Image Processing for Aerial Photography

ENEE499 – Independent Study

Jackie Weber

ABSTRACT

This project aimed to improve the process of identifying and classifying specific targets on the ground to help complete a component of a student competition. The goal was to create an automated image processing system with Matlab's Image Processing Toolbox that could detect colors, shapes, and letters within an image and use a GUI to output this data into a text box with the option to choose parameters to be output into a text box. Within this scope, the project was a partial success – being able to complete these different tasks to varying degrees of success.

TABLE OF CONTENTS

INTRODUCTION

Image processing is a very important capability in order to automate unmanned vehicles. Once a machine can recognize an object within its field of view, it gains the ability to sort through image data and determine visual characteristics of its environment without human input. While machine vision is growing, the size of some of the machines that require this capability is shrinking. The military is starting to turn to Unmanned Aerial Vehicles to do various different types of reconnaissance missions in warzones to reduce the risks that these missions could introduce to humans.

In this context, the Association for Unmanned Vehicle Systems International has been hosting a competition for student teams to create a UAS and complete a reconnaissance mission at the Patuxent River Naval Air Base. This reconnaissance mission includes the navigation of a search area in order to observe an unknown number of targets and determine their shapes, alphanumeric, the color of the shape and alphanumeric, GPS location, and heading. One of the bonus tasks available to the teams is automated detection and location of targets.

This project is significant because there are few many teams that compete in the competition that are capable of accurately locating and characterizing these targets autonomously. Therefore, I would like to create the backbone of this automated target recognition system and see if I can create an improved version of those that are currently in use by other teams.

METHODOLOGY

This project was completed in Matlab using the Image Processing Toolbox to achieve the major computational goals of the project, using GUI-building tutorials to help develop the GUI programmatically. Much of the development will be based on Matlab's online documentation and testing using multiple images created in Photoshop or found online. In this manner, a multi-featured recognition program was created that will be able to be used either indoors or outdoors to recognize a minimum of 8 colors and 8 shapes, after further development.

In order to start the process, I went through a number of online and print resources to see what tools were available to me to complete all of the goals of this project. Due to the simplicity and straightforward nature of the targets that I aimed to recognize, lower-level functions and procedures could be used to accomplish the tasks involved. After this research, an outline of how I could tackle each component was created as it was encountered. This outline allowed me to have a basis of what functions and transformations would be necessary for that component and guided my overall process.

After these plans were completed, I would begin the construction of my code in a new file so that new progress would not be hindered by less-than-adequate previous components. It also allowed me to remain more focused on each component and making sure that it worked in its entirety without having to sort through a more massive program. After a component was initially coded, I would then try it on a relevant test image with added print statements to ensure that every step worked as intended. A trial and error process was then used to further flesh out the program. If a component was not working as intended, new research was conducted on a potential replacement process which would go through the same testing.

RESULTS

SHAPE RECOGNITION

For the shape recognition program, I decided on an algorithm that would take a map of the distances from the edge of a shape to the centroid and compare this to known values for each potential shape. The original idea was to count the number of corners on each shape, using the findPeaks() function and counting the number of locations; however, some of the necessary shapes, such as the star and the pentagon, had the same number of corners, so further data on each shape needed to be taken. However, I was unable to find a robust way to use this data to accurately identify each shape, so below shows the furthest progress I was able to make.

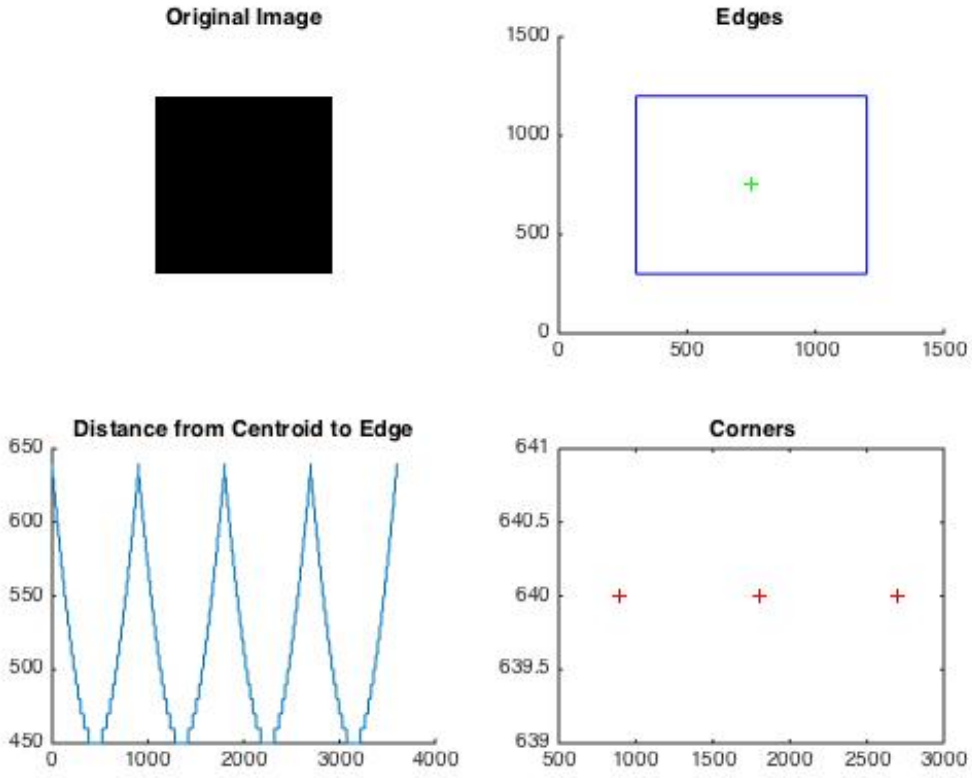

FIGURE 1 - SHAPE RECOGNITION PROGRAM OUTPUT

Color Recognition

Color recognition is a far simpler problem; however, I needed to be able to rank the colors that were within each image in order to display color of the shape and the color of the alphanumeric. For this problem, I employed a Matlab function file of my creation entitled color_picker.m that used the bin counts of the histogram of each image's pixel color values to rank the top 3 colors in each image. The top three were used because each shape lays on a solid background so I needed to be able to get this value so I knew it wasn't being used as the color of the shape.

Each image was first converted from the RGB color-space to the HSV color-space in order to easily differentiate between the six colors of the color wheel plus black and white. The function histcounts() was used for the hue and value channels of each image, input from the parent function color_rec.m, and a ranked system of if-else statements were used within a 3-step for-loop to determine and rank the top three colors based on these histograms. These colors were then output back to color_rec.m to be sent to the GUI.

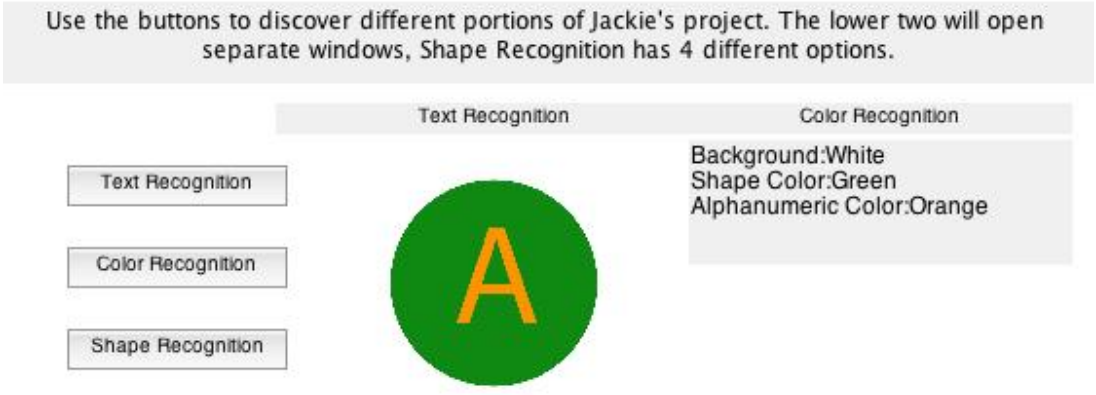

Figure 2 - View within GUI Window after Running Color Recognition

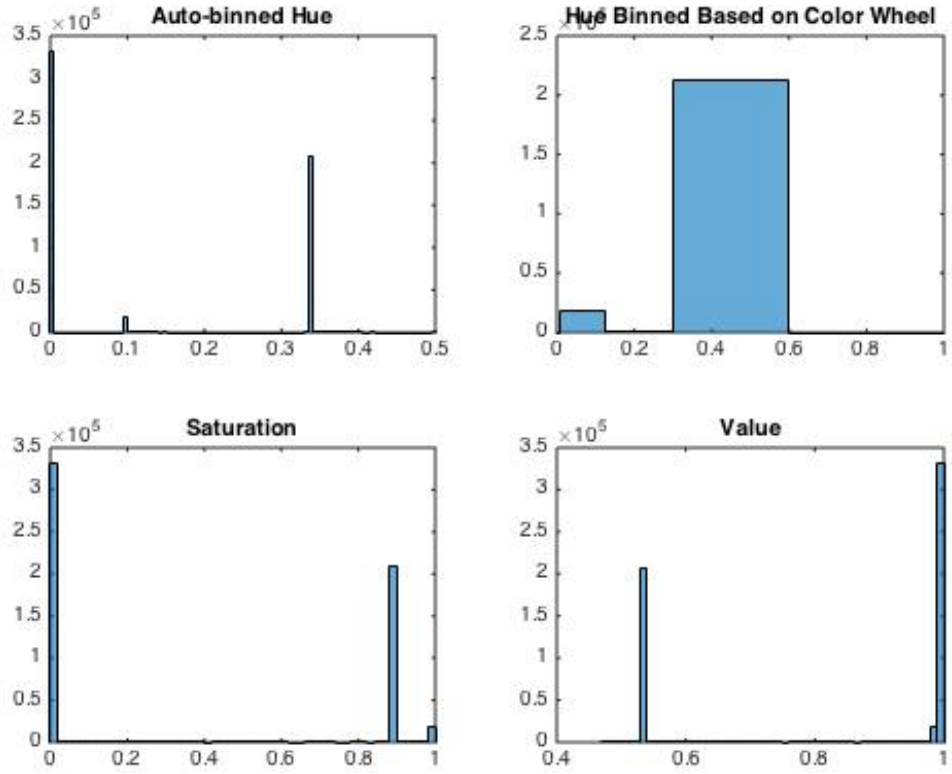

FIGURE 3 - SECOND FIGURE CREATED BY COLOR_REC.M

TEXT RECOGNITION

Text recognition was the simplest part of this project since Matlab has a build-it Optical Character Recognition (OCR) engine. As a result, the images simply needed to be contrast-adjusted and then converted to black and white prior to calling the function ocr(). I also used the optional Name-Value pair 'TextLayout','Block' in order to let the function know that it only needed to look in a single, confined region. The function ocr() outputs an object from which you can take the Text and CharacterConfidences properties to tell you the confidence that the function has in correctly identifying these characters.

The function alpha_rec.m was called by the GUI file to conduct both of these processes to be displayed; figure 4 shows the results of pressing the Text Recognition button.

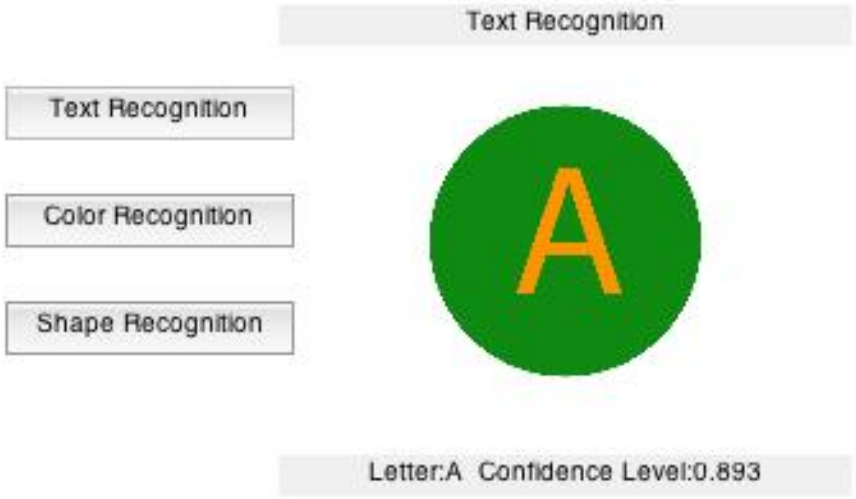

FIGURE 4 - TEXT RECOGNITION AND CONFIDENCE

MATLAB CODE

The GUI for this project, which conducts the previous three functions, uses the following m-files, to be included in the appendices: indyGUI.m, alpha_rec.m, shape_find.m, color_rec.m, color_picker.m. Additional files created throughout this process include outputtext.m, which formats and outputs a text file to be used at the competition; imconvert.m, which runs through a series of 5 images of fences and takes the gradients of each, cleaning up after to display the prominent edges as a side project for Dr. Blankenship. The image files necessary will be attached in a zip folder that can then be used in conjunction with the Matlab code.

CONCLUSION

Summarily, Matlab provides users with an excellent library of tools for processing and analyzing images. However, with the depth of the project attempted, it was very difficult to get every part working exactly as planned. Many of the portions of this project were anywhere from 75-90% completed but it was very difficult over the course of a semester to both learn and implement everything that would be necessary to successfully complete this project.

The shape recognition program was meant to be able to actually output the name of the shape in the picture. I was only able to get as far as pulling out the necessary information for that process before it became more prudent to move to the next item. The color recognition program works as expected; many of the commands that display figures and values will be removed for future use, they were simply for the illustrative purposes of this report. The text recognition program also works as expected. In the future, it will have the capability to recognize characters in rotated images. The text file output also works as expected; it will simply need tested with actual files. Finally, the GUI is currently configured for the illustrative purposes of this report. It will get reconfigured for use at the competition as well.

Overall, I would deem this project a success, but also a work in progress. I realized that I had committed to more than I realized I would be able to complete. This was an excellent exercise in image processing and I learned a lot more than I had planned on.

REFERENCES

Bennamoun, M., and G. J. Mamic. *Object Recognition: Fundamentals and Case Studies*. London: Springer, 2002. Print.

"Documentation." *MATLAB*. MathWorks, 1994-2015. Web. Feb-May. 2015.

Gonzalez, Rafael C., Richard E. Woods, and Steven L. Eddins. *Digital Image Processing Using MATLAB*. Upper Saddle River, NJ: Pearson Prentice Hall, 2004. Print.

Poon, Ting-Chung, and Partha P. Banerjee. *Contemporary Optical Image Processing with MATLAB*. New York: Elsevier Science, 2001. Print.

Semmlow, John L. *Biosignal and Biomedical Image Processing MATLAB-based Applications*. New York: Marcel Dekker, 2004. Print.

Weeks, Arthur R. *Fundamentals of Electronic Image Processing*. Bellingham, WA, USA: SPIE Optical Engineering, 1996. Print.

Independent Study Progress Display

Jackie Weber

```
function indyGUI()
% Structure to store object handles
h = struct();
global file;
file = 'OAGC.tif';

% Create the main window
hwindow = figure('Name','ENEE499 - Image Processing Project');

% Window Title
h.talk = uicontrol('Parent',hwindow,'Units','Normalized',...
    'Style','Text','FontSize',12,'FontName','Ariel','HorizontalAlignment','Center',...
    'Position',[0.0 0.75 1 0.1],...
    'String','Use the buttons to discover different portions of Jackie''s project. The lowe
r two will open separate windows, Shape Recognition has 4 different options.');

% Create a text box for conveying average ratio

h.button1 = uicontrol('Parent',hwindow,'Units','Normalized',...
    'Style','PushButton','Position',[0.06 0.6 0.2 0.05],...
    'HorizontalAlignment','Center','String','Text Recognition'); % A button

h.button2 = uicontrol('Parent',hwindow,'Units','Normalized',...
    'Style','PushButton','Position',[0.06 0.5 0.2 0.05],...
    'HorizontalAlignment','Center','String','Color Recognition'); % A button

h.button3 = uicontrol('Parent',hwindow,'Units','Normalized',...
    'Style','PushButton','Position',[0.06 0.4 0.2 0.05],...
    'HorizontalAlignment','Center','String','Shape Recognition'); % A button

% Create an axis
h.ax1 = axes('Parent',hwindow,'Position',[0.25 0.33 0.4 0.35]);
[txt, conf] = alpha_rec(file);
h.image = imshow(file);
set(h.image,'Visible','off')

h.alphalabel = uicontrol('Parent',hwindow,'Units','Normalized',...
    'Style','Text','HorizontalAlignment','Center',...
    'Position',[0.25 0.69 0.4 0.035],...
    'String','Text Recognition');

h.alphatext = uicontrol('Parent',hwindow,'Units','Normalized',...
    'Style','Text','HorizontalAlignment','Center',...
    'Position',[0.25 0.27 0.4 0.035],...
    'String',strcat('Letter: ',txt,'  Confidence Level: ',num2str(round(conf,3))));

set(h.alphalabel,'Visible','off')
set(h.alphatext,'Visible','off')

[h.button1.Callback,h.button2.Callback,h.button3.Callback] = deal({@GUI_Callbacks,h});
```

```
global x;
x = 1;
function GUI_Callbacks(hObject,eventdata,h)
    shapes = {strcat('Hexagon','.tif');...
        strcat('Circle','.tif');...
        strcat('Square','.tif');...
        strcat('Triangle','.tif')};
    switch hObject
        case h.button1
            h.image.Visible = 'on';
            set(h.alphalabel,'Visible','on')
            set(h.alphatext,'Visible','on')
        case h.button2
            h.image.Visible = 'on';
            colors = color_rec(file);
            h.colortext = uicontrol('Parent',hwindow,'Units','Normalized',...
                'Style','Text','FontSize',12,'HorizontalAlignment','Left',...
                'Position',[0.63 0.53 0.35 0.15],...
                'String',colors);
            h.colorlabel = uicontrol('Parent',hwindow,'Units','Normalized',...
                'Style','Text','HorizontalAlignment','Center',...
                'Position',[0.63 0.69 0.35 0.035],...
                'String','Color Recognition');
        case h.button3
            if (x > length(shapes))
                x = 1;
            end
            h.shape = shape_find(char(shapes(x)));
            x = x + 1;
    end
end

end
```

Shape Distinctions

This file takes in a color image, enhances the contrast, then makes it a binary image. It then uses this binary image to detect the border of the shape and then calculate the centroid and the distances from the centroid to compare to sample data in order to determine which shape it is an output that name.

```
function [window] = shape_find(filename)
X = imread(filename);
I = X(1:end, 1:end, 1:3);
I2 = rgb2gray(I);
I2 = imadjust(I2, [0;1], [1;0]);
% figure
% imshow(I2)
I2 = im2bw(I2);
% I3 = bwconncomp(I2);

% Pulling out boundary points
window = figure('Name','Shape Recognition Progress');
[B,L,N,A] = bwboundaries(I2);
stat = regionprops(I2, 'Centroid');
cent = stat.Centroid;
boundary = B{1};
subplot(2,2,2)
hold on
plot(boundary(:,2), boundary(:,1), 'b', 'LineWidth', 1)
plot(cent(1),cent(2), 'g+')
axis([0 length(I(1,:))/3 0 length(I(:,1))])
hold off
title('Edges')

% Finding Distances from Centroid
distPlot = [];
for i = 1:length(boundary)
%     distPlot = [distPlot cast(sqrt((cent(1)-boundary(i,1))^2 + (cent(2)-boundary(i,2))^2)
,'int32')];
    distPlot = [distPlot sqrt((cent(1)-boundary(i,1))^2 + (cent(2)-boundary(i,2))^2)];
    % does not cast data to integer for use later with findpeaks
end
distPlot2 = round(distPlot, -1);
distPlot = cast(distPlot2, 'like', distPlot);
subplot(2,2,3)
hold on
plot(distPlot)
% axis([0 length(boundary) 400 800])
hold off
title('Distance from Centroid to Edge')

% Showing the initial shape
subplot(2,2,1)
imshow(I);
title('Original Image')

% Shape Comparisons
```

```
[maxs locs] = findpeaks(distPlot2,'MinPeakDistance',cast(length(boundary)/12,'int16'));
subplot(2,2,4)
plot(locs, maxs, 'r+')
title('Corners')

end
```

```
Error using shape_find (line 10)
Not enough input arguments.
```

Published with MATLAB® R2014b

Color Recognition

uses color_picker.m to display colors within an image

```
function [ colors ] = color_rec( filename )

X = imread(filename);
I = X(1:end, 1:end, 1:3);
I2 = rgb2hsv(I);

% figure
% imshow(X)
I2_1 = I2(:,:,1);
I2_2 = I2(:,:,2);
I2_3 = I2(:,:,3);

figure('Name','Histograms of Image Pixels')

subplot(2,2,1)
histogram(I2_1)
title('Auto-binned Hue')

subplot(2,2,2)
histogram(I2_1,[1*10^-10,0.01,0.125,0.3,0.6,0.75,0.9,1])
title('Hue Binned Based on Color Wheel')

subplot(2,2,3)
histogram(I2_2)
title('Saturation')

subplot(2,2,4)
histogram(I2_3)
title('Value')

a = color_picker(I2_1,I2_3);
colors = [strcat('Background',': ',a(1));strcat('Shape Color',': ',a(2));strcat('Alphanumer
ic Color',': ', a(3))];
end

function [ colors ] = color_picker( hue, val)
%color_picker Chooses the top three colors in an image using the hue and
%value channels


counth = histcounts(hue,[0,1*10^-10,0.01,0.125,0.3,0.6,0.75,0.9,1]);
counth = [counth;1 2 3 4 5 6 7 8];
counth = counth.';
sorted_h = sortrows(counth);

countv = histcounts(val,[0,0.1,0.8,1]);
countv = [countv;1 2 3];
countv = countv.';
sorted_v = sort(countv);
```

```
chm = sorted_h(6:8,2);
cvm = sorted_v(:,2);

colors = {'';'';''};

for i = 1:3
    if (chm(4-i) == 2)
        colors(i) = {'Red'};
    elseif (chm(4-i) == 3)
        colors(i) = {'Orange'};
    elseif (chm(4-i) == 4)
        colors(i) = {'Yellow'};
    elseif (chm(4-i) == 5)
        colors(i) = {'Green'};
    elseif (chm(4-i) == 6)
        colors(i) = {'Blue'};
    elseif (chm(4-i) == 7)
        colors(i) = {'Purple'};
    elseif (cvm(4-i) == 1)
        colors(i) = {'Black'};
    elseif (cvm(4-i) == 3)
        colors(i) = {'White'};
    end
end
end
```

```
Error using color_rec (line 7)
Not enough input arguments.
```

Published with MATLAB® R2014b

alpha_rec.m

turns recog2.m into a function

```
function [ out_txt, out_conf ] = alpha_rec( filename )

I = imread(filename);
X = I(1:end, 1:end, 1:3);

I1 = rgb2gray(X);
I2 = imtophat(I1,strel('disk',30));
I3 = imadjust(I2, [0;0.4],[1;0]);
I4 = im2bw(I3);

out = ocr(I4 , 'TextLayout','Block');
out_txt = out.Text(1);
out_conf = out.CharacterConfidences(1);

end
```

```
Error using alpha_rec (line 7)
Not enough input arguments.
```

Published with MATLAB® R2014b

outputtxt.m

Outputs the text from the shape recognition and color recognition into a text file

```
fileIDs = {'OAGC.tif';'t.tif';'ncsu_N.jpg'};

t_ID = [1:numel(fileIDs)];
shapes = {'square';'circle';'triangle';'hexagon';'pentagon';'cross';'star';'square'};
colors = {'red';'orange';'yellow';'green';'blue';'purple';'black';'white'};
alphas = [];
confs = []; % confidence value for that letter

for i = t_ID
    ID = char(fileIDs(i));
    [txt conf] = alpha_rec(ID);
    alphas = [alphas;txt];
    confs = [confs;conf];
end
targID = t_ID.';
T = table(targID,alphas,confs);
writetable(T,'ENEE.txt','Delimiter','\t','WriteRowNames',true);
```

Published with MATLAB® R2014b

```
for i = 1:5
    filename = strcat('test',num2str(i),'.jpg');
    X = imread(filename);
    X = rgb2gray(X);
    [Gmag, Gdir] = imgradient(X,'prewitt');
    fence = (Gmag > 0.3*(max(max(Gmag))));
    subplot(3,2,i)
    imshow(fence); % Gmag./max(max(Gmag))
end
```

Published with MATLAB® R2014b