SOFTWARE OPERATIONS MANUAL

Contents

Introduction

There are three MATLAB associated files with the software build of the P13361. The first is the "IMGPRO_TOOL.fig" file. The .fig has the associated locations of each pushbutton. Any edits to position of buttons or options should be done to this file, which is further explained in section 1.

Section 2 is discusses the "IMGPRO_TOOL.m", this file is responsible for interacting with the fig file and triggering each pushbutton call. IMGPRO_TOOL keeps account for all the state variables and sends out the appropriate filter command to "Imgpro_filterfnc.m" function file

"Imgpro_filterfnc" is the last function file. Here, the state variables set from IMGPRO_TOOL are used to display the new picture on the screen.

Variables of interest

handles

Since each button in the GUI is its own function file, a method to pass one variable from one function to another. Handles variable is a predefined MATLAB variable that is used to pass variables from one function to another. After each function call, or when necessary, the sub variables of handles are updated into the main handles variable. Handles is of type structure.

currentfilter, oldfilter, Coff_new, and Coff_old. BW2

It is very important that after the picture is filtered or after the image has been converted to black and white, a variable, what this document will call state, is updated to account for current output display of the image.

currentfilter, oldfilter, Coff_new, and Coff_old. BW2 variables compare the new state of the output image to the old one currently on the screen. Currentfilter variable is responsible for being able to click from one filter to another and have it seamlessly switch. Coff_new, and coff_old, compares the current filter coefficient value, to the new one that was trigger by moving the slider. If a change is detected the output image is updated with the new filter coefficient. Note this comparison is triggered in the main GUI file under the slider call back function.

BW2 is used because there was an original BW handle variable that was used in a different way. The BW2 variable name was kept as legacy. The importance of this variable is defined in adjustment pushbuttons section.

Section 1: Editing Positions

Editing positions can be done by opening the GUI editor in MATLAB. This can be done by typing in guide in MATLAB's command prompt, then opening the .fig file in the editor.

Please refer to the help command or YouTube videos on editing positions on the GUI.

Section 2: IMGPRO_TOOL

"IMGPRO_TOOL" is the main function that deals with user inputs. The user inputs in this case are in the form of pushbuttons, sliders and check boxes. IMGPRO_TOOL is responsible for setting all the conditional/state variables used in the "Imgpro_filterfnc" function based on which button was pressed. There are three different types of user interactions, filtering, image input, and image adjustment. Other than the image input pushbuttons. All other inputs first check if there is an open image on the screen. If not a prompt appears on the screen indicating that there is no image open.

Open Image Push Button

Open Image Push Buttons are responsible for opening an image.

The "Open Image" Pushbutton allows the user to import their own image to the "IMGPRO_TOOL". As part of the function, the image is automatically resized so the largest dimension is 640x480.

For the viewer's convenience. Several stock images are placed as pushbuttons to the right of the open pushbutton.

Adjustment pushbutton

Black and White

The Black and White Pushbutton is responsible for converting the image to black and white. If the dimensionality of the image is 3, the image is in color, if its dimensionality 1, it is in black and white. (One dimension for each color layer, RGB)

If the current image is dimension 1, BW2 handle is set to 0, which is used as a conditional statement in "Imgpro_filterfnc" to set the image to color.

If the image is dimension 3, BW2 handle is set to 1, which is used as a conditional statement in MSD filter to set the image to Black and white.

Slider

When the slider is moved, the value of the slider is obtained to see if it has been changed. If it has been changed, the value of new_coff, is updated to the new slider value. The comparison between old_coff and new_coff are used as conditional statements in "Imgpro_filterfnc" to update the image if necessary.

If the coefficient is updated, the Imgpro_filterfnc function is iterated with the new coefficient value, and at the end of the filter, coff_new is written as coff_old in the handles parameters.

Filtering Push Buttons.

These are the “High Pass”, “Low Pass”, “Blur”, “Edge Detection”, and “Phase” pushbuttons in the panel.

Code breakdown for Filter Pushbuttons.

Each of the Filtering Pushbuttons, first compares the value of the currentfilter variables with the filter call. And if they are the same the filter is set to “off” and the image on the right is reset to the original image.

If not, the current filter is set to filter call.

After this the currentfilter is updated in handles and pass to the Imgpro_filterfnc function.

Section 3: Imgpro_filterfnc

```
function [hObject, handles]= MSD_filter(hObject, eventdata, handles)
```

Note it is important that the "Imgpro_filterfnc" function returns handles variables. If handles is not return, the updated handles variable is not returned.

```
%% Initial Conditions.
Image = handles.Current_image;
[~ , ~, c] = size(Image);
currentfilter = handles.currentfilter;
oldfilter= handles.oldfilter;
Coff_new = handles.Coff_new;
Coff_old = handles.Coff_old;
```

Figure 1: builds current and old conditions to be used later in code to determine what output should be.

```
switch currentfilter
    case('HP')
        %%

        %%
    case('LP')
        %%

        %%
    case('G')
        %%

        %%
    case('Edge')
        %%

        %%
    case('off')
        %%

        %%
    case('phase')

end
```

Figure 2: Break down of cases.

Depending on the button pressed, a comparison will take place. If high pass button was pressed the high pass case will be initiated. After that, it will compare the previous initialization files with the current initialization files to see what "case" needs to be performed. A description of each case will be shown in figure 3.

Note the special case of off. Off is not a button in the GUI window, however, if a filer push button is pressed twice, there is a conditional statement that will relabeled the current case to "off" to turn of the filtering on the screen. The output will be either a color or BW image of the original image depending on the current state of BW.

```
case('HP')
    % This section of the code handles if  high pass button is pressed.
    otherfilter = (strcmp(oldfilter,'LP') || strcmp(oldfilter,'Edge') || strcmp(oldfilter,'G'));
    %The above line generates true if previous filter is applied.
    if (c == 3) % Checks if image is in color

        if ((handles.Filter_on == 0 && Coff_new == Coff_old)) % check if the image is filtered and no change in filtering scale
            %Filters and displays filtered version of the colored
            %image.
            %Updates handles to account for new image properties.
            % Set Filter_on to 1.

        elseif(handles.Filter_on == 1 && Coff_new ~= Coff_old) %applies filter update based on slider update

            %Imports Original image and filters it with respect to the
            %new filter cofficent.
            %Updates handles to account for new image properties.
            % sets current filter as old filter coff.
            % sets filter_on to 1.

        elseif(handles.Filter_on == 1 && otherfilter) % used if another filter is applied
            %Imports Original image and filters it with respect to the
            %new filter.
            % sets current filter as old filter coff.
            % sets filter_on to 1.

        end
    elseif (c == 1)
        if (handles.Filter_on == 0 && Coff_new == Coff_old)
            %Filters and displays filtered version of the BW
            %image.
            %Updates handles to account for new image properties.
            % Set Filter_on to 1.

        elseif (handles.Filter_on == 1 && Coff_new ~= Coff_old)
            %Imports Original image and filters it with respect to the
            %new filter cofficent.
            %Updates handles to account for new image properties.
            % sets current filter as old filter coff.
            % sets filter_on to 1.

        elseif(handles.Filter_on == 1 && otherfilter)
            %Imports Original image and filters it with respect to the
            %new filter.
            % sets current filter as old filter coff.
            % sets filter_on to 1.
        end
    end
```

Figure 3: Note the condition of each conditional statement is explained in the comments. Each case is similar for LP, HP, Edge and Blur sections. Just the filter coefficient is changed.

Note there is an update of variables located in the end of the file. Which updates the new coefficient value.

Section 4: MISC.

Adding a new filter

Adding a new filter can be done very easily. A template for a new filter has been added to appropriate sections and commented out for the servicer's convenience. Multiple additions can be done by copying the associated commented code for each new filter.

1) First step to creating a new filter will be to create a box for that filter. This can be done by opening guide in the command prompt and opening the "IMGPRO_TOOL.fig" file. Control clicking on the any of the current filters (or hit control-c → control-v). By doing this you have you created a new pushbutton with the same size dimensions as previous one. After which double click on the pushbutton and edit the name and associated tag to the new name of your new filter.
2) Save and run. After which, open up the "IMGPRO_TOOL.m" file, go to the commented code, copy and paste the code into the pushbutton function. Note the name of the pushbutton will be named after the "tag" field in step 1.
3) Rename the "new filter" variable to the name of your new filter name.
4) Open up "Imgpro_filterfnc.m" file and copy and paste the code associated with a new filter in the switch statement. Uncomment the code and proceed to rename all the new filter variables names in all the sections. Please use the naming convention of coff_new and coff_old to vary your filter results. Please refer to 'HP' or 'LP' section to see different ways of varying your filter intensity.
5) Test functionality.

Adding a new predefined image.

Adding a new predefined image can be done very easily. A template for a new filter has been added to appropriate sections and commented out for the servicer's convenience. Multiple additions can be done by copying the associated commented code for each new image.

1) First step to creating a new predefined image will be to create a box for that image. This can be done by opening guide in the command prompt and opening the "IMGPRO_TOOL.fig" file. Control clicking on the any of the predefined image pushbuttons (or hit control-c → control-v). By doing this you have you created a new pushbutton with the same size dimensions as previous one. After which double click on the pushbutton and edit the name and associated tag to the new name of your new filter.
2) Save and run. After which, open up the "IMGPRO_TOOL.m" file, go to the commented code for a new predefined image, copy and paste the code into the pushbutton function. Note the name of the pushbutton will be named after the "tag" field in step 1.
3) Include a copy of the picture in the same directory as the figure is in, and rename the image names.
4) Test functionality.