Project 5 Report

Color Analysis and Image Segmentation

CPE428-01,02 - Computer Vision

Prepared for:	Professor Xiaozheng Zhang
Prepared By:	The “A” Team; Nikhil Patolia, Alec Hardy
Date Submitted:	Saturday, March 5, 2020

Table of Contents

Part 1 - Strawberry Color Analysis

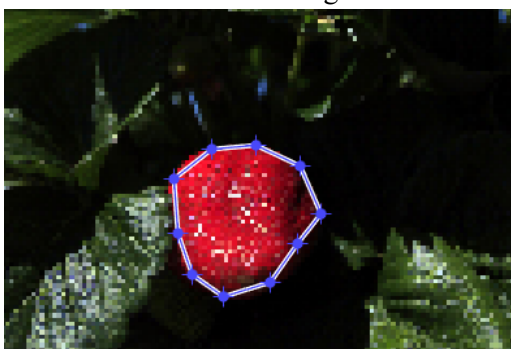
The goal of part 1 of the laboratory was to plot a histogram in different color spaces of the pixels in 20 images of strawberries - one histogram for the manually selected strawberry pixels and one histogram for the background pixels per each channel of each color space.

Procedure

All 20 images are loaded and subsampled such that they are within 150% the resolution of the smallest image. Then, masks are manually drawn on all 20 images such that the strawberry pixels, our region of interest, can be isolated from the background pixels. This is accomplished as shown in the code snippet below:

```
% Parameters to help resize images to nearly the same size
SMALLEST_IMG_SIZE = [259 194];
SIZE_THRESHOLD = 1.5;

%% Load images and determine ROIs
% Only do this if we haven't already...
if exist('ROI','var')==0 || exist('I','var')==0

    % Cell array of all image mats and ROIs
    I    = {20};
    ROI = {20};

    cwd = pwd;
    I_files = dir(fullfile(cwd+"/ImgA",'*.jpg'));

    % Iterate across all images for Part A
    for i = 1:size(I_files,1)
        i_filename = I_files(i).name;
        I{i} = imread("ImgA/"+i_filename);

        % Resize if necessary
        while SIZE_THRESHOLD*size(I{i},1) > SMALLEST_IMG_SIZE(1)
            I{i} = subsample(I{i});
        end

        % Perform ROI selection
        imshow(I{i});
        ROI{i} = roipoly();
    end
end
```

An example of the manual ROI selection is shown in the figure below:



Manual ROI is drawn around the strawberry in each of the 20 sample images.

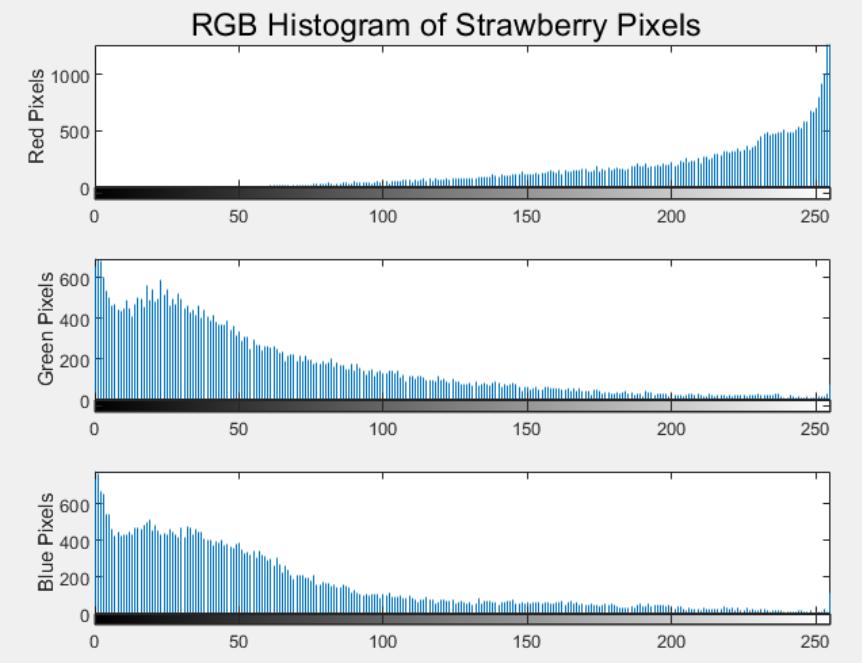
Next, for each of the the RGB, Normalized RGB, HSV, YCbCr, and LAB color spaces, each image is converted into the respective color space, and the ROI and background pixels are concatenated into a massive array specific to the respective color space channels. In the below code snippet, which takes a cell array, *I*, of all 20 images and a cell array, *ROI*, of 20 ROI masks, the construction of the six ROI/background 3-channel arrays is demonstrated and commented to explain the process for the RGB color space.

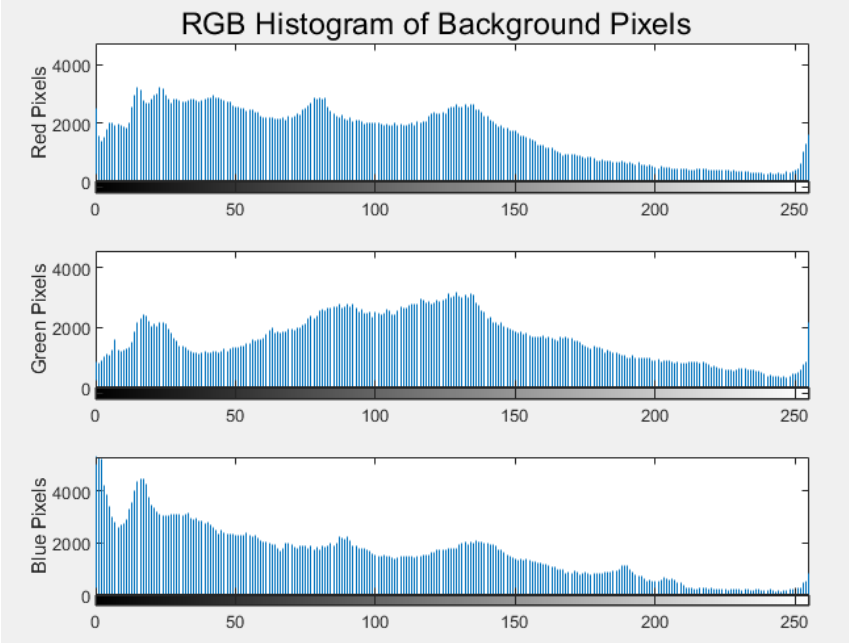
```
%% Draw histograms for the RGB spaces

    % Channel histogram
    R_obj = [];
    R_bg  = [];
    G_obj = [];
    G_bg  = [];
    B_obj = [];
    B_bg  = [];
    for i=1:size(I,2)
        cur_img = I{i};

        % Extract the R, G, B planes
        R = cur_img(:,:,1);
        G = cur_img(:,:,2);
        B = cur_img(:,:,3);

        % Extract the pixels of the R, G, B, planes that are masked or unmasked
        R_bg = vertcat(R_bg,R(ROI{i}==0));
        R_obj = vertcat(R_obj,R(ROI{i}==1));
        G_bg = vertcat(G_bg,G(ROI{i}==0));
        G_obj = vertcat(G_obj,G(ROI{i}==1));
        B_bg = vertcat(B_bg,B(ROI{i}==0));
        B_obj = vertcat(B_obj,B(ROI{i}==1));
    end
```

The rest of the code used to generate the histograms is included as an attachment to his report.

Histograms of the net result of each color channel for the background and ROI pixels are then generated and displayed. The results are shown in the figures below.

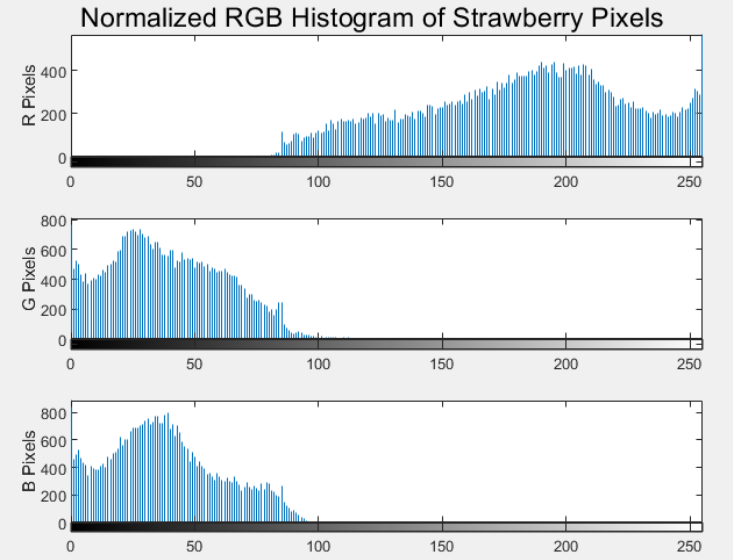
RGB Histograms

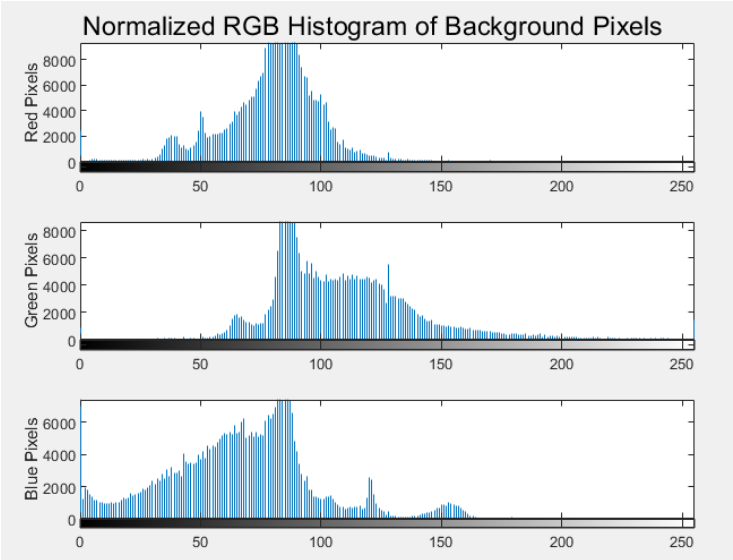




From the RGB histograms, it is apparent that among the strawberry ROI pixels there are a high number of pixels that have a high intensity value for the red channels and very few pixels that have intensity values below ~100 in the red channel. This is because the bright red strawberry pixels are very-intensely red. The shape of the histogram for the G and B channels of the strawberry pixels are similarly shaped and both have peaks in the very low intensity ranges. In the RGB histograms of the background pixels there is a much more uniform distribution of pixel intensities for all three channels. There is a small local maxima at intensity value 255 of the red channel from the background pixels caused by imperfect ROI drawing.

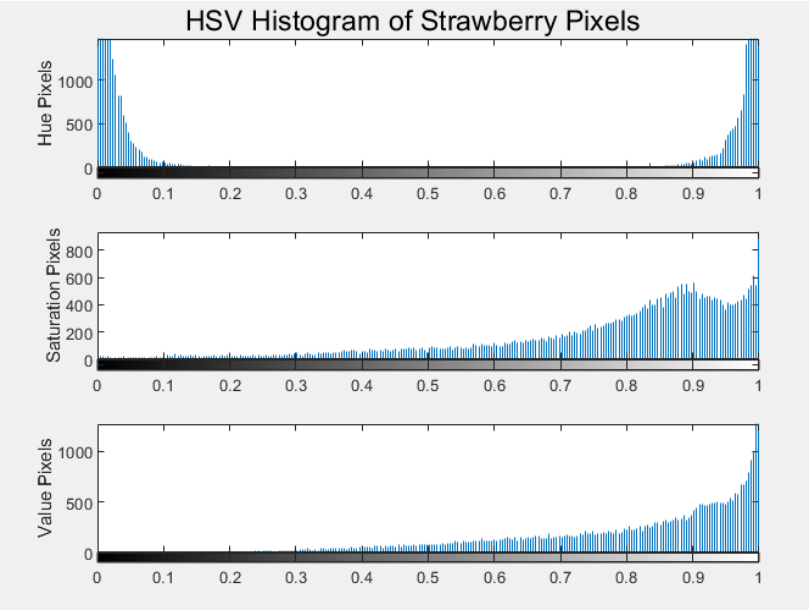
Normalized RGB Histograms

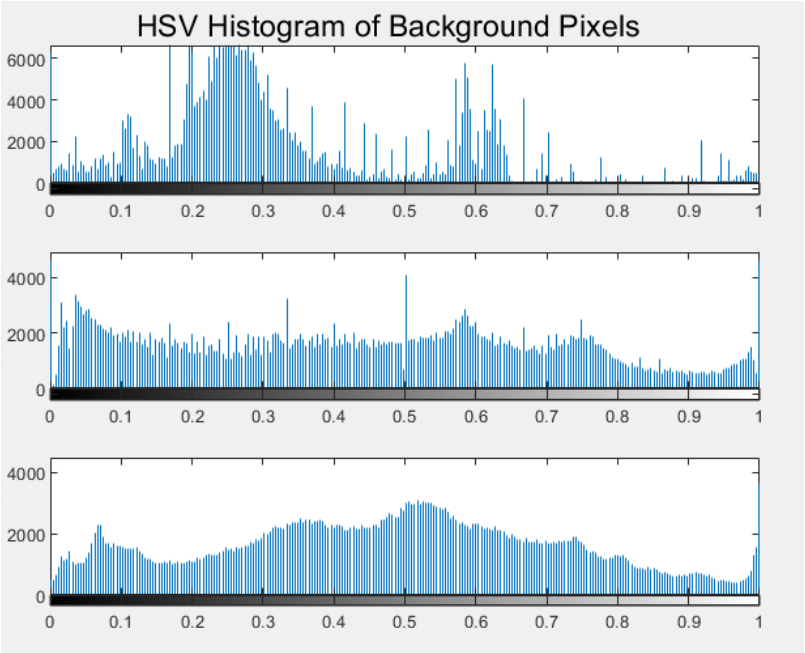




The observation from the normalized RGB histograms are similar to the observations from the (non-normalized) RGB histograms. However, the red-channel intensity distribution of the ROI regions doesn't have as sharp of a peak at the upper bound - rather the distribution is bimodal and has a maximum and mode value around 200. Accordingly, the range of intensity values (minus outliers) for the G and B channels in the strawberry pixels and the R, G, B channels of the background pixels are all much lower compared to the range of values in the (non-normalized) RGB histograms. This makes it significantly easier to observe the peaks of each histogram and to distinguish color, and this characterizing strawberry color is easier when comparing normalized RGB and non-normalized RGB histograms.

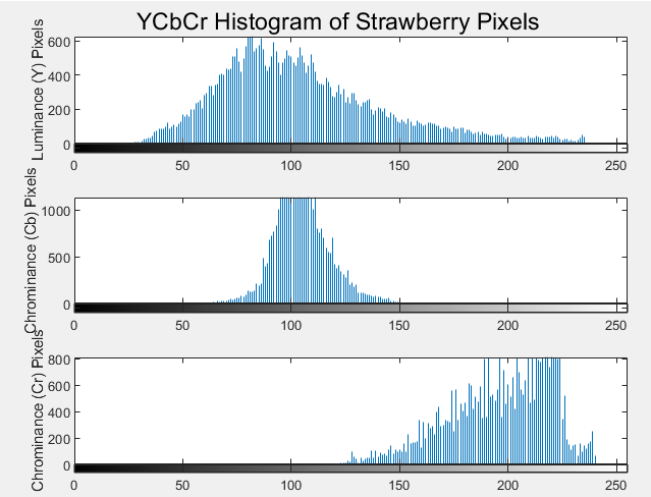
HSV Histograms

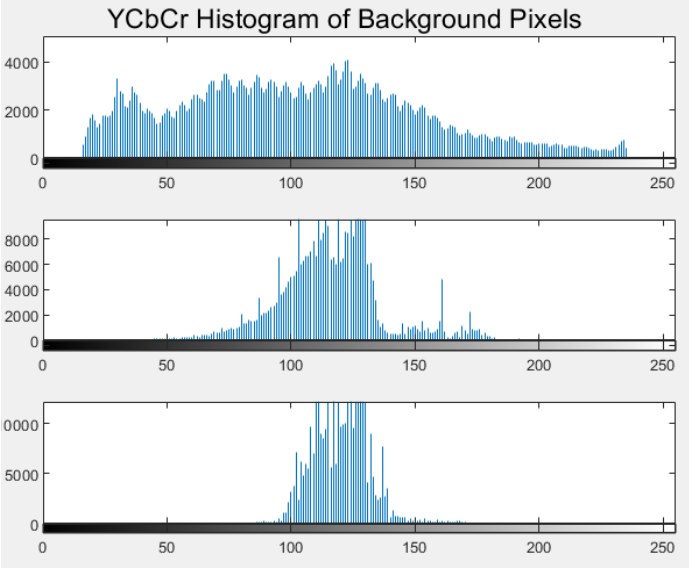




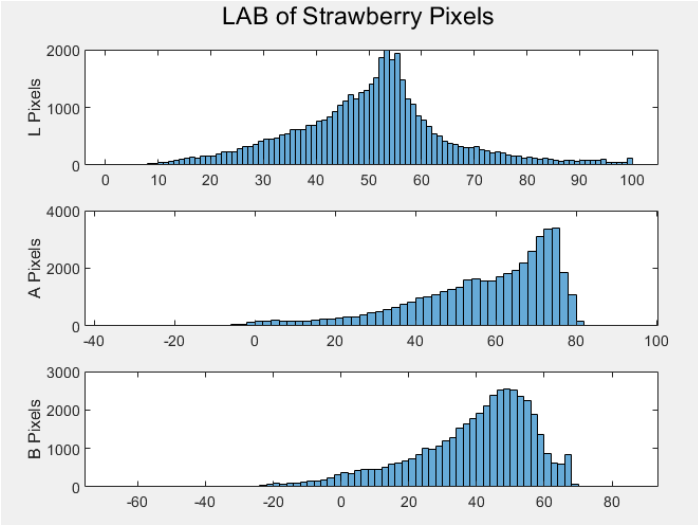
In the HSV color space we can observe the distribution of pixel colors in both the background and ROI pixels by looking at the hue values. In the hue space, hues are assigned a numerical value such that, on a continuous circle of colors, red would be represented at the 0 degree angle, green at the 120 degree angle, and blue at 240 degrees. In the histogram above, these angles are normalized between 0 and 1. Thus, in the strawberry ROI, we see a high intensity value at 0 and 1, the values corresponding close to the 0 degree color circle hues, which represents red. Because we know the background is mostly green, and because we see a peak around 0.2-0.3 in the hue channel of the background pixels, we know that green must have a huge value in this range. The other two channels provide saturation and value information, which is important in image display but not as important for color/hue segmentation and analysis. Compared to the RGB/normalized-RGB color spaces, the HSV color space is easiest to distinguish colors from pixel values.

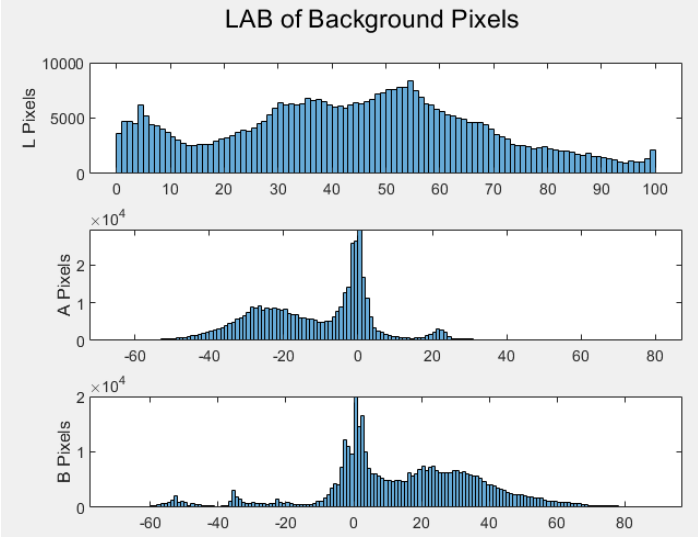
YCbCr Histograms





LAB Histograms





The YCbCr uses a channel for luminance (Y) and two channels for blue-offset and red-offsets. With this color space, there is not much distinguishable information from the luminance channel. However, among the strawberry pixels, there is a significant difference in the distribution within the Cb and Cr channel, while for the background pixels there is very little difference between the distributions. This may be useful for analysis and segmentation, however, because there is little information to be extracted from the luminance channel, this color space is not as useful for analysis as the other color spaces.

The LAB color space has an almost identical intensity distribution for the L (lightness) channel, and two color channels for red/green and blue/yellow values. This yields the same analysis results as from the YCbCr color space. However, unlike the YCbCr color space, it is the background pixels which show a difference in distribution compared to the strawberry ROI pixels.

If we were analyzing a different object/ROI with different colors we may have different results, however, the conclusion that RGB/HSV color spaces are more useful for color analysis still stands because there is additional data that can be analyzed specific to color for those color spaces.

Part 2 - Strawberry Finding/Counting Using K-Means and EM

Image Pre-Processing

For these images of the strawberries, we preprocessed these images with a gaussian filter in which the filter size was the area for the image to the power of 0.23. This value was determined after testing

multiple different values and this one allowed for adequate image smoothing for all the different image sizes that were given in this dataset. We first began by using a standard filter size of 3 but since image t1.JPG is so much larger than the other images, the gaussian filter of size 3 had almost no effect on the final result. That is why we chose to use a variable h-size for the gaussian filter.



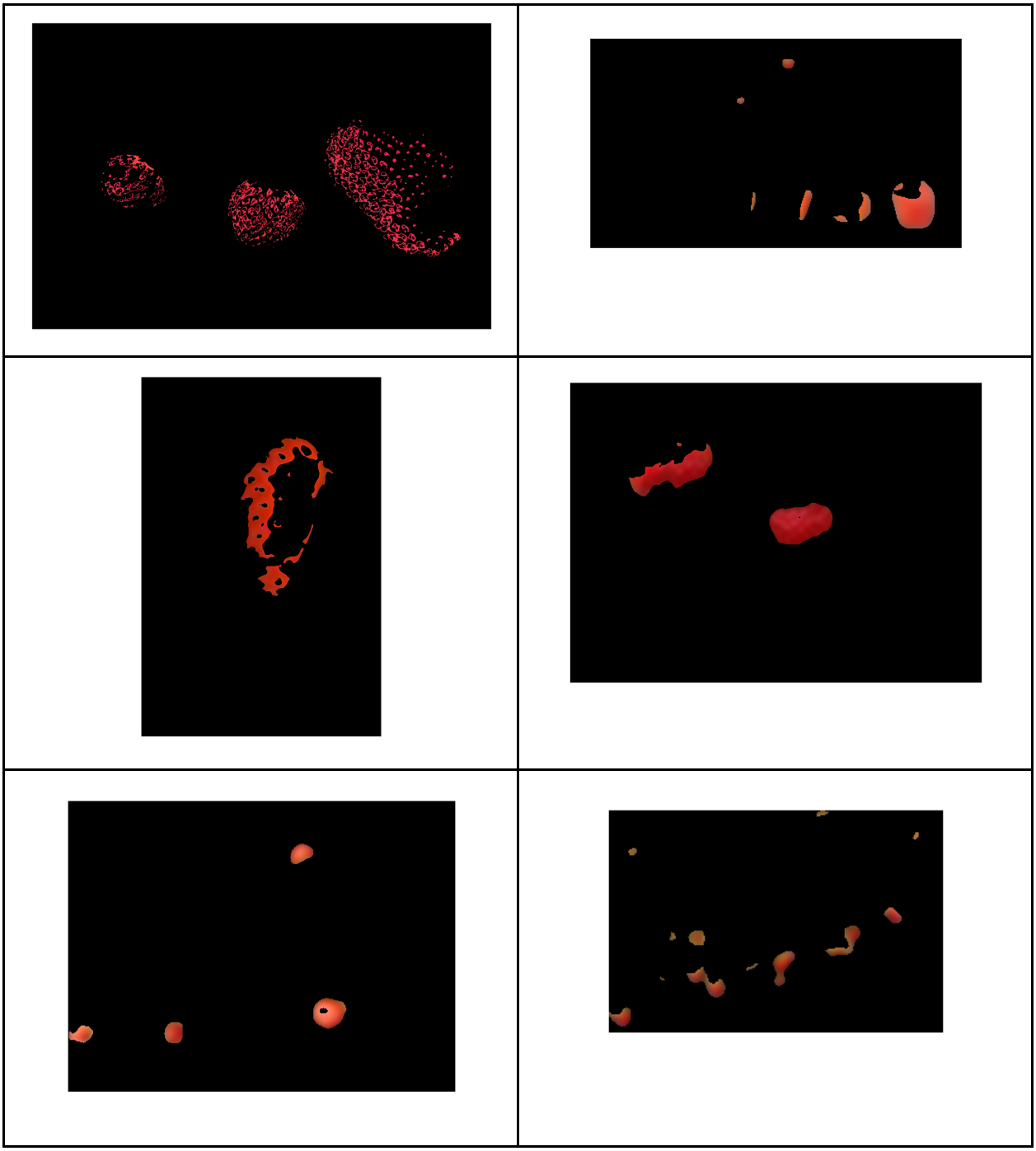



Determining the number of K Classes Needed

This was also determined on a variable basis. This was because the smaller images looked to be much noisier than the larger images and applying a heavy Gaussian filter would have smoothed the small strawberries out too much to the point of them being indistinguishable from the background. That is why for smaller images, we had to use more K Classes so that the different colors can be separated better.

Finding the Optimal K

To find the optimal K Class for the image, we went through each K Class grouping and first places a mask of the original image over the areas that were not 0 of the K Class grouping. K means groups by color so doing this step allowed us to shee what color K means grouped on. Each pixel of this masked image was counted if it resembled the red color of a strawberry and the K Class image that had the highest number of pixels that resembled the color of the strawberry was chosen as the best K Class image.



Connected Component Grouping

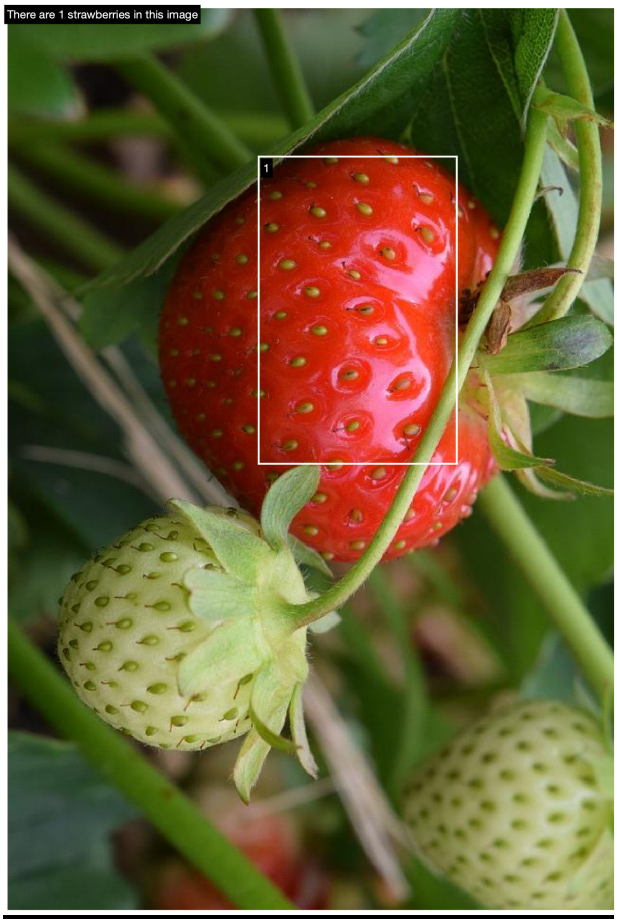
We then grouped each cluster together and kept the data of its Area and the BoundingBox. We used regionprops to get the connected components on the K Class Image without the original image masked over it, meaning that it is a binary image. This matrix was then sorted by Area in descending order because the larger groups of red colored connected components are more likely to be strawberries while the small grouping are likely to be anomalies.

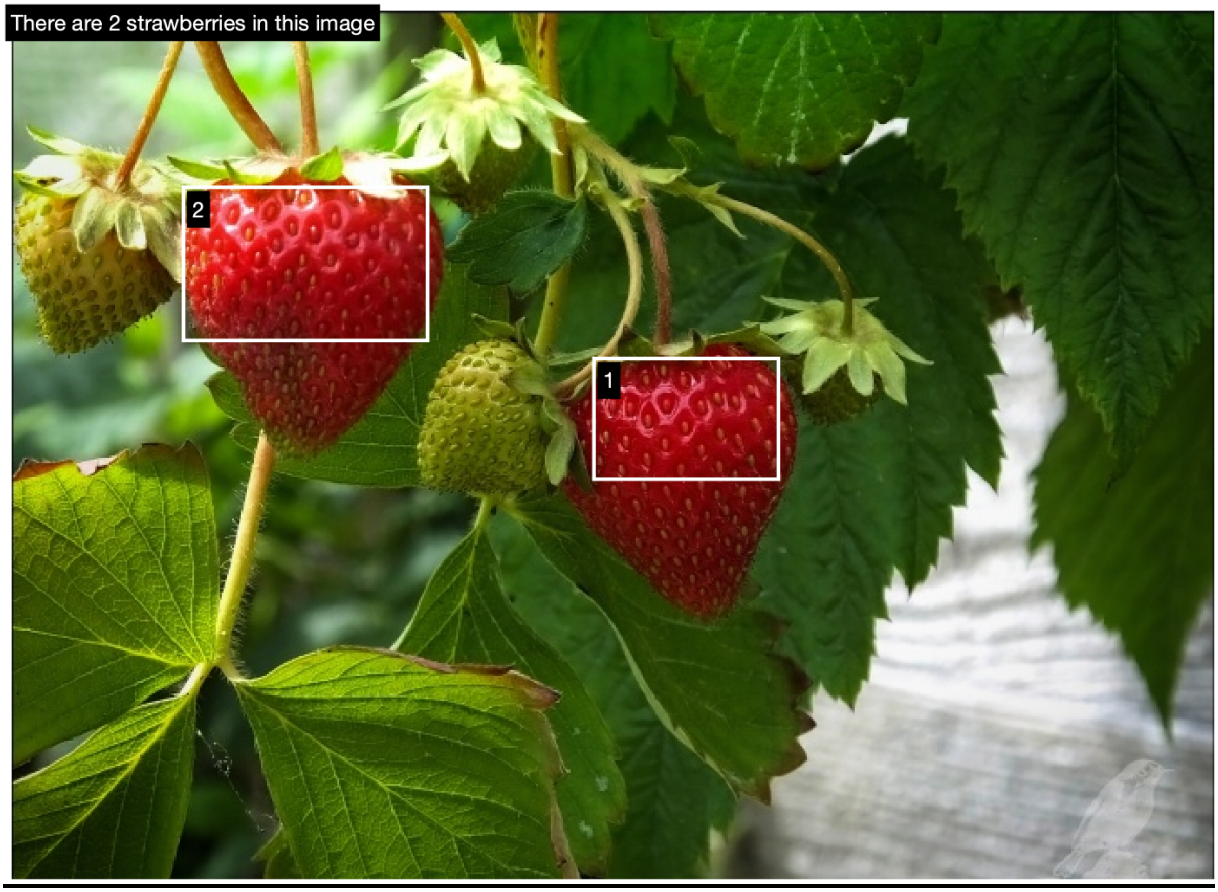
Pruning List of Potential Strawberries

With our list of groupings of red pixels in the image, we first started with the largest groupings and went down the list until we found one that was 20% of the max sized strawberry. The max size strawberry would be the first one in the matrix since the list matrix was sorted in descending order by area. For each connected component grouping that was larger that 20% of the max sized strawberry, we drew a white rectangle around it and numbered it. After finding all the large strawberries, we add text in the upper left hand side corner specifying how many strawberries were found in the image.


There are 3 strawberries in this image
1
2
3

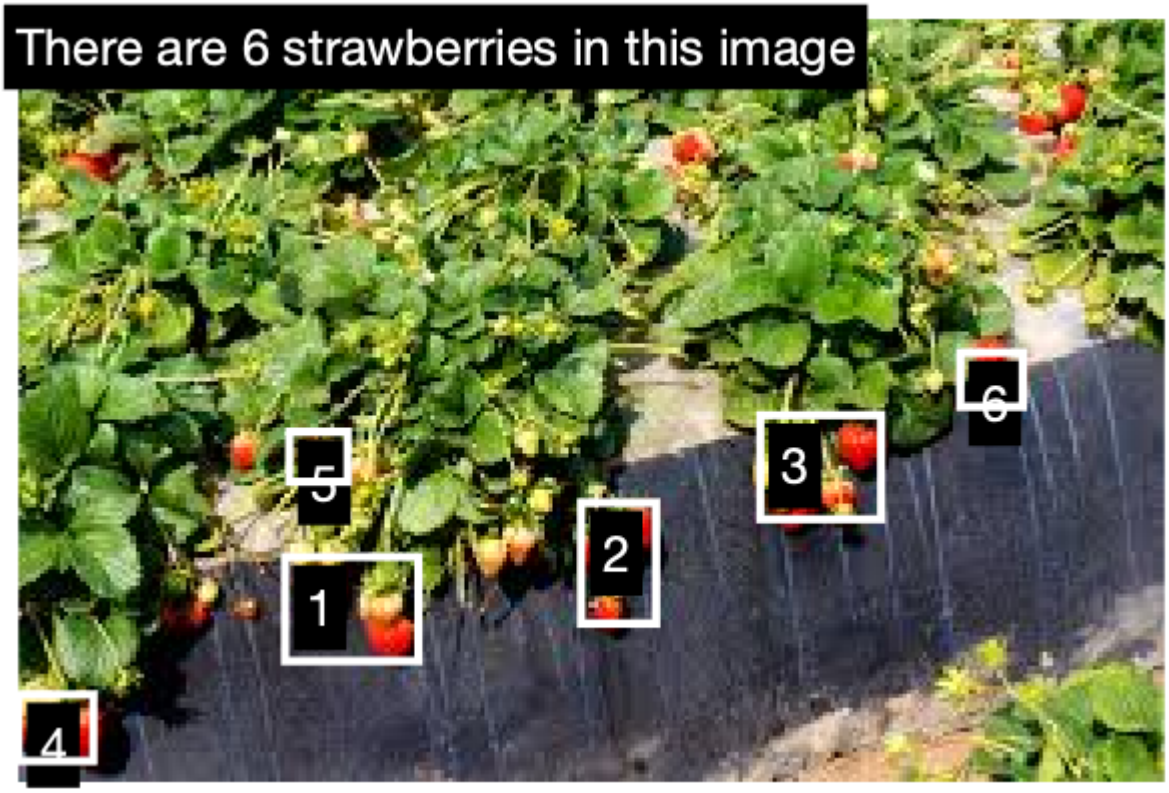

There are 1 strawberries in this image
1


There are 1 strawberries in this image
1


There are 2 strawberries in this image
2
1

There are 3 strawberries in this image

There are 6 strawberries in this image



Part 3 - K-Means and EM Clustering on Synthetic Data

Cluster Generation

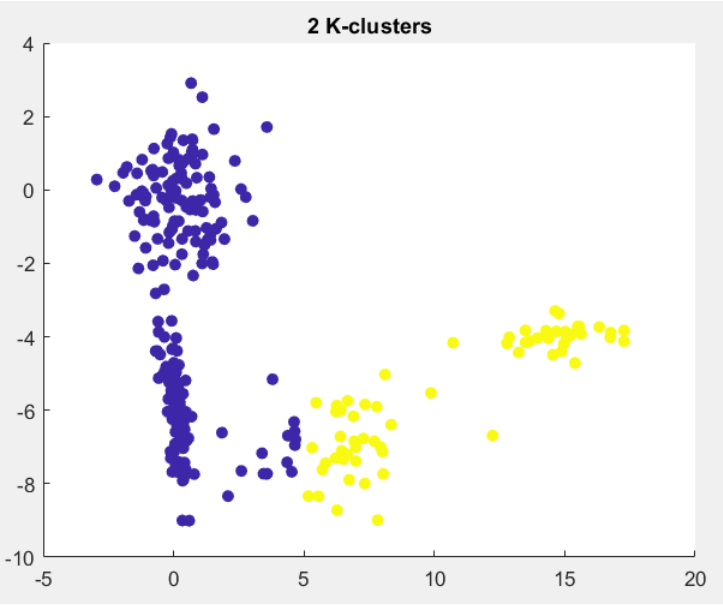
Synthetic cluster data was generated using MATLAB's mvnrnd() function. 5 2-D clusters, each with a different number of datapoints, means, and covariances, were generated using the code below.

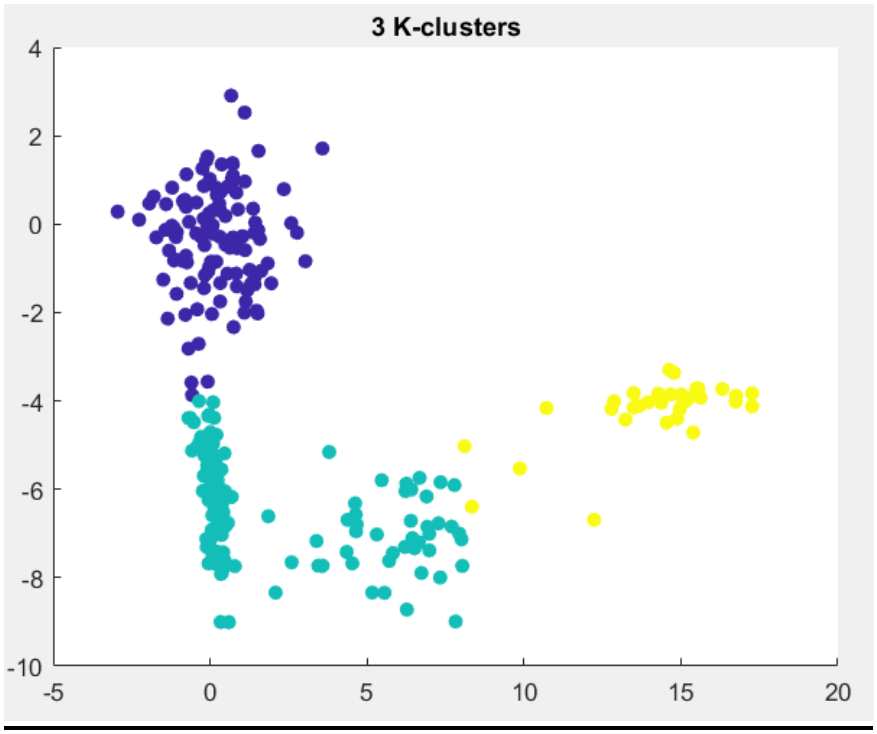
```
%% Parameters for the generation of the 5 random cluster points
% Vary the means and covariance matrices of the distributions as well as
% the number of points in each cluster.
NUM_CLUSTERS = 5;
NUM_PTS   = [100, 10, 30, 91, 50];
MU_PTS    = {[0 0],[1 -1],[15 -4],[.1 -6],[6 -7]};
SIGMA_PTS = {[1 0;0 1],[1 .75;.75 2],[2 -.1;-.1 .1],[.1 -.3;-.3 2],[5 .65;.65 1]};

%% Generate 2D clusters with different number of pts, means, and covariances
clusters = [];
for i=1:5
    clusters = [clusters; mvnrnd(MU_PTS{i}, SIGMA_PTS{i}, NUM_PTS(i))];
end
```

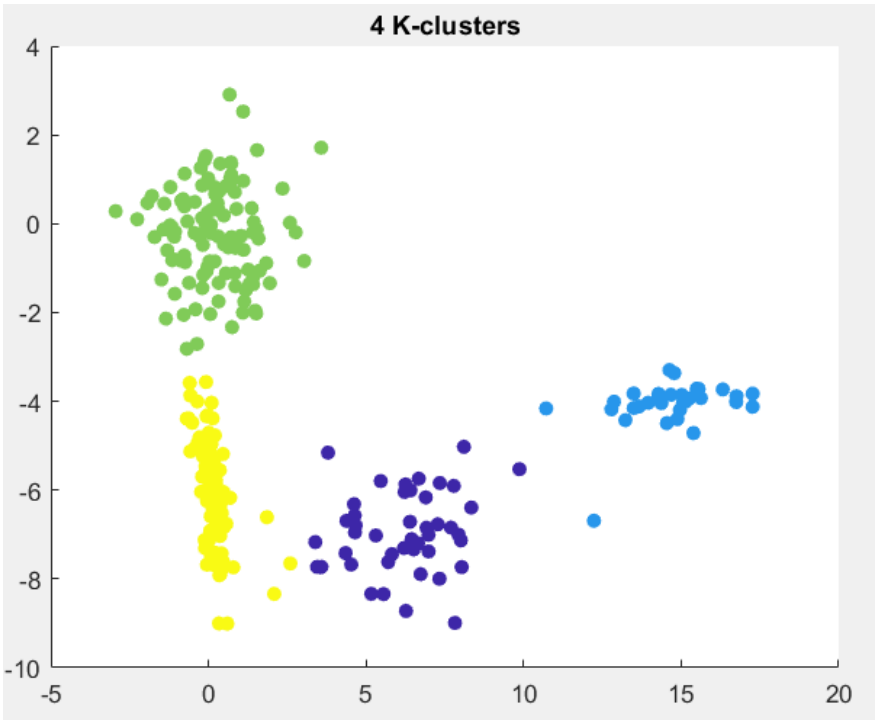
Next, the K-Means algorithm was run and displayed on the synthetic data using the code snippet below.

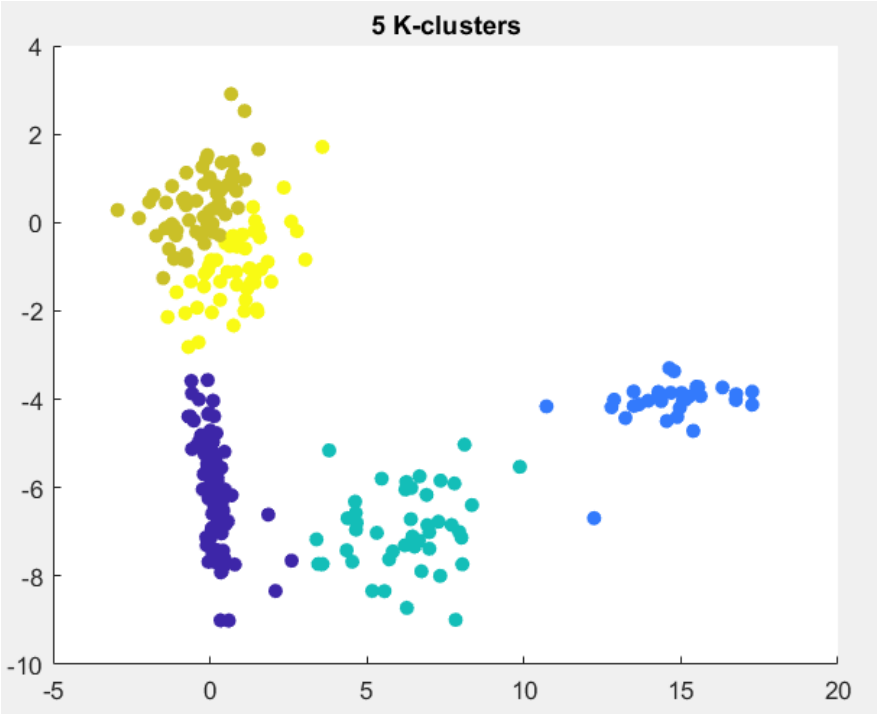
```
%% Run K-Means on the synthetic data generated in Step1.
% Does it always yield the optimal configuration? Discuss the results.
K_SIZES = [2 3 4 5];

for i=1:size(K_SIZES,2)
   figure();
   [IDX,C] = kmeans(clusters,K_SIZES(i));
   scatter(clusters(:,1),clusters(:,2),[],IDX,'filled');
   title(K_SIZES(i)+" K-clusters");
end
```

The results of the clustering algorithm are shown in the figures below for a k-size of 2, 3, 4, and 5.




3 K-clusters
4
2
0
-2
-4
-6
-8
-10
-5
0
5
10
15
20

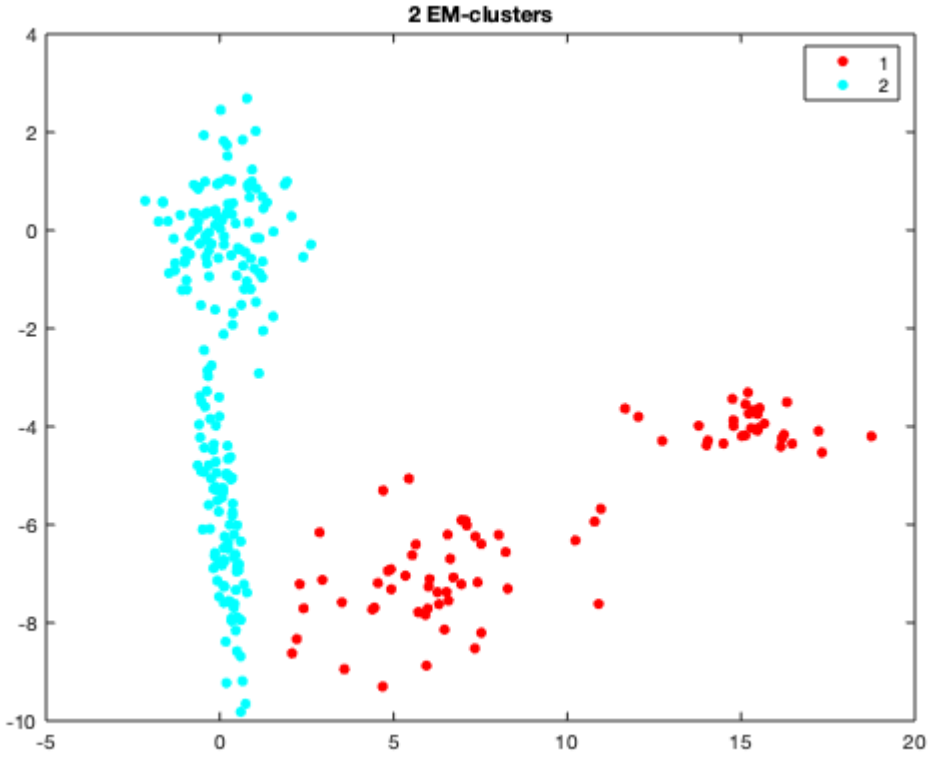

4 K-clusters
4
2
0
-2
-4
-6
-8
-10
-5
0
5
10
15
20

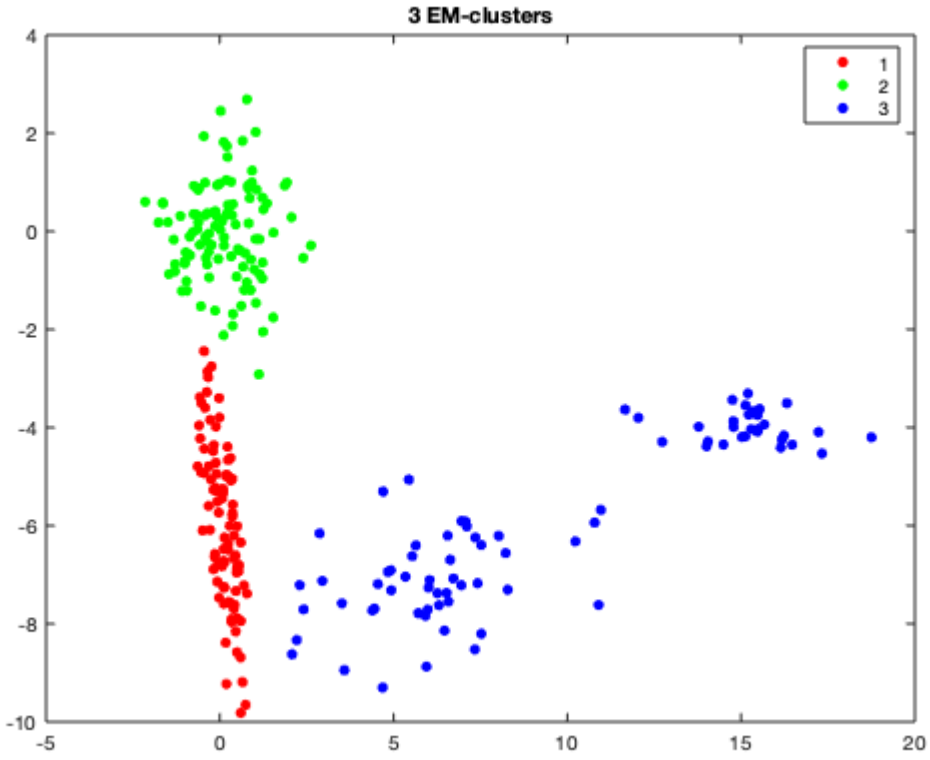


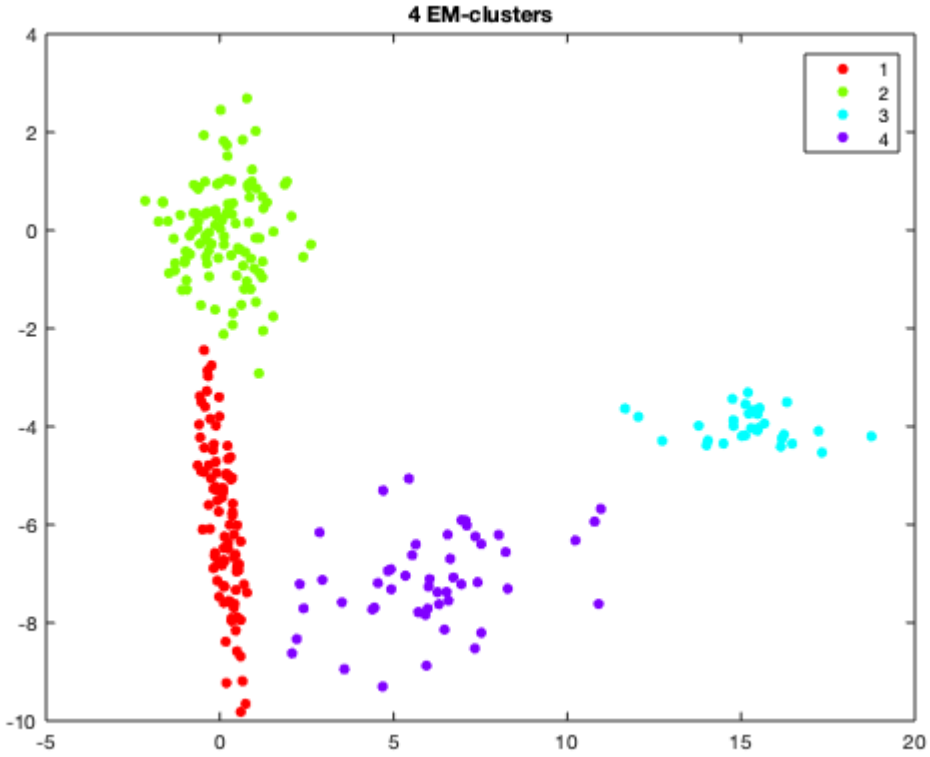
As we can see from the results, the algorithm does not always yield optimum results, especially if an incorrect number of clusters is used as a parameter for the K-means algorithm. In some cases, the data within a cluster is split within different K-clusters (if too-low of a k-value is used), and in other cases, there are multiple k-clusters assigned to a single cluster of data (if too-high of a k-value is used).

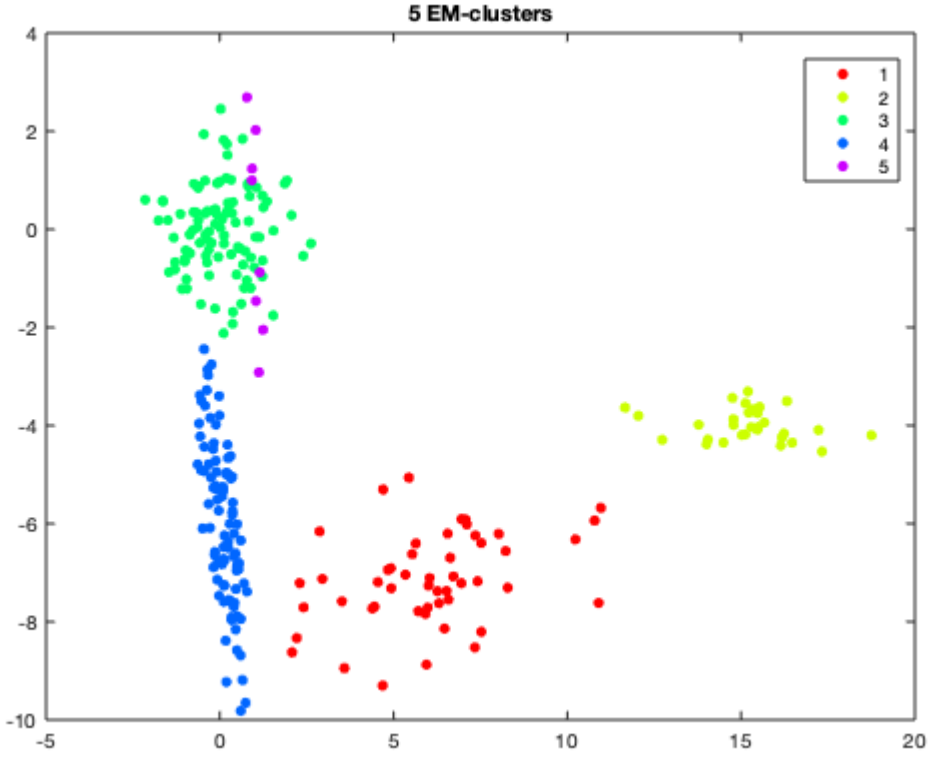
EM Clustering

EM Clustering is a soft clustering meaning that data points can be in multiple groups and that there is overlap of groups. This is different from K Means clustering which is a hard clustering algorithm meaning that a data point is either in one group or the other. The results of the EM Clustering Algorithm are shown below.


2 EM-clusters
1
2
4
2
0
-2
-4
-6
-8
-10
-5
0
5
10
15
20


3 EM-clusters
1
2
3
4
2
0
-2
-4
-6
-8
-10
-5
0
5
10
15
20


4 EM-clusters
1
2
3
4
4
2
0
-2
-4
-6
-8
-10
-5
0
5
10
15
20


5 EM-clusters
1
2
3
4
5
4
2
0
-2
-4
-6
-8
-10
-5
0
5
10
15
20

Comparison of Data Clustering Methods?

As we can see from these results, for the 2 group clusterings, there isn't much difference between the two clustering algorithms since with only 2 groups, the results are fairly binary. Once we get to 3 group clusterings , we can see how EM clusterings soft clustering approach affects the results as seen at the bottom where dark blue grouping differs from that of the K Means yellow and teal groupings even though those data points are the same. As we move to 4 and 5 group clusterings, the differences become more apparent especially with EM 5 group clustering having the 5th group straight through group 3 due to being able to overlap unlike in K Means.

Reflections

In this lab we learned color analysis techniques used to distinguish color-features within an image. By manually selecting regions-of-interest around a common object in many different pictures, in this case a strawberry, we were able to see the distribution of color intensities for the region's pixels and of the background's pixels. By converting between different color spaces, such as RGB, normalized RGB, HSV, YCbCr, and LAB, we were able to identify trends among the histograms that could be used for feature identification. Especially in the case of very red-intense strawberries, the HSV and normalized RGB color spaces were very useful in identifying the features. Furthermore, by using this data with a K-means and EM clustering algorithms, we were able to algorithmically identify these characteristics and features. The further analysis of the performance between K-Means and EM Clustering algorithms exposed the simplicity behind the K-means algorithm but also the greater power and performance of the EM algorithm. The K-means algorithm only worked ideally when initially provided the correct number of clusters to identify, and performance varied per run depending on the random initial conditions. The EM algorithm did not suffer from these problems.

-Alec Hardy

For this lab we learned more about segmenting an image based off of color which is very important in the real world. Previously we only worked with grayscale images and thresholded them to get image segmentation but for this lab, we used K Means and EM image segmentation algorithms that let us separate strawberries from their background of vegetation. We learned how each cluster of pixels was grouped for K Means and how if we place a mask of the original image on that K Means cluster that you can see what color that K Means Cluster represents. This allowed us to see which cluster corresponded to the image of the strawberry. For real world applications, this can be much better than using the image segmentation methods we used before since this worked when the subject of the image wasn't very clear which was interesting to work with. One of the workarounds that we had to discover to get the results we wanted was that we had to preprocess the image with a Gaussian filter so that there wouldn't be several image clusters which weren't strawberries. Also in this lab, I learned a lot more how to use regionprops to get the desired result. I previously used to only use 'Centroid' because I didn't understand how to use regionprops but this time around using Area and Bounding Box made the task of finding and labeling the strawberries much easier. As for part C, I learned how to implement EM clustering algorithm and how it affects synthetic data differently than it does with real images such as those of the strawberries.

-Nikhil Patolia

Attachment A - MATLAB Code

Part A Code:

```
clc
close all

% Parameters to help resize images to nearly the same size
SMALLEST_IMG_SIZE = [259 194];
SIZE_THRESHOLD = 1.5;

%% Load images and determine ROIs
% Only do this if we haven't already...
if exist('ROI','var')==0 || exist('I','var')==0

    % Cell array of all image mats and ROIs
    I    = {20};
    ROI = {20};

    cwd = pwd;
    I_files = dir(fullfile(cwd+"/ImgA",'*.jpg'));

    % Iterate across all images for Part A
    for i = 1:size(I_files,1)
        i_filename = I_files(i).name;
        I{i} = imread("ImgA/"+i_filename);

        % Resize if necessary
        while SIZE_THRESHOLD*size(I{i},1) > SMALLEST_IMG_SIZE(1)
            I{i} = subsample(I{i});
        end

        % Perform ROI selection
        imshow(I{i});
        ROI{i} = roipoly();
    end
end
% Now we have all of our images and ROIs

%% Plot our historgams for the foreground and background pixels

% RGB
planeHist(I, ROI, "RGB", "Red", "Green", "Blue");

% Normalized RGB
I_nRGB = {20};
for i=1:size(I,2)
   I_nRGB{i} =  normalizeRGB(I{i});
end
planeHist(I_nRGB, ROI, "Normalized RGB", "R", "G", "B");

% HSV
I_HSV = {20};
for i=1:size(I,2)
   I_HSV{i} =  rgb2hsv(I{i});
end
planeHist(I_HSV, ROI, "HSV", "Hue", "Saturation", "Value");

% Extra Credit
I_YCBCR = {20};
for i=1:size(I,2)
   I_YCBCR{i} =  rgb2ycbcr(I{i});
end
planeHist(I_YCBCR, ROI, "YCbCr", ...
    "Luminance (Y)", "Chrominance (Cb)", "Chrominance (Cr)");
```

```
%% LAB is going to be a pain because I can't use my planeHist() function
%% AGGHH!
I_LAB = {20};
for i=1:size(I,2)
   I_LAB{i} =  rgb2lab(I{i});
end
% Channel histogram, I'm not going to rename them to LAB
R_obj = [];
R_bg  = [];
G_obj = [];
G_bg  = [];
B_obj = [];
B_bg  = [];
for i=1:size(I_LAB,2)
    cur_img = I_LAB{i};

    % Extract the L, A, B planes
    L = cur_img(:,:,1);
    A = cur_img(:,:,2);
    B = cur_img(:,:,3);

    % Extract the pixels of the L, A, B, planes that are masked or unmasked
    R_bg = vertcat(R_bg,L(ROI{i}==0));
    R_obj = vertcat(R_obj,L(ROI{i}==1));
    G_bg = vertcat(G_bg,A(ROI{i}==0));
    G_obj = vertcat(G_obj,A(ROI{i}==1));
    B_bg = vertcat(B_bg,B(ROI{i}==0));
    B_obj = vertcat(B_obj,B(ROI{i}==1));
end
figure()
subplot(311);
histogram(R_obj);
ylabel("L Pixels");
subplot(312);
histogram(G_obj);
ylabel("A Pixels");
subplot(313);
histogram(B_obj);
ylabel("B Pixels");
sgtitle('LAB of Strawberry Pixels')

figure()
subplot(311);
histogram(R_bg);
ylabel("L Pixels");
subplot(312);
histogram(G_bg);
ylabel("A Pixels");
subplot(313);
histogram(B_bg);
ylabel("B Pixels");
sgtitle('LAB of Background Pixels')


function planeHist(I, ROI, name, redName, greenName, blueName)
%PLOTPLANES Summary of this function goes here

    %% Draw histograms for the RGB spaces

    % Channel histogram
    R_obj = [];
    R_bg  = [];
    G_obj = [];
    G_bg  = [];
    B_obj = [];
```

```
    B_bg  = [];
    for i=1:size(I,2)
        cur_img = I{i};

        % Extract the R, G, B planes
        R = cur_img(:,:,1);
        G = cur_img(:,:,2);
        B = cur_img(:,:,3);

        % Extract the pixels of the R, G, B, planes that are masked or unmasked
        R_bg = vertcat(R_bg,R(ROI{i}==0));
        R_obj = vertcat(R_obj,R(ROI{i}==1));
        G_bg = vertcat(G_bg,G(ROI{i}==0));
        G_obj = vertcat(G_obj,G(ROI{i}==1));
        B_bg = vertcat(B_bg,B(ROI{i}==0));
        B_obj = vertcat(B_obj,B(ROI{i}==1));
    end

    figure()
    subplot(311);
    imhist(R_obj);
    ylabel(redName + " Pixels");
    subplot(312);
    imhist(G_obj);
    ylabel(greenName + " Pixels");
    subplot(313);
    imhist(B_obj);
    ylabel(blueName + " Pixels");
    sgtitle(name + ' Histogram of Strawberry Pixels')

    figure()
    subplot(311);
    imhist(R_bg);
    ylabel("Red Pixels");
    subplot(312);
    imhist(G_bg);
    ylabel("Green Pixels");
    subplot(313);
    imhist(B_bg);
    ylabel("Blue Pixels");

    sgtitle(name + ' Histogram of Background Pixels')

end


function [ halved_mat ] = subsample( mat_input )
%Returns a "compressed" copy of the input image matrix
    halved_mat = mat_input(1:2:end,1:2:end,:);
end


function O = normalizeRGB(I)

    O = uint8(zeros(size(I)));

    for color=1:size(I,3)
        for row=1:size(I,1)
            for col=1:size(I,2)
                R = double(I(row,col,1))/255;
                G = double(I(row,col,2))/255;
                B = double(I(row,col,3))/255;
                cur_channel = double(I(row,col,color))/255;
                O(row,col,color) = uint8(255*(cur_channel / (R+B+G)));
            end
        end
```

```
    end
end
```

Part B Code

```
clc
close all
clear

I = {6};

cwd = pwd;
I_files = dir(fullfile(cwd+"/ImgB",'*.jpg'));

% Iterate across all images for Part B
for i = 1:size(I_files,1)
    i_filename = I_files(i).name;
    I{i} = imread("ImgB/"+i_filename);
end


for x = 1:6
    og = I{x};
    [row, col] = size(og);
    %Preprocess the image by Gaussian Filtering the image
    sizeBasedNum = round((row * col)^(0.23), 0);
    gaussianFilt = fspecial('gaussian', sizeBasedNum, 4);
    rgbImage = imfilter(og, gaussianFilt);



    %Choose the number of different K groups
    numKClasses = cast(sizeBasedNum * 0.7, 'uint8');
    k_vals = imsegkmeans(rgbImage,numKClasses);
    maxRed = 0;
    for n = 1 : numKClasses
        %For each K Group
        thisClass = k_vals == n;
        %Mask the original image over the parts over the K Group
        maskedRgbImage = bsxfun(@times, rgbImage, cast(thisClass, 'like', rgbImage));
        %See how many pixels are strawberry colored.  The one with the
        %highest count of strawberry colored pixels is chosen as the K
        %Group
        redPoints = maskedRgbImage(:,:,1)>=130 & maskedRgbImage(:,:,2)<=60 & maskedRgbImage(:,:,3)<=100;
        tempRed = sum(sum(redPoints));
        if tempRed > maxRed
            maxRed = tempRed;
            maxRedImage = thisClass;
            dur = maskedRgbImage;
        end
    end


    % Get connected components with area and bounding box
    rp = regionprops(maxRedImage, 'Area', 'BoundingBox');
    % Sort the connected components by area, descending
    [~, ind] = sort([rp.Area], 'descend');
    rp = rp(ind);

    figure();
    imshow(og);
    %The maxArea is the first in the list
    maxArea = rp(1).Area;
    i = 1;
    while i < length(rp)
        %Stop when you find a connected group that is 20% of the largest
        %strawberry
        currentArea = rp(i).Area;
```

```
        if currentArea < maxArea * 0.2
            break;
        end
        bb = rp(i).BoundingBox;
        %Counting number
        text(bb(1) + 5, bb(2) + 5, int2str(i) ,'Color', 'white', 'BackgroundColor', 'black', 'FontSize',
12, 'HorizontalAlignment','left','VerticalAlignment','top')
        %Box strawberry in white box
        rectangle('Position', [bb(1), bb(2), bb(3), bb(4)] ,'EdgeColor',[1 1 1], 'LineWidth' , 2)
        i = i + 1;
    end
    %Caption the image showing how many strawberries are in the image
    caption = sprintf("There are %d strawberries in this image", i - 1);
    text(0, 0, caption ,'Color', 'white', 'BackgroundColor', 'black', 'FontSize', 12,
'HorizontalAlignment','left','VerticalAlignment','top')
end
```

Part C Code

```
clc
close all

%% Parameters for the generation of the 5 random cluster points
% Vary the means and covariance matrices of the distributions as well as
% the number of points in each cluster.
NUM_CLUSTERS = 5;
NUM_PTS   = [100, 10, 30, 91, 50];
MU_PTS    = {[0 0],[1 -1],[15 -4],[.1 -6],[6 -7]};
SIGMA_PTS = {[1 0;0 1],[1 .75;.75 2],[2 -.1;-.1 .1],[.1 -.3;-.3 2],[5 .65;.65 1]};

%% Generate 2D clusters with different number of pts, means, and covariances
clusters = [];
for i=1:5
    clusters = [clusters; mvnrnd(MU_PTS{i}, SIGMA_PTS{i}, NUM_PTS(i))];
end

%% Run K-Means on the synthetic data generated in Step1.
% Does it always yield the optimal configuration? Discuss the results.
K_SIZES = [2 3 4 5];

for i=1:size(K_SIZES,2)
   figure();
   [IDX,C] = kmeans(clusters,K_SIZES(i));
   scatter(clusters(:,1),clusters(:,2),[],IDX,'filled');
   title(K_SIZES(i)+" K-clusters");
end


for i=1:size(K_SIZES,2)
    gmm = fitgmdist(clusters,K_SIZES(i));
    em_idx = cluster(gmm, clusters);
    figure()
    gscatter(clusters(:,1), clusters(:,2), em_idx)
    title(K_SIZES(i)+" EM-clusters");
end
```