2022

Assignment 2

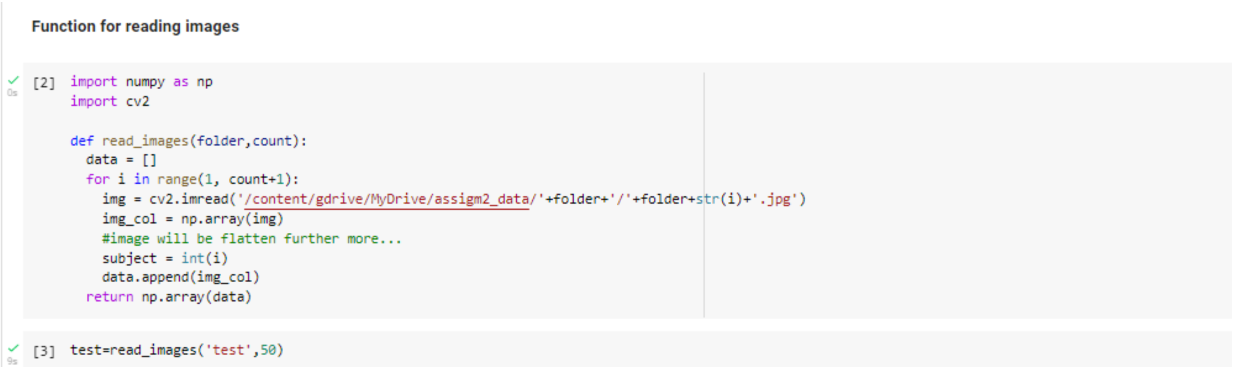
(Image Segmentation)

YOUSSEF HASSAN	6259
NOUR EL-DIN HAZEM	6261

1. Download the Dataset and Understand the Format:

[we work only on 50 images from the test images]

A function used to read images from a folder in the drive and save it in np array



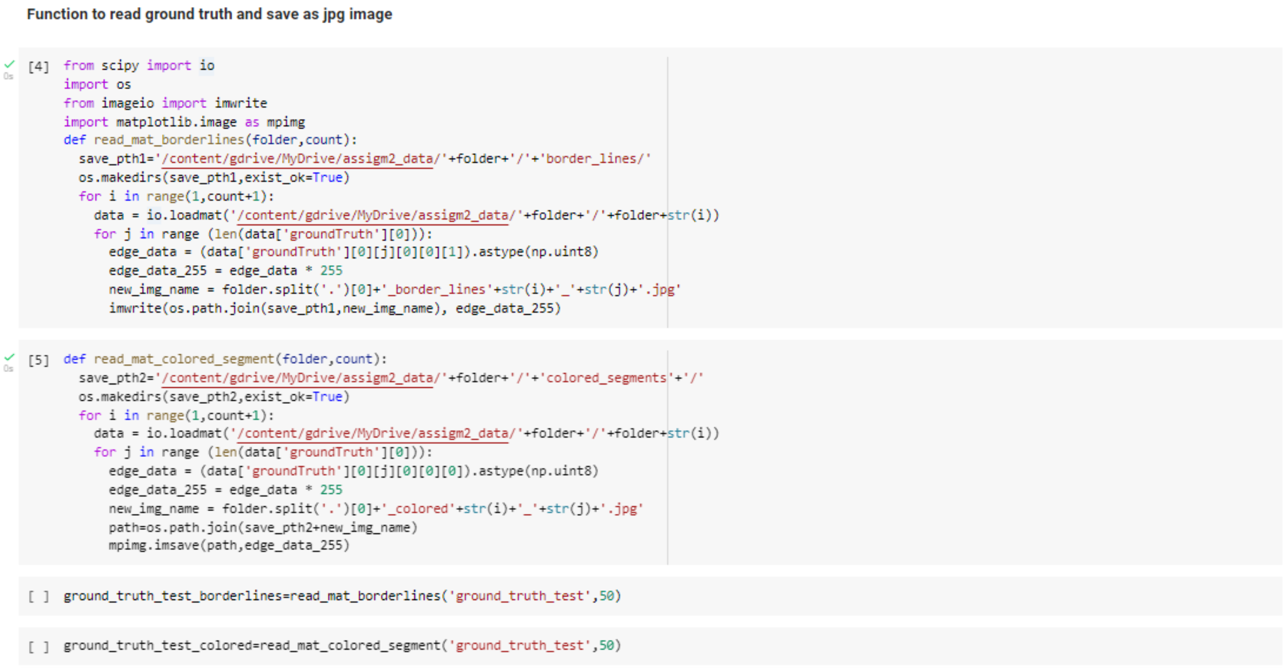
Function for reading images

```
[2] import numpy as np
    import cv2

    def read_images(folder,count):
      data = []
      for i in range(1, count+1):
        img = cv2.imread('/content/gdrive/MyDrive/assigm2_data/'+folder+'/'+folder+str(i)+'.jpg')
        img_col = np.array(img)
        #image will be flatten further more...
        subject = int(i)
        data.append(img_col)
      return np.array(data)
```

```
[3] test=read_images('test',50)
```

Read_mat_boderlines is a function used to load the borderlines of the ground truth from the mat file and save them as a jpg image

Read_mat_colored_segment is a function used to load the colored segments of the ground truth from the mat file and save them as a jpg image



Function to read ground truth and save as jpg image

```
[4] from scipy import io
    import os
    from imageio import imwrite
    import matplotlib.image as mpimg
    def read_mat_borderlines(folder,count):
      save_pth1='/content/gdrive/MyDrive/assigm2_data/'+folder+'/'+'border_lines/'
      os.makedirs(save_pth1,exist_ok=True)
      for i in range(1,count+1):
        data = io.loadmat('/content/gdrive/MyDrive/assigm2_data/'+folder+'/'+folder+str(i))
        for j in range (len(data['groundTruth'][0])):
          edge_data = (data['groundTruth'][0][j][0][0][1]).astype(np.uint8)
          edge_data_255 = edge_data * 255
          new_img_name = folder.split('.')[0]+'_border_lines'+str(i)+'_'+str(j)+'.jpg'
          imwrite(os.path.join(save_pth1,new_img_name), edge_data_255)
```

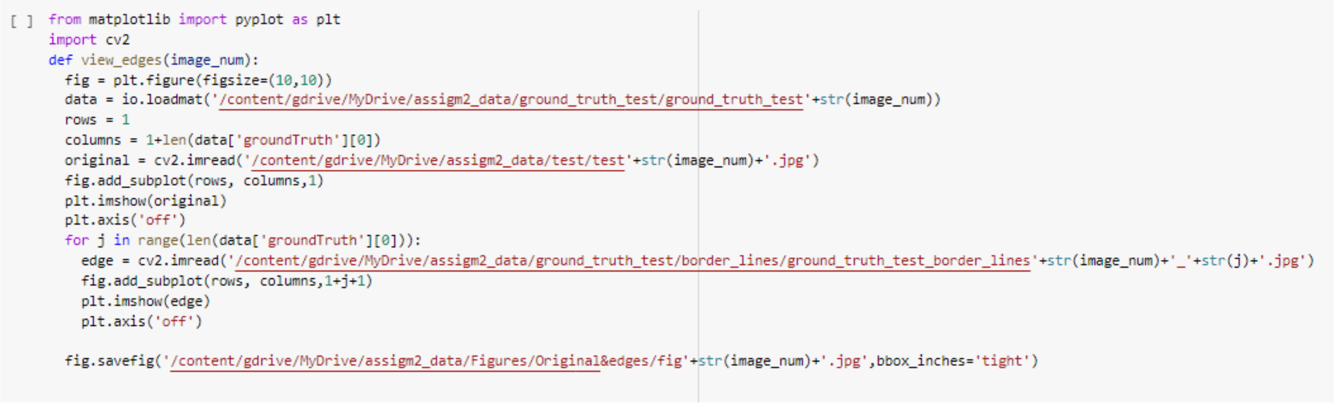
```
[5] def read_mat_colored_segment(folder,count):
      save_pth2='/content/gdrive/MyDrive/assigm2_data/'+folder+'/'+'colored_segments'+'/'
      os.makedirs(save_pth2,exist_ok=True)
      for i in range(1,count+1):
        data = io.loadmat('/content/gdrive/MyDrive/assigm2_data/'+folder+'/'+folder+str(i))
        for j in range (len(data['groundTruth'][0])):
          edge_data = (data['groundTruth'][0][j][0][0][0]).astype(np.uint8)
          edge_data_255 = edge_data * 255
          new_img_name = folder.split('.')[0]+'_colored'+str(i)+'_'+str(j)+'.jpg'
          path=os.path.join(save_pth2+new_img_name)
          mpimg.imsave(path,edge_data_255)
```

```
[ ] ground_truth_test_borderlines=read_mat_borderlines('ground_truth_test',50)
```

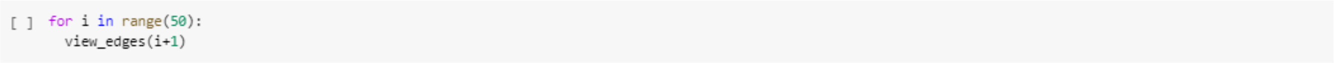
```
[ ] ground_truth_test_colored=read_mat_colored_segment('ground_truth_test',50)
```

2. Visualize the image and the ground truth segmentation

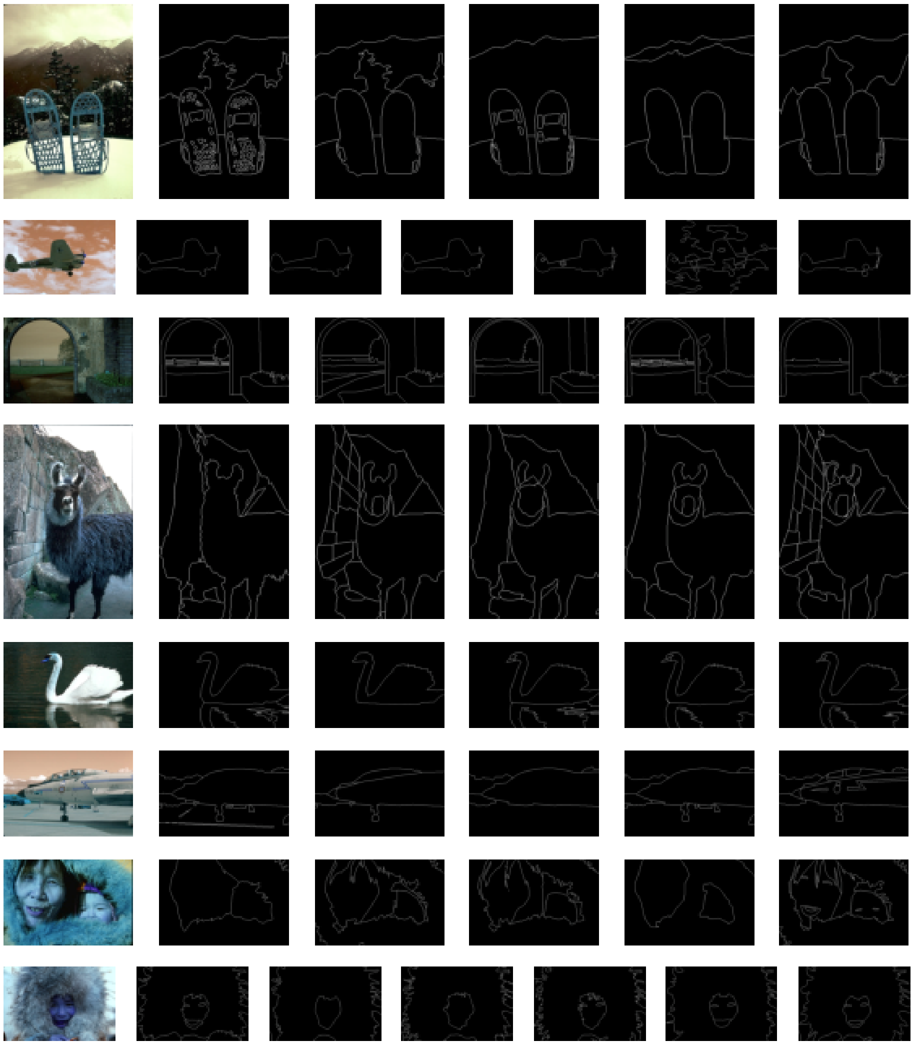
A function used to plot the image with all the corresponding borderlines ground truth and save them in the drive

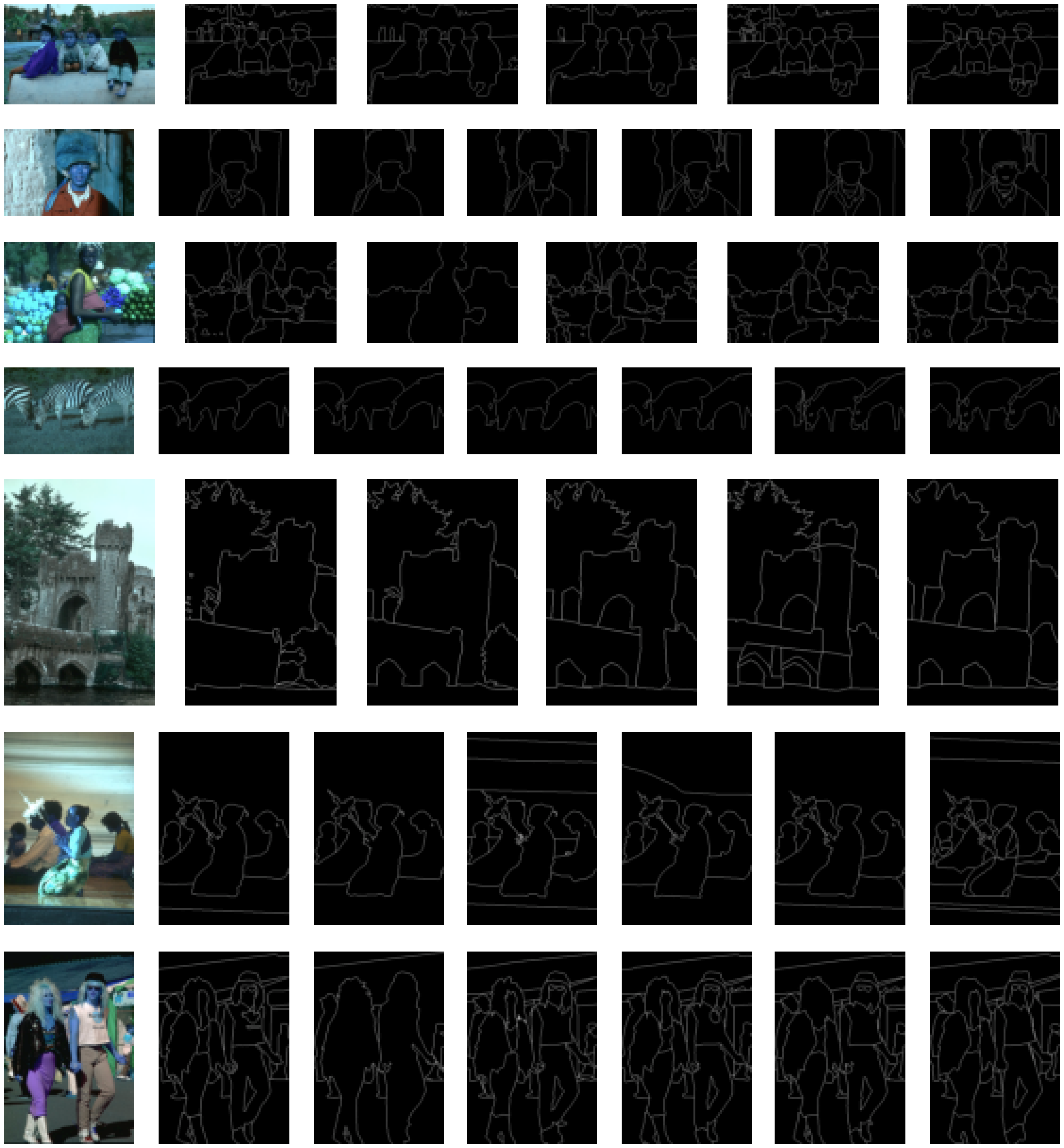


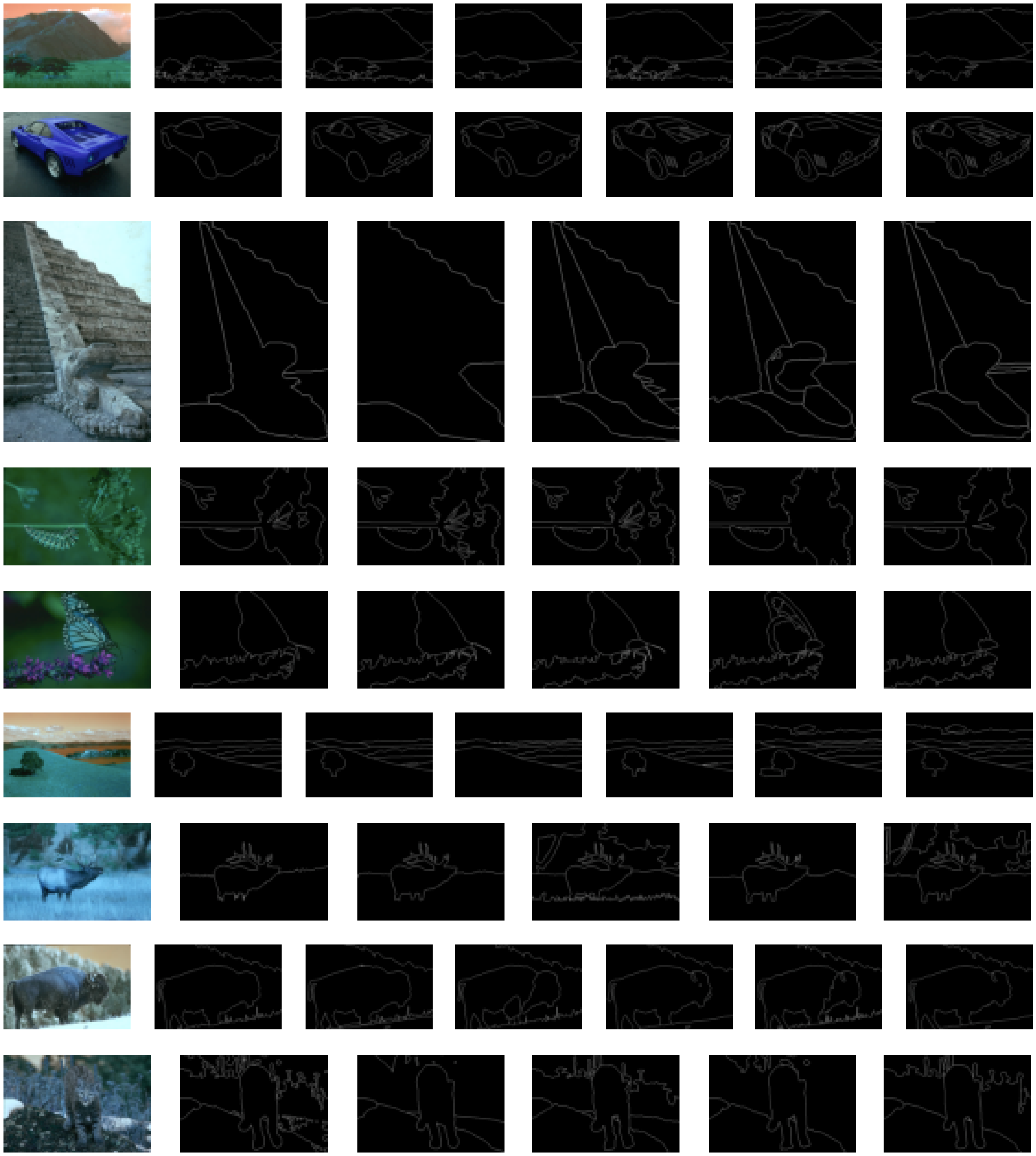
```
[ ] from matplotlib import pyplot as plt
    import cv2
    def view_edges(image_num):
      fig = plt.figure(figsize=(10,10))
      data = io.loadmat('/content/gdrive/MyDrive/assigm2_data/ground_truth_test/ground_truth_test'+str(image_num))
      rows = 1
      columns = 1+len(data['groundTruth'][0])
      original = cv2.imread('/content/gdrive/MyDrive/assigm2_data/test/test'+str(image_num)+'.jpg')
      fig.add_subplot(rows, columns,1)
      plt.imshow(original)
      plt.axis('off')
      for j in range(len(data['groundTruth'][0])):
        edge = cv2.imread('/content/gdrive/MyDrive/assigm2_data/ground_truth_test/border_lines/ground_truth_test_border_lines'+str(image_num)+'_'+str(j)+'.jpg')
        fig.add_subplot(rows, columns,1+j+1)
        plt.imshow(edge)
        plt.axis('off')

      fig.savefig('/content/gdrive/MyDrive/assigm2_data/Figures/Original&edges/fig'+str(image_num)+'.jpg',bbox_inches='tight')
```

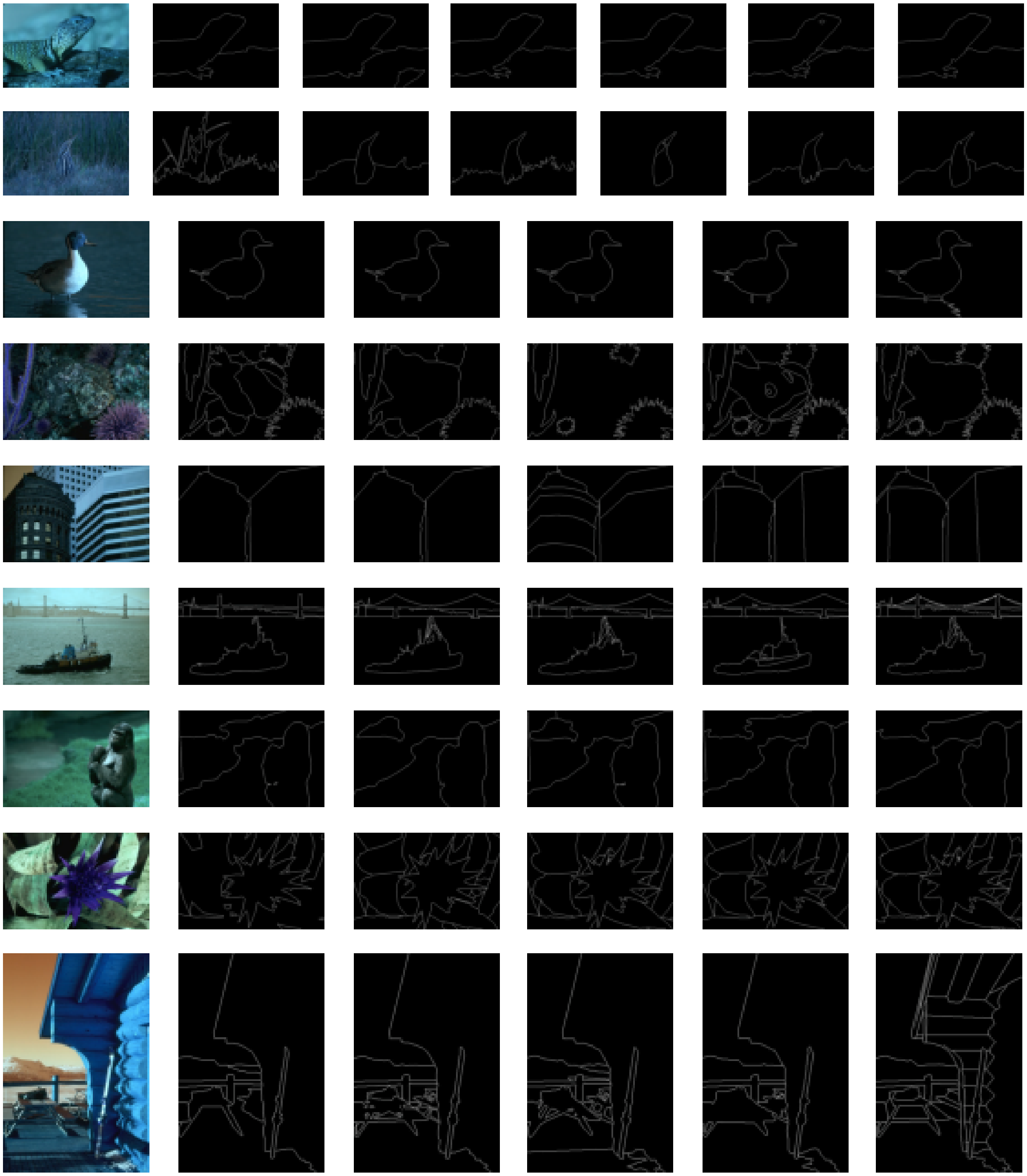


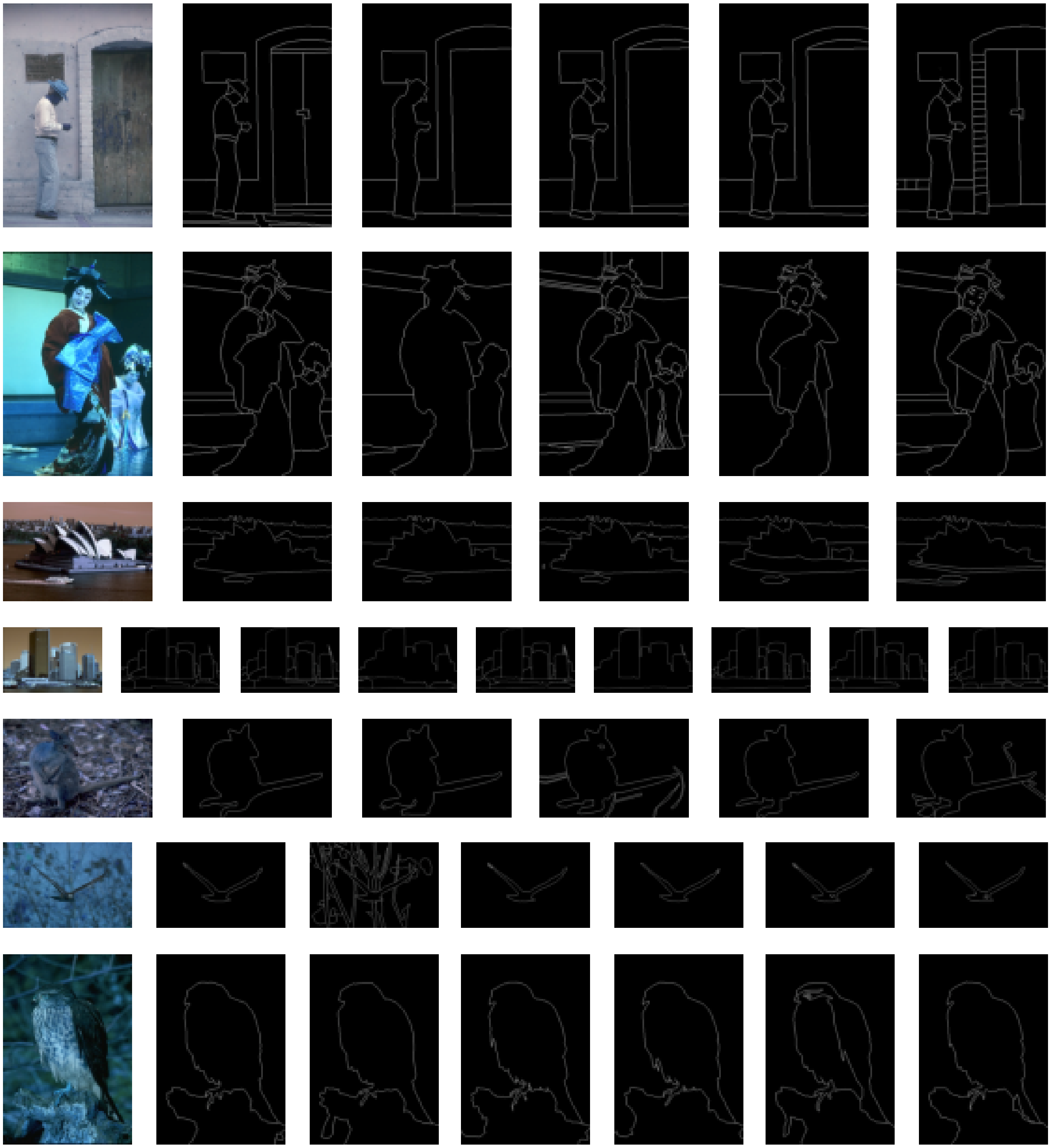
```
[ ] for i in range(50):
      view_edges(i+1)
```

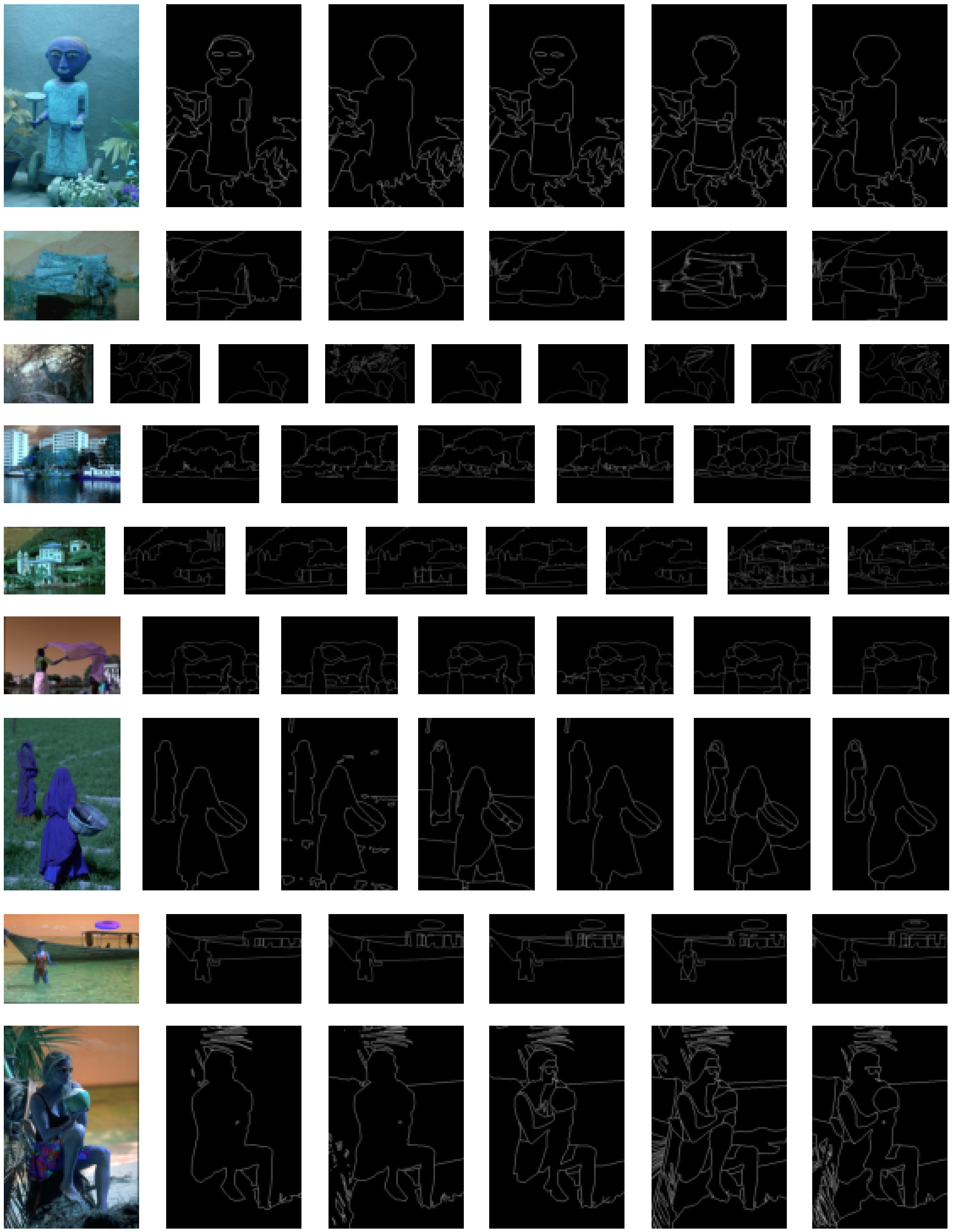


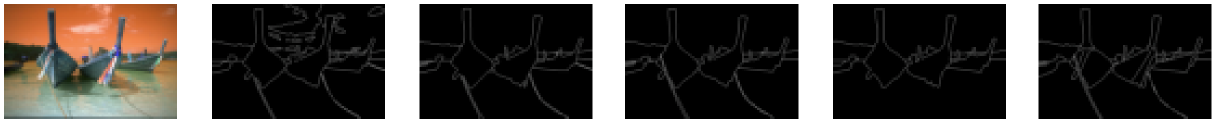










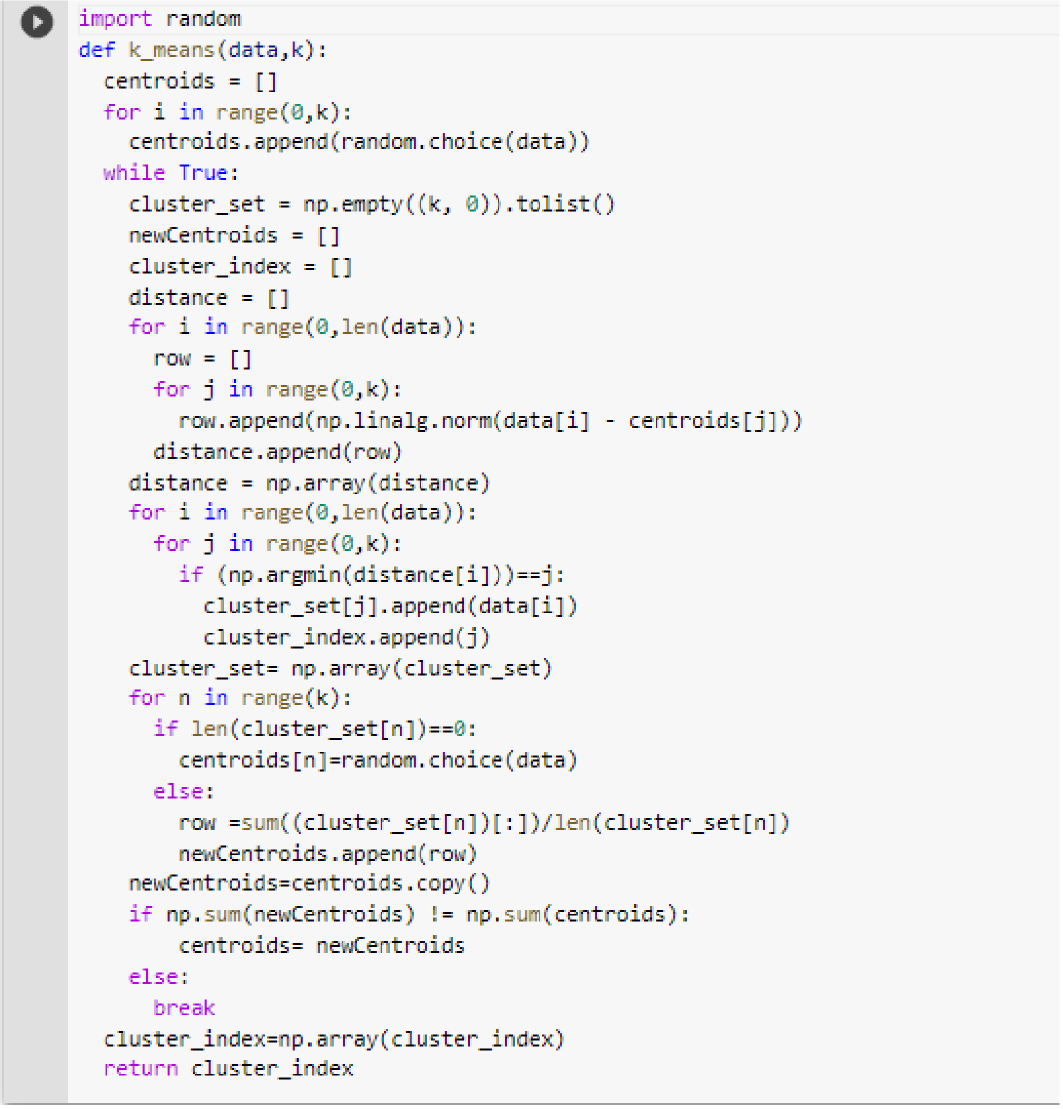


3. Segmentation using K-means

a)

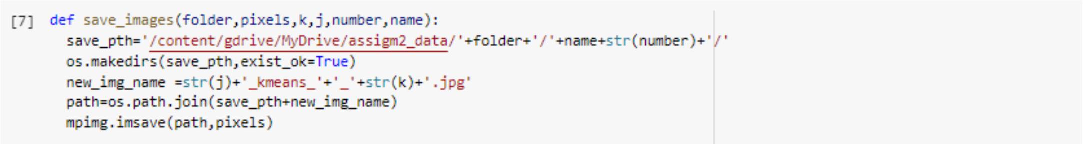
A function used to calculate the k means of given image and return the cluster index for every pixel in the image.

We initialize random centroids at the first run. The function keeps working until it finds that the centroids places remain the same for 2 loops. If we have an empty cluster, we initialize new random centroid for it.



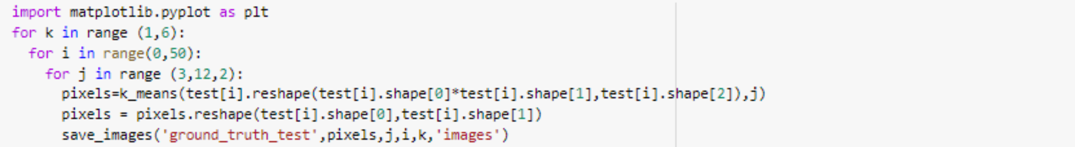
```
import random
def k_means(data,k):
  centroids = []
  for i in range(0,k):
    centroids.append(random.choice(data))
  while True:
    cluster_set = np.empty((k, 0)).tolist()
    newCentroids = []
    cluster_index = []
    distance = []
    for i in range(0,len(data)):
      row = []
      for j in range(0,k):
        row.append(np.linalg.norm(data[i] - centroids[j]))
      distance.append(row)
    distance = np.array(distance)
    for i in range(0,len(data)):
      for j in range(0,k):
        if (np.argmin(distance[i]))==j:
          cluster_set[j].append(data[i])
          cluster_index.append(j)
    cluster_set= np.array(cluster_set)
    for n in range(k):
      if len(cluster_set[n])==0:
        centroids[n]=random.choice(data)
      else:
        row =sum((cluster_set[n])[:])/len(cluster_set[n])
        newCentroids.append(row)
    newCentroids=centroids.copy()
    if np.sum(newCentroids) != np.sum(centroids):
        centroids= newCentroids
    else:
      break
  cluster_index=np.array(cluster_index)
  return cluster_index
```

A function used to save the images resulting from the k_means function in the drive to use them later



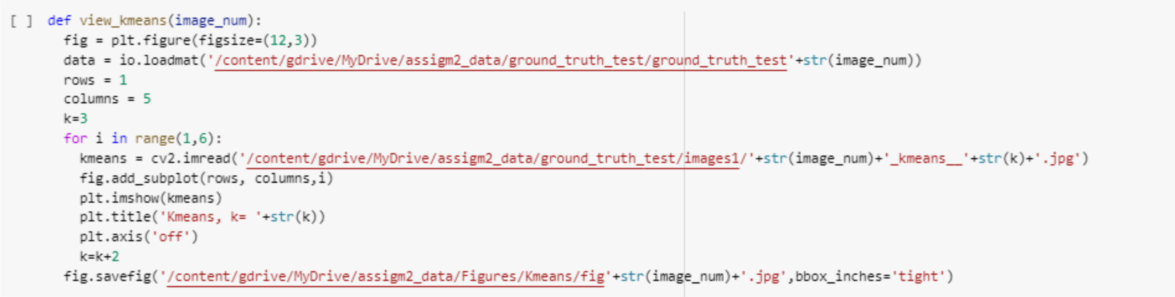
```
[7]  def save_images(folder,pixels,k,j,number,name):
       save_pth='/content/gdrive/MyDrive/assigm2_data/'+folder+'/'+name+str(number)+'/'
       os.makedirs(save_pth,exist_ok=True)
       new_img_name =str(j)+'_kmeans_'+'_'+str(k)+'.jpg'
       path=os.path.join(save_pth+new_img_name)
       mpimg.imsave(path,pixels)
```

We make the k means 5 times (M=5) to use them for calculating the average f measure and conditional entropy for every image

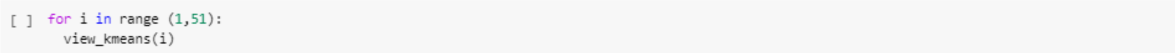


```
import matplotlib.pyplot as plt
for k in range (1,6):
  for i in range(0,50):
    for j in range (3,12,2):
      pixels=k_means(test[i].reshape(test[i].shape[0]*test[i].shape[1],test[i].shape[2]),j)
      pixels = pixels.reshape(test[i].shape[0],test[i].shape[1])
      save_images('ground_truth_test',pixels,j,i,k,'images')
```

A function used to plot the 5 k means (3-5-7-9-11) for every image

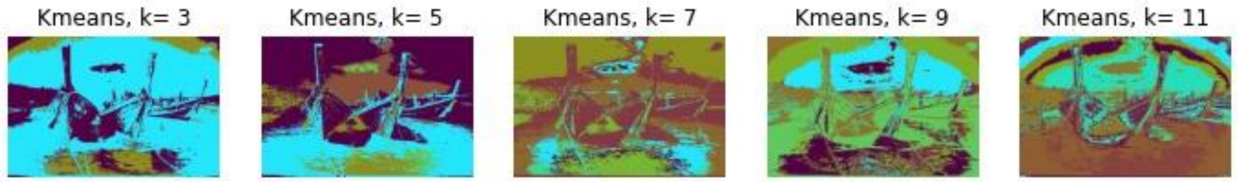


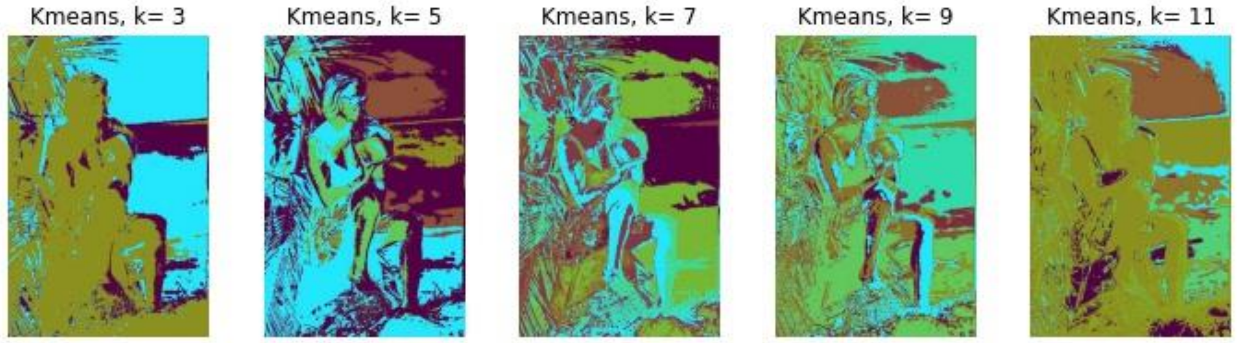
```
[ ]  def view_kmeans(image_num):
       fig = plt.figure(figsize=(12,3))
       data = io.loadmat('/content/gdrive/MyDrive/assigm2_data/ground_truth_test/ground_truth_test'+str(image_num))
       rows = 1
       columns = 5
       k=3
       for i in range(1,6):
         kmeans = cv2.imread('/content/gdrive/MyDrive/assigm2_data/ground_truth_test/images1/'+str(image_num)+'_kmeans__'+str(k)+'.jpg')
         fig.add_subplot(rows, columns,i)
         plt.imshow(kmeans)
         plt.title('Kmeans, k= '+str(k))
         plt.axis('off')
         k=k+2
       fig.savefig('/content/gdrive/MyDrive/assigm2_data/Figures/Kmeans/fig'+str(image_num)+'.jpg',bbox_inches='tight')
```

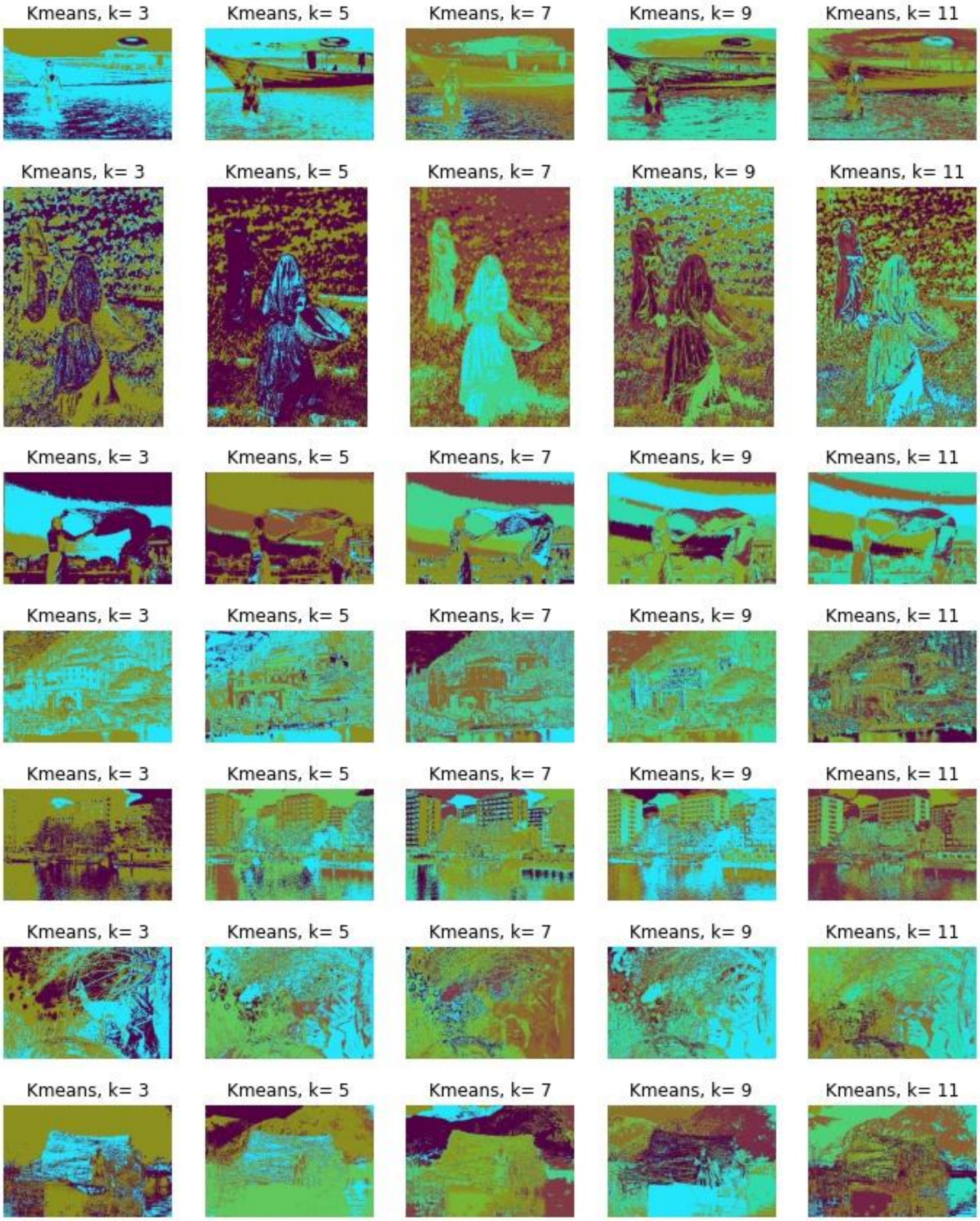


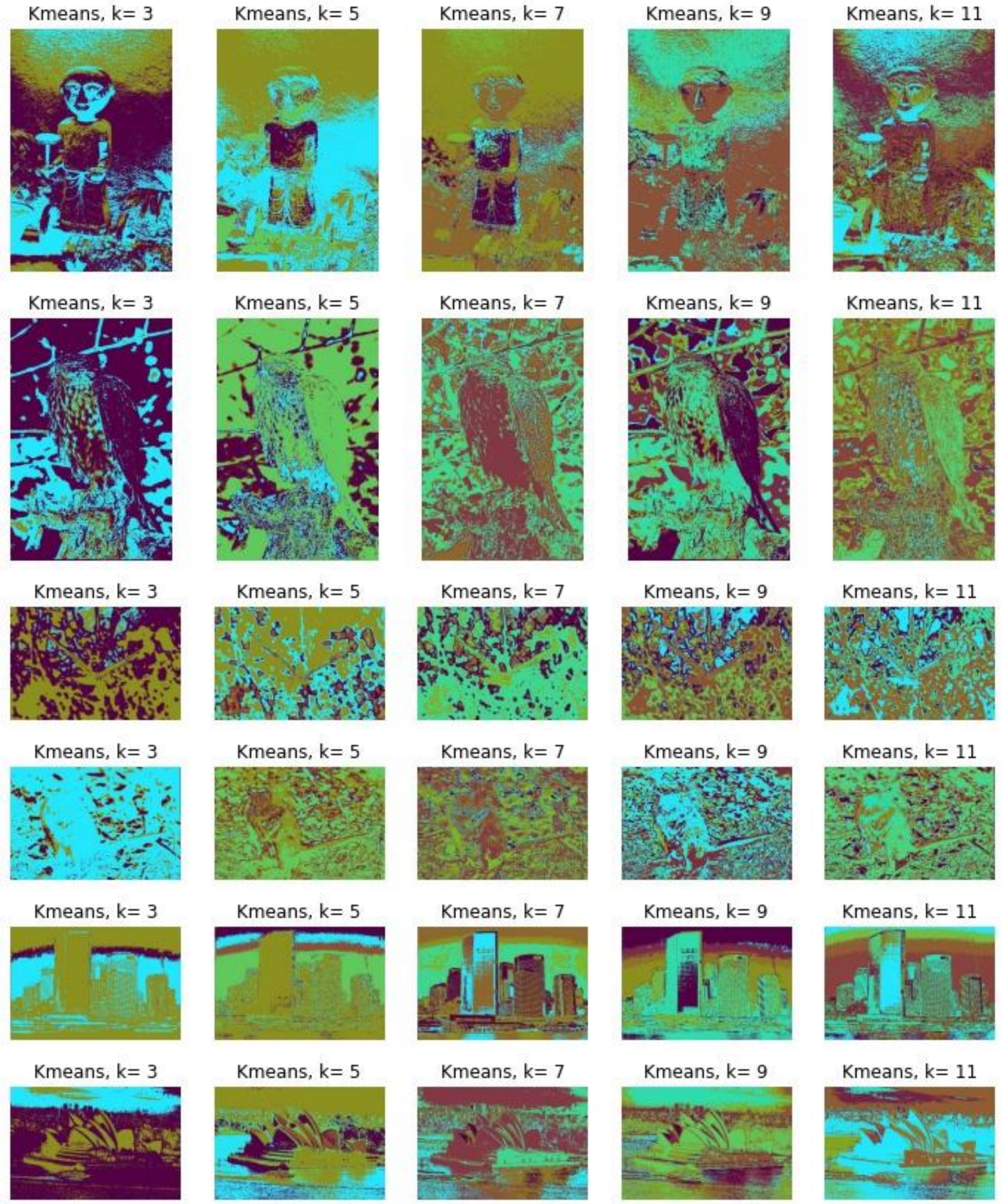
```
[ ]  for i in range (1,51):
       view_kmeans(i)
```

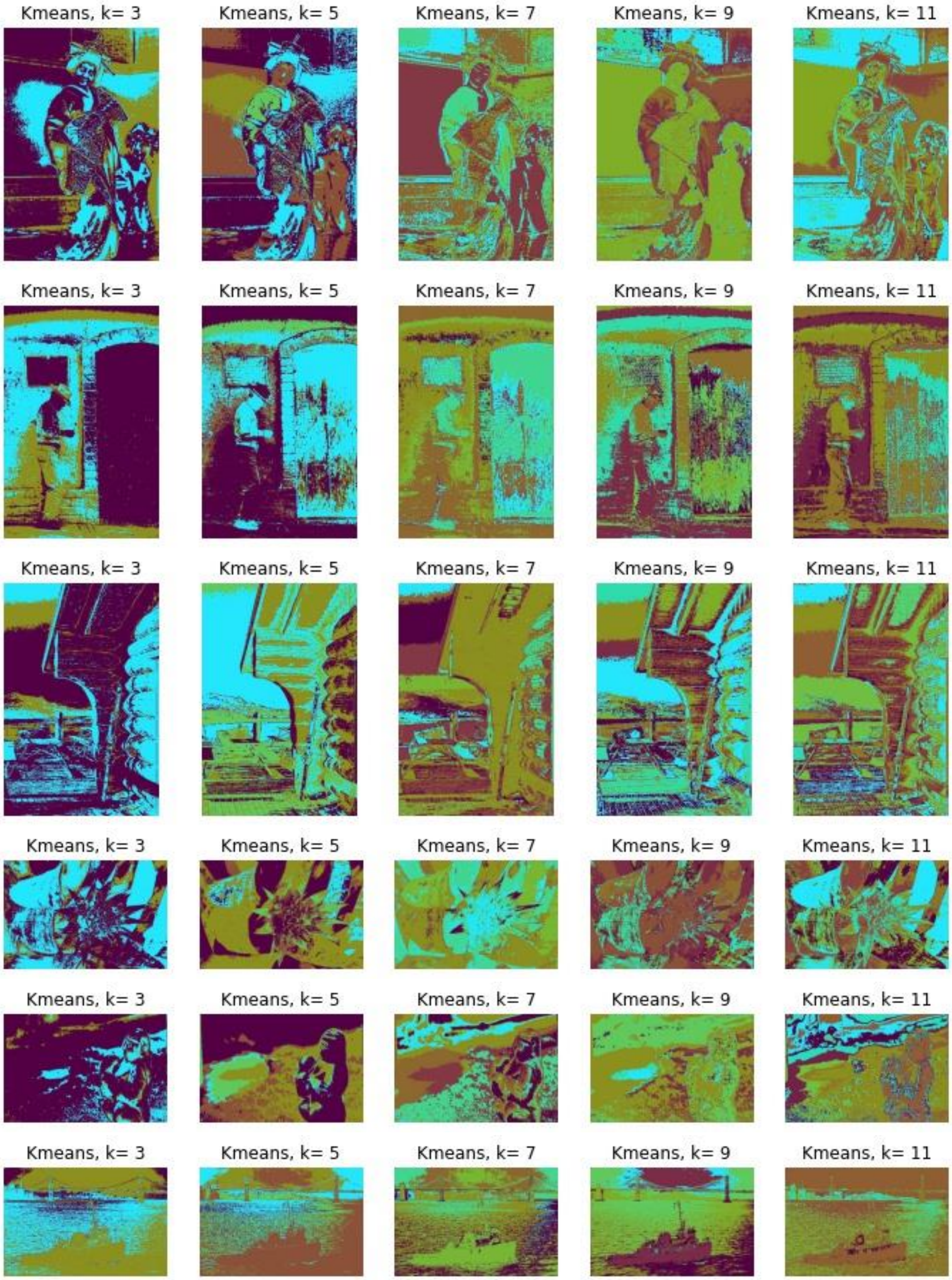
K-means for every image

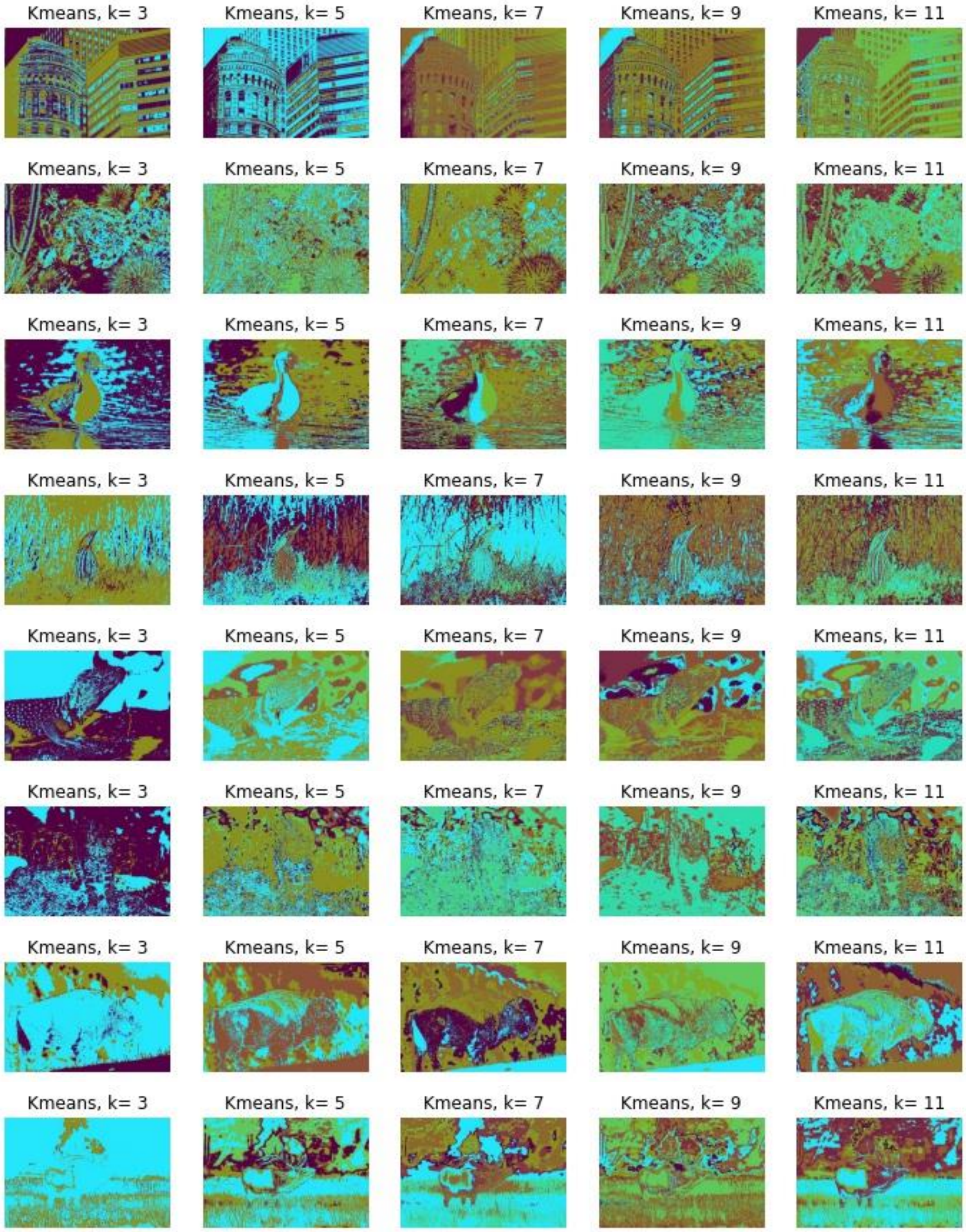


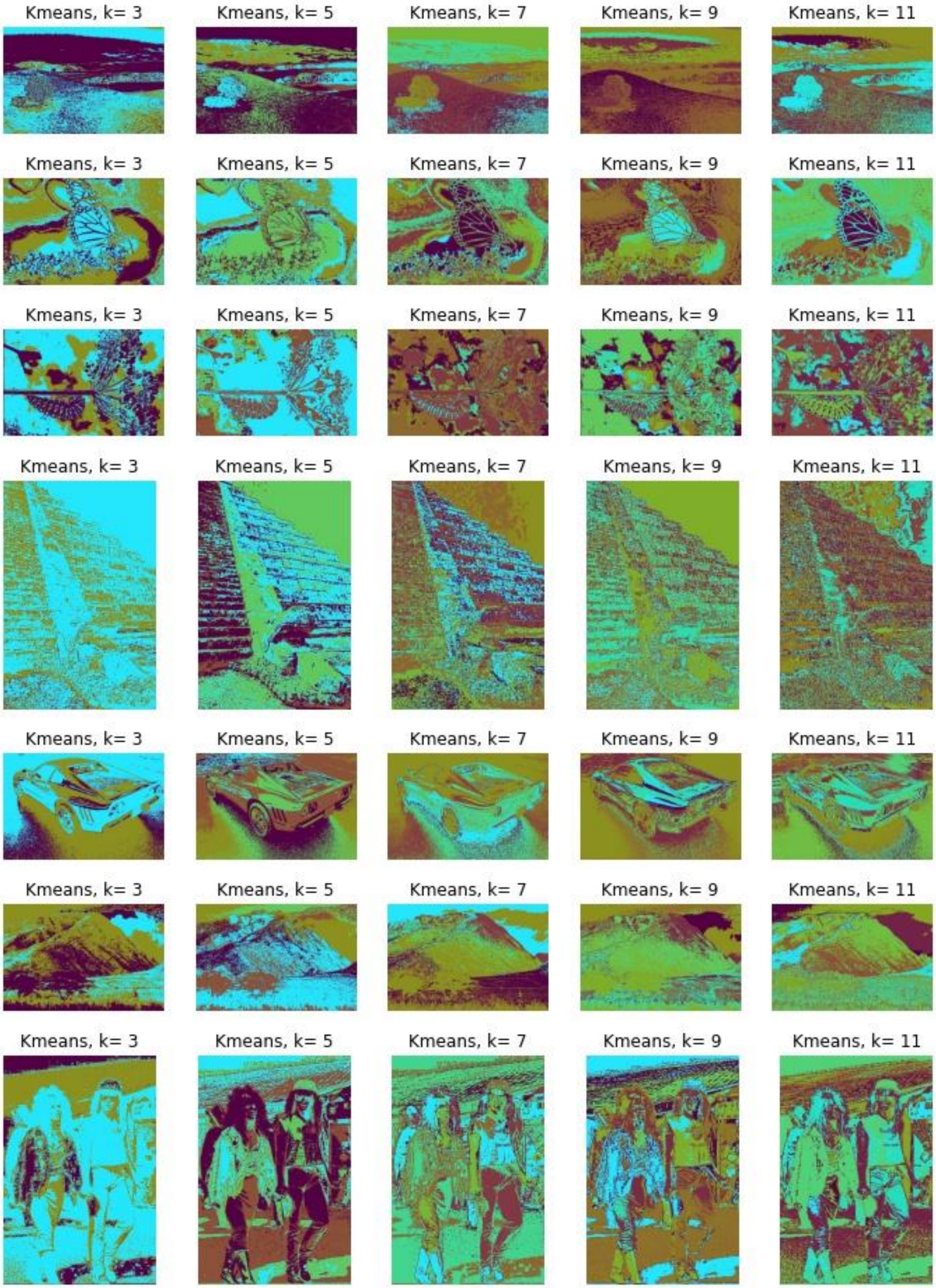



Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11


Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11


Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11


Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11



Kmeans, k= 3 Kmeans, k= 5 Kmeans, k= 7 Kmeans, k= 9 Kmeans, k= 11

Kmeans, k= 3 Kmeans, k= 5 Kmeans, k= 7 Kmeans, k= 9 Kmeans, k= 11

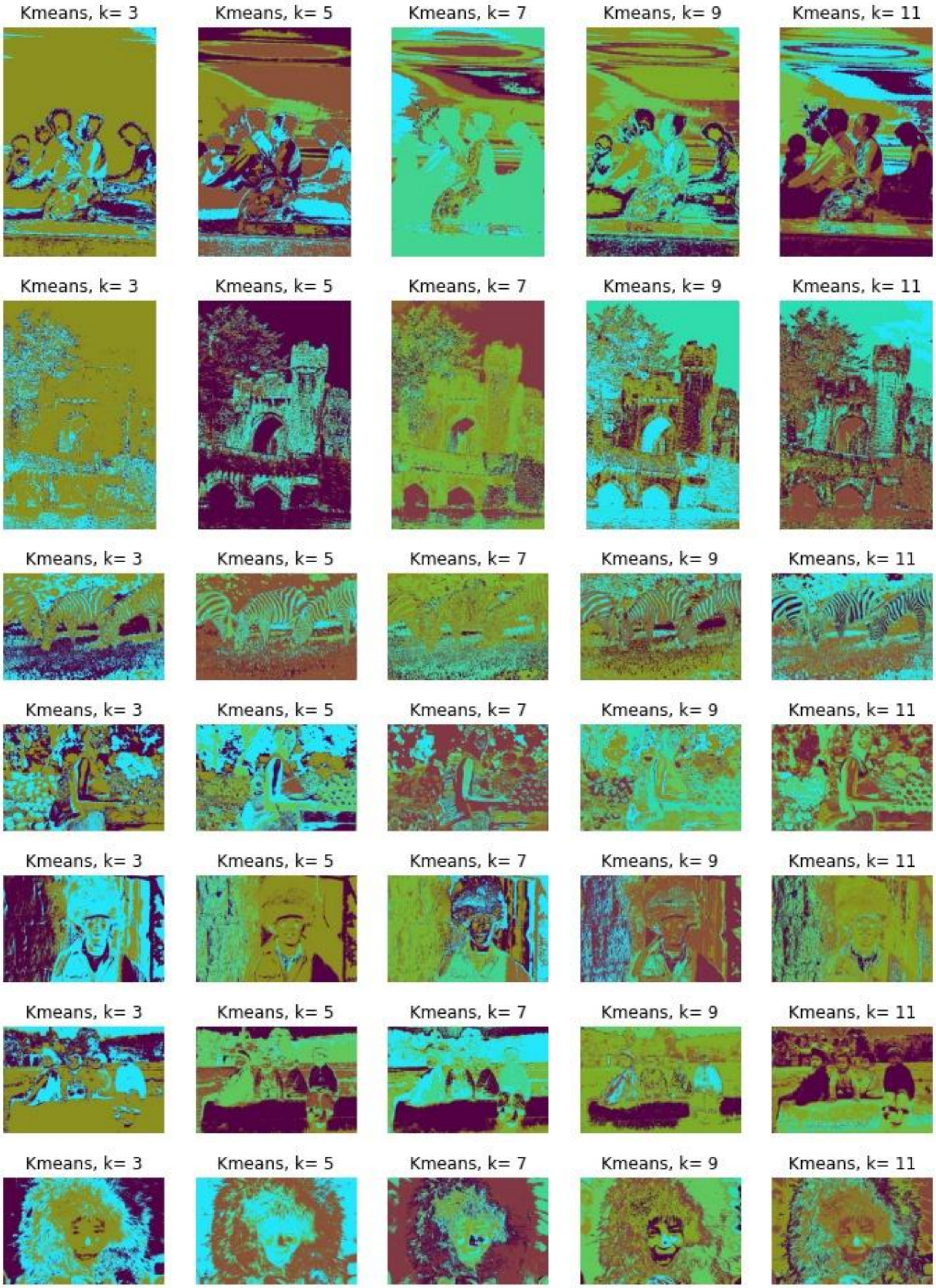
Kmeans, k= 3 Kmeans, k= 5 Kmeans, k= 7 Kmeans, k= 9 Kmeans, k= 11

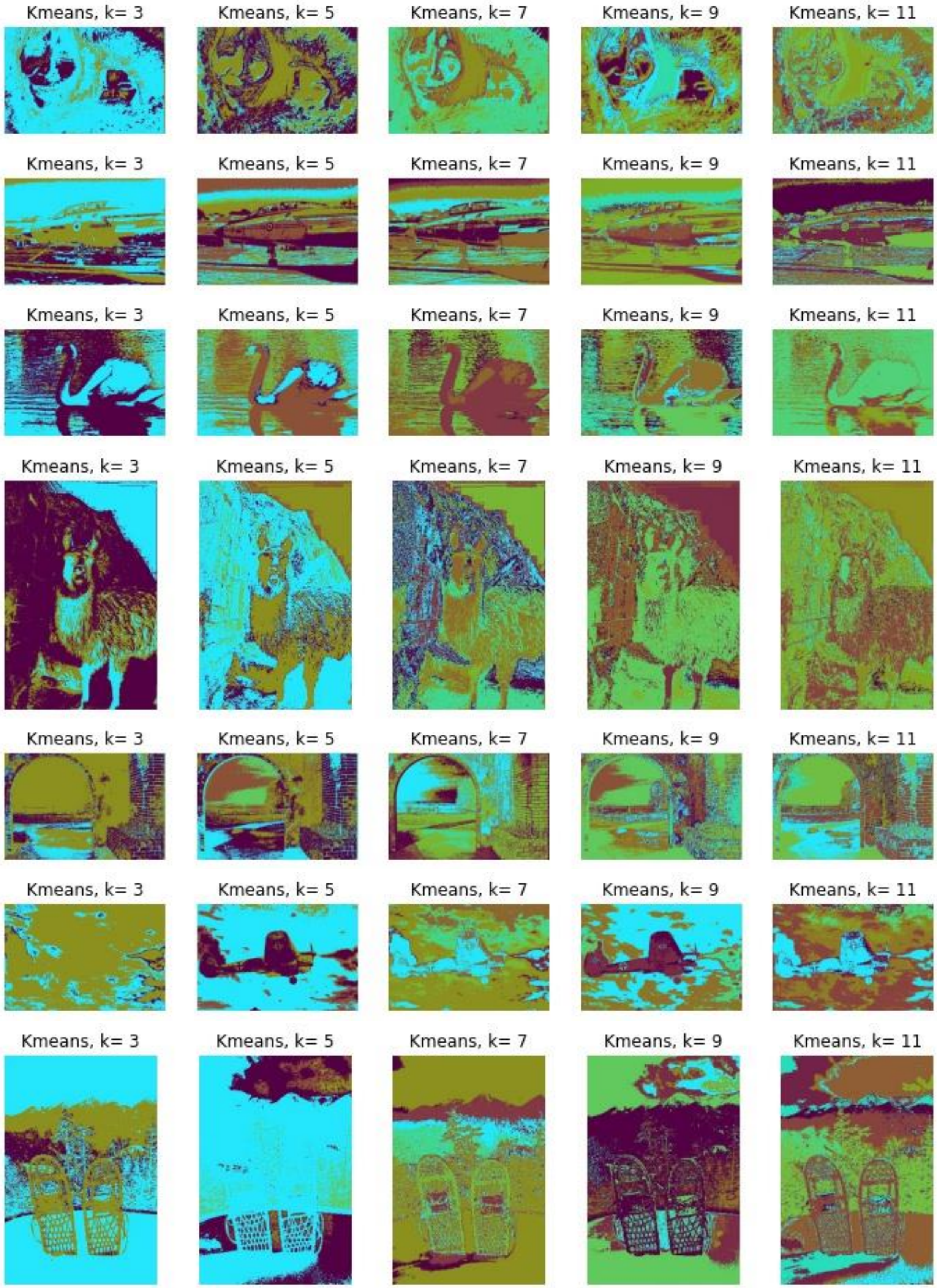
Kmeans, k= 3 Kmeans, k= 5 Kmeans, k= 7 Kmeans, k= 9 Kmeans, k= 11

Kmeans, k= 3 Kmeans, k= 5 Kmeans, k= 7 Kmeans, k= 9 Kmeans, k= 11

Kmeans, k= 3 Kmeans, k= 5 Kmeans, k= 7 Kmeans, k= 9 Kmeans, k= 11

Kmeans, k= 3 Kmeans, k= 5 Kmeans, k= 7 Kmeans, k= 9 Kmeans, k= 11


Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11



Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11

Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11

Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11

Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11

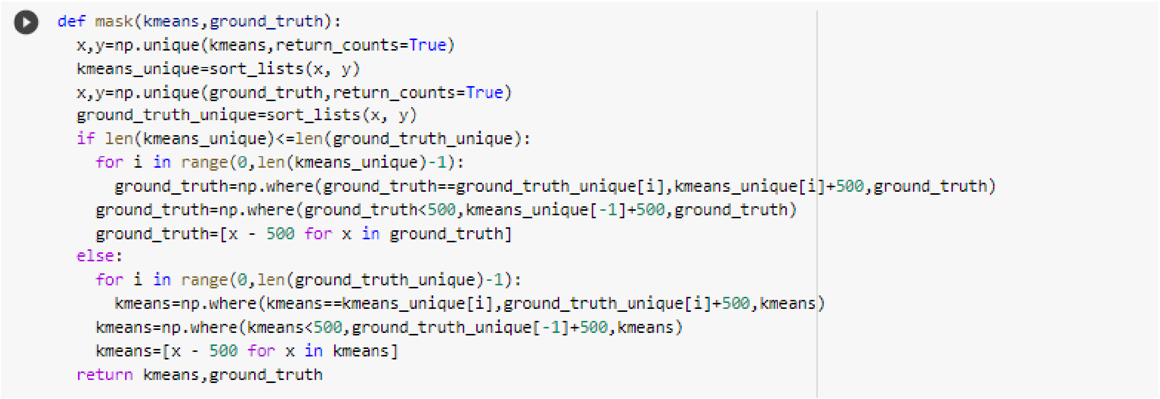
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11

Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11

Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11

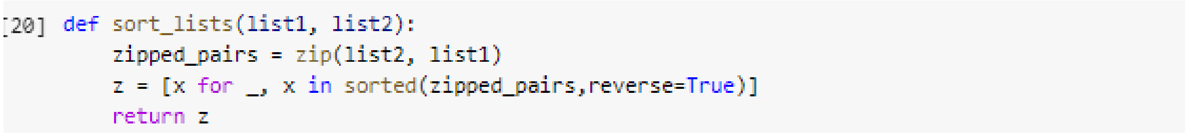
b)

A function used to make masking for the k means clusters and ground truth clusters we collect all the extra clusters we have in the last cluster so both k means and ground truth have the same number of clusters



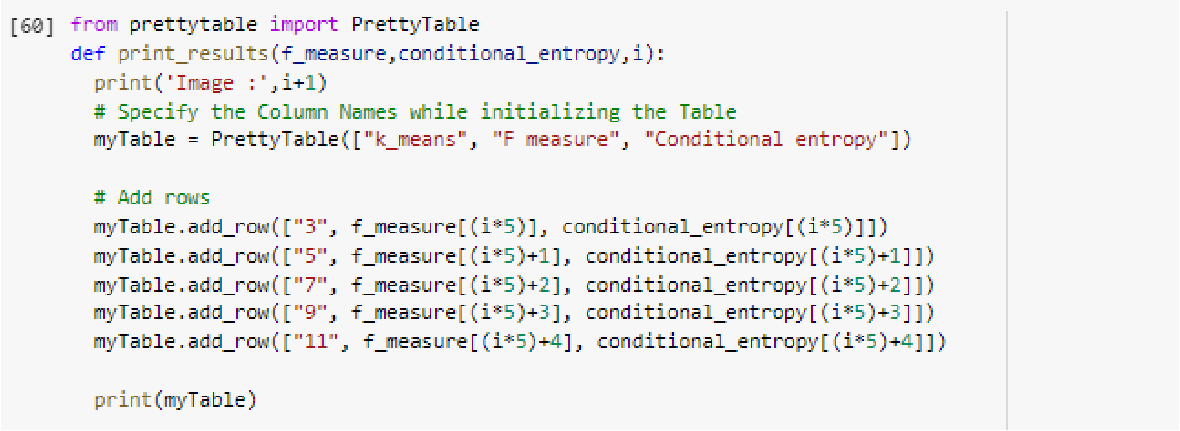
```
def mask(kmeans,ground_truth):
  x,y=np.unique(kmeans,return_counts=True)
  kmeans_unique=sort_lists(x, y)
  x,y=np.unique(ground_truth,return_counts=True)
  ground_truth_unique=sort_lists(x, y)
  if len(kmeans_unique)<=len(ground_truth_unique):
    for i in range(0,len(kmeans_unique)-1):
      ground_truth=np.where(ground_truth==ground_truth_unique[i],kmeans_unique[i]+500,ground_truth)
    ground_truth=np.where(ground_truth<500,kmeans_unique[-1]+500,ground_truth)
    ground_truth=[x - 500 for x in ground_truth]
  else:
    for i in range(0,len(ground_truth_unique)-1):
      kmeans=np.where(kmeans==kmeans_unique[i],ground_truth_unique[i]+500,kmeans)
    kmeans=np.where(kmeans<500,ground_truth_unique[-1]+500,kmeans)
    kmeans=[x - 500 for x in kmeans]
  return kmeans,ground_truth
```

A function used to sort 1 list according to the elements of the second list we used this function in the masking function



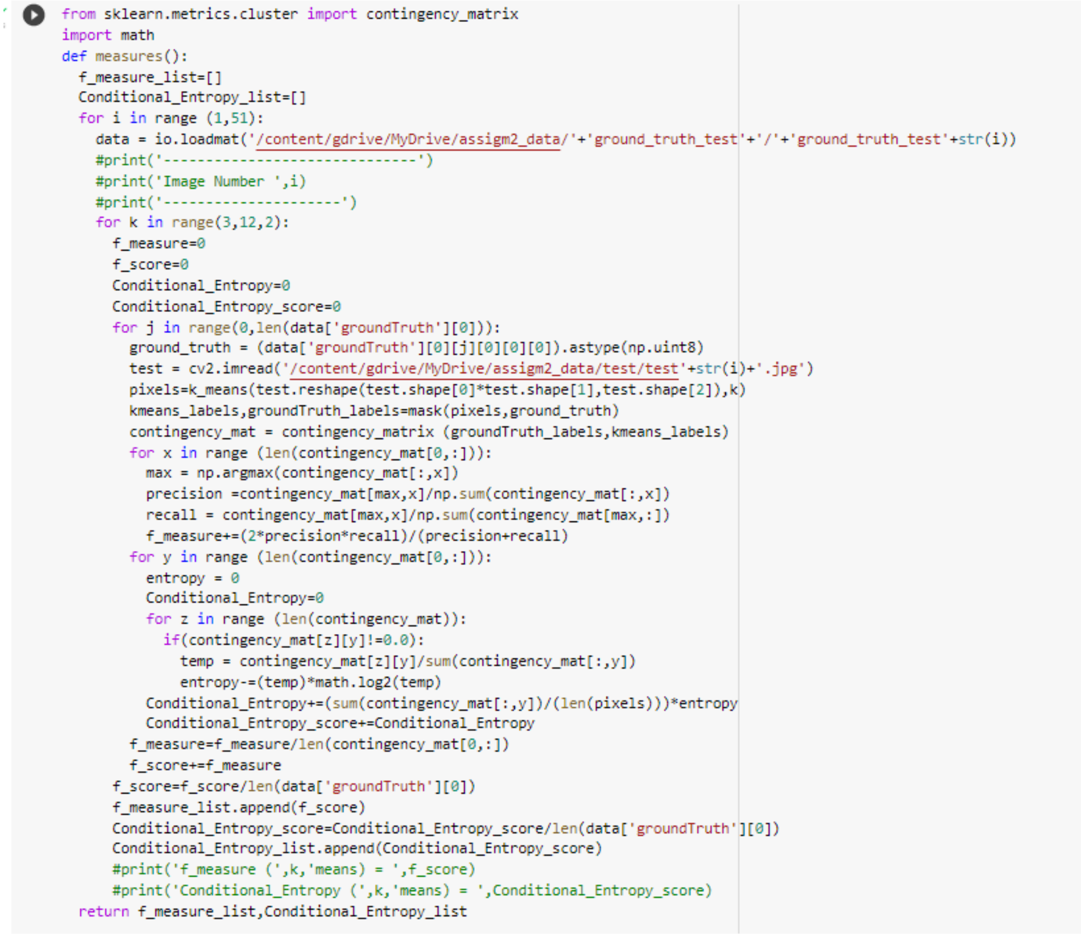
```
[20] def sort_lists(list1, list2):
       zipped_pairs = zip(list2, list1)
       z = [x for _, x in sorted(zipped_pairs,reverse=True)]
       return z
```

A function used to print the measure results in a good table view



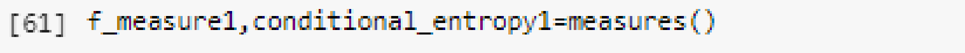
```
[60] from prettytable import PrettyTable
     def print_results(f_measure,conditional_entropy,i):
       print('Image :',i+1)
       # Specify the Column Names while initializing the Table
       myTable = PrettyTable(["k_means", "F measure", "Conditional entropy"])

       # Add rows
       myTable.add_row(["3", f_measure[(i*5)], conditional_entropy[(i*5)]])
       myTable.add_row(["5", f_measure[(i*5)+1], conditional_entropy[(i*5)+1]])
       myTable.add_row(["7", f_measure[(i*5)+2], conditional_entropy[(i*5)+2]])
       myTable.add_row(["9", f_measure[(i*5)+3], conditional_entropy[(i*5)+3]])
       myTable.add_row(["11", f_measure[(i*5)+4], conditional_entropy[(i*5)+4]])

       print(myTable)
```

A function used to calculate the f measure and conditional entropy measure for every image using the contingency matrix

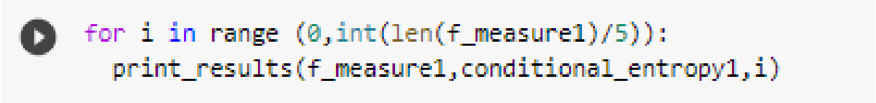


```
from sklearn.metrics.cluster import contingency_matrix
import math
def measures():
  f_measure_list=[]
  Conditional_Entropy_list=[]
  for i in range (1,51):
    data = io.loadmat('/content/gdrive/MyDrive/assigm2_data/'+'ground_truth_test'+'/'+'ground_truth_test'+str(i))
    #print('------------------------------')
    #print('Image Number ',i)
    #print('---------------------')
    for k in range(3,12,2):
      f_measure=0
      f_score=0
      Conditional_Entropy=0
      Conditional_Entropy_score=0
      for j in range(0,len(data['groundTruth'][0])):
        ground_truth = (data['groundTruth'][0][j][0][0][0]).astype(np.uint8)
        test = cv2.imread('/content/gdrive/MyDrive/assigm2_data/test/test'+str(i)+'.jpg')
        pixels=k_means(test.reshape(test.shape[0]*test.shape[1],test.shape[2]),k)
        kmeans_labels,groundTruth_labels=mask(pixels,ground_truth)
        contingency_mat = contingency_matrix (groundTruth_labels,kmeans_labels)
        for x in range (len(contingency_mat[0,:])):
          max = np.argmax(contingency_mat[:,x])
          precision =contingency_mat[max,x]/np.sum(contingency_mat[:,x])
          recall = contingency_mat[max,x]/np.sum(contingency_mat[max,:])
          f_measure+=(2*precision*recall)/(precision+recall)
        for y in range (len(contingency_mat[0,:])):
          entropy = 0
          Conditional_Entropy=0
          for z in range (len(contingency_mat)):
            if(contingency_mat[z][y]!=0.0):
              temp = contingency_mat[z][y]/sum(contingency_mat[:,y])
              entropy-=(temp)*math.log2(temp)
          Conditional_Entropy+=(sum(contingency_mat[:,y])/(len(pixels)))*entropy
          Conditional_Entropy_score+=Conditional_Entropy
        f_measure=f_measure/len(contingency_mat[0,:])
        f_score+=f_measure
      f_score=f_score/len(data['groundTruth'][0])
      f_measure_list.append(f_score)
      Conditional_Entropy_score=Conditional_Entropy_score/len(data['groundTruth'][0])
      Conditional_Entropy_list.append(Conditional_Entropy_score)
      #print('f_measure (',k,'means) = ',f_score)
      #print('Conditional_Entropy (',k,'means) = ',Conditional_Entropy_score)
  return f_measure_list,Conditional_Entropy_list
```

We calculate the f measures and conditional entropy and print them for all the images

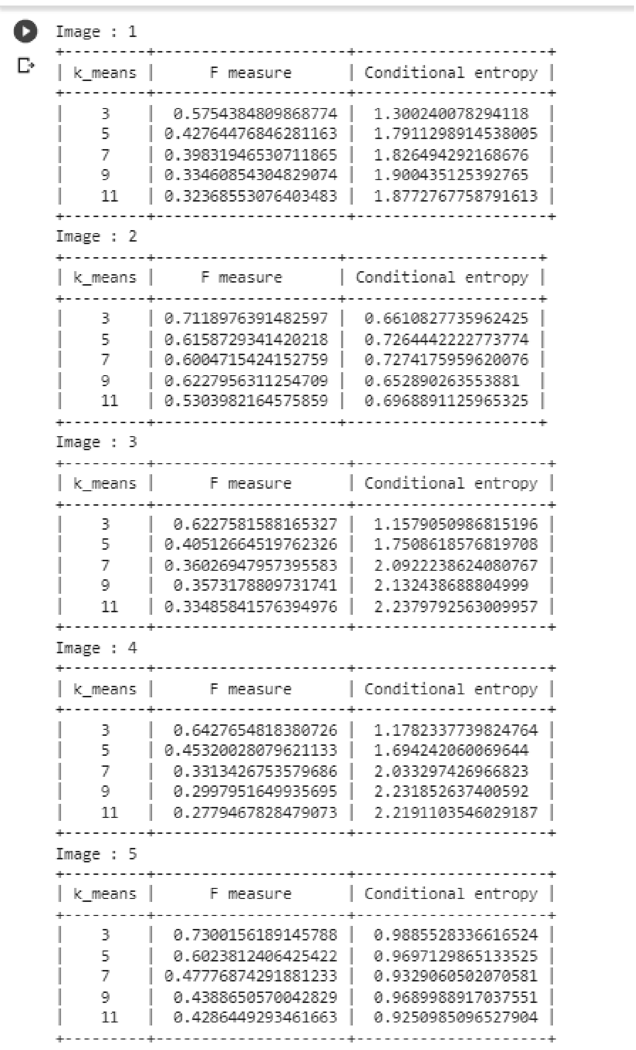


```
[61] f_measure1,conditional_entropy1=measures()
```

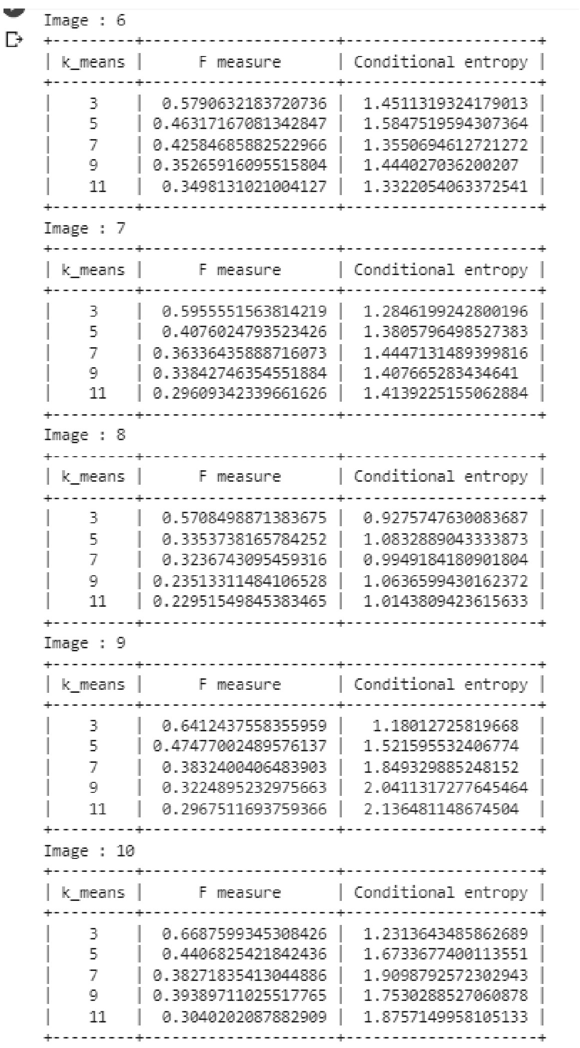


```
for i in range (0,int(len(f_measure1)/5)):
  print_results(f_measure1,conditional_entropy1,i)
```

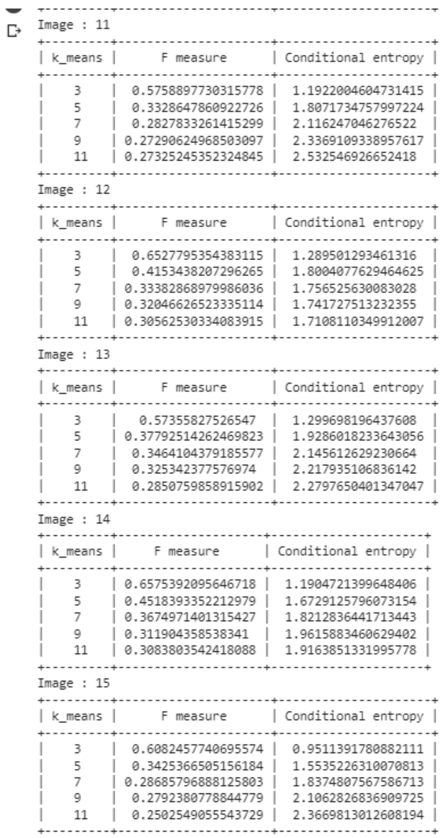
f measure and conditional entropy for every image



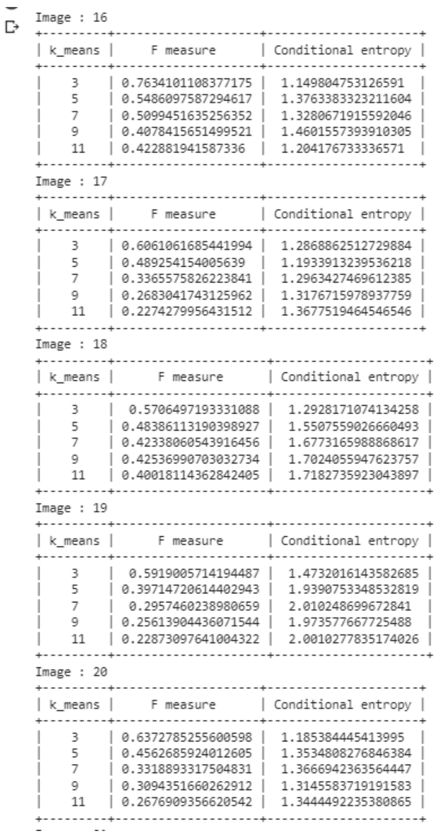
```
Image : 1
+----------+--------------------+---------------------+
| k_means  |     F measure      | Conditional entropy |
+----------+--------------------+---------------------+
|    3     | 0.5754384809868774 |  1.300240078294118  |
|    5     | 0.4276447684628116 3| 1.7911298914538005 |
|    7     | 0.39831946530711865 | 1.826494292168676  |
|    9     | 0.33460854304829074 | 1.900435125392765  |
|    11    | 0.32368553076403483 | 1.8772767758791613 |
+----------+--------------------+---------------------+
Image : 2
+----------+--------------------+---------------------+
| k_means  |     F measure      | Conditional entropy |
+----------+--------------------+---------------------+
|    3     | 0.7118976391482597 | 0.6610827735962425  |
|    5     | 0.6158729341420218 | 0.7264442222773774  |
|    7     | 0.6004715424152759 | 0.7274175959620076  |
|    9     | 0.6227956311254709 | 0.652890263553881   |
|    11    | 0.5303982164575859 | 0.6968891125965325  |
+----------+--------------------+---------------------+
Image : 3
+----------+---------------------+---------------------+
| k_means  |      F measure      | Conditional entropy |
+----------+---------------------+---------------------+
|    3     |  0.6227581588165327 |  1.1579050986815196 |
|    5     | 0.40512664519762326 |  1.7508618576819708 |
|    7     | 0.36026947957395583 |  2.0922238624080767 |
|    9     |  0.3573178809731741 |  2.132438688804999  |
|    11    | 0.33485841576394976 |  2.2379792563009957 |
+----------+---------------------+---------------------+
Image : 4
+----------+---------------------+---------------------+
| k_means  |      F measure      | Conditional entropy |
+----------+---------------------+---------------------+
|    3     |  0.6427654818380726 |  1.1782337739824764 |
|    5     | 0.45320028079621133 |  1.69424206006964 4 |
|    7     |  0.3313426753579686 |  2.033297426966823  |
|    9     |  0.2997951649935695 |  2.231852637400592  |
|    11    |  0.2779467828479073 |  2.2191103546029187 |
+----------+---------------------+---------------------+
Image : 5
+----------+---------------------+---------------------+
| k_means  |      F measure      | Conditional entropy |
+----------+---------------------+---------------------+
|    3     |  0.7300156189145788 |  0.9885528336616524 |
|    5     |  0.6023812406425422 |  0.9697129865133525 |
|    7     | 0.47776874291881233 |  0.9329060502070581 |
|    9     |  0.4388650570042829 |  0.9689988917037551 |
|    11    |  0.4286449293461663 |  0.9250985096527904 |
+----------+---------------------+---------------------+
```



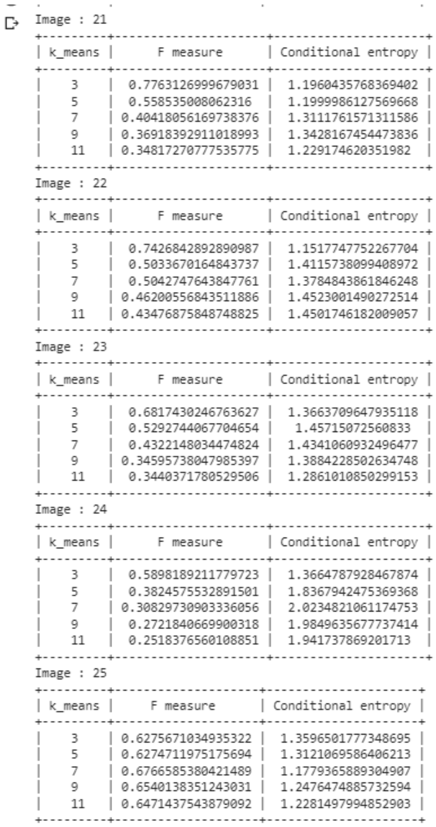
```
Image : 6
+----------+---------------------+---------------------+
| k_means  |      F measure      | Conditional entropy |
+----------+---------------------+---------------------+
|    3     |  0.5790632183720736 |  1.4511319324179013 |
|    5     | 0.46317167081342847 |  1.5847519594307364 |
|    7     | 0.42584685882522966 |  1.3550694612721272 |
|    9     | 0.35265916095515804 |  1.444027036200207  |
|    11    |  0.3498131021004127 |  1.3322054063372541 |
+----------+---------------------+---------------------+
Image : 7
+----------+---------------------+---------------------+
| k_means  |      F measure      | Conditional entropy |
+----------+---------------------+---------------------+
|    3     |  0.5955551563814219 |  1.2846199242800196 |
|    5     |  0.4076024793523426 |  1.3805796498527383 |
|    7     | 0.36336435888716073 |  1.4447131489399816 |
|    9     | 0.33842746354551884 |  1.407665283434641  |
|    11    | 0.29609342339661626 |  1.4139225155062884 |
+----------+---------------------+---------------------+
Image : 8
+----------+---------------------+---------------------+
| k_means  |      F measure      | Conditional entropy |
+----------+---------------------+---------------------+
|    3     |  0.5708498871383675 |  0.9275747630083687 |
|    5     |  0.3353738165784252 |  1.0832889043333873 |
|    7     |  0.3236743095459316 |  0.9949184180901804 |
|    9     | 0.23513311484106528 |  1.0636599430162372 |
|    11    | 0.22951549845383465 |  1.0143809423615633 |
+----------+---------------------+---------------------+
Image : 9
+----------+---------------------+---------------------+
| k_means  |      F measure      | Conditional entropy |
+----------+---------------------+---------------------+
|    3     |  0.6412437558355959 |   1.18012725819668  |
|    5     | 0.47477002489576137 |  1.521595532406774  |
|    7     |  0.3832400406483903 |  1.849329885248152  |
|    9     |  0.3224895232975663 |  2.0411317277645464 |
|    11    |  0.2967511693759366 |  2.136481148674504  |
+----------+---------------------+---------------------+
Image : 10
+----------+---------------------+---------------------+
| k_means  |      F measure      | Conditional entropy |
+----------+---------------------+---------------------+
|    3     |  0.6687599345308426 |  1.2313643485862689 |
|    5     |  0.4406825421842436 |  1.6733677400113551 |
|    7     | 0.38271835413044886 |  1.9098792572302943 |
|    9     | 0.39389711025517765 |  1.7530288527060878 |
|    11    |  0.3040202087882909 |  1.8757149958105133 |
+----------+---------------------+---------------------+
```



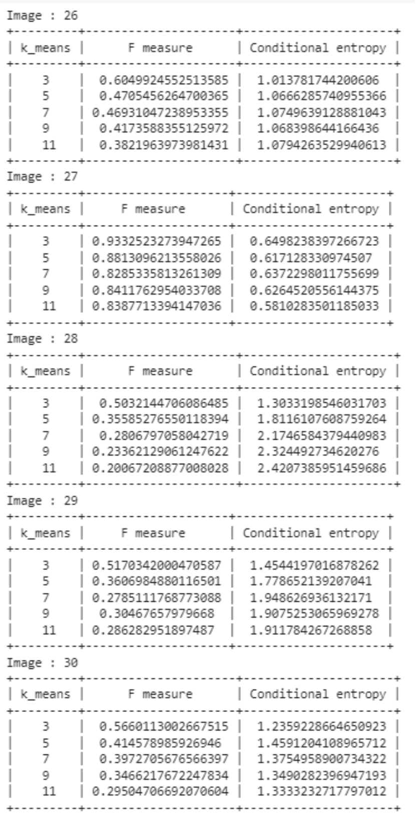
```
+---------+--------------------+---------------------+
Image : 11
+---------+--------------------+---------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+---------------------+
|    3    | 0.5758897730315778 |  1.1922004604731415 |
|    5    | 0.3328647860922726 |  1.8071734757997224 |
|    7    | 0.2827833261415299 |  2.116247046276522  |
|    9    | 0.27290624968503097|  2.3369109338957617 |
|    11   | 0.27325245352324845|  2.532546926652418  |
+---------+--------------------+---------------------+
Image : 12
+---------+--------------------+---------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+---------------------+
|    3    | 0.6527795354383115 |  1.289501293461316  |
|    5    | 0.415343820729265  |  1.8004077629464625 |
|    7    | 0.33382868979986036|  1.756525630083028  |
|    9    | 0.32046626523335114|  1.741727513232355  |
|    11   | 0.30562530334083915|  1.7108110349912007 |
+---------+--------------------+---------------------+
Image : 13
+---------+--------------------+---------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+---------------------+
|    3    |  0.57355827526547  |  1.299698196437608  |
|    5    | 0.37792514262469823|  1.9286018233643056 |
|    7    | 0.3464104379185577 |  2.145612629230664  |
|    9    |  0.325342377576974 |  2.217935106836142  |
|    11   | 0.2850759858915902 |  2.2797650401347047 |
+---------+--------------------+---------------------+
Image : 14
+---------+--------------------+--------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+--------------------+
|    3    | 0.6575392095646718 | 1.1904721399648406 |
|    5    | 0.4518393352212979 | 1.6729125796073154 |
|    7    | 0.3674971401315427 | 1.8212836441713443 |
|    9    | 0.311904358538341  | 1.9615883460629402 |
|    11   | 0.3083803542418088 | 1.9163851331995778 |
+---------+--------------------+--------------------+
Image : 15
+---------+--------------------+---------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+---------------------+
|    3    | 0.6082457740695574 |  0.9511391780882111 |
|    5    | 0.3425366505156184 |  1.5535226310070813 |
|    7    | 0.28685796888125803|  1.8374807567586713 |
|    9    | 0.2792380778844779 |  2.1062826836909725 |
|    11   | 0.2502549055543729 |  2.3669813012608194 |
+---------+--------------------+---------------------+
```



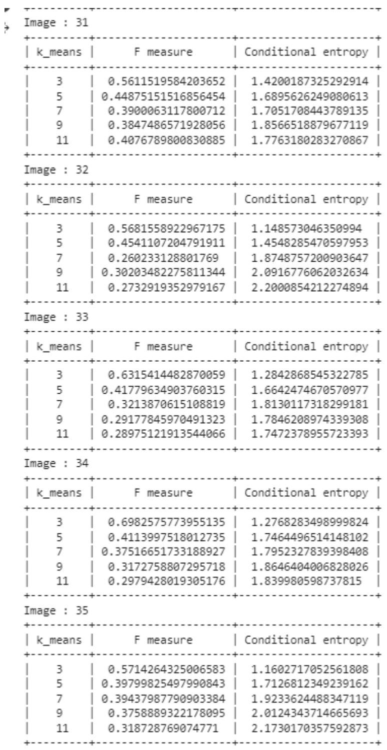
```
Image : 16
+---------+--------------------+--------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+--------------------+
|    3    | 0.7634101108377175 | 1.149804753126591  |
|    5    | 0.5486097587294617 | 1.3763383323211604 |
|    7    | 0.5099451635256352 | 1.3280671915592046 |
|    9    | 0.4078415651499521 | 1.4601557393910305 |
|    11   | 0.422881941587336  | 1.204176733336571  |
+---------+--------------------+--------------------+
Image : 17
+---------+--------------------+--------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+--------------------+
|    3    | 0.6061061685441994 | 1.2868862512729884 |
|    5    | 0.489254154005639  | 1.1933913239536218 |
|    7    | 0.3365575826223841 | 1.2963427469612385 |
|    9    | 0.2683041743125962 | 1.3176715978937759 |
|    11   | 0.2274279956431512 | 1.3677519464546546 |
+---------+--------------------+--------------------+
Image : 18
+---------+---------------------+--------------------+
| k_means |      F measure      | Conditional entropy |
+---------+---------------------+--------------------+
|    3    |  0.5706497193331088 | 1.2928171074134258 |
|    5    | 0.48386113190398927 | 1.5507559026660493 |
|    7    | 0.42338060543916456 | 1.6773165988868617 |
|    9    | 0.42536990703032734 | 1.7024055947623757 |
|    11   | 0.40018114362842405 | 1.7182735923043897 |
+---------+---------------------+--------------------+
Image : 19
+---------+---------------------+--------------------+
| k_means |      F measure      | Conditional entropy |
+---------+---------------------+--------------------+
|    3    |  0.5919005714194487 | 1.4732016143582685 |
|    5    | 0.39714720614402943 | 1.9390753348532819 |
|    7    |  0.2957460238980659 | 2.010248699672841  |
|    9    | 0.25613904436071544 | 1.973577667725488  |
|    11   | 0.22873097641004322 | 2.0010277835174026 |
+---------+---------------------+--------------------+
Image : 20
+---------+--------------------+--------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+--------------------+
|    3    | 0.6372785255600598 | 1.185384445413995  |
|    5    | 0.4562685924012605 | 1.3534808276846384 |
|    7    | 0.3318893317504831 | 1.3666942363564447 |
|    9    | 0.3094351660262912 | 1.3145583719191583 |
|    11   | 0.2676909356620542 | 1.3444492235380865 |
+---------+--------------------+--------------------+
```



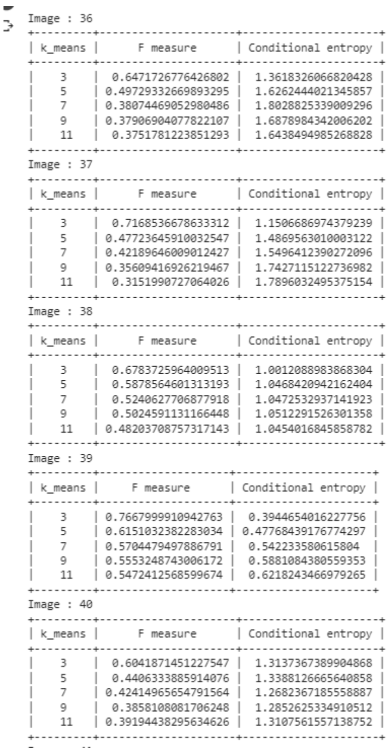
```
Image : 21
+---------+--------------------+---------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+---------------------+
|    3    | 0.7763126999679031 |  1.1960435768369402 |
|    5    |  0.558535008062316 |  1.1999986127569668 |
|    7    | 0.40418056169738376|  1.3111761571311586 |
|    9    | 0.36918392911018993|  1.3428167454473836 |
|    11   | 0.34817270777535775|  1.229174620351982  |
+---------+--------------------+---------------------+
Image : 22
+---------+--------------------+---------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+---------------------+
|    3    | 0.7426842892890987 |  1.1517747752267704 |
|    5    | 0.5033670164843737 |  1.4115738099408972 |
|    7    | 0.5042747643847761 |  1.3784843861846248 |
|    9    | 0.46200556843511886|  1.4523001490272514 |
|    11   | 0.43476875848748825|  1.4501746182009057 |
+---------+--------------------+---------------------+
Image : 23
+---------+--------------------+---------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+---------------------+
|    3    | 0.6817430246763627 |  1.3663709647935118 |
|    5    | 0.5292744067704654 |   1.45715072560833  |
|    7    | 0.4322148034474824 |  1.4341060932496477 |
|    9    | 0.34595738047985397|  1.3884228502634748 |
|    11   | 0.3440371780529506 |  1.2861010850299153 |
+---------+--------------------+---------------------+
Image : 24
+---------+--------------------+---------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+---------------------+
|    3    | 0.5898189211779723 |  1.3664787928467874 |
|    5    | 0.3824575532891501 |  1.8367942475369368 |
|    7    | 0.30829730903336056|  2.0234821061174753 |
|    9    | 0.2721840669900318 |  1.9849635677737414 |
|    11   | 0.2518376560108851 |  1.941737869201713  |
+---------+--------------------+---------------------+
Image : 25
+---------+--------------------+--------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+--------------------+
|    3    | 0.6275671034935322 | 1.3596501777348695 |
|    5    | 0.6274711975175694 | 1.3121069586406213 |
|    7    | 0.6766585380421489 | 1.1779365889304907 |
|    9    | 0.6540138351243031 | 1.2476474885732594 |
|    11   | 0.6471437543879092 | 1.2281497994852903 |
+---------+--------------------+--------------------+
```



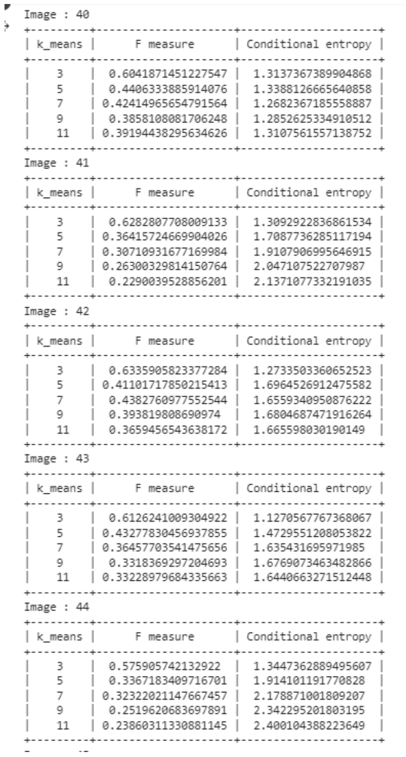
```
Image : 26
+---------+--------------------+---------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+---------------------+
|    3    | 0.6049924552513585 |  1.013781744200606  |
|    5    | 0.4705456264700365 |  1.0666285740955366 |
|    7    | 0.46931047238953355|  1.0749639128881043 |
|    9    | 0.4173588355125972 |  1.068398644166436  |
|    11   | 0.3821963973981431 |  1.0794263529940613 |
+---------+--------------------+---------------------+
Image : 27
+---------+--------------------+--------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+--------------------+
|    3    | 0.9332523273947265 | 0.6498238397266723 |
|    5    | 0.8813096213558026 | 0.617128330974507  |
|    7    | 0.8285335813261309 | 0.6372298011755699 |
|    9    | 0.8411762954033708 | 0.6264520556144375 |
|    11   | 0.8387713394147036 | 0.5810283501185033 |
+---------+--------------------+--------------------+
Image : 28
+---------+--------------------+---------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+---------------------+
|    3    | 0.5032144706086485 |  1.3033198546031703 |
|    5    | 0.35585276550118394|  1.8116107608759264 |
|    7    | 0.2806797058042719 |  2.1746584379440983 |
|    9    | 0.23362129061247622|  2.324492734620276  |
|    11   | 0.20067208877008028|  2.4207385951459686 |
+---------+--------------------+---------------------+
Image : 29
+---------+--------------------+--------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+--------------------+
|    3    | 0.5170342000470587 | 1.4544197016878262 |
|    5    | 0.3606984880116501 | 1.778652139207041  |
|    7    | 0.2785111768773088 | 1.948626936132171  |
|    9    |  0.30467657979668  | 1.9075253065969278 |
|    11   | 0.286282951897487  | 1.911784267268858  |
+---------+--------------------+--------------------+
Image : 30
+---------+--------------------+---------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+---------------------+
|    3    | 0.5660113002667515 |  1.2359228664650923 |
|    5    | 0.414578985926946  |  1.4591204108965712 |
|    7    | 0.3972705676566397 |  1.3754958900734322 |
|    9    | 0.3466217672247834 |  1.3490282396947193 |
|    11   | 0.29504706692070604|  1.3333232717797012 |
+---------+--------------------+---------------------+
```



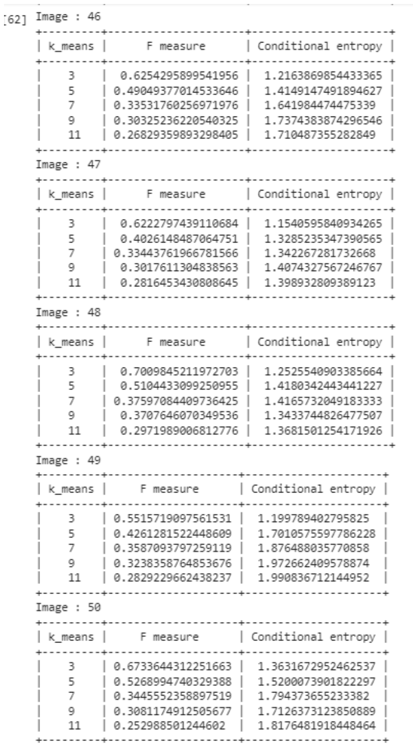
```
+---------+---------------------+----------------------+
Image : 31
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.5611519584203652 |  1.4200187325292914  |
|    5    | 0.44875151516856454 |  1.6895626249080613  |
|    7    |  0.3900063117800712 |  1.7051708443789135  |
|    9    |  0.3847486571928056 |  1.8566518879677119  |
|   11    |  0.4076789800830885 |  1.7763180283270867  |
+---------+---------------------+----------------------+
Image : 32
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.5681558922967175 |  1.148573046350994   |
|    5    |  0.4541107204791911 |  1.4548285470597953  |
|    7    |  0.260233128801769  |  1.8748757200903647  |
|    9    | 0.30203482275811344 |  2.0916776062032634  |
|   11    |  0.2732919352979167 |  2.2000854212274894  |
+---------+---------------------+----------------------+
Image : 33
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.6315414482870059 |  1.2842868545322785  |
|    5    | 0.41779634903760315 |  1.6642474670570977  |
|    7    |  0.3213870615108819 |  1.8130117318299181  |
|    9    | 0.29177845970491323 |  1.7846208974339308  |
|   11    | 0.28975121913544066 |  1.7472378955723393  |
+---------+---------------------+----------------------+
Image : 34
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.6982575773955135 |  1.2768283498999824  |
|    5    |  0.4113997518012735 |  1.7464496514148102  |
|    7    | 0.37516651733188927 |  1.7952327839398408  |
|    9    |  0.3172758807295718 |  1.8646404006828026  |
|   11    |  0.2979428019305176 |  1.839980598737815   |
+---------+---------------------+----------------------+
Image : 35
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.5714264325006583 |  1.1602717052561808  |
|    5    | 0.39799825497990843 |  1.7126812349239162  |
|    7    | 0.39437987790903384 |  1.9233624488347119  |
|    9    |  0.3758889322178095 |  2.0124343714665693  |
|   11    |  0.318728769074771  |  2.1730170357592873  |
+---------+---------------------+----------------------+
```



```
Image : 36
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.6471726776426802 |  1.3618326066820428  |
|    5    | 0.49729332669893295 |  1.6262444021345857  |
|    7    | 0.38074469052980486 |  1.8028825339009296  |
|    9    | 0.37906904077822107 |  1.6878984342006202  |
|   11    |  0.3751781223851293 |  1.6438494985268828  |
+---------+---------------------+----------------------+
Image : 37
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.7168536678633312 |  1.1506686974379239  |
|    5    | 0.47723645910032547 |  1.4869563010003122  |
|    7    | 0.42189646009012427 |  1.5496412390272096  |
|    9    | 0.35609416926219467 |  1.7427115122736982  |
|   11    |  0.3151990727064026 |  1.7896032495375154  |
+---------+---------------------+----------------------+
Image : 38
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.6783725964009513 |  1.0012088983868304  |
|    5    |  0.5878564601313193 |  1.0468420942162404  |
|    7    |  0.5240627706877918 |  1.0472532937141923  |
|    9    |  0.5024591131166448 |  1.0512291526301358  |
|   11    | 0.48203708757317143 |  1.0454016845858782  |
+---------+---------------------+----------------------+
Image : 39
+---------+--------------------+---------------------+
| k_means |     F measure      | Conditional entropy |
+---------+--------------------+---------------------+
|    3    | 0.7667999910942763 |  0.3944654016227756 |
|    5    | 0.6151032382283034 | 0.47768439176774297 |
|    7    | 0.5704479497886791 |  0.542233580615804  |
|    9    | 0.5553248743006172 |  0.5881084380559353 |
|   11    | 0.5472412568599674 |  0.6218243466979265 |
+---------+--------------------+---------------------+
Image : 40
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.6041871451227547 |  1.3137367389904868  |
|    5    |  0.4406333885914076 |  1.3388126665640858  |
|    7    | 0.42414965654791564 |  1.2682367185558887  |
|    9    |  0.3858108081706248 |  1.2852625334910512  |
|   11    | 0.39194438295634626 |  1.3107561557138752  |
+---------+---------------------+----------------------+
```



```
Image : 40
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.6041871451227547 |  1.3137367389904868  |
|    5    |  0.4406333885914076 |  1.3388126665640858  |
|    7    | 0.42414965654791564 |  1.2682367185558887  |
|    9    |  0.3858108081706248 |  1.2852625334910512  |
|   11    | 0.39194438295634626 |  1.3107561557138752  |
+---------+---------------------+----------------------+
Image : 41
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.6282807708009133 |  1.3092922836861534  |
|    5    | 0.36415724669904026 |  1.7087736285117194  |
|    7    | 0.30710931677169984 |  1.9107906995646915  |
|    9    | 0.26300329814150764 |  2.047107522707987   |
|   11    |  0.2290039528856201 |  2.1371077332191035  |
+---------+---------------------+----------------------+
Image : 42
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.6335905823377284 |  1.2733503360652523  |
|    5    | 0.41101717850215413 |  1.6964526912475582  |
|    7    |  0.4382760977552544 |  1.6559340950876222  |
|    9    |  0.393819808690974  |  1.6804687471916264  |
|   11    |  0.3659456543638172 |  1.665598030190149   |
+---------+---------------------+----------------------+
Image : 43
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.6126241009304922 |  1.1270567767368067  |
|    5    | 0.43277830456937855 |  1.4729551208053822  |
|    7    | 0.36457703541475656 |  1.635431695971985   |
|    9    |  0.3318369297204693 |  1.6769073463482866  |
|   11    | 0.33228979684335663 |  1.6440663271512448  |
+---------+---------------------+----------------------+
Image : 44
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.575905742132922  |  1.3447362889495607  |
|    5    |  0.3367183409716701 |  1.914101191770828   |
|    7    | 0.32322021147667457 |  2.178871001809207   |
|    9    |  0.2519620683697891 |  2.342295201803195   |
|   11    | 0.23860311330881145 |  2.400104388223649   |
+---------+---------------------+----------------------+
```



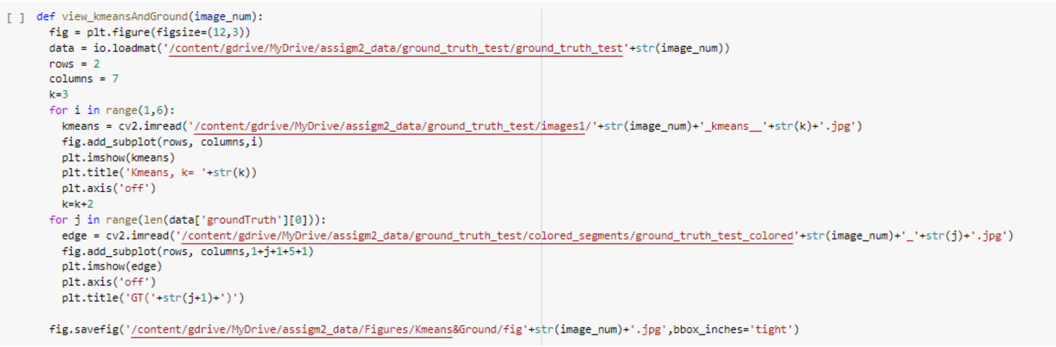
```
[62] Image : 46
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.6254295899541956 |  1.2163869854433365  |
|    5    | 0.49049377014533646 |  1.4149147491894627  |
|    7    | 0.33531760256971976 |  1.641984474475339   |
|    9    | 0.30325236220540325 |  1.7374383874296546  |
|   11    | 0.26829359893298405 |  1.710487355282849   |
+---------+---------------------+----------------------+
Image : 47
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.6222797439110684 |  1.1540595840934265  |
|    5    |  0.4026148487064751 |  1.3285235347390565  |
|    7    | 0.33443761966781566 |  1.342267281732668   |
|    9    |  0.3017611304838563 |  1.4074327567246767  |
|   11    |  0.2816453430808645 |  1.398932809389123   |
+---------+---------------------+----------------------+
Image : 48
+---------+---------------------+----------------------+
| k_means |      F measure      | Conditional entropy  |
+---------+---------------------+----------------------+
|    3    |  0.7009845211972703 |  1.2525540903385664  |
|    5    |  0.5104433099250955 |  1.4180342443441227  |
|    7    | 0.37597084409736425 |  1.4165732049183333  |
|    9    |  0.3707646070349536 |  1.3433744826477507  |
|   11    |  0.2971989006812776 |  1.3681501254171926  |
+---------+---------------------+----------------------+
Image : 49
+---------+--------------------+----------------------+
| k_means |     F measure      | Conditional entropy  |
+---------+--------------------+----------------------+
|    3    | 0.5515719097561531 |  1.199789402795825   |
|    5    | 0.4261281522448609 |  1.7010575597786228  |
|    7    | 0.3587093797259119 |  1.876488035770858   |
|    9    | 0.3238358764853676 |  1.972662409578874   |
|   11    | 0.2829229662438237 |  1.990836712144952   |
+---------+--------------------+----------------------+
Image : 50
+---------+--------------------+----------------------+
| k_means |     F measure      | Conditional entropy  |
+---------+--------------------+----------------------+
|    3    | 0.6733644312251663 |  1.3631672952462537  |
|    5    | 0.5268994740329388 |  1.5200073901822297  |
|    7    | 0.3445552358897519 |  1.794373655233382   |
|    9    | 0.3081174912505677 |  1.7126373123850889  |
|   11    | 0.252988501244602  |  1.8176481918448464  |
+---------+--------------------+----------------------+
```

c)we displayed all the results for the 50 images (good & bad results)

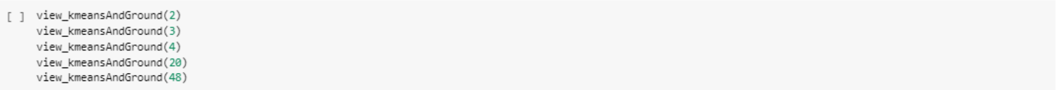
4. Big picture

a)

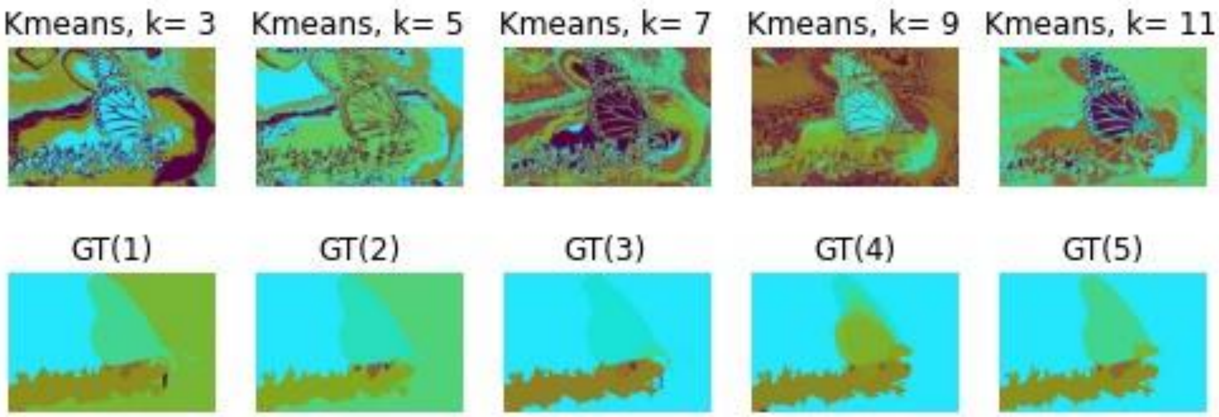
A function used to plot the 5 k means (3-5-7-9-11) and the ground truth for 5 selected images

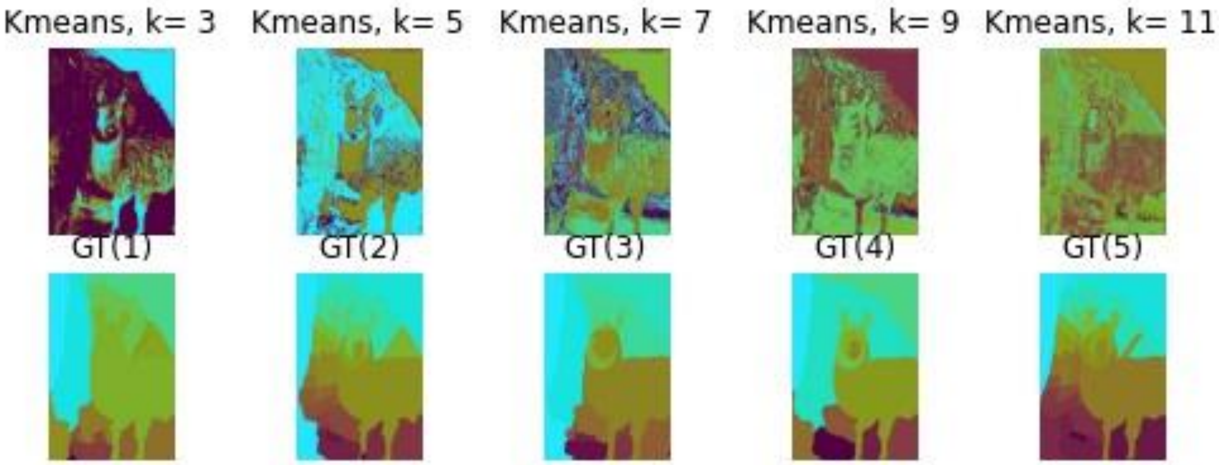


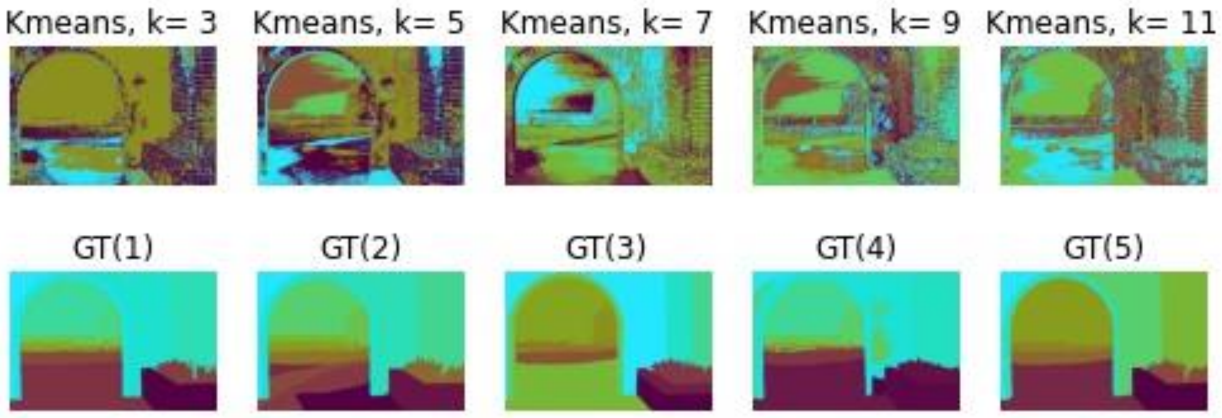
```
[ ] def view_kmeansAndGround(image_num):
      fig = plt.figure(figsize=(12,3))
      data = io.loadmat('/content/gdrive/MyDrive/assigm2_data/ground_truth_test/ground_truth_test'+str(image_num))
      rows = 2
      columns = 7
      k=3
      for i in range(1,6):
        kmeans = cv2.imread('/content/gdrive/MyDrive/assigm2_data/ground_truth_test/images1/'+str(image_num)+'_kmeans__'+str(k)+'.jpg')
        fig.add_subplot(rows, columns,i)
        plt.imshow(kmeans)
        plt.title('Kmeans, k= '+str(k))
        plt.axis('off')
        k=k+2
      for j in range(len(data['groundTruth'][0])):
        edge = cv2.imread('/content/gdrive/MyDrive/assigm2_data/ground_truth_test/colored_segments/ground_truth_test_colored'+str(image_num)+'_'+str(j)+'.jpg')
        fig.add_subplot(rows, columns,1+j+1+5+1)
        plt.imshow(edge)
        plt.axis('off')
        plt.title('GT('+str(j+1)+')')

      fig.savefig('/content/gdrive/MyDrive/assigm2_data/Figures/Kmeans&Ground/fig'+str(image_num)+'.jpg',bbox_inches='tight')
```

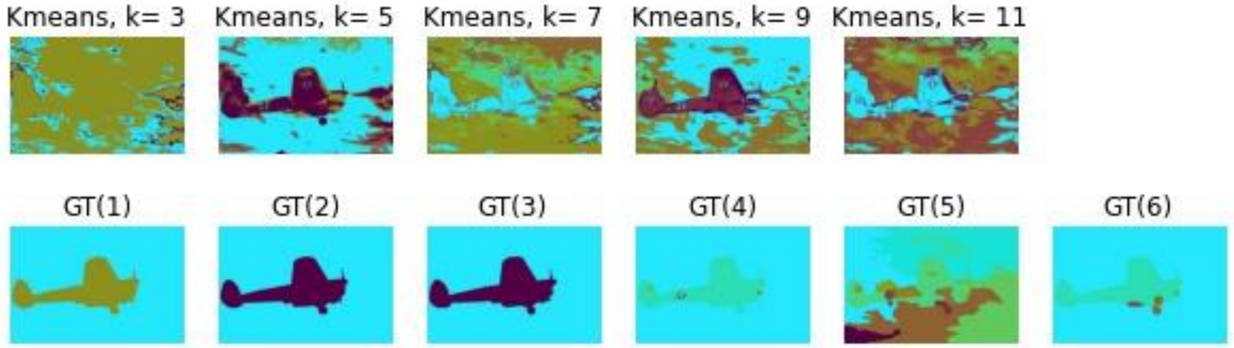


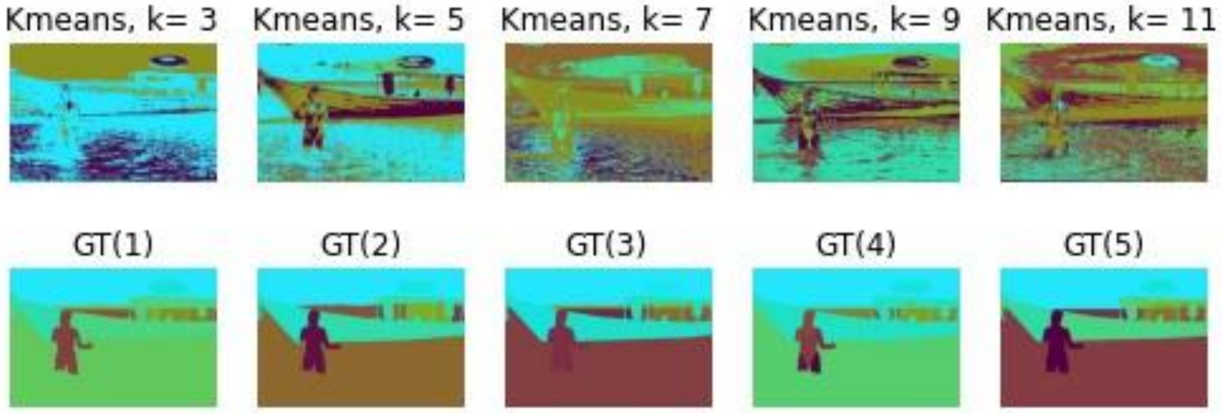
```
[ ] view_kmeansAndGround(2)
    view_kmeansAndGround(3)
    view_kmeansAndGround(4)
    view_kmeansAndGround(20)
    view_kmeansAndGround(48)
```





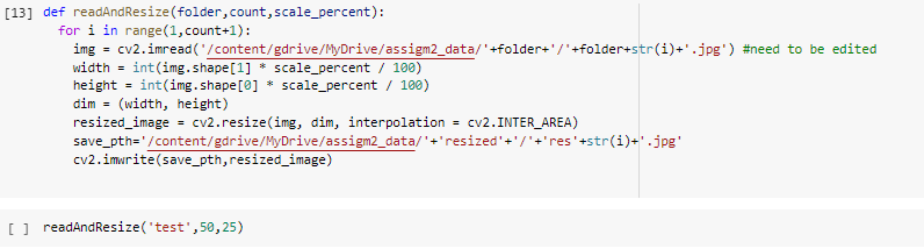

Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
GT(1)
GT(2)
GT(3)
GT(4)
GT(5)


Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
GT(1)
GT(2)
GT(3)
GT(4)
GT(5)
GT(6)


Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
GT(1)
GT(2)
GT(3)
GT(4)
GT(5)

b)

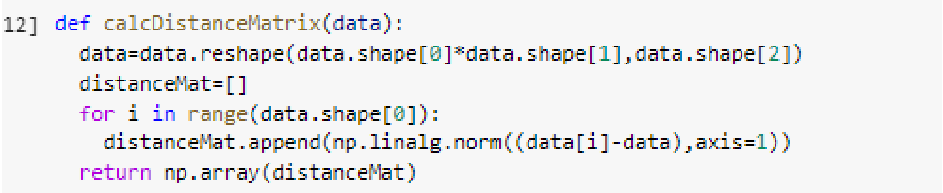
A function used to resize the original image by taking 25% of the width and 25% of the length to use it in the normalized cut



```
[13] def readAndResize(folder,count,scale_percent):
       for i in range(1,count+1):
         img = cv2.imread('/content/gdrive/MyDrive/assigm2_data/'+folder+'/'+folder+str(i)+'.jpg') #need to be edited
         width = int(img.shape[1] * scale_percent / 100)
         height = int(img.shape[0] * scale_percent / 100)
         dim = (width, height)
         resized_image = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
         save_pth='/content/gdrive/MyDrive/assigm2_data/'+'resized'+'/'+'res'+str(i)+'.jpg'
         cv2.imwrite(save_pth,resized_image)
```

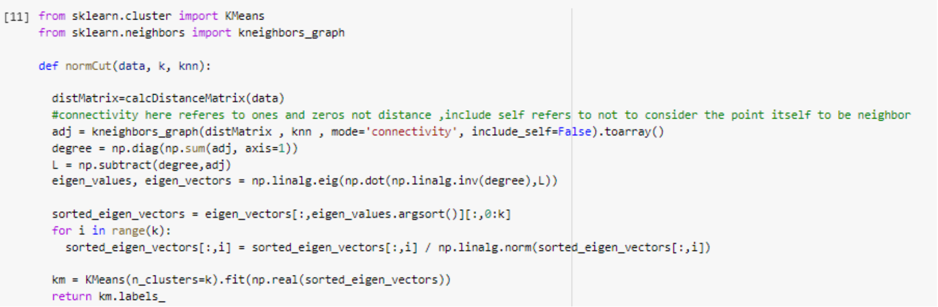
```
[ ]  readAndResize('test',50,25)
```

A function used to calculate the distance matrix



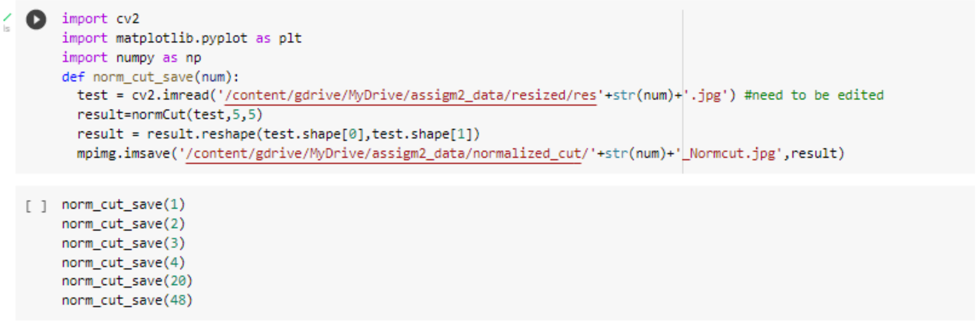
```
12] def calcDistanceMatrix(data):
      data=data.reshape(data.shape[0]*data.shape[1],data.shape[2])
      distanceMat=[]
      for i in range(data.shape[0]):
        distanceMat.append(np.linalg.norm((data[i]-data),axis=1))
      return np.array(distanceMat)
```

A function used to calculate the normalization cut



```
[11] from sklearn.cluster import KMeans
     from sklearn.neighbors import kneighbors_graph

     def normCut(data, k, knn):

       distMatrix=calcDistanceMatrix(data)
       #connectivity here referes to ones and zeros not distance ,include self refers to not to consider the point itself to be neighbor
       adj = kneighbors_graph(distMatrix , knn , mode='connectivity', include_self=False).toarray()
       degree = np.diag(np.sum(adj, axis=1))
       L = np.subtract(degree,adj)
       eigen_values, eigen_vectors = np.linalg.eig(np.dot(np.linalg.inv(degree),L))

       sorted_eigen_vectors = eigen_vectors[:,eigen_values.argsort()][:,0:k]
       for i in range(k):
         sorted_eigen_vectors[:,i] = sorted_eigen_vectors[:,i] / np.linalg.norm(sorted_eigen_vectors[:,i])

       km = KMeans(n_clusters=k).fit(np.real(sorted_eigen_vectors))
       return km.labels_
```

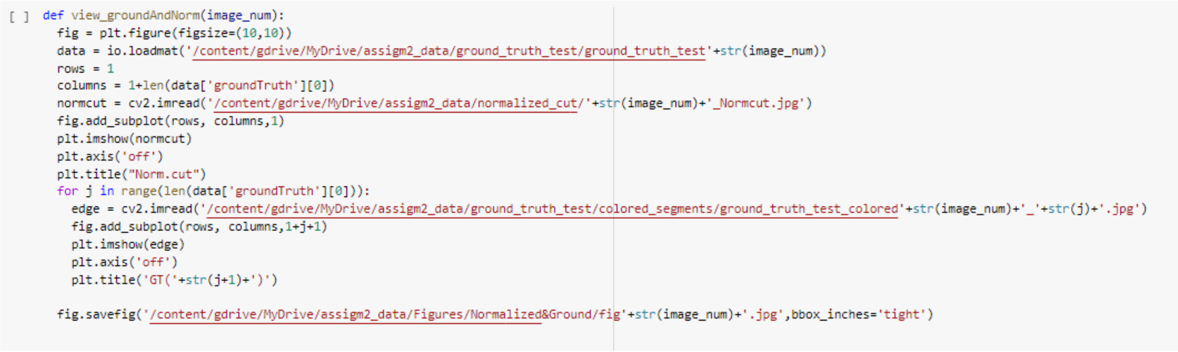
A function used to calculate the normalized cut and save the image resulted from it in the drive



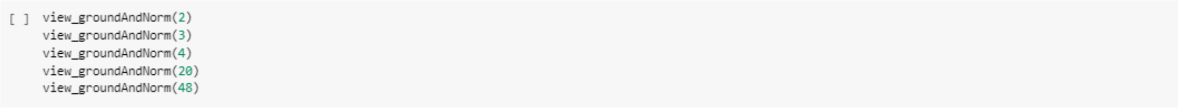
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
def norm_cut_save(num):
  test = cv2.imread('/content/gdrive/MyDrive/assigm2_data/resized/res'+str(num)+'.jpg') #need to be edited
  result=normCut(test,5,5)
  result = result.reshape(test.shape[0],test.shape[1])
  mpimg.imsave('/content/gdrive/MyDrive/assigm2_data/normalized_cut/'+str(num)+'_Normcut.jpg',result)
```

```
[ ]  norm_cut_save(1)
     norm_cut_save(2)
     norm_cut_save(3)
     norm_cut_save(4)
     norm_cut_save(20)
     norm_cut_save(48)
```

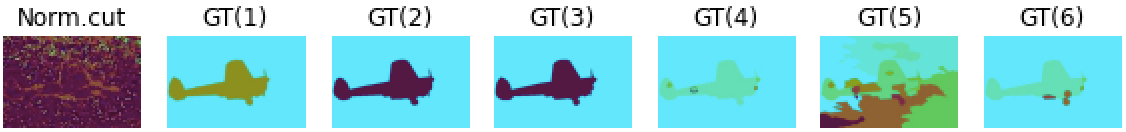
A function used to plot the normalized cut and the ground truth for 5 selected images

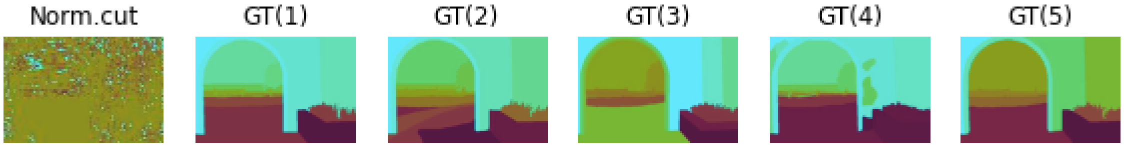


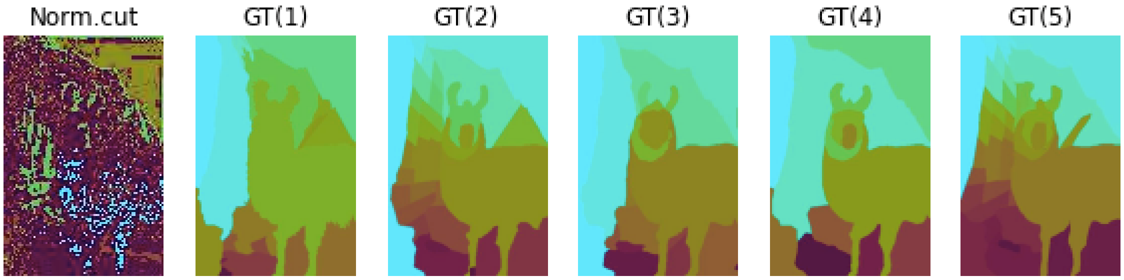
```
[ ] def view_groundAndNorm(image_num):
      fig = plt.figure(figsize=(10,10))
      data = io.loadmat('/content/gdrive/MyDrive/assigm2_data/ground_truth_test/ground_truth_test'+str(image_num))
      rows = 1
      columns = 1+len(data['groundTruth'][0])
      normcut = cv2.imread('/content/gdrive/MyDrive/assigm2_data/normalized_cut/'+str(image_num)+'_Normcut.jpg')
      fig.add_subplot(rows, columns,1)
      plt.imshow(normcut)
      plt.axis('off')
      plt.title("Norm.cut")
      for j in range(len(data['groundTruth'][0])):
        edge = cv2.imread('/content/gdrive/MyDrive/assigm2_data/ground_truth_test/colored_segments/ground_truth_test_colored'+str(image_num)+'_'+str(j)+'.jpg')
        fig.add_subplot(rows, columns,1+j+1)
        plt.imshow(edge)
        plt.axis('off')
        plt.title('GT('+str(j+1)+')')

      fig.savefig('/content/gdrive/MyDrive/assigm2_data/Figures/Normalized&Ground/fig'+str(image_num)+'.jpg',bbox_inches='tight')
```

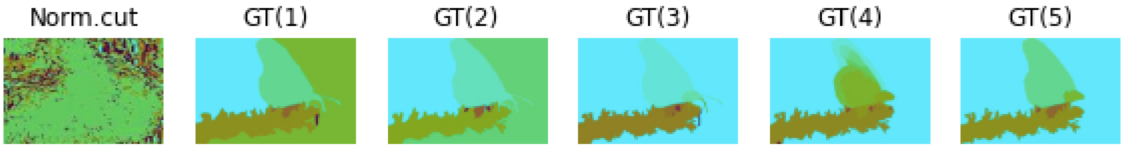


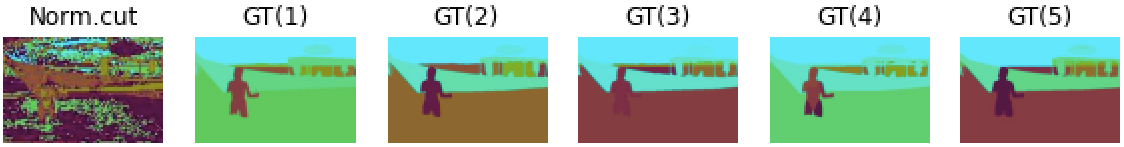
```
[ ] view_groundAndNorm(2)
    view_groundAndNorm(3)
    view_groundAndNorm(4)
    view_groundAndNorm(20)
    view_groundAndNorm(48)
```





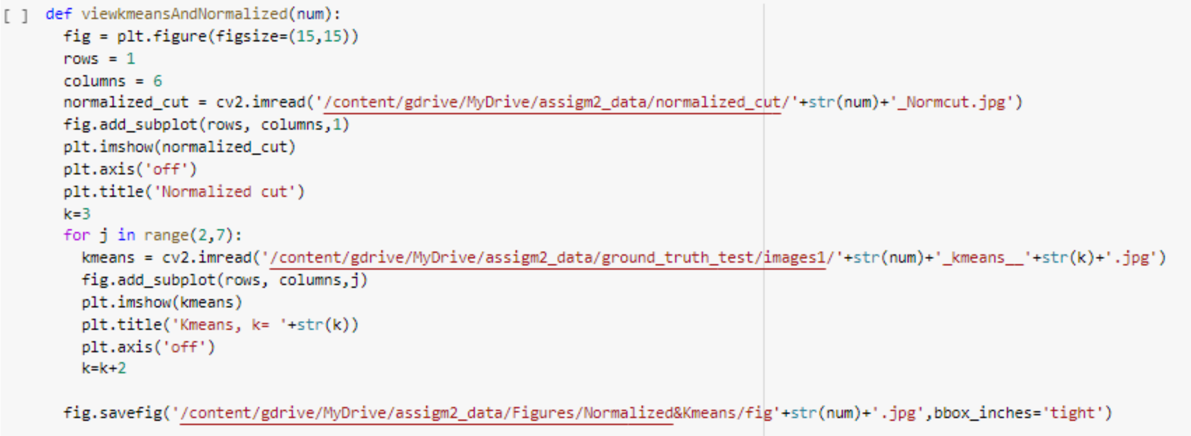




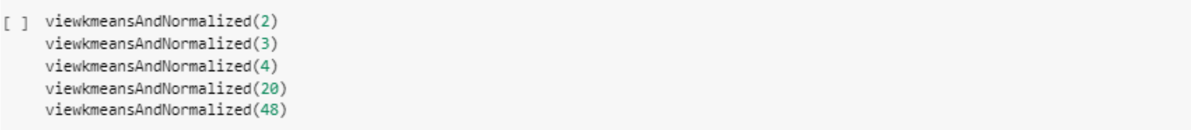


c)

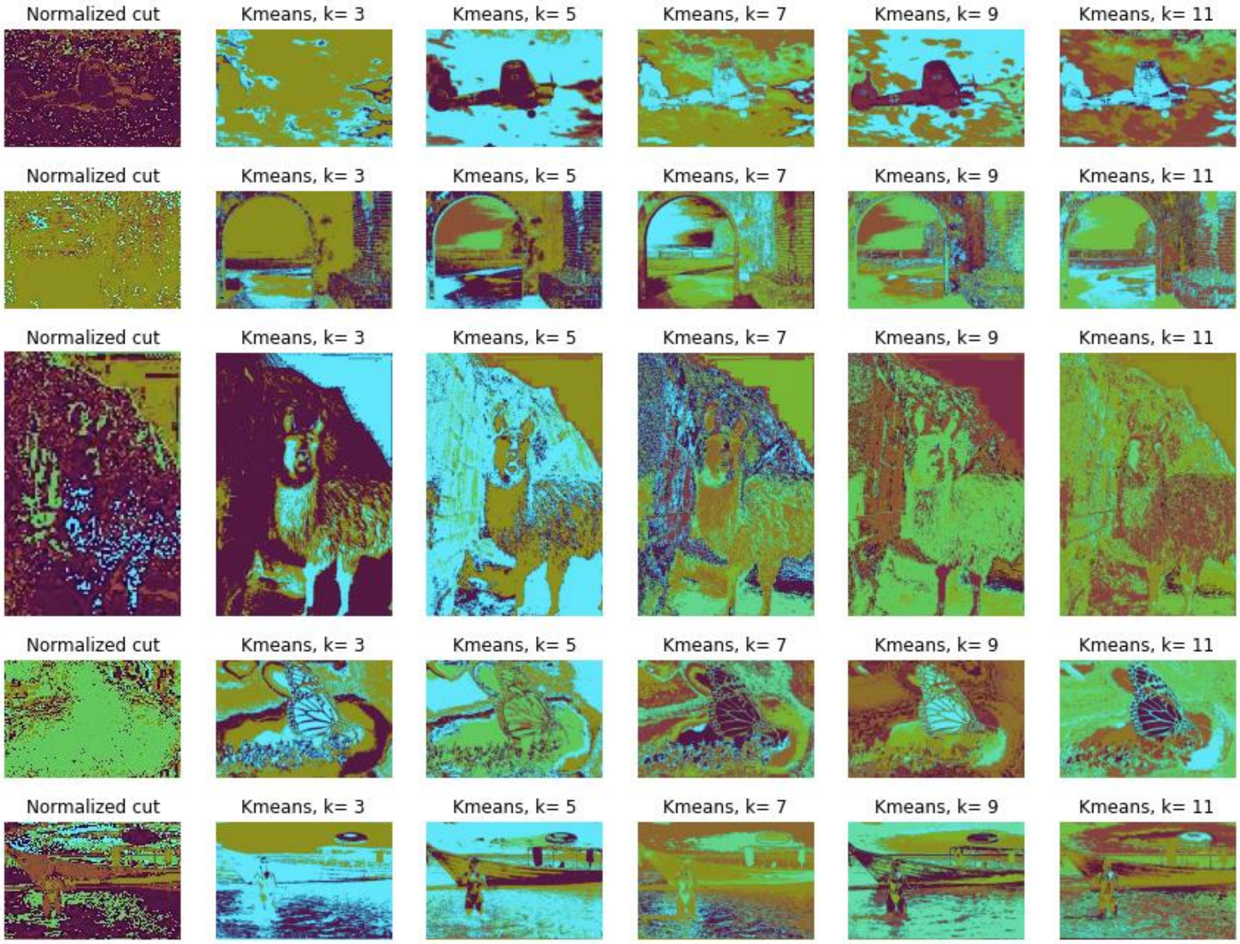
A function used to plot the 5 k means (3-5-7-9-11) and the normalized cut for 5 selected images



```
[ ] def viewkmeansAndNormalized(num):
      fig = plt.figure(figsize=(15,15))
      rows = 1
      columns = 6
      normalized_cut = cv2.imread('/content/gdrive/MyDrive/assigm2_data/normalized_cut/'+str(num)+'_Normcut.jpg')
      fig.add_subplot(rows, columns,1)
      plt.imshow(normalized_cut)
      plt.axis('off')
      plt.title('Normalized cut')
      k=3
      for j in range(2,7):
        kmeans = cv2.imread('/content/gdrive/MyDrive/assigm2_data/ground_truth_test/images1/'+str(num)+'_kmeans__'+str(k)+'.jpg')
        fig.add_subplot(rows, columns,j)
        plt.imshow(kmeans)
        plt.title('Kmeans, k= '+str(k))
        plt.axis('off')
        k=k+2

      fig.savefig('/content/gdrive/MyDrive/assigm2_data/Figures/Normalized&Kmeans/fig'+str(num)+'.jpg',bbox_inches='tight')
```



```
[ ] viewkmeansAndNormalized(2)
    viewkmeansAndNormalized(3)
    viewkmeansAndNormalized(4)
    viewkmeansAndNormalized(20)
    viewkmeansAndNormalized(48)
```



Normalized cut
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11

Normalized cut
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11

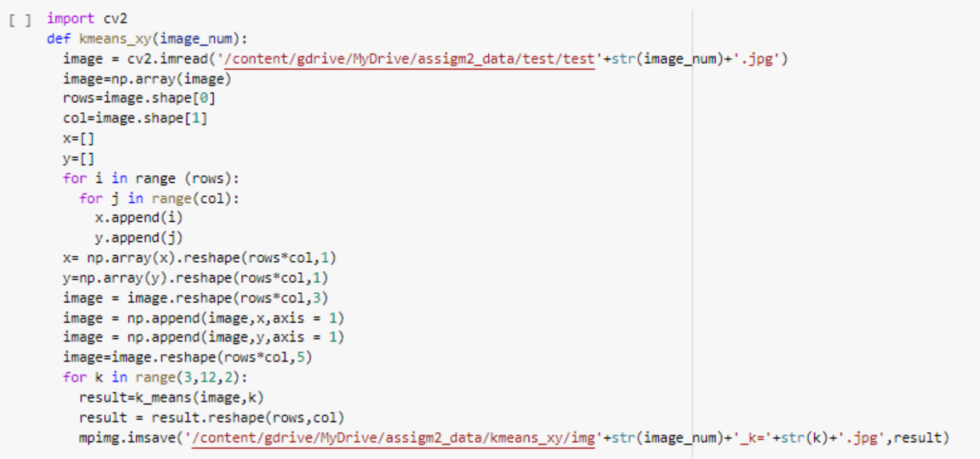
Normalized cut
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11

Normalized cut
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11

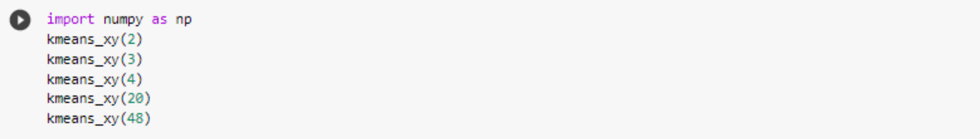
Normalized cut
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11

5. Extra

A function used to bring the x, y coordinates for every point in the image and calculate the k means for this image after modification (adding x , y to it)

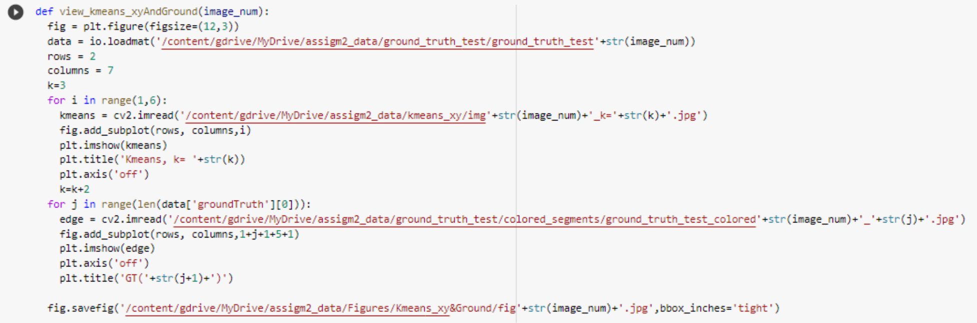


```
import cv2
def kmeans_xy(image_num):
  image = cv2.imread('/content/gdrive/MyDrive/assigm2_data/test/test'+str(image_num)+'.jpg')
  image=np.array(image)
  rows=image.shape[0]
  col=image.shape[1]
  x=[]
  y=[]
  for i in range (rows):
    for j in range(col):
      x.append(i)
      y.append(j)
  x= np.array(x).reshape(rows*col,1)
  y=np.array(y).reshape(rows*col,1)
  image = image.reshape(rows*col,3)
  image = np.append(image,x,axis = 1)
  image = np.append(image,y,axis = 1)
  image=image.reshape(rows*col,5)
  for k in range(3,12,2):
    result=k_means(image,k)
    result = result.reshape(rows,col)
    mpimg.imsave('/content/gdrive/MyDrive/assigm2_data/kmeans_xy/img'+str(image_num)+'_k='+str(k)+'.jpg',result)
```

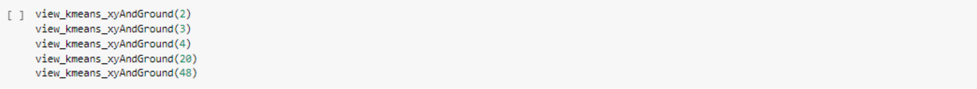


```
import numpy as np
kmeans_xy(2)
kmeans_xy(3)
kmeans_xy(4)
kmeans_xy(20)
kmeans_xy(48)
```

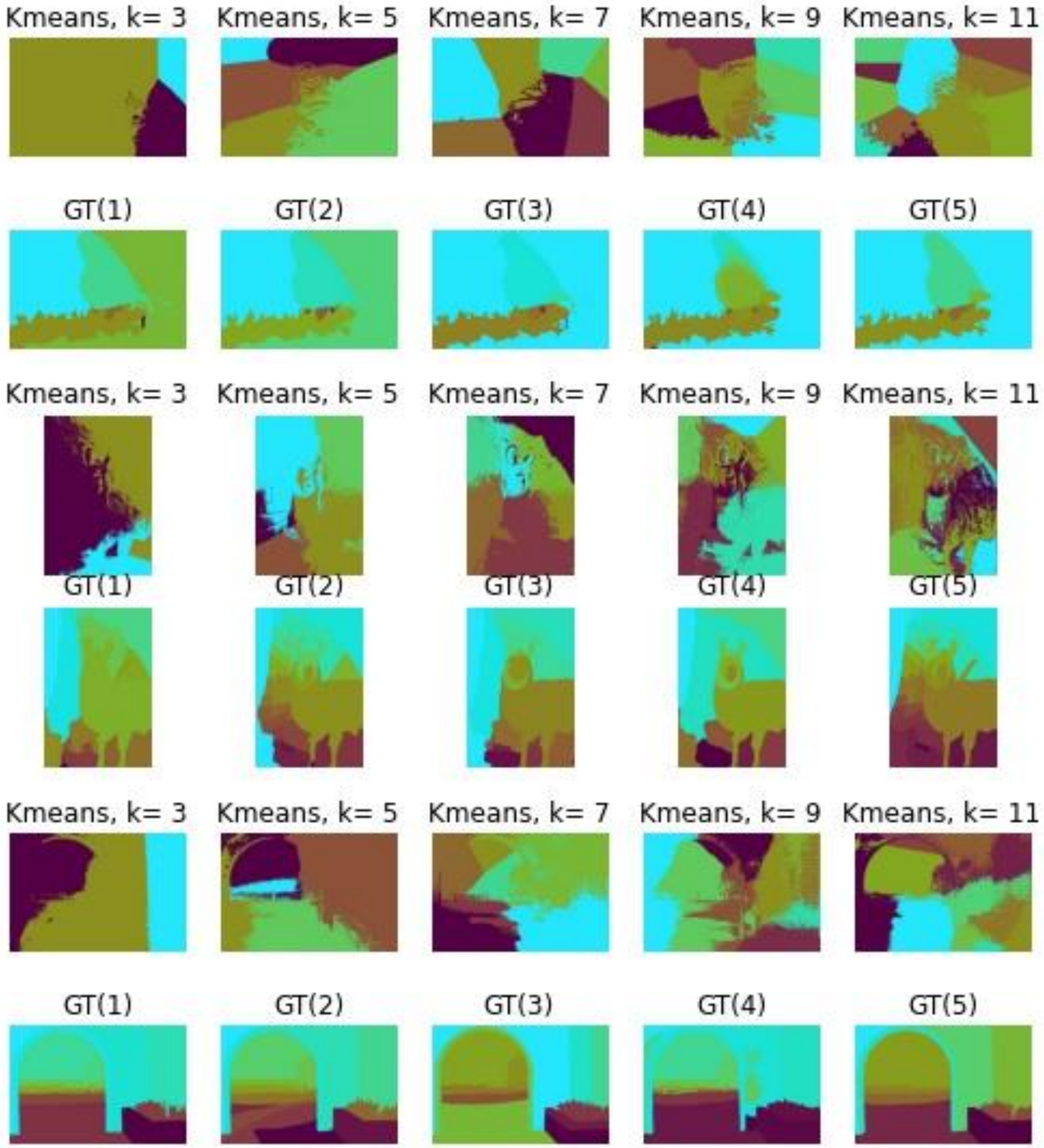
A function used to plot the resulted images with the ground truth for 5 selected images

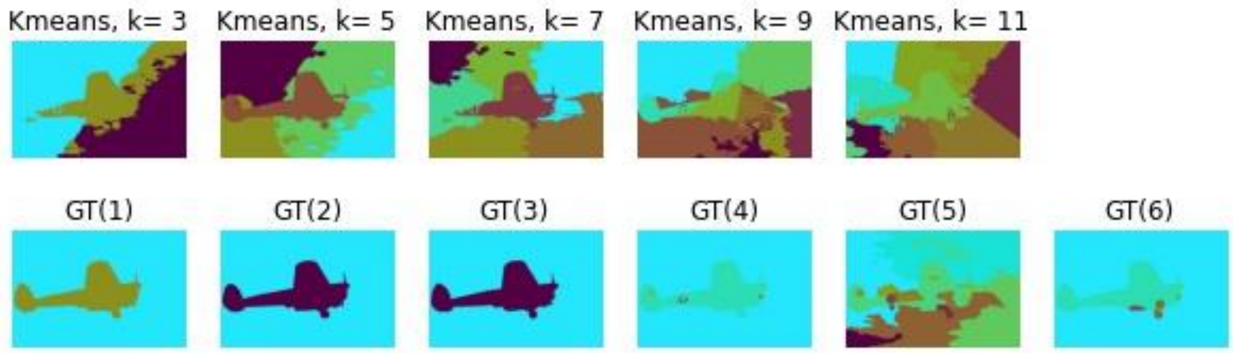


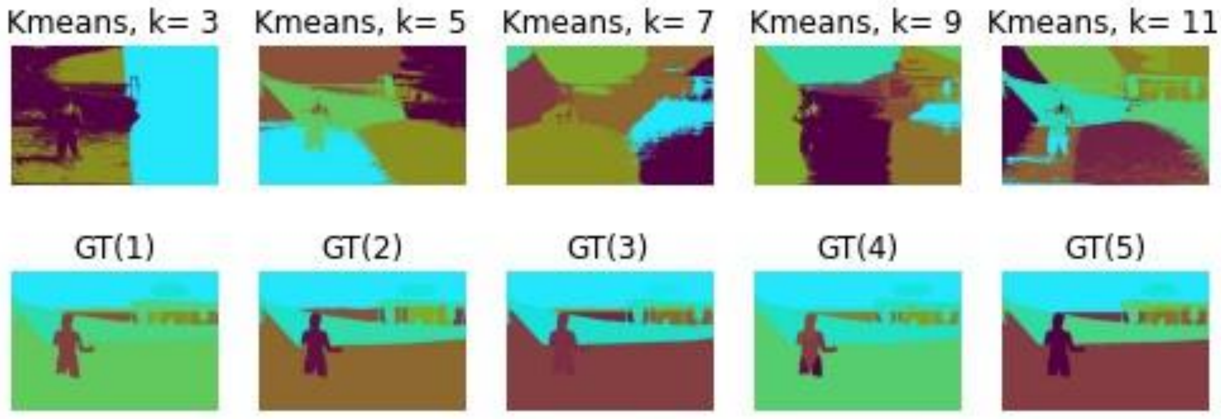
```
def view_kmeans_xyAndGround(image_num):
  fig = plt.figure(figsize=(12,3))
  data = io.loadmat('/content/gdrive/MyDrive/assigm2_data/ground_truth_test/ground_truth_test'+str(image_num))
  rows = 2
  columns = 7
  k=3
  for i in range(1,6):
    kmeans = cv2.imread('/content/gdrive/MyDrive/assigm2_data/kmeans_xy/img'+str(image_num)+'_k='+str(k)+'.jpg')
    fig.add_subplot(rows, columns,i)
    plt.imshow(kmeans)
    plt.title('Kmeans, k= '+str(k))
    plt.axis('off')
    k=k+2
  for j in range(len(data['groundTruth'][0])):
    edge = cv2.imread('/content/gdrive/MyDrive/assigm2_data/ground_truth_test/colored_segments/ground_truth_test_colored'+str(image_num)+'_'+str(j)+'.jpg')
    fig.add_subplot(rows, columns,1+j+1+5+1)
    plt.imshow(edge)
    plt.axis('off')
    plt.title('GT('+str(j+1)+')')

  fig.savefig('/content/gdrive/MyDrive/assigm2_data/Figures/Kmeans_xy&Ground/fig'+str(image_num)+'.jpg',bbox_inches='tight')
```



```
view_kmeans_xyAndGround(2)
view_kmeans_xyAndGround(3)
view_kmeans_xyAndGround(4)
view_kmeans_xyAndGround(20)
view_kmeans_xyAndGround(48)
```


Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
GT(1)
GT(2)
GT(3)
GT(4)
GT(5)
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
GT(1)
GT(2)
GT(3)
GT(4)
GT(5)
Kmeans, k= 3
Kmeans, k= 5
Kmeans, k= 7
Kmeans, k= 9
Kmeans, k= 11
GT(1)
GT(2)
GT(3)
GT(4)
GT(5)





project on google colab

Drive having all the results