**CPE 428: Project 5**

Team Name

Yu Asai

Alan Nonaka

Luis Gomez

February 29, 2020

**Objective: Project 5 is an exercise in Color Analysis and Image Segmentation.**

Table of Contents

[**Part 1: Strawberry Color Analysis**](#_m9z428ynz6ru) **3**

[Part 1.1: ROI Masks Strawberry Image Segmentation](#_38wbtsv1sds) 3

[Figure 1: Original Image (left) and Resized Mask (right)](#_vhoey3e3l4sf) 3

[Figure 2: Resized Image with Strawberry Segmentation - Strawberry (left) Background (right)](#_tjqzl744q0ca) 3

[Part 1.2: Strawberry Image Segments Separate and Combined Histograms](#_835dg0p2fou3) 4

[Figure 3: RGB Histogram of All Pixels (left), Strawberry Pixels (center), and Background Pixels (right)](#_idgngkltt0fa) 4

[Figure 4: Strawberry Photo 8, Cool Lighting](#_6hdda4mzp8mh) 4

[Part 1.3-5: Color Space Experimentation](#_6i1qzi3htznv) 5

[Figure 5: Normalized RGB Histogram of All Pixels (left), Strawberry Pixels (center), and Background Pixels (right)](#_gegcdid9fw5g) 5

[Figure 6: HSV Histogram of All Pixels (left), Strawberry Pixels (center), and Background Pixels (right)](#_2km96e4ioqwp) 5

[**Part 2: Strawberry Finding/Counting Using K-Means and EM**](#_xdmztr39yp9j) **6**

[Part 2.1: K-Means and EM Clustering Algorithms](#_u0jbdgpc9cj1) 6

[Figure 7: Original Image of “t1.JPG” (left) and its corresponding segmentation image using K = 3](#_uc8yd9m4mno8) 6

[Figure 8: Original Image of “t2.JPG” (left) and its corresponding segmentation image (right).](#_kfzxm2nl4vy8) 7

[Figure 9: Original Image of “t3.JPG” (left) and its corresponding segmentation image (right).](#_lt67mkt3iopv) 8

[Figure 10: Original Image of “t4.JPG” (left) and its corresponding segmentation image (right).](#_g74w8bl16n5s) 8

[Figure 11: Original Image of “t5.JPG” (left) and its corresponding segmentation image (right).](#_t28rd0x9s02h) 9

[Figure 12: Original Image of “t6.JPG” (left) and its corresponding segmentation image (right).](#_oa7phns2gtvg) 9

[Part 2.2: Team Strawberry Identification and Counting Algorithm](#_lusj3fc4e04z) 12

[Part 2.3: Strawberry Counting Algorithm using best feature from PartA](#_nhwi6ycy47v) 13

[**Part 3: K-Means and EM Clustering on Synthetic data**](#_o7xchezfys6p) **14**

[Part 2.1: K-Means Algorithms](#_3uguymk3z8wk) 14

[Part 2.2: EM Clustering Algorithms](#_nua8vapm1kp0) 14

[Part 2.3: K-Means vs EM Clustering Comparison](#_4zj05mql8pay) 14

[**Conclusions**](#_gi6408jtt1y9) **15**

[**Appendix A: Image**](#_r7h5lmmr9zjj) **16**

[Appendix A: Python and Matlab ROI Segmentation Code](#_itpyxfwt5wfo) 16

# 

# **Part 1: Strawberry Color Analysis**

In this assignment, you will analyze strawberry pixels and non-strawberry pixels in different color spaces and identify the best color component(s) to perform strawberry detection.

## **Part 1.1: ROI Masks Strawberry Image Segmentation**

To build the strawberry‐color model for strawberry detection, use all 20 images from the folder PartA, and manually draw a polygonal region of interest (ROI) around the strawberry to include as many strawberry pixels as possible in the ROI while leave non-strawberry pixels outside the ROI. Create a set of 20 mask images in which 1’s indicate strawberry pixels and 0’s indicate non‐strawberry pixels. Label and save those mask images. Since the 20 images are of different sizes, you might want to resize images with high resolutions to keep all images at around the same size of the smallest image.

|  |  |
| --- | --- |
|  |  |

### *Figure 1: Original Image (left) and Resized Mask (right)*

|  |  |
| --- | --- |
|  |  |

### *Figure 2: Resized Image with Strawberry Segmentation - Strawberry (left) Background (right)*

## **Part 1.2: Strawberry Image Segments Separate and Combined Histograms**

Examine color distributions for strawberry and non‐strawberry pixels in RGB color space. For this, construct a histogram of all strawberry pixels from the images in folder PartA with the help of the mask images and compare it with the histogram of all non-strawberry pixels from the same set of images. You need to construct a total of 3 sets of histograms. Examine color distributions between strawberry and non-strawberry pixels. What do you observe?

|  |  |  |
| --- | --- | --- |
|  |  |  |

### *Figure 3: RGB Histogram of All Pixels (left), Strawberry Pixels (center), and Background Pixels (right)*

After calculating the RGB histogram of Strawberry pixels, Background pixels, and both pixels, we come to intuitive conclusions. The strawberry segmented pixels have a very strong red response rate at peak saturation. This means the pixels are dominated by red, the color of strawberries. Inversely, the background pixels do not have a strong red contribution rate, with equivalent contributions of red, green, and blue. With the combined images, we see that red, green, and blue have similar contributions until peak saturation where there is a high red response represented by the strawberries.

The only surprising result was the peak of blue pixels near zero saturation. However, upon examination of the strawberry photos, I noticed a trend in a couple photos. Some of the images were taken with cool lighting. This term refers to the lighting color, here it appears most photos have much less red and green lightning undertones. This is one hypothesis for why there is a large blue response at low saturation, but we are not certain why this peak occurs.

|  |
| --- |
|  |

### *Figure 4: Strawberry Photo 8 - Cool Lighting*

## **Part 1.3-5: Color Space Experimentation**

Experiment with two other color spaces- normalized rgb and HSV (Here since hue is defined on a ring, it might be advantageous to represent strawberry hue in a continuous range, how can you do it?). Again, examine color distributions for the strawberry and non‐strawberry pixels in those color spaces. 4. What is/are the best feature(s) for characterizing strawberry‐color? Why? 5. Extra credit: Examine additional color spaces such as YCbCr, and Lab.

|  |  |  |
| --- | --- | --- |
|  |  |  |

### *Figure 5: Normalized RGB Histogram of All Pixels (left), Strawberry Pixels (center), and Background Pixels (right)*

In the Normalized RGB color space. We observe a very distinct histogram shape. There is a large saturation of red, but not at 255, because it is normalized so all RGB add to one (255). We observe how background pixels dominate the total histogram, but isolating strawberry pixels creates a unique histogram with significantly higher than average red.

|  |  |  |
| --- | --- | --- |
|  |  |  |

### *Figure 6: HSV Histogram of All Pixels (left), Strawberry Pixels (center), and Background Pixels (right)*

In the HSV (hue saturation, value) color space, we notice a similar trend on the hue channel. In strawberry images there was a very high response near zero, which corresponds to a red hue in HSV color space. This was a very high peak we should investigate more.

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

Write a function that extract the feature vector from the detected keypoints. The function should have the format: [extracted\_features] = my\_extractFeatures\_a/b(image, detected\_pts) For this part, you can use MATLAB’s built-in feature detectors (such as FASTFirst use raw pixel data in a small square window (say 5x5) around the keypoint as the feature descriptor. This should work well when the images you are comparing are related by only a translation.

## **Part 2.1: K-Means and EM Clustering Algorithms**

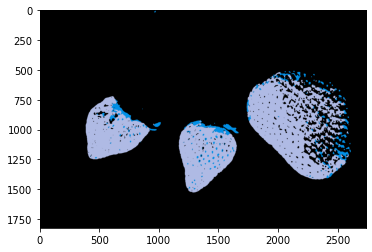
Run K-Means and EM clustering algorithms using RGB values of a color image. Perform image segmentation on images in folder PartB. Display and discuss the segmentation results with different initializations and different values of K. Show the best result.

**K means Clustering**

Souce: https://towardsdatascience.com/introduction-to-image-segmentation-with-k-means-clustering-83fd0a9e2fc3

Changing values of K based on the number of clusters in the image.





### 

### 

### 

### *Figure 7: Original Image of “t1.JPG” (left) and its corresponding segmentation image using K = 3*



### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

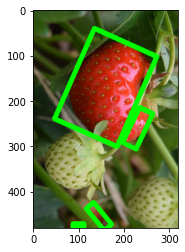
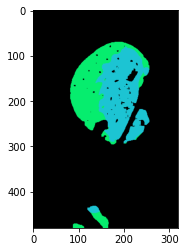
### 

### 

### 

### *Figure 8: Original Image of “t2.JPG” (left) and its corresponding segmentation image (right).*

You can observe different values of K. K = 4, you can clearly see all the different shades of the strawberries. K=6 you can distinguish the different colors of the strawberry leaves.



### 

### 

### 

### *Figure 9: Original Image of “t3.JPG” (left) and its corresponding segmentation image (right).*

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

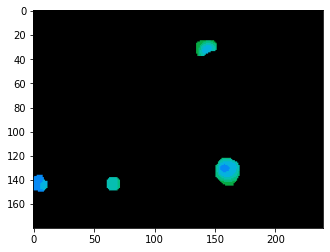
### 

### 

### 

### 

### *Figure 10: Original Image of “t4.JPG” (left) and its corresponding segmentation image (right).*



### 

### 

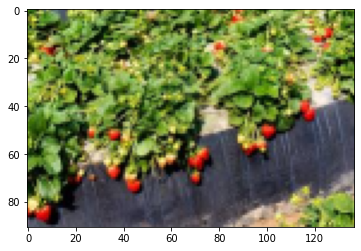
### 

### 

### 

### 

### *Figure 11: Original Image of “t5.JPG” (left) and its corresponding segmentation image (right).*



### 

### 

### 

### 

### 

### 

### 

### 

## 

## 

## 

## *Figure 12: Original Image of “t6.JPG” (left) and its corresponding segmentation image (right).*

**Python code**

|  |
| --- |
| *# Helper Functions for displaying image and scaling*  *def scale\_img(img, scale\_percent):*  *#print("\nscale\_img()")*  *#print(img.shape)*    *# Scaled Image Dimensions*  *width = int(img.shape[1] \* scale\_percent / 100)*  *height = int(img.shape[0] \* scale\_percent / 100)*  *dim = (width, height)*    *# resize image*  *img = cv2.resize(img, dim, interpolation = cv2.INTER\_AREA)*  *#print(img.shape)*  *return img*  *# show grayscale image*  *def show\_grayscale( my\_image ):*  *plt.figure(figsize = (10,10))*  *plt.imshow(my\_image, cmap = plt.get\_cmap(name = 'gray'))*  *plt.show()*  *# horizontally concatenate 2 images*  *def concat\_horiz(img1, img2):*  *vis = np.concatenate((img1, img2), axis=1)*  *show\_grayscale(vis)*  *return vis*  Cool Medium article on K means  # https://towardsdatascience.com/introduction-to-image-segmentation-with-k-means-clustering-83fd0a9e2fc3  # Opencv guide to changing colorspaces  # https://docs.opencv.org/trunk/df/d9d/tutorial\_py\_colorspaces.html    # Conducts Image Segmentation using K Means Clustering, shows result  # Input: Image File, Number of Clusters Desired, Number of attemps, print debug bool  def image\_seg\_Kmeans( file, K=3, attempts=10,print=False):  # read image & Scale down  img = cv2.imread(file)  img = scale\_img(img, 50)    # convert to RGB  #print('RGB')  rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  plt.imshow(rgb)  plt.show()  # convert to HSV & Blur  #print("HSV")  blurred = cv2.GaussianBlur(img, (15,15),0)  hsv = cv2.cvtColor(blurred, cv2.COLOR\_BGR2HSV)  plt.imshow(hsv)  plt.show()    # Threshold the HSV image to get only red colors  # Lower mask then upper mask  mask1 = cv2.inRange(hsv, (0, 100, 0), (10, 255, 255))  mask2 = cv2.inRange(hsv, (170 ,100 ,0), (180 ,255 ,255))    # Erode Masks  mask1 = cv2.erode(mask1, None, iterations=2)  mask1 = cv2.dilate(mask1, None, iterations=2)  mask2 = cv2.erode(mask2, None, iterations=2)  mask2 = cv2.dilate(mask2, None, iterations=2)  #mask1 = mask1.astype('bool')  #mask2 = mask2.astype('bool')    # Combine Lower and Upper Masks  mask = mask1 + mask2  plt.imshow(mask)  plt.show()    # find countours in the mast and init current (x,y) of strawberry  cnts = cv2.findContours(mask.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  cnts = imutils.grab\_contours(cnts)  center = None  NumStraw = 0  for cnt in cnts:  NumStraw = NumStraw + 1  rect = cv2.minAreaRect(cnt)  box = cv2.boxPoints(rect)  box = np.int0(box)  cv2.drawContours(rgb,[box],0,(0,255,0),10)    #print ("Detected %d Strawberries" %NumStraw)  plt.imshow(rgb)  plt.show()    ## only proceed if at least one contour was found  #if len(cnts) > 0:  # find the largest contour in the mask, then use  # it to compute the minimum enclosing circle and  # centroid  #c = max(cnts, key=cv2.contourArea)  #((x, y), radius) = cv2.minEnclosingCircle(c)  #M = cv2.moments(c)  #center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))  # only proceed if the radius meets a minimum size  #if radius > 10:  # draw the circle and centroid on the frame,  # then update the list of tracked points  #cv2.circle(frame, (int(x), int(y)), int(radius),  # (0, 255, 255), 2)  #cv2.circle(frame, center, 5, (0, 0, 255), -1)    # Bitwise-AND mask and original image  res = cv2.bitwise\_and(hsv, hsv, mask= mask)  plt.imshow(res)  plt.show()    # vector Z  Z = res.reshape((-1,3))  # convert to np.float32  Z = np.float32(Z)  if(print):  print(Z.shape)  # define criteria, number of clusters(K) and apply kmeans()  criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, attempts, 0.2)  # number of clusters  ret,label,center=cv2.kmeans(Z,K,None,criteria,10,cv2.KMEANS\_RANDOM\_CENTERS)  # Now convert back into uint8, and make original image  center = np.uint8(center)  # flatten labels array  res2 = center[label.flatten()]  # reshape back to original image dimension  res3 = res2.reshape((img.shape))  plt.imshow(res3)  plt.show()  # find countours in the mast and init current (x,y) of strawberry  #cnts = cv2.findContours(res3.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  #cnts = imutils.grab\_contours(cnts)  #center = None  #for cnt in cnts:  # rect = cv2.minAreaRect(cnt)  # box = cv2.boxPoints(rect)  # box = np.int0(box)  # cv2.drawContours(rgb,[box],0,(0,255,0),10)    plt.imshow(rgb)  plt.show() |

## **Part 2.2: Team Strawberry Identification and Counting Algorithm**

Based on the segmentation results in the last step, develop a strawberry finding and counting algorithm. Place a white bounding box around the detected strawberry

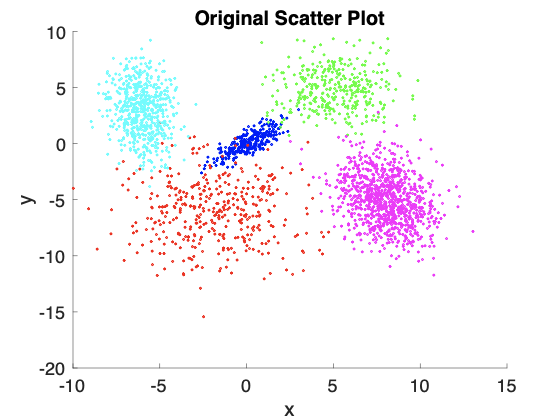
|  |
| --- |
| See Part A Code |

## **Part 2.3: Strawberry Counting Algorithm using best feature from PartA**

|  |
| --- |
| See Part A Code |

# **Part 3: K-Means and EM Clustering on Synthetic data**

Generate several (3-5) clusters of 2D data points by using mvnrnd(). Data in each cluster follow a multivariate Gaussian distribution. Vary the means and covariance matrices of the distributions as well as the number of points in each cluster.

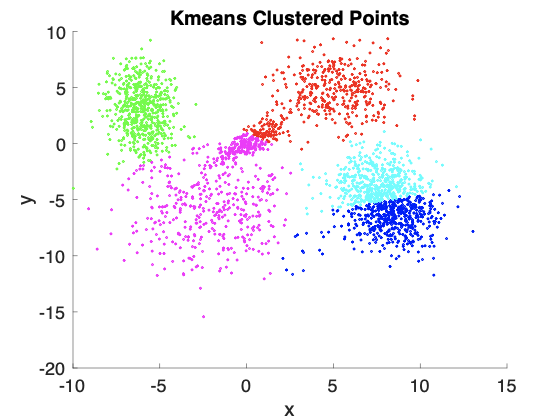


### *Figure 1x: Original Scatter Plotted by Group Color*

## 

## **Part 3.1: K-Means Algorithms**

Run K-Means on the synthetic data generated in Step1. Does it always yield the optimal configuration? Discuss the results.



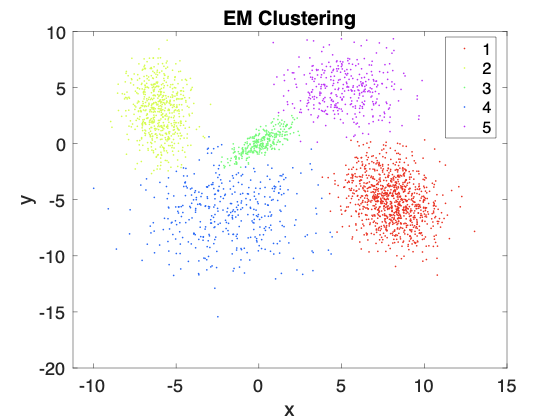
### *Figure 1x: K-Means Clustering Points Scatter Plotted by Color*

Above is the K-Means clustered points of our randomly generated synthetic data. Unfortunately, we see how k-means fails to cluster points correctly. It is unable to separate the diagonal three points into three different groups, and thus only sees two groups. The final cluster is then split in two to make 5 groups. This is likely due to k-means struggle with clustering elliptical shapes like the center grouping.

## 

## **Part 3.2: EM Clustering Algorithms**

Run EM clustering on the synthetic data generated in Step1.



### *Figure 1x: EM Clustering Points Scatter Plotted by Color*

## **Part 2.3: K-Means vs EM Clustering Comparison**

How do the two data clustering techniques compare?

EM cluster was significantly more robust to non-circular shapes than K-Means clustering. This is demonstrated by EM’s ability to distinctly separate the three diagonal distributions into three groups. K-Means was unable to identify three groups and split the group into two parts, leaving another very distinct cluster to incorrectly be split in two. EM is still not able to be 100% accurate, but it is very good at tracking and separating points within reasonable standard deviations.

# **Conclusions**

**Yu Asai**:

From this lab, I learned that by extracting the characteristics from the image we can determine which characteristic is the best to analyze clustering. The most difficult part of this project was finding the right documentation for counting the strawberries when there is many noise.

**Luis Gomez**:

**Alan Nonaka**:

This lab instructed how to produce image grouping and identification with k-means and EM-clustering. The main conclusion is EM-clustering can better segment data grouping due to its ability to draw elliptical grouping. This was best demonstrated with the synthetic data analysis, where k-means failed to converge to a meaningful answer. Additionally we observed how transforming images into different color spaces may have advantages with image segmentation, by producing higher contrast between the relevant strawberry pixels and backgrounds.

# 

# **Appendix A: Image**

## **Appendix A: Python and Matlab ROI Segmentation Code**

p5a\_resize.py - This code resizes the images to the same size as the smallest image

|  |
| --- |
| *# EE 428 Computer Vision*  *# Winter 2020 Cal Poly*  *# Yu Asai, Luis Gomez, Alan Nonaka*  import sys  import os  from PIL import Image, ImageOps  import numpy as np  from PIL import Image  *# this function opens a file and resizes it to a given width and height*  def resize\_image(filename, w, h):  image\_array = Image.open(filename)  resized\_image = np.asarray(ImageOps.fit(image\_array, (w,h), Image.ANTIALIAS))  return resized\_image  *# this function uses the resize function, then saves the image as a seperate file*  def resize\_and\_save\_image(filename):  *# (259, 194) is the smallest image, resize all image to this size*  w = 256; h = 194;  resized\_image = resize\_image(filename, w, h)  *# split the given filename to generate new filename*  x = filename.split(".")  poop = '\_resized.'  resized\_filename = x[0] + poop + x[1]    *#resize image then save as a different image name*  resized\_IMAGE = Image.fromarray(resized\_image)  resized\_IMAGE = resized\_IMAGE.save(resized\_filename)  *# for all 20 images, resize the image and save it*  for i in range(1,21):  filename = 's' + str(i) + '.jpg'  resize\_and\_save\_image(filename) |

p5a\_segment.mat - This code segments the resized strawberry image into strawberry and background pixels

|  |
| --- |
| for i=1:20  *% index through all 20 images, resize them, segment them, and save them*    *% generate filename for the resized image*  filename = horzcat('s', int2str(i), '\_resized.jpg');  *% split filename for saving images later*  split\_name = split(filename, '.');  *% open strawberry image file*  im = imread(filename);    *% open image for ROI polygon tool*  mask = roipoly(im);  *% generate a name to save the image mask*  mask\_filename = horzcat(char(split\_name(1)), '\_mask.', char(split\_name(2)));  *% save the mask of the resized image*  imwrite(mask, mask\_filename);      int\_mask = uint8(mask); *% cast the mask to uint8 then mask the image*  strawberry(:,:,1) = im(:,:,1) .\* int\_mask; *% red pixel masking*  strawberry(:,:,2) = im(:,:,2) .\* int\_mask; *% green pixel masking*  strawberry(:,:,3) = im(:,:,3) .\* int\_mask; *% blue pixel masking*    *% generate name of strawberry segmented image*  strawberry\_filename = horzcat(char(split\_name(1)), '\_strawberry.', char(split\_name(2)));  *% save strawberry image*  imwrite(strawberry, strawberry\_filename);    *% get inverse of mask to get background of strawberry image*  not\_int\_mask = uint8(not(mask));  background(:,:,1) = im(:,:,1) .\* not\_int\_mask; *% red pixel masking*  background(:,:,2) = im(:,:,2) .\* not\_int\_mask; *% green pixel masking*  background(:,:,3) = im(:,:,3) .\* not\_int\_mask; *% blue pixel masking*    *% generate name of background segmented image*  background\_filename = horzcat(char(split\_name(1)), '\_background.', char(split\_name(2)));  *% save background image*  imwrite(background, background\_filename);    end |

p5a\_histogram.py - This code generates histograms for segmented and combined images for strawberry and background

|  |
| --- |
| *# EE 428 Computer Vision*  *# Winter 2020 Cal Poly*  *# Yu Asai, Luis Gomez, Alan Nonaka*  import sys  import os  from PIL import Image, ImageOps  import numpy as np  import cv2 as cv  from matplotlib import pyplot as plt  from PIL import Image  def get\_histogram(image\_filename, mask\_filename, use\_mask):  *# opens image and its mask*  img = cv.imread(image\_filename);  mask = cv.imread(mask\_filename,0);    *# decide if the mask is for background, foreground, or no mask*  if use\_mask:  if use\_mask == 2:  mask = 255 - mask;  else:  mask = None    *# generate histogram for each color using a mask*  blue = cv.calcHist([img],[0],mask,[256],[0,256]);  green = cv.calcHist([img],[1],mask,[256],[0,256]);  red = cv.calcHist([img],[2],mask,[256],[0,256]);  return blue, green, red  def plot\_RGB\_histogram(blue, green, red, plt\_title, filename):  *# plot all three RGB colors on the same plot*  plt.plot(blue,color = 'b');  plt.plot(green,color = 'g');  plt.plot(red,color = 'r');    *# plot labels and making it pretty*  plt.xlabel('Intensity')  plt.ylabel('Bins')  plt.title(plt\_title)  plt.xlim([0,256])  plt.savefig(filename,dpi=300,bbox\_inches='tight',pad\_inches=0.1)  plt.show()  *# generate first histogram for all pixels*  blue, green, red = get\_histogram('s1\_resized.jpg', 's1\_resized\_mask.jpg', 0)  *# add the histogram of the other 19 strawberry images*  for i in range(2,21):  b, g, r = get\_histogram('s' + str(i) + '\_resized.jpg', 's' + str(i) + '\_resized\_mask.jpg', 0)  blue = blue + b  green = green + g  red = red + r  *# save and display the histogram for all pixels*  plot\_RGB\_histogram(blue, green, red, 'Total Pixel Histogram','total\_histogram.jpg')  *# generate first histogram for strawberry pixels*  blue, green, red = get\_histogram('s1\_resized.jpg', 's1\_resized\_mask.jpg', 1)  *# add the histogram of the other 19 strawberry segmented images*  for i in range(2,21):  b, g, r = get\_histogram('s' + str(i) + '\_resized.jpg', 's' + str(i) + '\_resized\_mask.jpg', 1)  blue = blue + b  green = green + g  red = red + r  *# save and display the histogram for strawberry segmented pixels*  plot\_RGB\_histogram(blue, green, red, 'Strawberry Pixel Histogram','strawberry\_histogram.jpg')  *# generate first histogram for background pixels*  blue, green, red = get\_histogram('s1\_resized.jpg', 's1\_resized\_mask.jpg', 2)  *# add the histogram of the other 19 background segmented images*  for i in range(1,21):  b, g, r = get\_histogram('s' + str(i) + '\_resized.jpg', 's' + str(i) + '\_resized\_mask.jpg', 2)  blue = blue + b  green = green + g  red = red + r  *# save and display the histogram for background segmented pixels*  plot\_RGB\_histogram(blue, green, red, 'Background Pixel Histogram','background\_histogram.jpg') |

p5a\_HSV\_histogram.m

|  |
| --- |
| *# EE 428 Computer Vision*  *# Winter 2020 Cal Poly*  *# Yu Asai, Luis Gomez, Alan Nonaka*  *%--------------------- All Pixels ----------------------*  h = zeros(256,1); s = zeros(256,1); v = zeros(256,1);  for i=1:20 *% iterate through all pixels*  *% open strawberry image file and convery to HSV*  im = imread(horzcat('s', int2str(i), '\_resized.jpg'));  im\_hsv = rgb2hsv(im);  *% tally up all HSV pixels*  h = h + imhist(im\_hsv(:,:,1));  s = s + imhist(im\_hsv(:,:,2));  v = v + imhist(im\_hsv(:,:,3));  end  *% plot HSV total pixels together on one histogram*  x=1:256; figure(1); clf(1);  plot(x, h, 'm', 'DisplayName', 'Hue'); hold on  plot(x, s, 'c', 'DisplayName', 'Saturation'); hold on  plot(x, v, 'r', 'DisplayName', 'Intensity'); hold on  xlim([1,256]); ylim([0,30000]);  xlabel('magnitude'); ylabel('bins');  set(gca,'FontSize',18);  title('HSV Total Pixel Histogram', 'FontSize', 20);  legend('Location','northeast','Orientation','vertical');  *%----------------- Strawberry Pixels ------------------*  h = zeros(256,1); s = zeros(256,1); v = zeros(256,1);  for i=1:20 *% iterate through all pixels*  *% open strawberry image file and convery to HSV*  im = imread(horzcat('s', int2str(i), '\_resized\_strawberry.jpg'));  im\_hsv = rgb2hsv(im);  *% tally up all HSV pixels*  h = h + imhist(im\_hsv(:,:,1));  s = s + imhist(im\_hsv(:,:,2));  v = v + imhist(im\_hsv(:,:,3));  end  *% plot HSV total pixels together on one histogram*  x=1:256; figure(2); clf(2);  plot(x, h, 'm', 'DisplayName', 'Hue'); hold on  plot(x, s, 'c', 'DisplayName', 'Saturation'); hold on  plot(x, v, 'r', 'DisplayName', 'Intensity'); hold on  xlim([1,256]); ylim([0,30000]);  xlabel('magnitude'); ylabel('bins');  set(gca,'FontSize',18);  title('HSV Strawberry Pixel Histogram', 'FontSize', 20);  legend('Location','northeast','Orientation','vertical');  *%---------------- Background Pixels -----------------*  h = zeros(256,1); s = zeros(256,1); v = zeros(256,1);  for i=1:20 *% iterate through all pixels*  *% open strawberry image file and convery to HSV*  im = imread(horzcat('s', int2str(i), '\_resized\_background.jpg'));  im\_hsv = rgb2hsv(im);  *% tally up all HSV pixels*  h = h + imhist(im\_hsv(:,:,1));  s = s + imhist(im\_hsv(:,:,2));  v = v + imhist(im\_hsv(:,:,3));  end  *% plot HSV total pixels together on one histogram*  x=1:256; figure(3); clf(3);  plot(x, h, 'm', 'DisplayName', 'Hue'); hold on  plot(x, s, 'c', 'DisplayName', 'Saturation'); hold on  plot(x, v, 'r', 'DisplayName', 'Intensity'); hold on  xlim([1,256]); ylim([0,30000]);  xlabel('magnitude'); ylabel('bins');  set(gca,'FontSize',18);  title('HSV Background Pixel Histogram', 'FontSize', 20);  legend('Location','northeast','Orientation','vertical'); |