Experiment No. 8

Aim: Using MATLAB to Measure the Diameter, Radius and Centroid of an Object within an Image.

Requirements:
Windows XP or better, Intel or AMD x86 processor, 4 GB **Disk Space,** 2048 MB **RAM** at least, hardware accelerated graphics card supporting OpenGL 3.3 with 1 GB GPU memory, MATLAB 2014a, Image Processing Toolbox .

Theory
Measuring objects within an image or frame can be an important capability for many applications where computer vision is required instead of making physical measurements. This application note has basic step-by-step algorithm for isolating a desired object and measuring its diameter. Through this application one can import an image, segment the image in order to isolate the desired object from its background and then use the MATLAB functions that come with the Image Processing Toolbox to determine the objects diameter.

Methodology

1) Import Image

Open the MATLAB software and in the application section; download the Image Processing Tool Box. Create a new MATLAB script file. Refer to *Figure 2* to begin adding code to import the desired image to measure into the MATLAB workspace. The first few lines clear the workspace to remove any previous variables and clear the command window. It is important that the Current Folder that you are working out of be the folder that contains both the script file and image. The command *imread* reads an image and converts it into a "3-dimensional" matrix in the RGB color space. The image used in this tutorial is *ball.jpg (Figure 1),* which is a 534 by 401 pixel image. The *imread* function converts this into a matrix that is 401x534x3 (Rows x Columns x RGB). The final dimension (RGB) corresponds to a red, green and blue intensity level. Use *imshow* to view the produced image in a new window.

Figure 1; Original Image, ball.jpg

Editor - D:\Docs\Avi\4.BE\Sem7\DSP\Diameter.m

Diameter.m

```
1      %Clear previous variables and Import Image
2
3 -    clear;
4 -    clc;
5
6 -    obj=imread('ball.jpg');
7 -    imshow(obj)
```

Figure 2; Code to Import an Image

Workspace

Name	Value
obj	401x534x3 uint8

Figure 3; obj variable in workspace

2) Segment Image

Follow the code in *Figure 4* below to segment the image into a binary image to differentiate the background from the desired objects. The first step taken is to divide the image into three images based on the intensities of each red, green and blue component within the image. This is Color Based Image Segmentation. You can see from *Figure 5* that the blue plane is the best choice to use for Image Thresholding because it provides the most contrast between the desired object (foreground) and the background. Image Thresholding takes an intensity image and converts it into a binary image based on the *level* desired (See line 25). A value between 0 and 1 determines which pixels (based on their value) will be set to a 1 (white) or 0 (black)). To choose the best value suited for your application right-click on the value and at the top of the menu and select "Increment Value and Run Section". Set the increment value to 0.01 and choose the best value at which to threshold. *Figure 5* shows the result of the Image Thresholding at 0.37. You can see that the image (Top-right of *Figure* 6) has been segmented between the object we desire to measure and the background.

```
10      %Segment Image
11      %divide image "obj" into its respective RGB intensites
12
13 -    red=obj(:,:,1);
14 -    green=obj(:,:,2);
15 -    blue=obj(:,:,3);
16
17 -    figure(1)
18 -    subplot(2,2,1); imshow(obj); title('Original Image');
19 -    subplot(2,2,2); imshow(red); title('Red Plane');
20 -    subplot(2,2,3); imshow(green); title('Green Plane');
21 -    subplot(2,2,4); imshow(blue); title('Blue Plane');
22
23      %Threshold the blue plane
24
25 -    figure(2)
26 -    level=0.37;
27 -    bw2=im2bw(blue,level);
28 -    subplot(2,2,1); imshow(bw2); title('Blue plane thresholded');
```

Figure 7; Code to Cleanup Image/Remove Noise

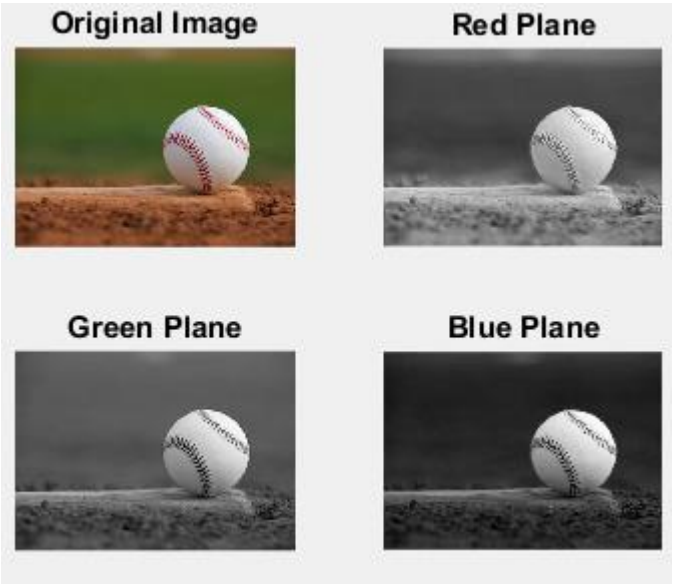

Figure 5; Color Thresholding

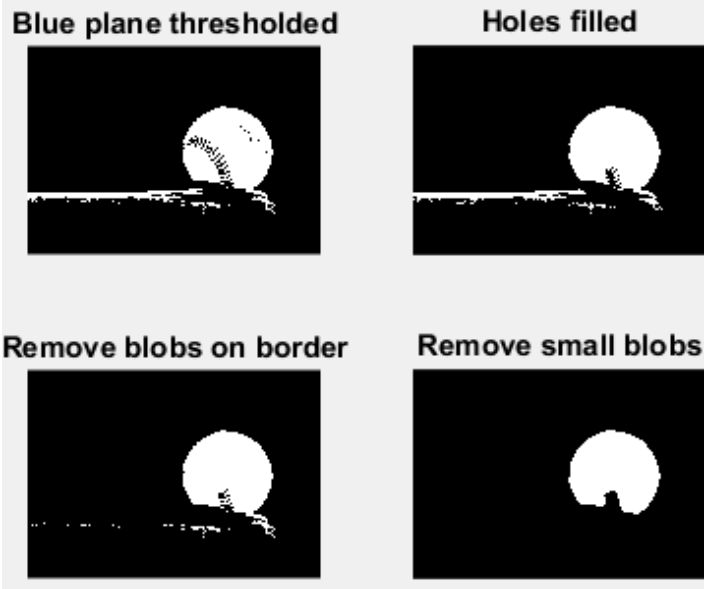

Figure 6; Complete Segmentation and Cleanup Image

3) Segmentation Continued (Remove Noise)

As you can see from the top-left image in *Figure 6* there is quite a bit of "noise" and we need to clean the image up significantly to improve the accuracy of our diameter measurement. Refer to *Figures 6 & 7* on the procedures taken to clean up the image and provide a more uniform blob to analyze. Blobs in this document are any collection of white pixels that touch to create a cohesive and distinct object.

```
30      %%Remove Noise
31
32      %Fill any holes
33 -    fill=imfill(bw2,'holes');
34 -    subplot(2,2,2); imshow(fill); title('Holes filled');
35
36      %Remove any blobs on the border of the image
37 -    clear=imclearborder(fill);
38 -    subplot(2,2,3); imshow(clear); title('Remove blobs on border');
39
40      %Remove blobs that are smaller than 10 pixels across
41 -    se=strel('disk',10);
42 -    open=imopen(fill,se);
43 -    subplot(2,2,4); imshow(open); title('Remove small blobs');
```

Figure 7; Code to Cleanup Image/Remove Noise

4) Measuring Image

The image in the bottom-right corner of *Figure 6* is the result of all image segmentation and cleanup procedures to provide one distinct and cohesive blob, which represents the ball in the original image. Having the original image in a binary form such as this will make it easy for other functions built into MATLAB to quickly analyze the region and a host of different information. The *regionprops* function is the tool that will provide the Centroid, MajorAxisLength, MinorAxisLength of the blob in the image. As you can see, by not suppressing line 45 *(Figure 8)* with a semi-colon, the diameter is the mean of MajorAxisLength, MinorAxisLength ,the radius is further calculated using the diameter information and is displayed in the Command Window *(Figure 10)*.

```
45      %Calculate properties of regions in the image and return the data in a table.
46
47 -    stats = regionprops('table',open,'Centroid',...
48      'MajorAxisLength','MinorAxisLength')
49
50      %Get centers and radii of the circles.
51
52 -    diameters = mean([stats.MajorAxisLength stats.MinorAxisLength],2);
53 -    radii = diameters/2;
54 -    fprintf('Diameter=%f and Radius=%f',diameters,radii);
55
56      %Show Result
57 -    figure(3)
58 -    imshow(obj)
59 -    d=imdistline; %Include a line to physically measure the ball
```

Figure 8; Code to Measure the Object Diameter, Radius and Centroid

Figure 9; Original Image Manually Measure

Command Window

```
stats =

       Centroid          MajorAxisLength    MinorAxisLength
    _______________    _________________    _______________

    365.6    185.18    170.97               143.41

Diameter=157.190355 and Radius=78.595178>>
```

Figure 10; Diameter, Radius and Centroid Output in the Command Window

5) Results

The diameter, radius and centroid is now displayed in the Command Window to be approx. 170 pixels across. This was verified in *Figure 9* by using the *imdistline* function in line 50 *(Figure 8).* As you can see between the two figures, the value calculated by the code was very close to the manual measurement in *Figure 9.*

Conclusion

Thus, using MATLAB successfully measured the Diameter, Radius and Centroid of an Object within an Image.