because images are matrices populated with values, we can manipulate those values to change the way an image looks. There are two main types of images, color, and grayscale. Grayscale images are one-dimensional, whereas RBG color images are three-dimensional. This makes image editing a very easy process as we essentially just perform elementary math operations on images to change their properties. Image processing is very useful for many applications such as extracting more information from a given image.

2.2 MATLAB GUIDE

MATLAB GUIDE is a built in GUI editor much like QT. You can convert scripts into simple apps interactively or programmatically. It is a powerful tool for implementing code into a more useable form for the end user. GUIDE has many features including a drag-and-drop for different components for easier development. It also has all the features MATLAB does so error detection is built in. Continually, adding functionality is simple since you just click add and write your function. The files are saved as a .mlapp and can be converted into an App Designer App. Overall it is very useful tool for implementing GUIs quickly and efficiently.

2.3 Contrast / Noise / Sharpening

Contrast is the action of enhancing luminance or color which allows the viewer to distinguish between different objects. Increasing contrast gives the image more depth, decreasing does the opposite. Noise is a random variation of brightness or color in images, and results in a "grainy" image. All images have some level of grain in them, but by adjusting the pixel values of an image we can add more grain or take some

away, but this is very difficult to do. In this code I will implement salt and pepper noise. Sharpening is the action of increasing the acutance of an image. This does not change resolution but modifies the apparent sharpness of the image. Acutance essentially sharpens / adds contras to the edges in an image.

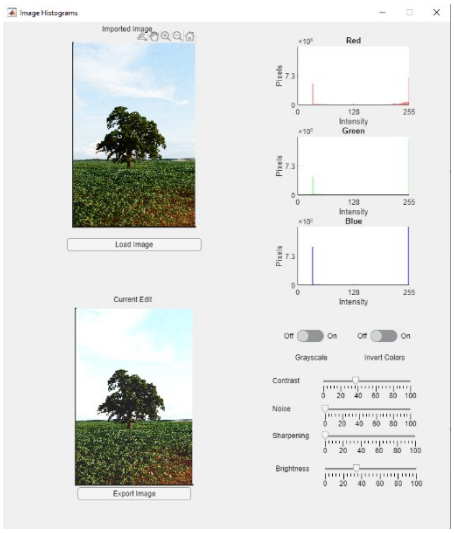
2.4 Grayscale / Inverting Colors / Brightness

Inverting colors is simply that, you just take each pixel value and completely invert it. To change from RGB to grayscale you use the following formula: 30% x Red Pixels + 60% x Green Pixels + 10% x Blue Pixels. In MATLAB use the formula: grayImage = .299*double(redChannel) + .587*double(greenChannel) + .114*double(blueChannel). This simply changes the values of each pixel and then stores it into a new image. Adjusting brightness is a simple math operation. Just add the value to the original image and it will adjust every pixel by that value.

2.5 RGB Channel Graphs

The RGB graph channels represent each color pixel value at a certain location in the image. Some areas may have more blue than green represented, therefore the graphs will show bluer for that area. Implementation is straightforward as well as you just display a histogram for each RGB value of the photo.

3. Simulation



As this is an application, simulation is straightforward. First store the code in an .mlapp file and then open it in MATLAB. Press the Run button and then import an image. Adjust using the sliders and switches, and then export the image once you are finished.

4. Problems I Faced

Initially I was able to write the code for all the features with the built-in functions (other than the grayscale) but had a lot of difficulty writing my own functions. Namely contrast was a difficult one as the histogram equalization function would always make the image completely black. Continually, the noise function was difficult to implement and had issues where it would make the image white or even duplicating the image. The biggest issue I ran into is not being able to undo edits. I attempted to write some code for it, however, undoing edits in MATLAB is very difficult and often would completely freeze the program, make the image worse, or just not do anything all

together. I also attempted to implement it outside of the GUI and got the same results. Overall, the project was relatively smooth with a few minor hiccups.

5. Results

Before

Contrast / Sharpening

Inverting Colors and Adding Noise

Grayscale & Brightness

As you can see, the app successfully edits the images. The top left image is before, and the rest are after with the changes noted below. One odd thing is the grayscale inverts the colors for some reason. Even using the built in rgb2gray function in MATLAB the app produces the same results, so it may be a limitation of the GUIDE tool. Another issue I had was undoing edits of an image. Unfortunately, I have only been able to add contrast, brightness etc. to the image. Doing some research, it seems very difficult to take away these properties once they have been added to MATLAB, especially without using the built-in functions. The app can get rid of grayscale and invert colors, however. Overall, while I was successful in editing the photos using MATLAB, the app has some significant limitations.

6. Demonstration / Files

All code and before / after images will be uploaded in this zip file and on the Google Drive. Continually, a demonstration will be placed in both locations as well.

Google Drive:
https://drive.google.com/drive/folders/142TJ5uZcltJPvqbKI2hc4i10-QoDEbeN?usp=sharing

7. Appendix

***** Make sure to copy paste and save as a .mlapp file ******

```
classdef Anderson_edit_FinalProject < matlab.apps.AppBase


% Properties that correspond to app components
properties (Access = public)
UIFigure matlab.ui.Figure
LoadButton matlab.ui.control.Button
ImportedImageLabel matlab.ui.control.Label
CurrentEditLabel matlab.ui.control.Label
ExportImageButton matlab.ui.control.Button
ContrastSliderLabel matlab.ui.control.Label
ContrastSlider matlab.ui.control.Slider
GrayscaleSwitchLabel matlab.ui.control.Label
GrayscaleSwitch matlab.ui.control.Switch
NoiseSliderLabel matlab.ui.control.Label
NoiseSlider matlab.ui.control.Slider
SharpeningSliderLabel matlab.ui.control.Label
SharpeningSlider matlab.ui.control.Slider
InvertColorsSwitchLabel matlab.ui.control.Label
InvertColorsSwitch matlab.ui.control.Switch
BrightnessSliderLabel matlab.ui.control.Label
BrightnessSlider matlab.ui.control.Slider
```

```
ImageAxes matlab.ui.control.UIAxes
GreenAxes matlab.ui.control.UIAxes
BlueAxes matlab.ui.control.UIAxes
RedAxes matlab.ui.control.UIAxes
ImageAxes_2 matlab.ui.control.UIAxes
end


methods (Access = private)
function updateimage(app,imagefile)
if strcmp(imagefile,'corn.tif')
im = imread('corn.tif', 2);
else
try
im = imread(imagefile);
catch ME
uialert(app.UIFigure, ME.message, 'Image Error');
return;
end
end
switch size(im,3)
case 1
imagesc(app.ImageAxes,im);
imagesc(app.ImageAxes_2,im);
histr = histogram(app.RedAxes, im, 'FaceColor',[1 0 0],'EdgeColor', 'none');
histg = histogram(app.GreenAxes, im, 'FaceColor',[0 1 0],'EdgeColor', 'none');
histb = histogram(app.BlueAxes, im, 'FaceColor',[0 0 1],'EdgeColor', 'none');
case 3
imagesc(app.ImageAxes,im);
imagesc(app.ImageAxes_2,im);
histr = histogram(app.RedAxes, im(:,:,1), 'FaceColor', [1 0 0], 'EdgeColor',
'none');
histg = histogram(app.GreenAxes, im(:,:,2), 'FaceColor', [0 1 0], 'EdgeColor',
'none');
histb = histogram(app.BlueAxes, im(:,:,3), 'FaceColor', [0 0 1], 'EdgeColor',
'none');
otherwise
uialert(app.UIFigure, 'Image must be grayscale or truecolor.', 'Image Error');
return;
end
maxr = max(histr.BinCounts);
maxg = max(histg.BinCounts);
maxb = max(histb.BinCounts);
maxcount = max([maxr maxg maxb]);
app.RedAxes.YLim = [0 maxcount];
app.RedAxes.YTick = round([0 maxcount/2 maxcount], 2, 'significant');
app.GreenAxes.YLim = [0 maxcount];
app.GreenAxes.YTick = round([0 maxcount/2 maxcount], 2, 'significant');
app.BlueAxes.YLim = [0 maxcount];
app.BlueAxes.YTick = round([0 maxcount/2 maxcount], 2, 'significant');
```

```
end
function exportimage(app)
[Filename, Filepath] = uiputfile('*.jpg','*.tif', 'Save as');
Name = fullfile(Filepath, Filename);
axes2 = get(app.ImageAxes_2, 'Children');
W = get(axes2(1), 'CData');
imwrite(W, Name, 'jpg');
end
function applycontrast(app, value)
L = get(app.ContrastSlider, 'Value');
im = getimage(app.ImageAxes_2);
imgMin = double(min(im(:)));
imgMax = double(max(im(:)));
img = (im - imgMin) / (imgMax - imgMin) * L;
img = my_histeq(app, im, 255);
imh = imhandles(app.ImageAxes_2);
img = imadjust(im, [L, 255]/255, [0, 255] / 255);
set(imh, 'CData', img);
updatergbaxes(app);
end
function grayscale(app, value)
val = get(app.GrayscaleSwitch, 'Value');
if (val == 1)
axes2 = get(app.ImageAxes_2, 'Children');
axes2 = get(app.ImageAxes_2);
redChannel = axes2(:, :, 1);
greenChannel = axes2(:, :, 2);
blueChannel = axes2(:, :, 3);
grayImage = .299*double(redChannel) + .587*double(greenChannel) +
.114*double(blueChannel);
grayImage = uint8(grayImage);
imh = imhandles(app.ImageAxes_2);
set(imh, 'CData', grayImage);
else
axes2 = get(app.ImageAxes_2, 'Children');
W = get(axes2(1), 'CData');
invertgray = uint8(255) - W;
imh = imhandles(app.ImageAxes_2);
set(imh, 'CData', invertgray);
end
updatergbaxes(app);
end
function sharpening(app, value)
L = get(app.SharpeningSlider, 'Value');
A = getimage(app.ImageAxes_2);
W = [0 1 0;1 -4 1; 0 1 0];
A = padarray(A,[1,1]);
IMG1 = A;
I = zeros(size(A));
```

```
A = double(A);
for i = 2:size(A,1)-1
for j = 2:size(A,2)-1
I(i,j) = sum(sum(W .* A(i-1:i+1,j-1:j+1)));
end
end
I = uint8(I);
B = IMG1 - I;
imh = imhandles(app.ImageAxes_2);
set(imh, 'CData', B);
updatergbaxes(app);
end
function invertcolors(app, value)
axes2 = get(app.ImageAxes_2, 'Children');
W = get(axes2(1), 'CData');
Wnew = 255-W;
imh = imhandles(app.ImageAxes_2);
set(imh, 'CData', Wnew);
updatergbaxes(app);
end
function hist_img = my_histeq(app, im, L)
% Forming 0's of length L.
c = zeros(255, 1);
for i = 0:255 - 1
c(i+1) = sum(sum(255 == i));
end
p = c / (size(im, 1) * 255); % Size of image
s = (L-1) * cumsum(p); % Cumulative hist of each pixel
s = round(s); % CDF Of each pixel
hist_img = uint8(zeros(255)); % Convert size.
for k=1:size(s,1) % Replace final value of each pixel
hist_img(255 == k-1) = s(k) / 255;
end
end
function noise(app, value)
L = get(app.NoiseSlider, 'Value');
my_image = getimage(app.ImageAxes_2);
image_thresholded = my_image;
a=0.05; b=0.05;
X=rand(size(my_image,1),size(my_image,2));
c= X<=a;
image_thresholded(c)=0;
u=a+b;
c=find(X>a & X<=u);
image_thresholded(c)=1;
imh = imhandles(app.ImageAxes_2);
set(imh, 'CData', image_thresholded);
updatergbaxes(app);
end
```

```
function updatergbaxes(app)
im = getimage(app.ImageAxes_2);
imagesc(app.ImageAxes_2,im);
histr = histogram(app.RedAxes, im(:,:,1), 'FaceColor', [1 0 0], 'EdgeColor',
'none');
histg = histogram(app.GreenAxes, im(:,:,2), 'FaceColor', [0 1 0], 'EdgeColor',
'none');
histb = histogram(app.BlueAxes, im(:,:,3), 'FaceColor', [0 0 1], 'EdgeColor',
'none');
maxr = max(histr.BinCounts);
maxg = max(histg.BinCounts);
maxb = max(histb.BinCounts);
maxcount = max([maxr maxg maxb]);
app.RedAxes.YLim = [0 maxcount];
app.RedAxes.YTick = round([0 maxcount/2 maxcount], 2, 'significant');
app.GreenAxes.YLim = [0 maxcount];
app.GreenAxes.YTick = round([0 maxcount/2 maxcount], 2, 'significant');
app.BlueAxes.YLim = [0 maxcount];
app.BlueAxes.YTick = round([0 maxcount/2 maxcount], 2, 'significant');
end
function adjustbrightness(app, value)
L = get(app.BrightnessSlider, 'Value');
im = getimage(app.ImageAxes_2);
imb = im + L;
imh = imhandles(app.ImageAxes_2);
set(imh, 'CData', imb);
updatergbaxes(app);
end
end


% Callbacks that handle component events
methods (Access = private)


% Code that executes after component creation
function startupFcn(app)
% Configure image axes
app.ImageAxes.Visible = 'on';
app.ImageAxes_2.Visible = 'on';
app.ImageAxes.Colormap = gray(256);
axis(app.ImageAxes, 'image');
axis(app.ImageAxes_2, 'image');
end


% Callback function
function DropDownValueChanged(app, event)
% Update the image and histograms
updateimage(app, app.DropDown.Value);
```

```
end


% Button pushed function: LoadButton
function LoadButtonPushed(app, event)
% Display uigetfile dialog
filterspec = {'*.jpg;*.tif;*.png;*.gif; *.bmp','All Image Files'};
[f, p] = uigetfile(filterspec);
% Make sure user didn't cancel uigetfile dialog
if (ischar(p))
fname = [p f];
updateimage(app, fname);
end
end


% Button pushed function: ExportImageButton
function ExportImageButtonPushed(app, event)
exportimage(app);
end


% Value changed function: ContrastSlider
function ContrastSliderValueChanged(app, event)
value = app.ContrastSlider.Value;
applycontrast(app, value);
end


% Value changed function: GrayscaleSwitch
function GrayscaleSwitchValueChanged(app, event)
value = app.GrayscaleSwitch.Value;
grayscale(app, value);
end


% Value changed function: InvertColorsSwitch
function InvertColorsSwitchValueChanged(app, event)
value = app.InvertColorsSwitch.Value;
invertcolors(app, value);
end


% Value changed function: SharpeningSlider
function SharpeningSliderValueChanged(app, event)
value = app.SharpeningSlider.Value;
sharpening(app, value);
end
```

```
% Value changed function: NoiseSlider
function NoiseSliderValueChanged(app, event)
value = app.NoiseSlider.Value;
noise(app, value);
end


% Value changed function: BrightnessSlider
function BrightnessSliderValueChanged(app, event)
value = app.BrightnessSlider.Value;
adjustbrightness(app, value);
end
end


% Component initialization
methods (Access = private)


% Create UIFigure and components
function createComponents(app)


% Create UIFigure and hide until all components are created
app.UIFigure = uifigure('Visible', 'off');
app.UIFigure.AutoResizeChildren = 'off';
app.UIFigure.Position = [100 100 750 849];
app.UIFigure.Name = 'Image Histograms';
app.UIFigure.Resize = 'off';


% Create LoadButton
app.LoadButton = uibutton(app.UIFigure, 'push');
app.LoadButton.ButtonPushedFcn = createCallbackFcn(app, @LoadButtonPushed,
true);
app.LoadButton.Position = [109 466 225 22];
app.LoadButton.Text = 'Load Image';


% Create ImportedImageLabel
app.ImportedImageLabel = uilabel(app.UIFigure);
app.ImportedImageLabel.Position = [168 827 90 22];
app.ImportedImageLabel.Text = 'Imported Image';


% Create CurrentEditLabel
app.CurrentEditLabel = uilabel(app.UIFigure);
app.CurrentEditLabel.Position = [187 374 70 22];
app.CurrentEditLabel.Text = 'Current Edit';
```

```
            % Create ExportImageButton
            app.ExportImageButton = uibutton(app.UIFigure, 'push');
            app.ExportImageButton.ButtonPushedFcn = createCallbackFcn(app,
@ExportImageButtonPushed, true);
            app.ExportImageButton.Position = [127 49 190 22];
            app.ExportImageButton.Text = 'Export Image';


            % Create ContrastSliderLabel
            app.ContrastSliderLabel = uilabel(app.UIFigure);
            app.ContrastSliderLabel.Position = [453 238 51 22];
            app.ContrastSliderLabel.Text = 'Contrast';


            % Create ContrastSlider
            app.ContrastSlider = uislider(app.UIFigure);
            app.ContrastSlider.ValueChangedFcn = createCallbackFcn(app,
@ContrastSliderValueChanged, true);
            app.ContrastSlider.Position = [538 247 146 3];


            % Create GrayscaleSwitchLabel
            app.GrayscaleSwitchLabel = uilabel(app.UIFigure);
            app.GrayscaleSwitchLabel.HorizontalAlignment = 'center';
            app.GrayscaleSwitchLabel.Position = [488 277 60 22];
            app.GrayscaleSwitchLabel.Text = 'Grayscale';


            % Create GrayscaleSwitch
            app.GrayscaleSwitch = uiswitch(app.UIFigure, 'slider');
            app.GrayscaleSwitch.ValueChangedFcn = createCallbackFcn(app,
@GrayscaleSwitchValueChanged, true);
            app.GrayscaleSwitch.Position = [494 314 45 20];


            % Create NoiseSliderLabel
            app.NoiseSliderLabel = uilabel(app.UIFigure);
            app.NoiseSliderLabel.Position = [453 191 36 22];
            app.NoiseSliderLabel.Text = 'Noise';


            % Create NoiseSlider
            app.NoiseSlider = uislider(app.UIFigure);
            app.NoiseSlider.ValueChangedFcn = createCallbackFcn(app,
@NoiseSliderValueChanged, true);
            app.NoiseSlider.Position = [541 200 143 3];
```

```
            % Create SharpeningSliderLabel
            app.SharpeningSliderLabel = uilabel(app.UIFigure);
            app.SharpeningSliderLabel.Position = [453 146 73 22];
            app.SharpeningSliderLabel.Text = 'Sharpening';


            % Create SharpeningSlider
            app.SharpeningSlider = uislider(app.UIFigure);
            app.SharpeningSlider.ValueChangedFcn = createCallbackFcn(app,
@SharpeningSliderValueChanged, true);
            app.SharpeningSlider.Position = [541 155 151 3];


            % Create InvertColorsSwitchLabel
            app.InvertColorsSwitchLabel = uilabel(app.UIFigure);
            app.InvertColorsSwitchLabel.HorizontalAlignment = 'center';
            app.InvertColorsSwitchLabel.Position = [604 277 74 22];
            app.InvertColorsSwitchLabel.Text = 'Invert Colors';


            % Create InvertColorsSwitch
            app.InvertColorsSwitch = uiswitch(app.UIFigure, 'slider');
            app.InvertColorsSwitch.ValueChangedFcn = createCallbackFcn(app,
@InvertColorsSwitchValueChanged, true);
            app.InvertColorsSwitch.Position = [617 314 45 20];


            % Create BrightnessSliderLabel
            app.BrightnessSliderLabel = uilabel(app.UIFigure);
            app.BrightnessSliderLabel.HorizontalAlignment = 'right';
            app.BrightnessSliderLabel.Position = [453 91 62 22];
            app.BrightnessSliderLabel.Text = 'Brightness';


            % Create BrightnessSlider
            app.BrightnessSlider = uislider(app.UIFigure);
            app.BrightnessSlider.ValueChangedFcn = createCallbackFcn(app,
@BrightnessSliderValueChanged, true);
            app.BrightnessSlider.Position = [541 100 153 3];


            % Create ImageAxes
            app.ImageAxes = uiaxes(app.UIFigure);
            app.ImageAxes.XTick = [];
            app.ImageAxes.XTickLabel = {'[ ]'};
            app.ImageAxes.YTick = [];
            app.ImageAxes.LineWidth = 2;
            app.ImageAxes.Position = [43 502 357 318];
```

```
            % Create GreenAxes
            app.GreenAxes = uiaxes(app.UIFigure);
            title(app.GreenAxes, 'Green')
            xlabel(app.GreenAxes, 'Intensity')
            ylabel(app.GreenAxes, 'Pixels')
            app.GreenAxes.XLim = [0 255];
            app.GreenAxes.XTick = [0 128 255];
            app.GreenAxes.Position = [456 525 236 152];


            % Create BlueAxes
            app.BlueAxes = uiaxes(app.UIFigure);
            title(app.BlueAxes, 'Blue')
            xlabel(app.BlueAxes, 'Intensity')
            ylabel(app.BlueAxes, 'Pixels')
            app.BlueAxes.XLim = [0 255];
            app.BlueAxes.XTick = [0 128 255];
            app.BlueAxes.Position = [456 374 236 152];


            % Create RedAxes
            app.RedAxes = uiaxes(app.UIFigure);
            title(app.RedAxes, 'Red')
            xlabel(app.RedAxes, 'Intensity')
            ylabel(app.RedAxes, 'Pixels')
            app.RedAxes.XLim = [0 255];
            app.RedAxes.XTick = [0 128 255];
            app.RedAxes.Position = [456 676 236 152];


            % Create ImageAxes_2
            app.ImageAxes_2 = uiaxes(app.UIFigure);
            app.ImageAxes_2.XTick = [];
            app.ImageAxes_2.XTickLabel = {'[ ]'};
            app.ImageAxes_2.YTick = [];
            app.ImageAxes_2.LineWidth = 2;
            app.ImageAxes_2.Position = [43 70 357 305];


            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end


    % App creation and deletion
    methods (Access = public)


        % Construct app
```

```
        function app = Anderson_edit_FinalProject


            % Create UIFigure and components
            createComponents(app)


            % Register the app with App Designer
            registerApp(app, app.UIFigure)


            % Execute the startup function
            runStartupFcn(app, @startupFcn)


            if nargout == 0
                clear app
            end
        end


        % Code that executes before app deletion
        function delete(app)


            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```