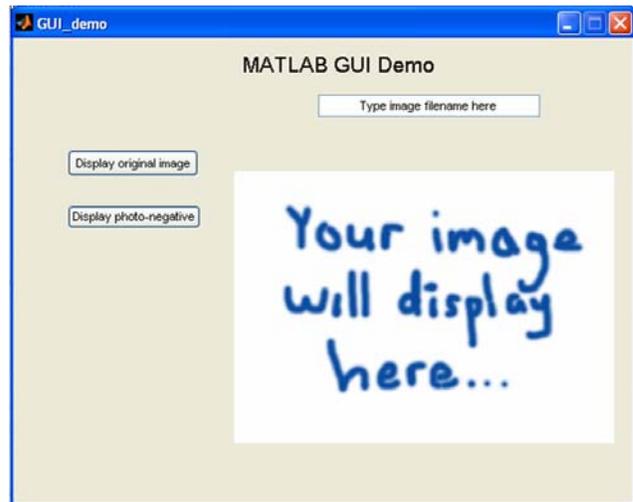


EE435: Biometric Signal Processing MATLAB GUI Tutorial

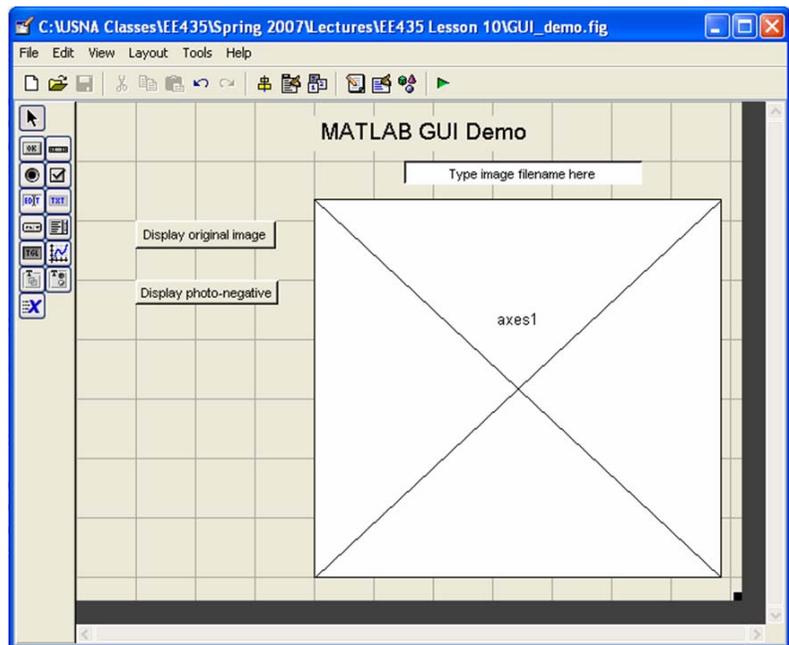
This tutorial will lead you through the creation of a simple MATLAB GUI. The GUI will allow a user to enter the name of an image file, and then enable the display of the image, and also its photo-negative. A picture of what your end product should look like is shown to the right. Each GUI is composed of two parts: a .fig file (which determines what the GUI looks like) and an m-file (which determines what happens in the GUI). The process starts by creating the layout of the GUI using the .fig file; the functionality is added later in the m-file.



1. Before we start building the GUI, download the following images from the course website (under *In-Class Activities*): *Default-image.bmp*, *Error-image.png*, *barbara.png*, *afghan2.jpg*, and *soccer.jpg*. These will be used in the GUI you develop, when the GUI is completed.
2. At the command line, type `>> guide`
This starts the MATLAB GUI Development Environment.
3. Select the type of GUI. For this tutorial, choose “Blank GUI (Default)”. This will create a blank GUI (a working .fig file).
4. On the left side of the palette, there are buttons which represent the different types of objects that can be added to the GUI. To begin, drag one of the “Static Text” objects over to the top of the grid. *Static Text* is text that never changes. You will use this for the title of your GUI. Place this object at the top of the grid.
 - a. Double-click on this Static Text object, and you will see the properties associated with it (the “Property Inspector” window will appear). The properties associated with this object include settings such as its location on the grid, font type, font size, and other such information.
 - b. Using the Property Inspector window, change the font size to 14 and make it bold (“font weight”).
 - c. Change the “String” property to “MATLAB GUI DEMO” instead of “Static Text”...this is the title of your GUI. The font size is too large for the object you originally created, so drag the corner of the *Static Text* object to make it bigger.
4. Drag an “Axes” box onto the grid, and enlarge it to what seems a reasonable size. Note the name of this object: “Axes1”. If we added more axes, they would be named “Axes2”, “Axes3”, etc.
5. Drag an “Edit Text” box onto the grid. “Edit Text” is different than “Static Text” because it can change. We will use this so that the user can enter the image filename to display.

- a. Set the *Edit Text* box on top of *Axes1*. Change its “String” property to the text that will appear when the GUI is started up...such as “Type image filename here”.
 - b. Enlarge the Edit Text box to be as wide as the axes below it.
6. Drag two “Push Buttons” to the grid, one on top of the other. One will be named “pushbutton1” and the other “pushbutton2”.
- a. Double-click on the top push button and change the “String” to “Display original image”.
 - b. Double-click on the bottom push button and change its “String” to “Display photo-negative”.
- c. Elongate the boxes so the text fits.

7. Save the GUI figure (currently called “untitled.fig”). Save it as “GUI_demo.fig”. Your end product should look something like the figure to the right. At this point (after you save the .fig file), MATLAB will create an *m*-file that specifies the functionality of the GUI: what each button does, what the axes is supposed to show, etc. Now we edit the *m*-file to complete the GUI.



8. Most of the code is background code MATLAB needs to run the GUI, and should not be altered. It consists of a number of MATLAB functions specific to this GUI, and in order to make the GUI work as advertised, we need to add some lines to some of the functions.
9. The function called “GUI_demo_OpeningFcn” (at approximately line 50 of the *m*-file) controls what happens when the GUI is started. For example, we could have the GUI display a default image on *axes1*. For this GUI, let’s initially display an image I made up called “Default-image.bmp”. Add the following lines at the end of this function:

```
axes(handles.axes1);
imshow('Default-image.bmp');
```

10. When the user pushes the “Display original image” button, the image should display on *axes1*, as long as the image file exists. To accomplish this, add the following lines at the end of the *pushbutton1_callback* function:

```
image_file = get(handles.edit1,'String'); % this "gets" the image file name the user typed in
                                                % and calls it "image_file"
if exist(image_file) % check if image exists
    im_original=imread(image_file); % read in the image
```

```

        axes(handles.axes1);           % will display on axes1
        imshow(im_original,[]);
    else                               % if image doesn't exist
        imshow('Error-image.png')     % Display the "Error image"
        disp(sprintf('Error in GUI_demo.m: could not find image called "%s" !!',image_file));
        % display error message in command window
    end
end

```

11. When the user pushes the “Display photo-negative” button, the photo-negative image should display on axes1, again as long as the image file exists. To accomplish this, add the following lines at the end of the *pushbutton2_callback* function:

```

    image_file = get(handles.edit1,'String'); % this "gets" the image file name
                                                % and calls it "image_file"
    if exist(image_file)                       % check if image exists
        im_original=imread(image_file); % read in the image
        axes(handles.axes1);
        imshow(imcomplement(im_original),[]);
    else                                       % if image doesn't exist
        imshow('Error-image.png')         % Display the "Error image"
        disp(sprintf('Error in GUI_demo.m: could not find image called "%s" !!',image_file));
        % display error message in command window
    end
end

```

12. After the m-file is saved, you can run the GUI at the command line with
 >> GUI_demo.

13. This concludes this tutorial. You can change the GUI *m*-file as desired, and as long as you save the *m*-file, the changes will be incorporated. You can add things to the .fig file (such as more push buttons, or slider bars, etc.), and when you save the .fig file, the *m*-file is updated to add functions for these. If you add additional buttons, etc., you need to add code to the *m*-file to give the additional blocks some functionality.

14. If you add another axes to display two images at once, then before you use a command like *imshow* in the 2nd axes, you need to let the GUI know where to display it. For example, if you created a 2nd axes in the .fig file, you'd use code like this in the *m*-file to display on that axes:

```

    axes(handles.axes2);
    imshow(threshold(im_original,200),[ ]);           % display thresholded image on axes2

```

15. One thing you probably don't want to do is to rename the .fig file or the .m file after you initially save it; this will cause problems, as various properties of each object in the GUI refer to the original name of the GUI. For example, on GUI_demo.fig, double-click on the “Display original image” pushbutton. Notice that the call-back functions for this pushbutton is “GUI_demo(...)”. Depending on the complexity of the GUI, it can be challenging to rename a GUI and have it work as advertised, since a good number of the properties of each object may need to be changed, NOT ONLY in the .fig file, but ALSO in the *m*-file.

16. Finally, if you want to come back tomorrow to change the look of the GUI, you can edit the .fig file using the command: `>> guide GUI_demo.fig`.
17. Once you're done with the GUI, compile the MATLAB code into a Windows executable (.exe) file so that any Windows machine, even w/o MATLAB can run it. At the command line, type:

```
>> mcc -m -g -v GUI_demo
```

This command may take a few minutes to run, and you might need to select which C++ compiler to have MATLAB use (choose Visual C++ if offered), but when done, you should see a file called "GUI_demo.exe" in your directory. It will bring up something that looks like your GUI in MATLAB, but Windows will be running it, not MATLAB. Not all MATLAB code can be compiled into a Windows executable file, but most can.