Assignment 2

Image Segmentation

Pattern Recognition

Omar Khaled - 5674
Noureen Mahmoud - 5810

1. Load Datasets :

Drive authorisation

```
[ ]  from google.colab import drive
     drive.mount('/content/drive')
```

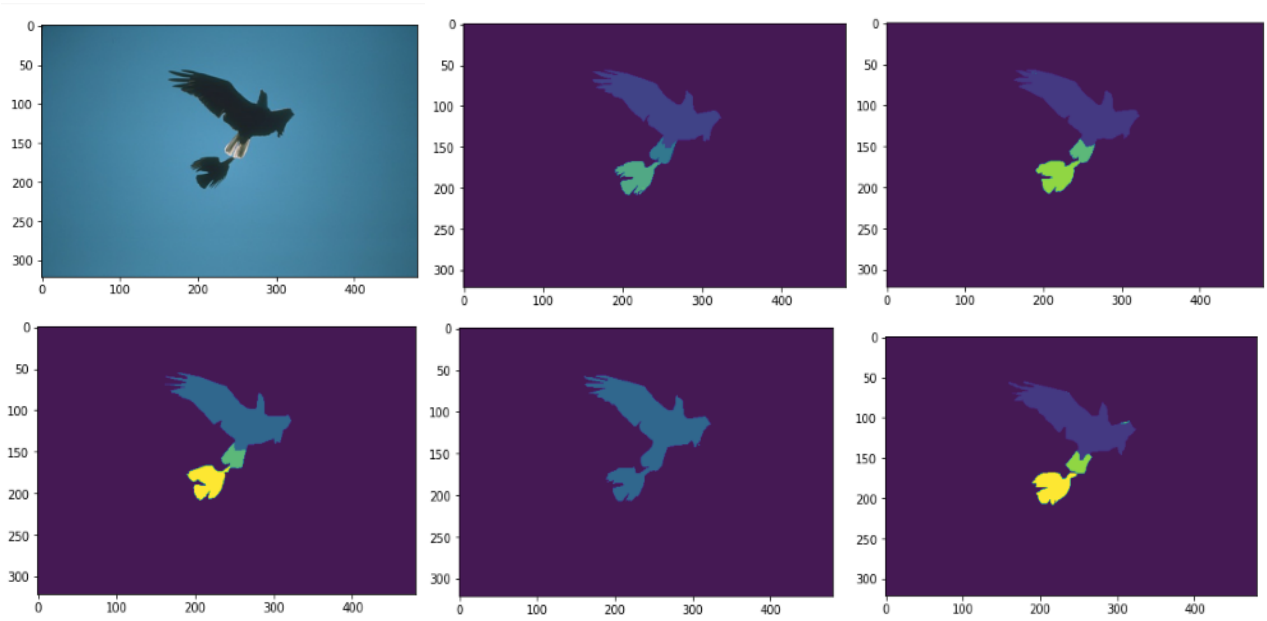
```
[ ]  TrainPath = "/content/drive/MyDrive/data/images/train/"
     GroundTruthPath = "/content/drive/MyDrive/data/ground_truth/train/"
     NumOfPixels = 481*321
     Dimensions = 3
     NumOfImages=200
     Train, GroundTruth = DataLoading(TrainPath,GroundTruthPath,(NumOfPixels*NumOfImages,Dimensions))
```

2. Visualize image and groundtruth segmentation:

Function that reads an image and display an image with its associated ground truth segmentation.

```
[ ]  def DataLoading(ImagesPath,GroundTruthPath,MatrixDimensions):
         GroundTruthMatrix = []
         ImagesList = listdir(ImagesPath)
         ImagesMatrix = np.arange(MatrixDimensions[0] * MatrixDimensions[1]).reshape(MatrixDimensions)
         NumOfPixels = MatrixDimensions[0] // len(ImagesList)
         Dimensions = MatrixDimensions[1]
         Count = 0
         for Image in ImagesList:
             GroundTruth = scipy.io.loadmat(GroundTruthPath+Image[:-4])
             Img = PImage.open(ImagesPath+Image)
             ImagesMatrix[NumOfPixels*Count:NumOfPixels+(NumOfPixels*Count),:] = np.array(Img).reshape(NumOfPixels,Dimensions)
             for j in range(0,GroundTruth['groundTruth'].shape[1]):
                 GroundTruthMatrix.append(GroundTruth['groundTruth'][0][j][0][0][0])
             Count += 1
         return ImagesMatrix,GroundTruthMatrix
```

```
[ ]  plt.imshow(Train[0:NumOfPixels].reshape(321,481,3));
     plt.figure();
     plt.imshow(GroundTruth[0]);
     plt.figure();
     plt.imshow(GroundTruth[1]);
     plt.figure();
     plt.imshow(GroundTruth[2]);
     plt.figure();
     plt.imshow(GroundTruth[3]);
     plt.figure();
     plt.imshow(GroundTruth[4]);
```



3. Segmentation using K-means :

k-means implementation

```
[ ]  def OwnKMeans(Data, NumOfClusters):
         NumberOfData = Data.shape[0]
         NumberOfFeatures = Data.shape[1]
         Mean = np.mean(Data, axis = 0)
         STD = np.std(Data, axis = 0)
         Centroids = np.random.randn(NumOfClusters,NumberOfFeatures)*STD + Mean
         OldCentroids = np.zeros(Centroids.shape)
         Clusters = np.zeros(NumberOfData)
         Distances = np.zeros((NumberOfData,NumOfClusters))
         for i in range(NumOfClusters):
             Distances[:,i] = np.linalg.norm(Data - Centroids[i], axis=1)
             Clusters = np.argmin(Distances, axis = 1)
             OldCentroids = deepcopy(Centroids)
             for i in range(NumOfClusters):
                 Centroids[i] = np.mean(Data[Clusters == i], axis=0)

         return Clusters
```

a. Changing the K of the K-means algorithm between {3,5,7,9,11} clusters. Producing different segmentations and saving them as colored images {R,G,B} in ColorizedImage. Every color represents a certain group (cluster) of pixels.

```
[ ]  ColorizedImages = []
     Clusters = []
     for K in range(3,12,2):
         for ImageNumber in range(5):
             Image = Train[NumOfPixels*ImageNumber:NumOfPixels+(NumOfPixels*ImageNumber)]
             #Model = OwnKMeans(Image,K)
             Model = ImplementedKMeans(n_clusters=K, random_state=0).fit(Image)
             Clusters.append(Model.labels_)
             ColorizedImages.append(Colorize(Model.labels_,Image,K))
```

```
[ ]  def Colorize(Clusters,Image,NumOfClusters):
         ColorizedImage= np.zeros(Image.shape,dtype=np.uint8)
         Colors = []
         for i in range(NumOfClusters):
             Colors.append(np.array([randint(0, 255),randint(0, 255),randint(0, 255)]))
         for i in range(Image.shape[0]):
             ColorizedImage[i] = Colors[Clusters[i]]
         return ColorizedImage
```

b)i) F-measure

- $prec_i = purity_i = \frac{1}{n_i} \max_j^k n_{ij}$
- $rec_i = \frac{n_{ij_i}}{|Tj_i|}$
- $F_i = \frac{2 * prec_i * rec_i}{prec_i + rec_i}$
- $F = \frac{1}{r} \sum_{i=1}^{r} F_i$

Precision: dividing the maximum element in each row over all elements in the same row(represents cluster)
MaximumPosition: finds the maximum element in which column
SummationOverPosition: sums the elements in the column determined by MaxPosition
Recall: divide maximum element in row over sum of the column
Applying the FMeasure rule then dividing it by ContingencyTable.shape which is the total number of elements (r)

```
[ ]  def FMeasure(Clusters,Variables):
         FMeasure = 0
         ContingencyTable =contingency_matrix(Clusters,Variables)
         for i in range(ContingencyTable.shape[0]):
             SummationOverColumns = 0
             Precision = ContingencyTable.max(axis=1)[i] / np.sum(ContingencyTable[i])
             MaximumPosition = ContingencyTable.argmax(axis=1)[i]
             for j in range(ContingencyTable.shape[0]):
              SummationOverColumns += ContingencyTable[j][MaximumPosition]
             Recall = ContingencyTable.max(axis=1)[i] / SummationOverColumns
             FMeasure += 2*Precision*Recall / (Precision+Recall)
         return FMeasure/ContingencyTable.shape[0]
```

```
[ ]  K = [3,5,7,9,11]
     FMeasures = []
     Count = 0
     for j in range(0,25,5):
         for i in range(0,5):
             GroundTruthImage = GroundTruth[i].reshape(481*321)
             FMeasureValue = FMeasure(Clusters[j],GroundTruthImage)
             print("For K = ",K[Count]," and Ground Truth Image = ",i+1," F-Measure is ",FMeasureValue)
         Count +=1
         FMeasures.append(FMeasureValue)
     print("Average of the M trials = ",sum(FMeasures)/len(FMeasures))
     plt.scatter(K,FMeasures)
     plt.title('F-Measure versus K for Ground Truth Image 5');
     plt.gcf().set_size_inches(15,5);
     plt.xlabel('K');
     plt.ylabel('F-measure');
```

```
For K =  3  and Ground Truth Image =  1  F-Measure is  0.733609265856848
For K =  3  and Ground Truth Image =  2  F-Measure is  0.7295116629968853
For K =  3  and Ground Truth Image =  3  F-Measure is  0.7302320535106962
For K =  3  and Ground Truth Image =  4  F-Measure is  0.7497413292857241
For K =  3  and Ground Truth Image =  5  F-Measure is  0.7275946767018934
For K =  5  and Ground Truth Image =  1  F-Measure is  0.4912085358951834
For K =  5  and Ground Truth Image =  2  F-Measure is  0.4887564511051098
For K =  5  and Ground Truth Image =  3  F-Measure is  0.4891618523419341
For K =  5  and Ground Truth Image =  4  F-Measure is  0.5001493558707896
For K =  5  and Ground Truth Image =  5  F-Measure is  0.48757425890930106
For K =  7  and Ground Truth Image =  1  F-Measure is  0.4437953671489502
For K =  7  and Ground Truth Image =  2  F-Measure is  0.4339851856086162
For K =  7  and Ground Truth Image =  3  F-Measure is  0.43569961244645333
For K =  7  and Ground Truth Image =  4  F-Measure is  0.37503069922219556
For K =  7  and Ground Truth Image =  5  F-Measure is  0.4309020247250897
For K =  9  and Ground Truth Image =  1  F-Measure is  0.35863754662925157
For K =  9  and Ground Truth Image =  2  F-Measure is  0.34976384523644577
For K =  9  and Ground Truth Image =  3  F-Measure is  0.34491786390119633
For K =  9  and Ground Truth Image =  4  F-Measure is  0.30899422568959384
For K =  9  and Ground Truth Image =  5  F-Measure is  0.3477837148026481
For K =  11  and Ground Truth Image =  1  F-Measure is  0.34973526200935334
For K =  11  and Ground Truth Image =  2  F-Measure is  0.34142771854398285
For K =  11  and Ground Truth Image =  3  F-Measure is  0.3412101629572837
For K =  11  and Ground Truth Image =  4  F-Measure is  0.25993233981646496
For K =  11  and Ground Truth Image =  5  F-Measure is  0.34028302502015095
Average of the M trials =  0.46682754003181665
```

b)ii) Conditional Entropy

$$H(\mathcal{T}|C_i) = -\sum_{j=1}^{k}\left(\frac{n_{ij}}{n_i}\right)\log\left(\frac{n_{ij}}{n_i}\right)$$

$$H(\mathcal{T}|\mathcal{C}) = \sum_{i=1}^{r}\frac{n_i}{n}H(\mathcal{T}|C_i)$$

Created if condition for ContingencyTable so it skip if nij = 0

```
def ConditionalEntropy(Clusters,Variables):
    ContingencyTable = contingency_matrix(Clusters, Variables)
    ConditionalEntropies = []
    for i in range(ContingencyTable.shape[0]):
        ConditionalEntropy = 0
        NumberOfElementsInCluster = np.sum(ContingencyTable[i])
        for j in range(ContingencyTable.shape[1]):
            if(ContingencyTable[i][j] != 0): ConditionalEntropy -= (ContingencyTable[i][j]/NumberOfElementsInCluster) * (np.log2(ContingencyTable[i][j]/NumberOfElementsInCluster))
        ConditionalEntropies.append(ConditionalEntropy*NumberOfElementsInCluster)
    return sum(ConditionalEntropies)/np.sum(ContingencyTable)
```

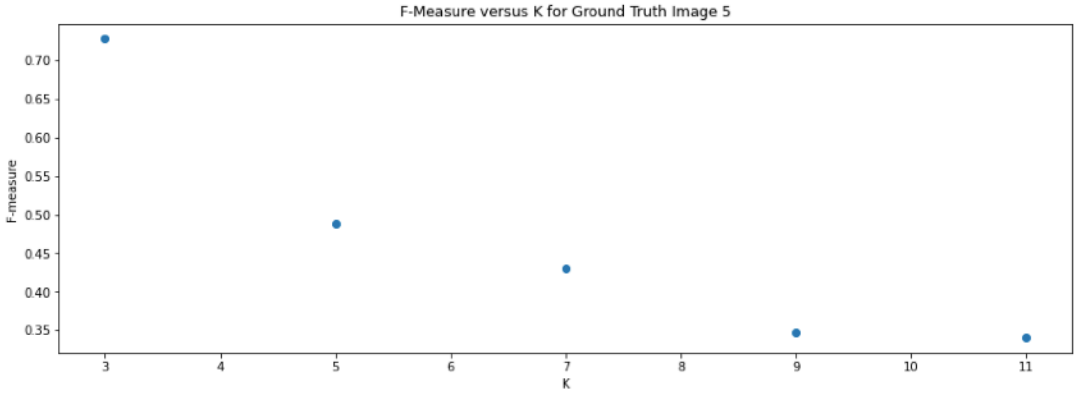
```
[ ] K = [3,5,7,9,11]
    ConditionalEntropies = []
    Count=0
    for j in range(0,25,5):
        for i in range(0,5):
            GroundTruthImage = GroundTruth[i].reshape(481*321)
            ConditionalEntropyValue = ConditionalEntropy(Clusters[j],GroundTruthImage)
            print("For K = ",K[Count]," and Ground Truth Image = ",i+1," Conditional Entropy is ",ConditionalEntropyValue)
        Count +=1
        ConditionalEntropies.append(ConditionalEntropyValue)
    print("Average of the M trials = ",sum(ConditionalEntropies)/len(ConditionalEntropies))
    plt.scatter(K,ConditionalEntropies)
    plt.title('Conditional Entropy versus K for Ground Truth Image 5');
    plt.gcf().set_size_inches(15,5);
    plt.xlabel('K');
    plt.ylabel('Conditional Entropy');
```

```
For K =  3  and Ground Truth Image =  1  Conditional Entropy is  0.10152969371736463
For K =  3  and Ground Truth Image =  2  Conditional Entropy is  0.10779588196287074
For K =  3  and Ground Truth Image =  3  Conditional Entropy is  0.10324334469251822
For K =  3  and Ground Truth Image =  4  Conditional Entropy is  0.06480372543451053
For K =  3  and Ground Truth Image =  5  Conditional Entropy is  0.11565521755228422
For K =  5  and Ground Truth Image =  1  Conditional Entropy is  0.09849018683546802
For K =  5  and Ground Truth Image =  2  Conditional Entropy is  0.10470293857867158
For K =  5  and Ground Truth Image =  3  Conditional Entropy is  0.10017939207788484
For K =  5  and Ground Truth Image =  4  Conditional Entropy is  0.062107291192656726
For K =  5  and Ground Truth Image =  5  Conditional Entropy is  0.11198678655047652
For K =  7  and Ground Truth Image =  1  Conditional Entropy is  0.08658652772909427
For K =  7  and Ground Truth Image =  2  Conditional Entropy is  0.0943329422294311
For K =  7  and Ground Truth Image =  3  Conditional Entropy is  0.08936518644072566
For K =  7  and Ground Truth Image =  4  Conditional Entropy is  0.051666739996496944
For K =  7  and Ground Truth Image =  5  Conditional Entropy is  0.10219320314991216
For K =  9  and Ground Truth Image =  1  Conditional Entropy is  0.07394001624664394
For K =  9  and Ground Truth Image =  2  Conditional Entropy is  0.0827527282300966
For K =  9  and Ground Truth Image =  3  Conditional Entropy is  0.07861393835166973
For K =  9  and Ground Truth Image =  4  Conditional Entropy is  0.04079864010489978
For K =  9  and Ground Truth Image =  5  Conditional Entropy is  0.0905134569039672
For K =  11  and Ground Truth Image =  1  Conditional Entropy is  0.06558941674308778
For K =  11  and Ground Truth Image =  2  Conditional Entropy is  0.07405629508892825
For K =  11  and Ground Truth Image =  3  Conditional Entropy is  0.06931670151042064
For K =  11  and Ground Truth Image =  4  Conditional Entropy is  0.03476522903902387
For K =  11  and Ground Truth Image =  5  Conditional Entropy is  0.08203258984609278
Average of the M trials =  0.10047625080054656
```

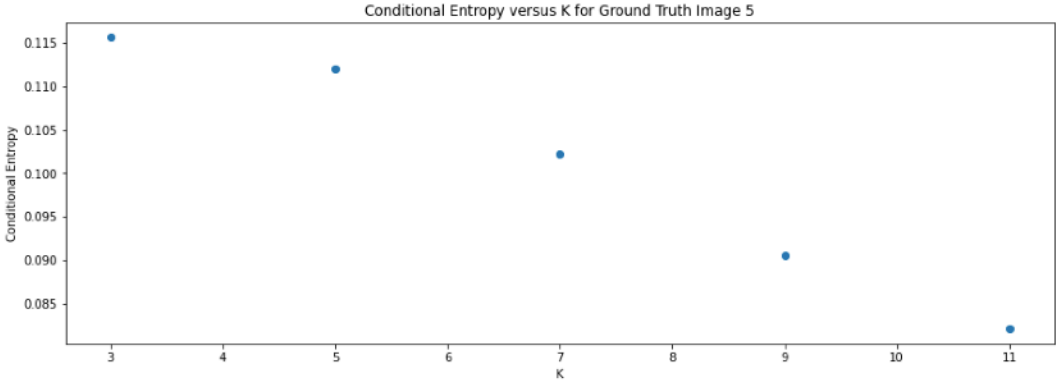
Comment on F-measure & Entropy:

Both, F-measure and Conditional Entropy, decreases as the K for GroundTruth increases.

In F-measure it started from 0.7 and decreased till 0.3 due to oversampling



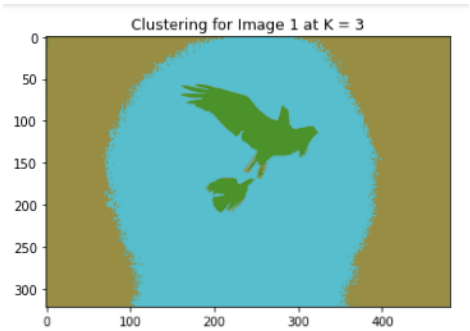
Conditional Entropy, measure of uncertainty, started 0.1 and decreased till 0.03

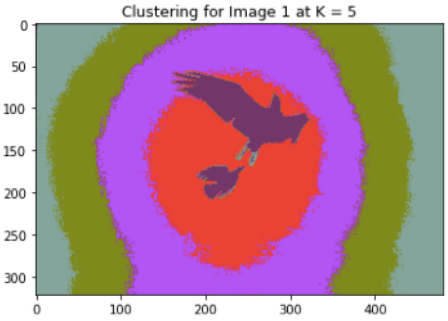


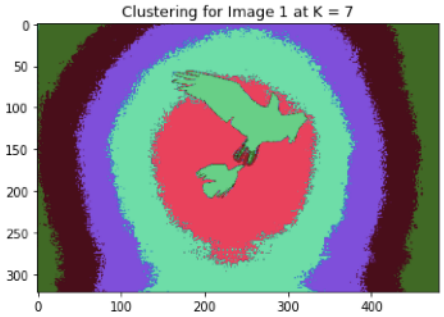
4. Big Picture

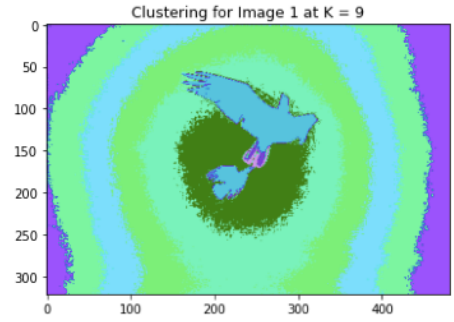
a. Select the first five images and display their corresponding groundtruth against segmentation results using K-means at K=5.
   Comment on the results
   Disparity between results, when k increases over-clustering occurs

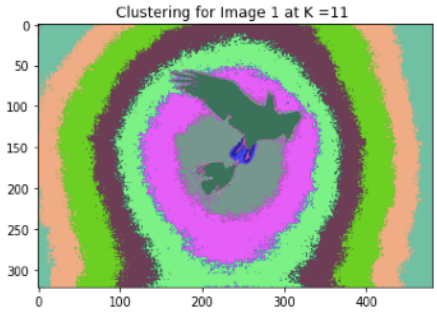
Image 1:

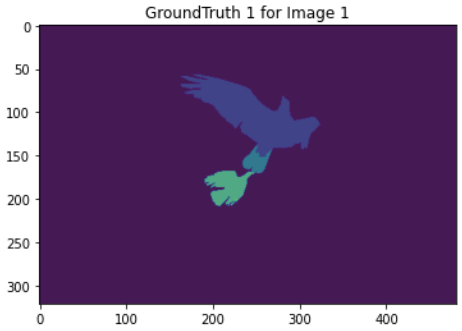


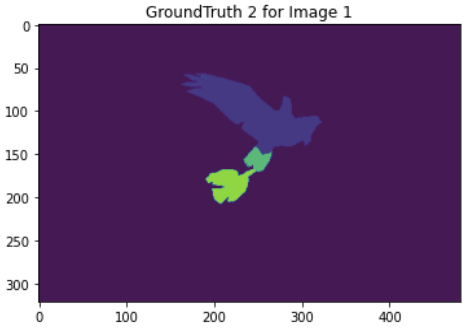


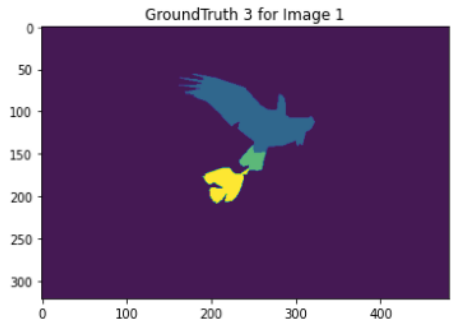


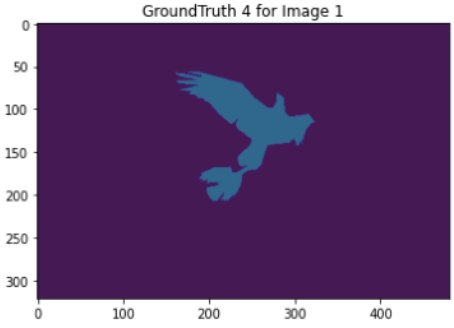












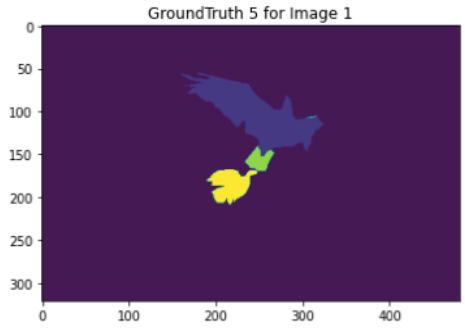
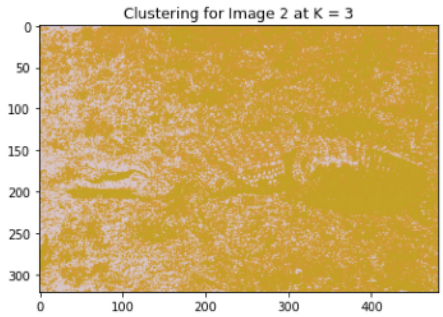
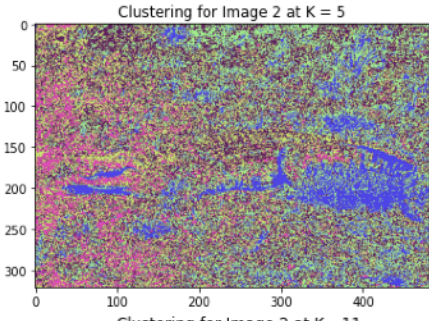
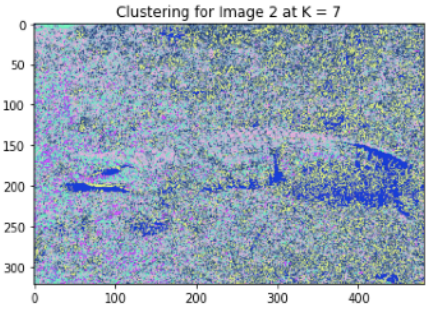
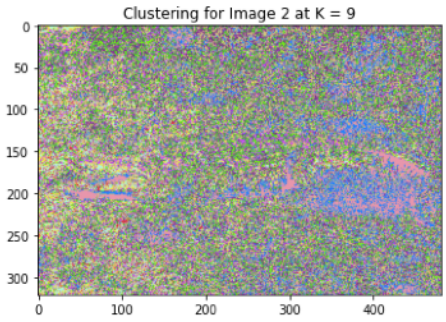


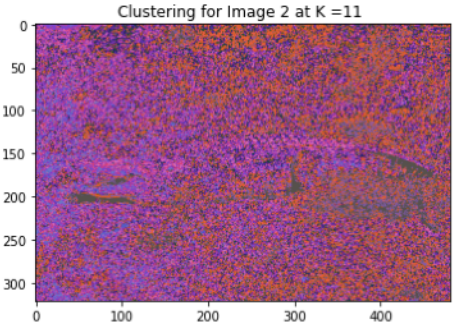
Image 2:

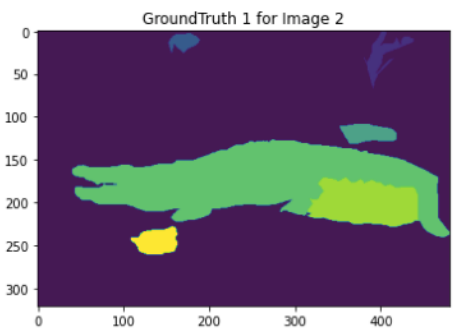

Clustering for Image 2 at K = 3

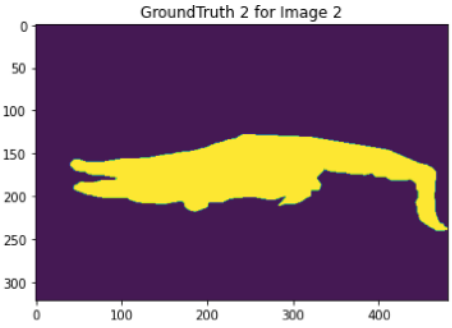

Clustering for Image 2 at K = 5

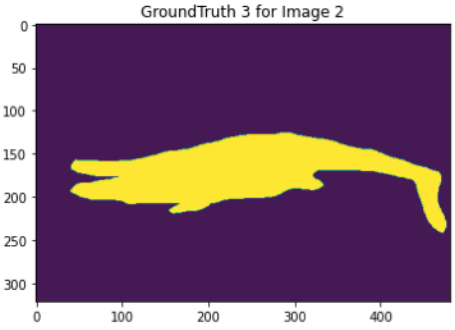

Clustering for Image 2 at K = 7

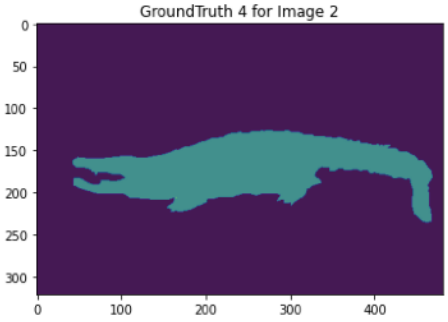

Clustering for Image 2 at K = 9


Clustering for Image 2 at K =11


GroundTruth 1 for Image 2


GroundTruth 2 for Image 2


GroundTruth 3 for Image 2


GroundTruth 4 for Image 2

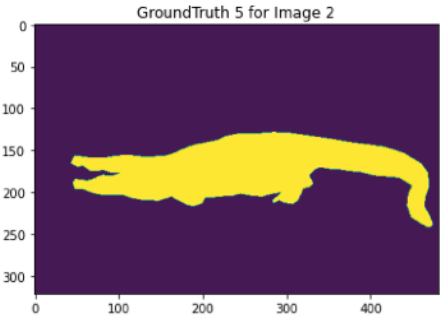
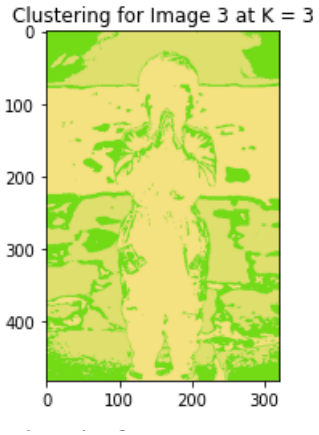
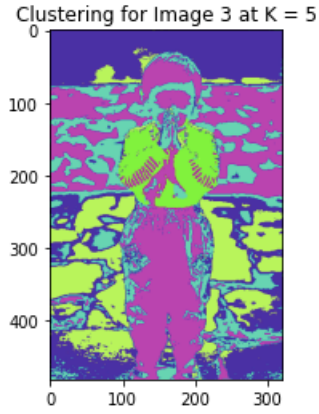
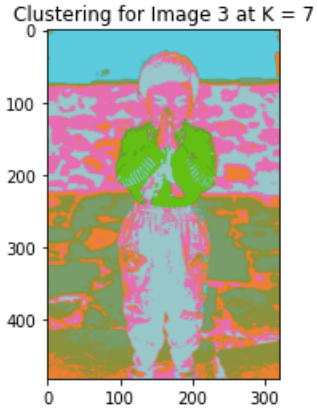
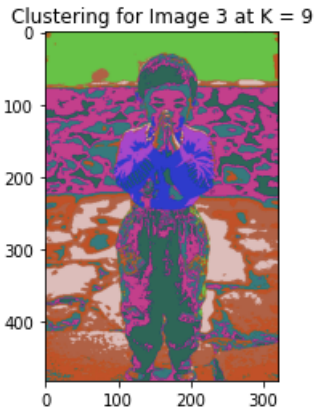

GroundTruth 5 for Image 2

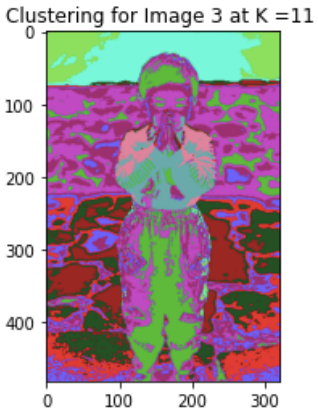
Image 3:

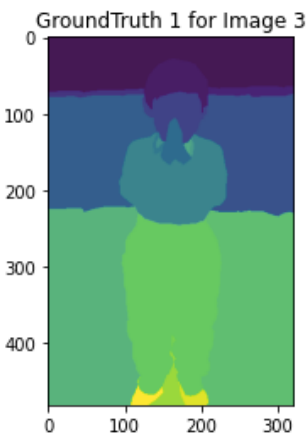

Clustering for Image 3 at K = 3

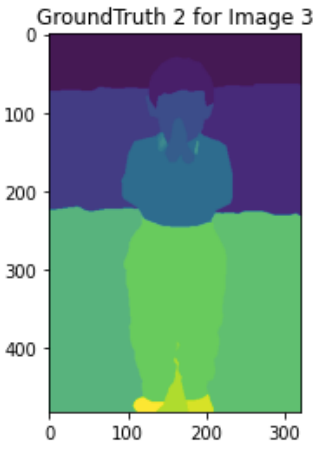

Clustering for Image 3 at K = 5

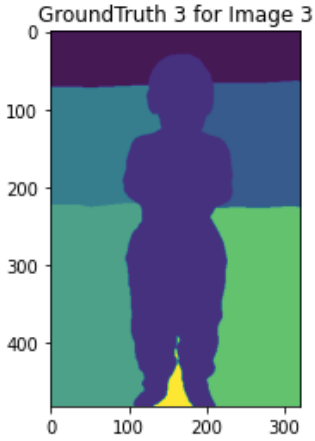

Clustering for Image 3 at K = 7

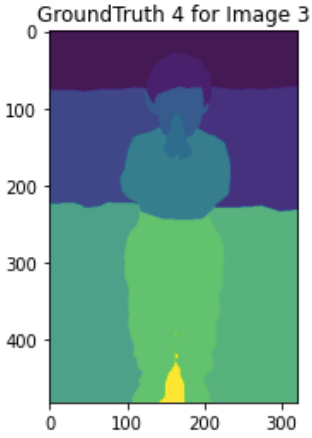

Clustering for Image 3 at K = 9


Clustering for Image 3 at K =11


GroundTruth 1 for Image 3


GroundTruth 2 for Image 3


GroundTruth 3 for Image 3


GroundTruth 4 for Image 3

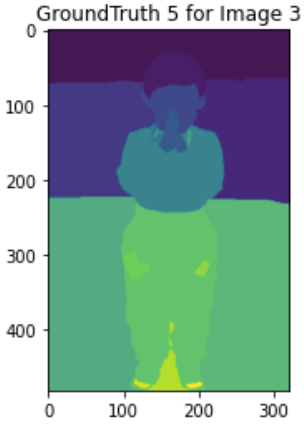
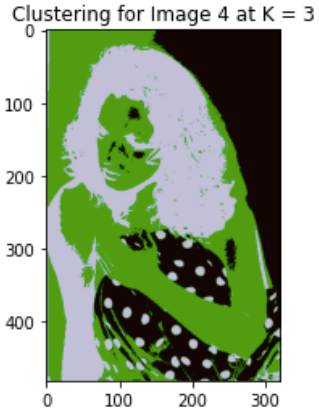
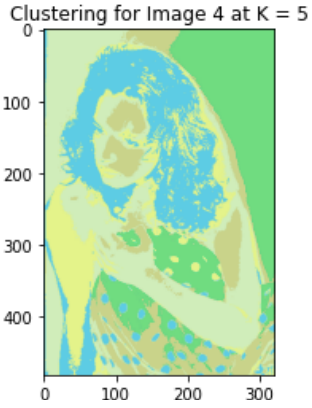
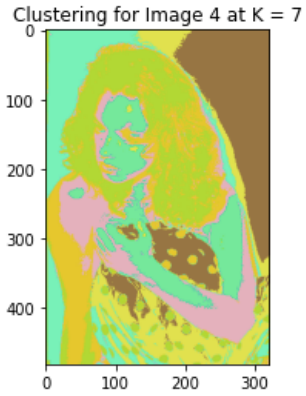
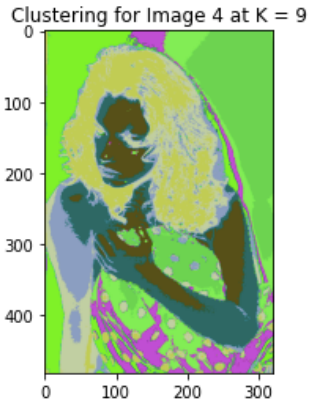

GroundTruth 5 for Image 3

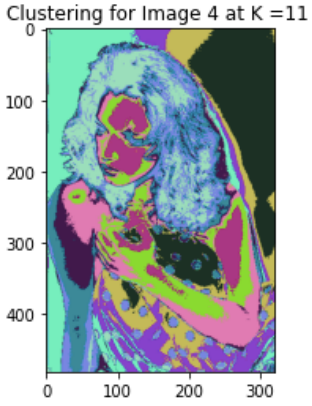
Image 4:

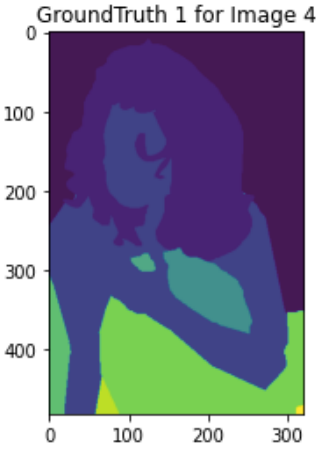

Clustering for Image 4 at K = 3

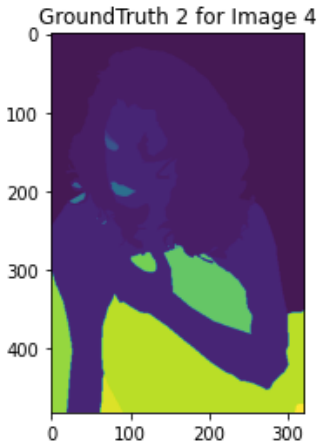

Clustering for Image 4 at K = 5

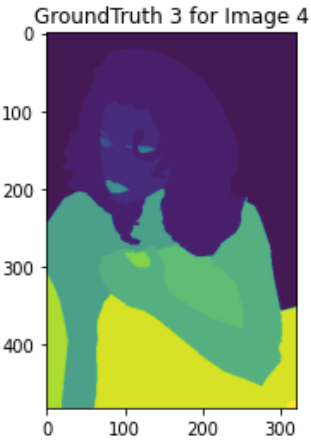

Clustering for Image 4 at K = 7

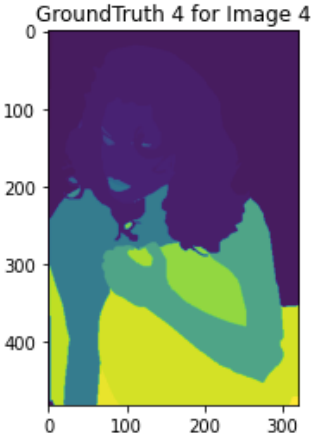

Clustering for Image 4 at K = 9


Clustering for Image 4 at K =11


GroundTruth 1 for Image 4


GroundTruth 2 for Image 4


GroundTruth 3 for Image 4


GroundTruth 4 for Image 4

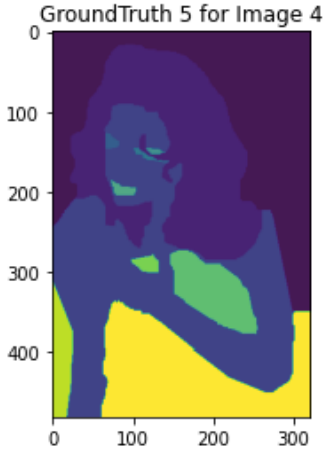
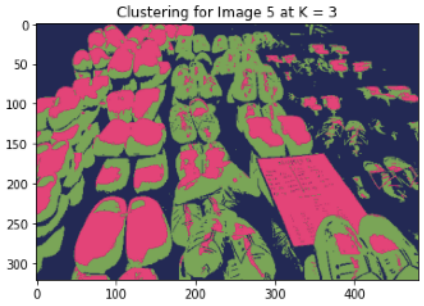
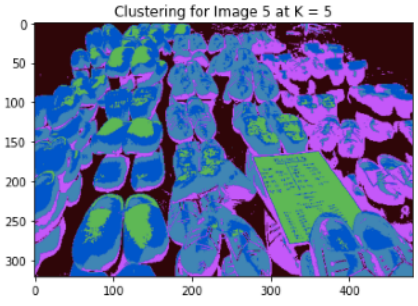
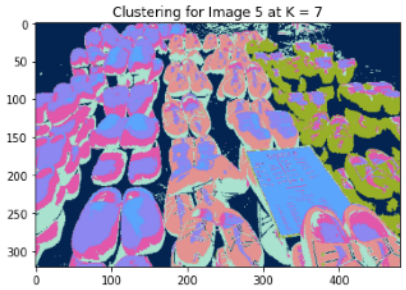
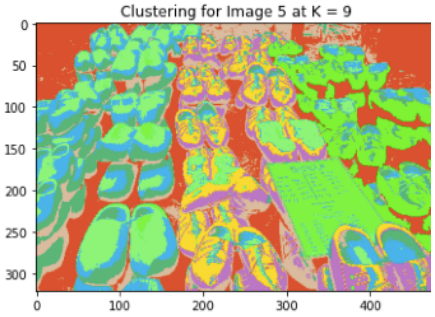

GroundTruth 5 for Image 4

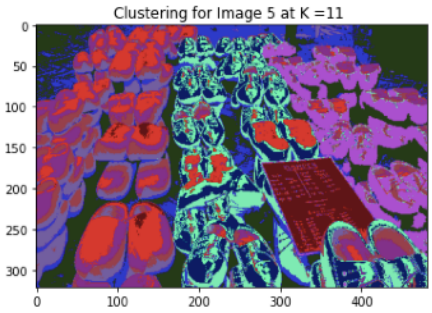
Image 5:

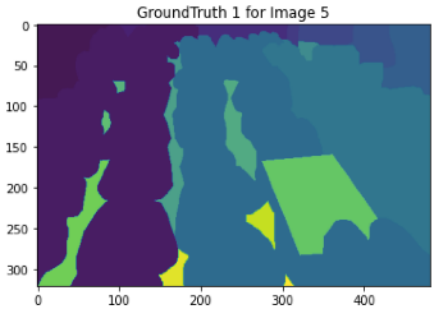

Clustering for Image 5 at K = 3

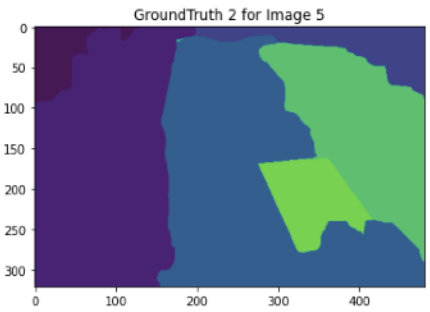

Clustering for Image 5 at K = 5

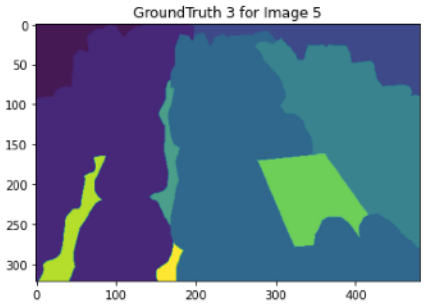

Clustering for Image 5 at K = 7

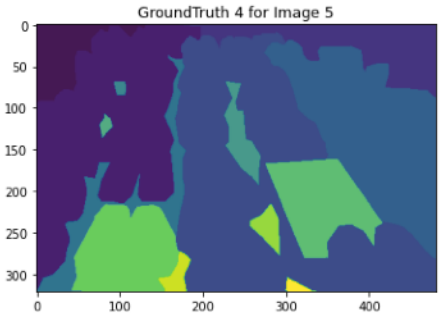

Clustering for Image 5 at K = 9

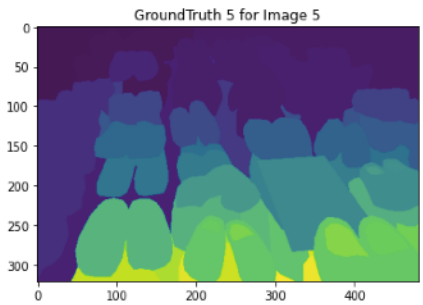

Clustering for Image 5 at K =11


GroundTruth 1 for Image 5


GroundTruth 2 for Image 5


GroundTruth 3 for Image 5


GroundTruth 4 for Image 5


GroundTruth 5 for Image 5

b. Select same five images and display their corresponding ground truth against segmentation results using Normalized-cut for the 5- NN graph, at K=5.

When running the code the first time, it didn't work hence reduced the images size by 25%.

```
[ ] TrainPath = "/content/drive/MyDrive/dataResized/images/"
    GroundTruthPath = "/content/drive/MyDrive/dataResized/ground_truth/train/"
    NumOfPixels = 120*80
    Dimensions = 3
    NumOfImages=200
    Train, GroundTruth = DataLoading(TrainPath,GroundTruthPath,(NumOfPixels*NumOfImages,Dimensions))
```

NormalisedCut algorithm

ALGORITHM 16.1. Spectral Clustering Algorithm

SPECTRAL CLUSTERING $(\mathbf{D}, k)$:

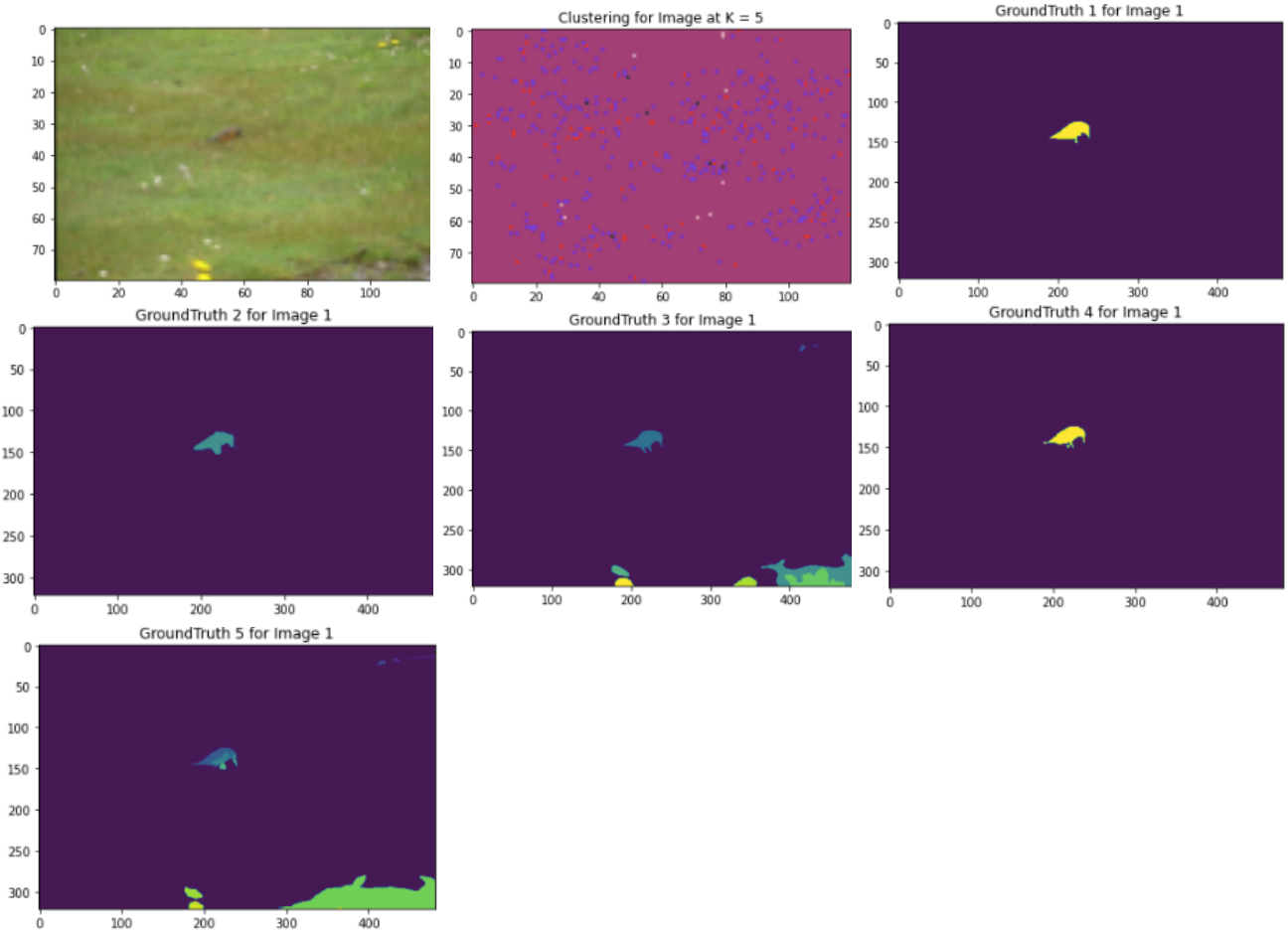
1 Compute the similarity matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$
2 **if** *ratio cut* **then** $\mathbf{B} \leftarrow \mathbf{L}$
3 **else if** *normalized cut* **then** $\mathbf{B} \leftarrow \mathbf{L}^s$ or $\mathbf{L}^a$
4 Solve $\mathbf{B}\mathbf{u}_i = \lambda_i \mathbf{u}_i$ for $i = n, \ldots, n-k+1$, where $\lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_{n-k+1}$
5 $\mathbf{U} \leftarrow (\mathbf{u}_n \quad \mathbf{u}_{n-1} \quad \cdots \quad \mathbf{u}_{n-k+1})$
6 $\mathbf{Y} \leftarrow$ normalize rows of $\mathbf{U}$ using Eq. (16.19)
7 $\mathcal{C} \leftarrow \{C_1, \ldots, C_k\}$ via K-means on $\mathbf{Y}$

```
[ ] def NormalizedCut(Data, NumOfClusters, Mode, Gamma=1.0, NumOfNeighbors=1):
        if(Mode == 'rbf'):
            similarityMatrix = rbf(Data,Data,Gamma)
        elif(Mode == 'knn'):
            nearest_neigbhor = nn(n_neighbors=NumOfNeighbors)
            nearest_neigbhor.fit(Data)
            similarityMatrix = nearest_neigbhor.kneighbors_graph(Data, mode='connectivity').toarray()

        degreeMmatrix = np.diag(np.sum(similarityMatrix, axis=1))
        laplacianMatrix = degreeMmatrix - similarityMatrix
        nomalizedAsymmertircLaplacianMatrix = np.dot(np.linalg.pinv(degreeMmatrix),laplacianMatrix)
        values,vectors = np.linalg.eig(nomalizedAsymmertircLaplacianMatrix)
        idx = np.real(values).argsort()[:NumOfClusters]
        vectors = np.real(vectors[:,idx])
        Normalization = np.linalg.norm(vectors, axis=1)
        NormalizedVectors = (vectors.T / Normalization).T
        Clusters = ImplementedKMeans(n_clusters=NumOfClusters,n_jobs=-1).fit_predict(NormalizedVectors)
        return Clusters
```

Cutting image before clustering

```
[ ] ColorizedImages = []
    Count = 2
    #Clusters=NormalizedCut(Train[NumOfPixels*Count:NumOfPixels+(NumOfPixels*Count)],5,'knn',NumOfNeighbors=5)
    Clusters=SpectralClustering(n_clusters=5,affinity='nearest_neighbors',n_neighbors=5,n_jobs=-1).fit_predict(Train[NumOfPixels*Count:NumOfPixels+(NumOfPixels*Count)])
    ColorizedImages.append(Colorize(Clusters,Train[NumOfPixels*Count:NumOfPixels+(NumOfPixels*Count)],5))
```


Clustering for Image at K = 5
GroundTruth 1 for Image 1
GroundTruth 2 for Image 1
GroundTruth 3 for Image 1
GroundTruth 4 for Image 1
GroundTruth 5 for Image 1

5. Extra

i) Suggest a way to modify the feature vector to include spatial layout.

By adding 2 more dimensions, the position of each pixel, {R,G,B,X,Y}

ii) 4a

For loop to determine j = x-axis , k = y-axis for the first 5 images

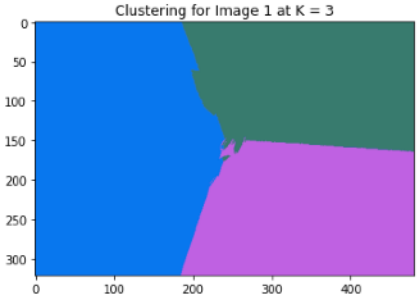
```
[ ] SpatialTrain=np.arange(NumOfPixels*NumOfImages*5).reshape(NumOfPixels*NumOfImages,5)
    SpatialPosition = []
    for i in range(5):
        for j in range(321):
            for k in range(481):
                SpatialPosition.append([j,k])
    TrainData = Train[0:NumOfPixels*5]
    SpatialTrain =np.hstack([TrainData,SpatialPosition])
```

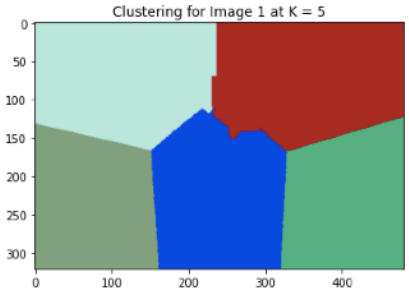
WithoutPosition removes last 2 column, as it won't matter in the colorizing process
In spatial layout, k-means take into consideration each pixel's position not only the colours {R,G,B}

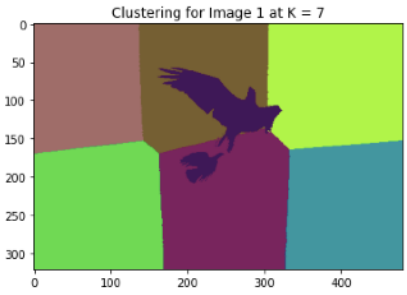
```
[ ] ColorizedImages = []
    Clusters = []
    for K in range(3,12,2):
        for Count in range(5):
            Image = SpatialTrain[NumOfPixels*Count:NumOfPixels+(NumOfPixels*Count)]
            WithoutPosition = SpatialTrain[NumOfPixels*Count:NumOfPixels+(NumOfPixels*Count),0:3]
            #Model = OwnKMeans(Image,K)
            Model = ImplementedKMeans(n_clusters=K, random_state=0).fit(Image)
            Clusters.append(Model.labels_)
            ColorizedImages.append(ColorizeSpatial(Model.labels_,WithoutPosition,K))
```

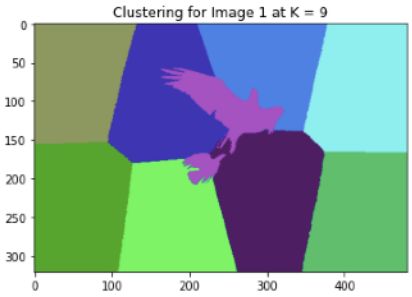
```
[ ] def ColorizeSpatial(Clusters,Image,NumOfClusters):
        ColorizedImages= np.zeros(Image.shape,dtype=np.uint8)
        Colors = []
        for i in range(NumOfClusters):
            Colors.append(np.array([randint(0, 255),randint(0, 255),randint(0, 255)]))
        for i in range(Image.shape[0]):
            ColorizedImages[i] = Colors[Clusters[i]]
        return ColorizedImages
```

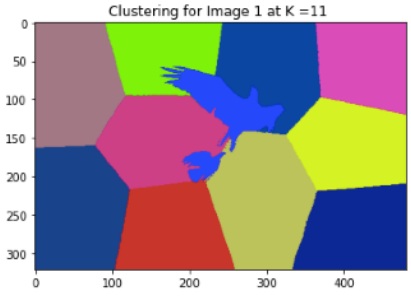
Image 1:

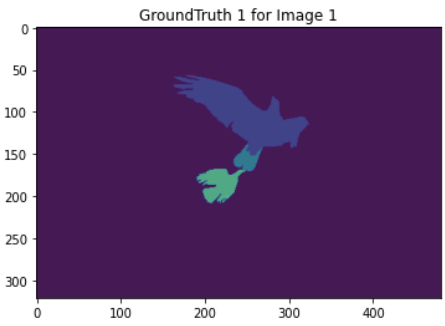

Clustering for Image 1 at K = 3

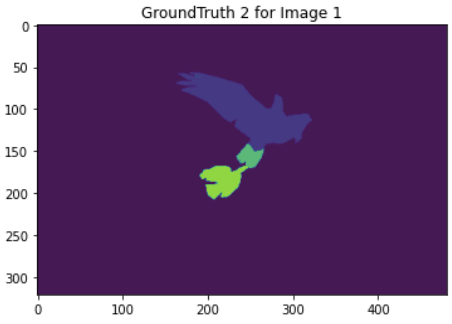

Clustering for Image 1 at K = 5

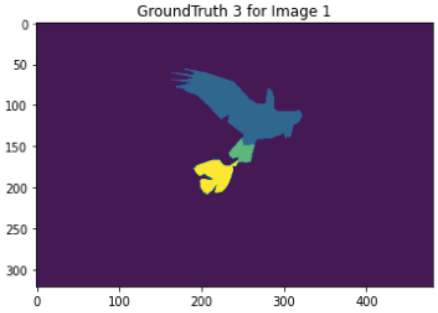

Clustering for Image 1 at K = 7

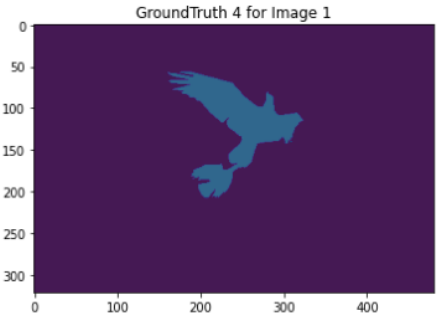

Clustering for Image 1 at K = 9


Clustering for Image 1 at K =11


GroundTruth 1 for Image 1


GroundTruth 2 for Image 1


GroundTruth 3 for Image 1


GroundTruth 4 for Image 1

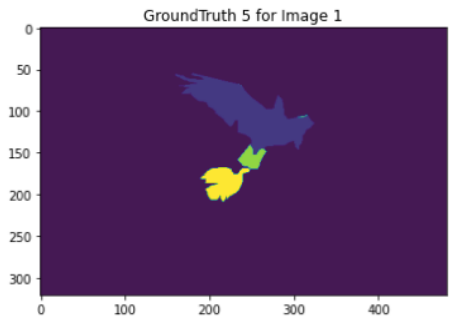
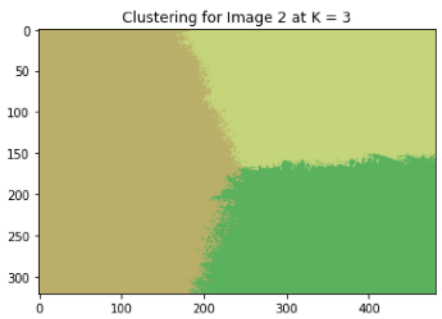
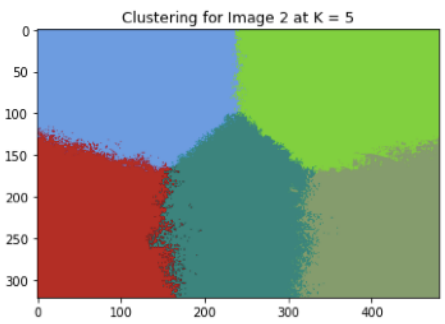
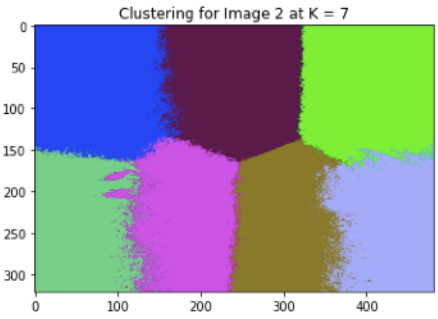
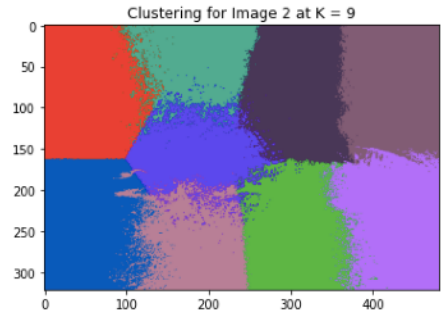

GroundTruth 5 for Image 1

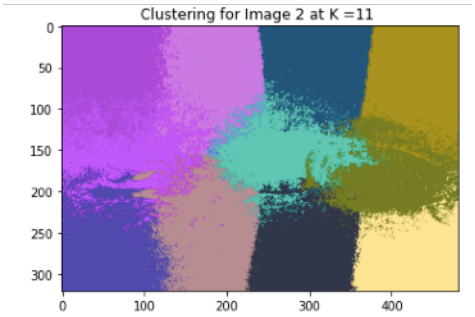
Image 2:

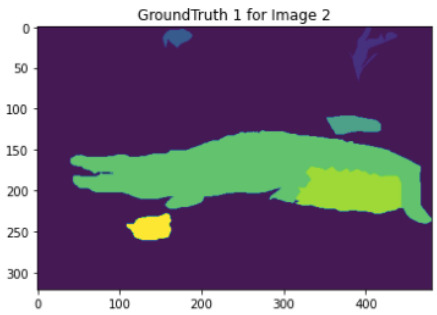

Clustering for Image 2 at K = 3

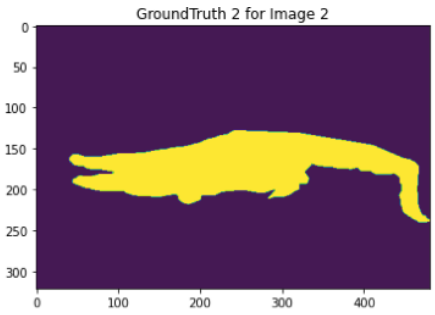

Clustering for Image 2 at K = 5

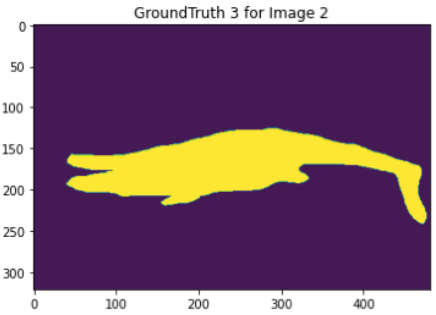

Clustering for Image 2 at K = 7

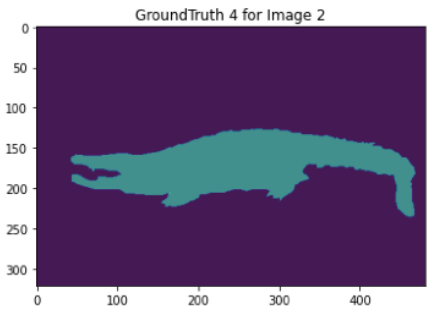

Clustering for Image 2 at K = 9


Clustering for Image 2 at K =11


GroundTruth 1 for Image 2


GroundTruth 2 for Image 2


GroundTruth 3 for Image 2


GroundTruth 4 for Image 2

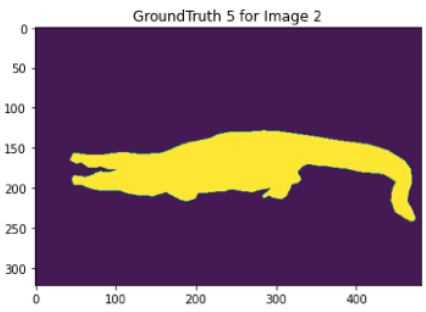
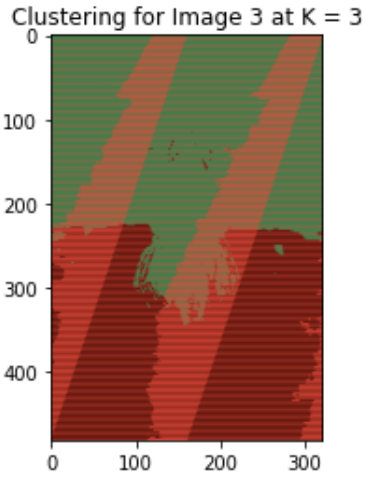
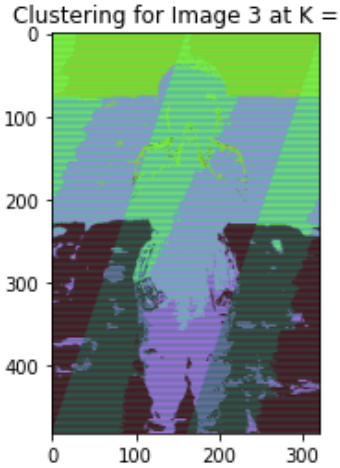
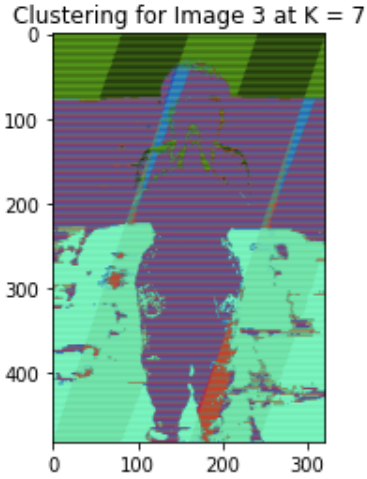
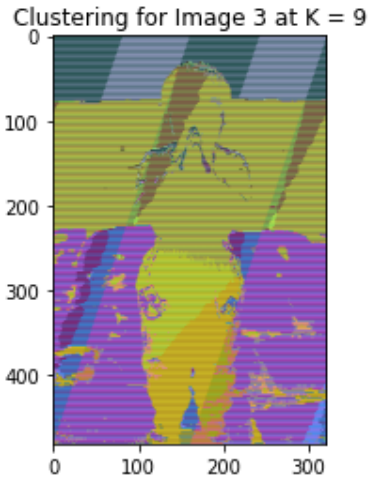

GroundTruth 5 for Image 2

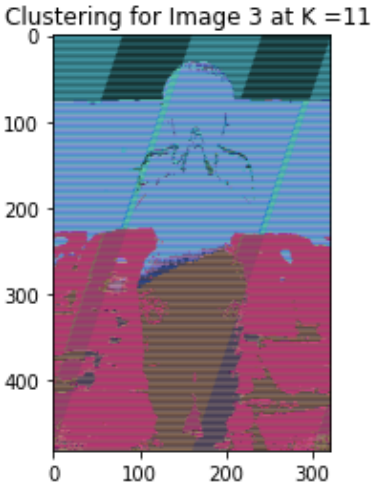
Image 3:

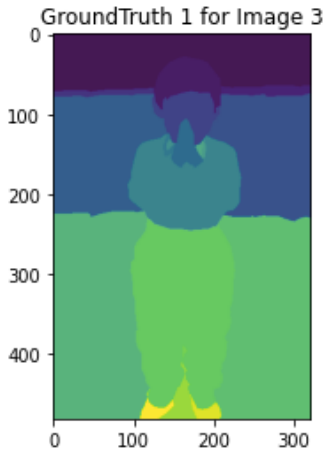

Clustering for Image 3 at K = 3

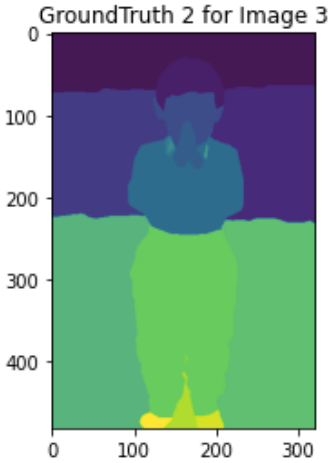

Clustering for Image 3 at K = 5

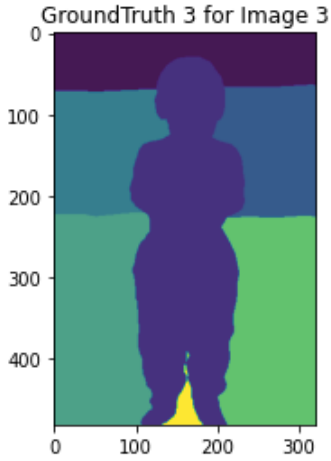

Clustering for Image 3 at K = 7

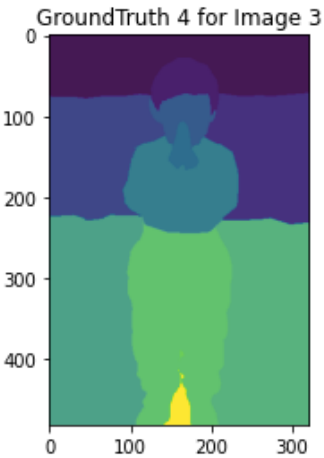

Clustering for Image 3 at K = 9


Clustering for Image 3 at K =11


GroundTruth 1 for Image 3


GroundTruth 2 for Image 3


GroundTruth 3 for Image 3


GroundTruth 4 for Image 3

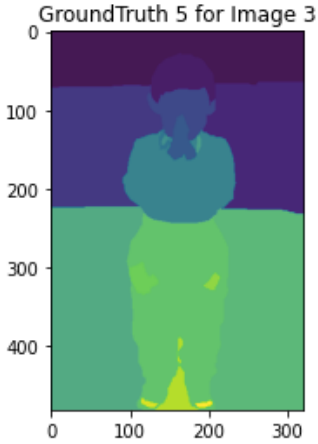

GroundTruth 5 for Image 3

Image 4:

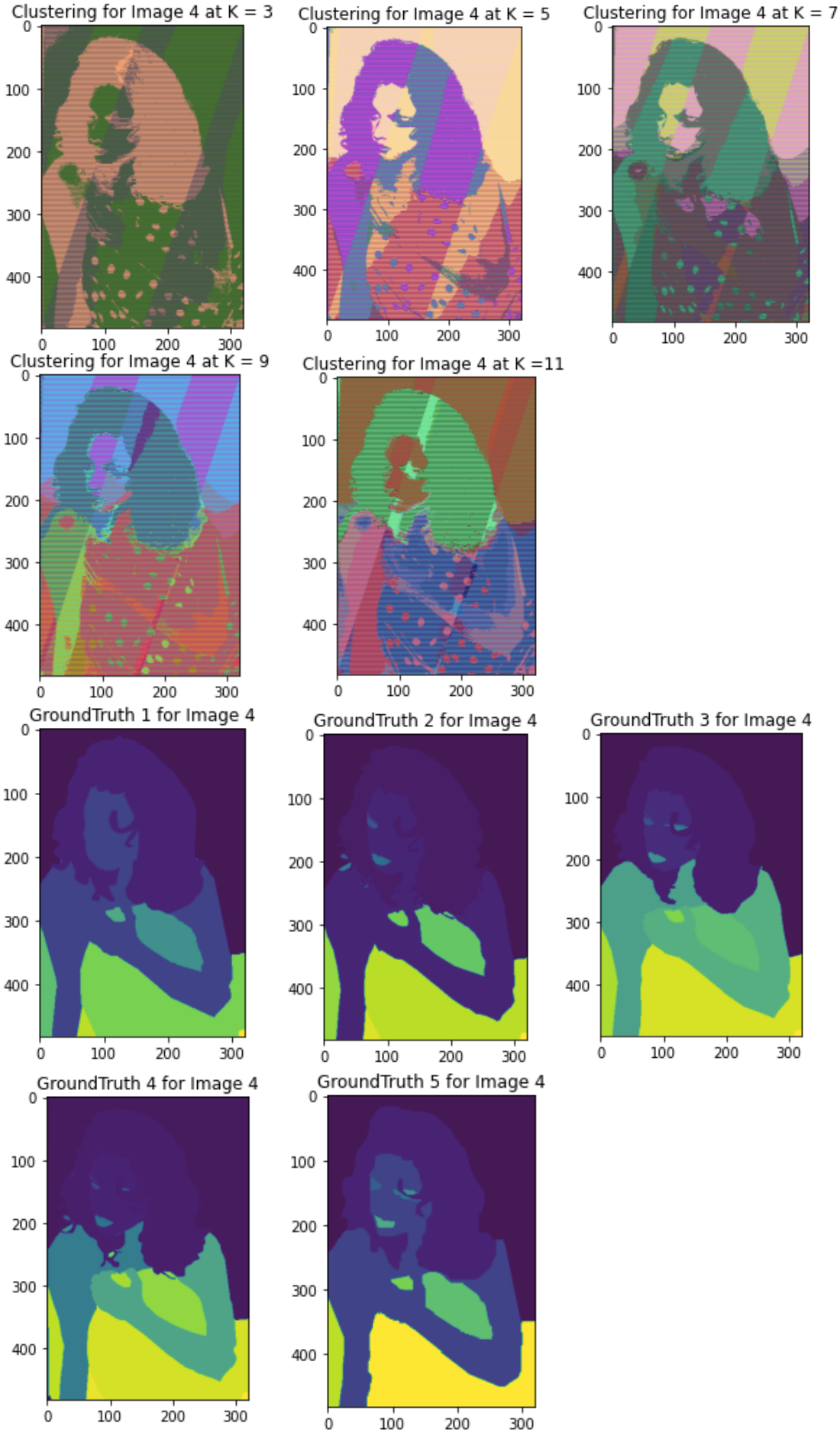
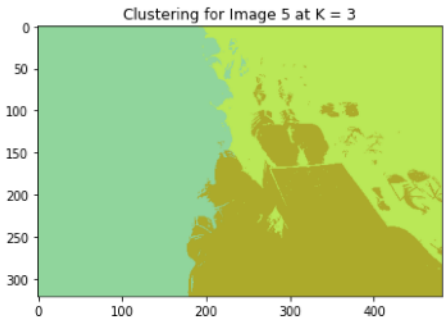
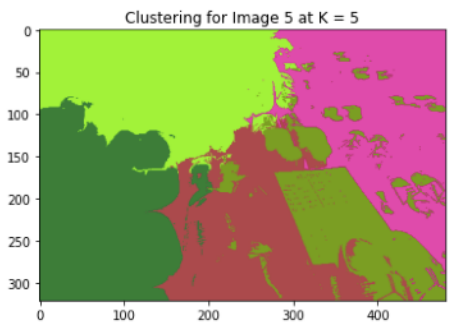
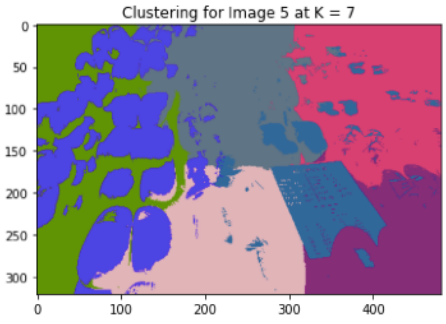
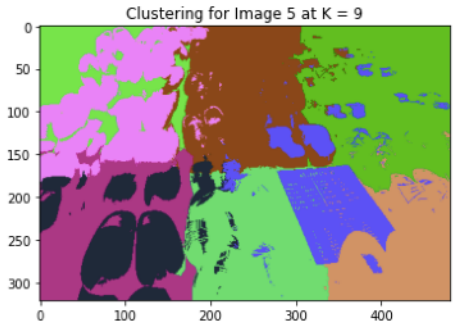

Clustering for Image 4 at K = 3
Clustering for Image 4 at K = 5
Clustering for Image 4 at K = 7
Clustering for Image 4 at K = 9
Clustering for Image 4 at K =11
GroundTruth 1 for Image 4
GroundTruth 2 for Image 4
GroundTruth 3 for Image 4
GroundTruth 4 for Image 4
GroundTruth 5 for Image 4

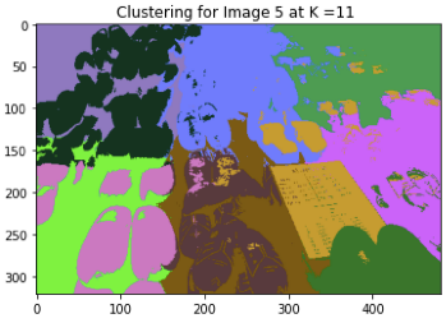
Image 5:

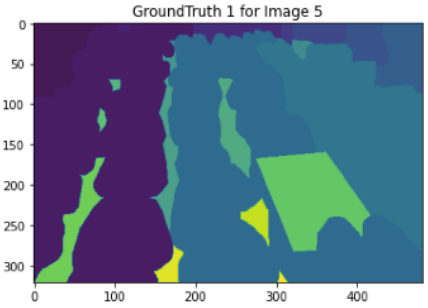

Clustering for Image 5 at K = 3

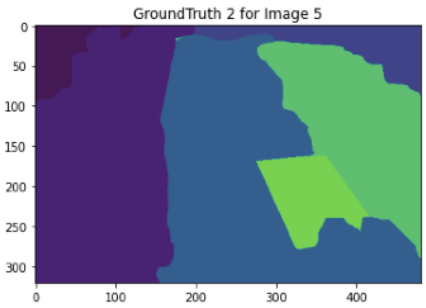

Clustering for Image 5 at K = 5

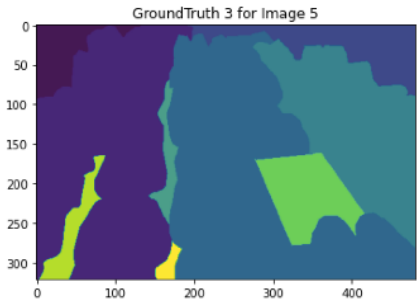

Clustering for Image 5 at K = 7

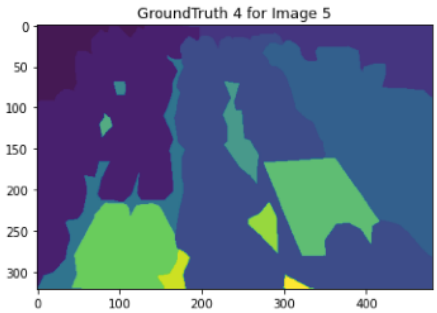

Clustering for Image 5 at K = 9

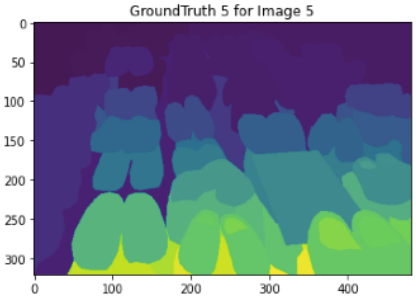

Clustering for Image 5 at K =11


GroundTruth 1 for Image 5


GroundTruth 2 for Image 5


GroundTruth 3 for Image 5


GroundTruth 4 for Image 5


GroundTruth 5 for Image 5