REPORT ON

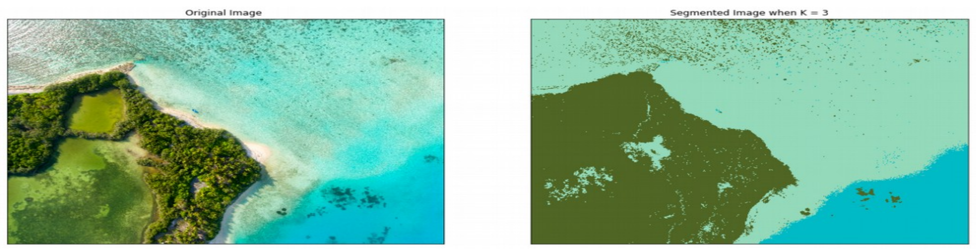
IMAGE SEGMENTATION

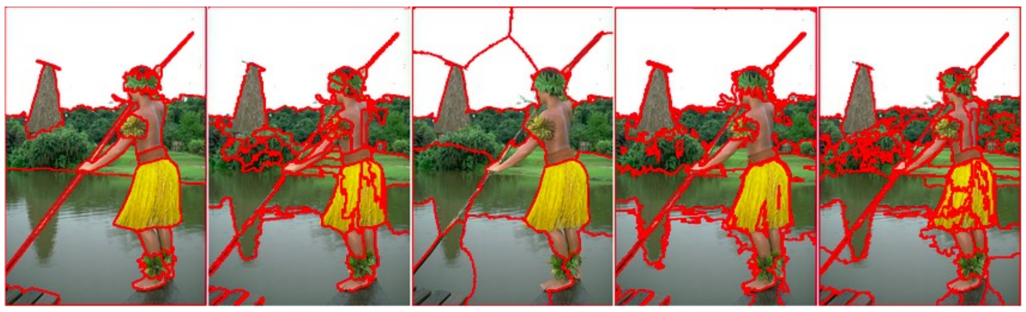
SUBARNA KANTI SAMANTA : MT2019523

UNDER GUIDANCE OF:
PROF. NEELAM SINHA

Theory:

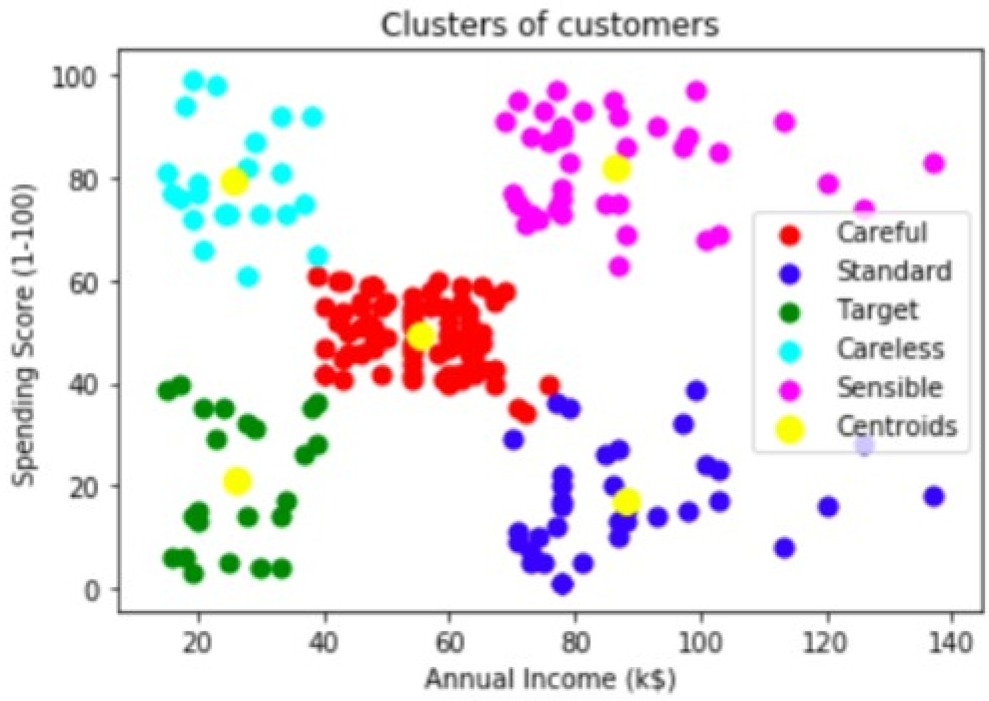
- Image Segmentation:
    - In digital image processing and computer vision, image segmentation is the process of partitioning a digital image into multiple segments (set of pixels, also known as image objects).
    - The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.
    - Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images.
    - More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.
    - The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image



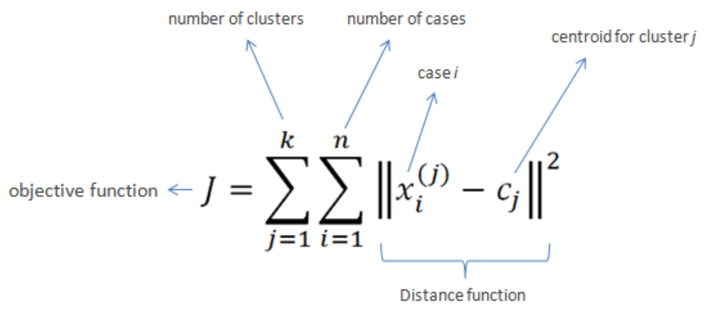


-
    - Applications:
        - CBIR(Content Based Image Retrieval)
        - Machine Vision
        - Medical imaging
        - Object detection
        - Recognition tasks
        - Traffic control systems
        - Video surveillance
    - There are many techniques of Image Segmentation, we are going to discuss and compare here two methods:
        - k-means
        - Normalized cut

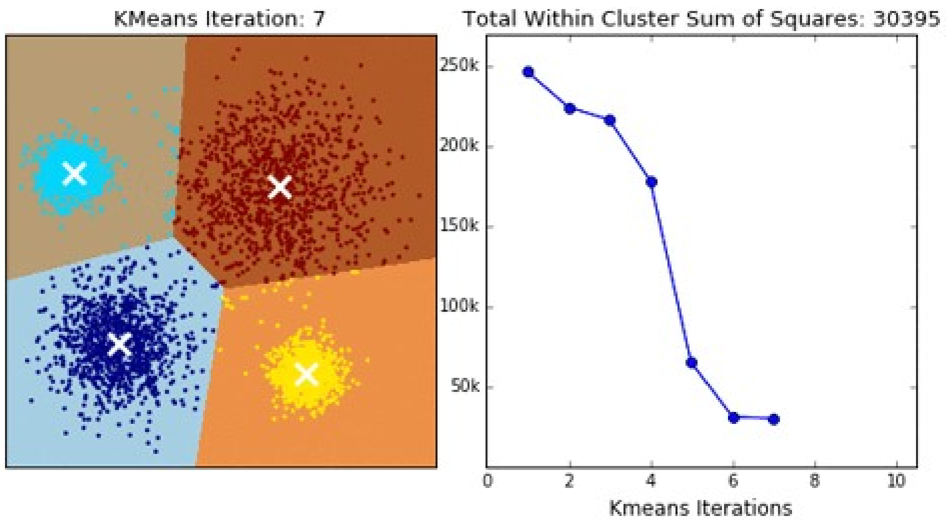
- K-means:
  - ***K***-Means clustering algorithm is an unsupervised algorithm and it is used to segment the interest area from the background. It clusters, or partitions the given data into K-clusters or parts based on the K-centroids.
  - The algorithm is used when we have unlabeled data(i.e. data without defined categories or groups). The goal is to find certain groups based on some kind of similarity in the data with the number of groups represented by K.



  - In the above figure, Customers of a shopping mall have been grouped into 5 clusters based on their income and spending score. Yellow dots represent the Centroid of each cluster.
  - The objective of K-Means clustering is to minimize the sum of squared distances between all points and the cluster center.


$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \underbrace{\left\| x_i^{(j)} - c_j \right\|^2}_{\text{Distance function}}$$

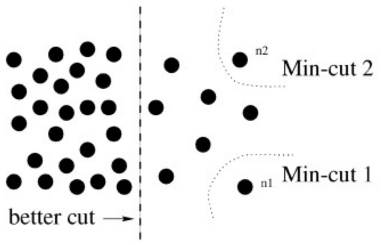
  - **Steps in K-Means algorithm:**
    - Choose the number of clusters K.
    - Select at random K points, the centroids
    - Assign each data point to the closest centroid → that forms K clusters.
    - Compute and place the new centroid of each cluster.
    - Reassign each data point to the new closest centroid. If any reassignment . took place, go to step 4, otherwise, the model is ready.



- Normalized Cut:
  - The segmentation approach given by Jianbo Shi and Jitendra Malik is Normalized cut approach which is basically graph partitioning approach also known as Shi and Malik approach
  - They propose a novel approach for solving the perceptual grouping problem in vision.
  - Where kmeans is basically deal with focusing on local features and their consistencies in the image data, this approach aims at extracting the global impression of an image
  - The normalized cut criterion measures both the total dissimilarity between the different groups as well as the total similarity within the groups.
  - Here image is treated as graph with each pixel as single vertex of graph and each vertex is joined by edges with some weights which show similarity and dissimilarity between them, which is the basic requirement of segmentation
  - Let us go through Graph theory and how graph partitioning works
- Graph Theory:
  - A graph G = (V, E)   where V represents vertices and E represent edges
  - Graph G can be partitioned into two disjoint sets, A,B, A ∪ B = V , A ∩ B = ϕ , by simply removing edges connecting the two parts.
  - The degree of dissimilarity between these two pieces can be computed as total weight of the edges that have been removed.
  - In graph theoretic language, it is called the cut:

$$cut(A,B) = \sum_{u \in A, v \in B} w(u,v).$$

  - Our aim is to minimize this cut



- But cut method lead to biasing problem, it totally seperate out single nodes instead prefering alike vertices
- To overcome this normalized cut approach is proposed which computes the cut cost as a fraction of the total edge connections to all the nodes in the graph.

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

- where $assoc(A,V) = \sum_{u \in A, t \in V} w(u,t)$ is the total connection from nodes in A to all nodes in the graph and assoc(B,V) is similarly defined.
- So now our aim is to optimize Ncut
- Solution of above equation is related to eigen vectors of Graph Laplacian which is also known as **Spectral Clustering**
- **Graph Laplacian:**
  - L = D – W , where D is degree matrix, W is weight matrix and L is our Laplacian matrix
  - The diagonal elements of laplacian matrix is our weights associated to each node or pixel in case of image
  - Properties:
    - Symmetric
    - Real valued non negative eigen values and real valued orthogonal eigen vectors

- Grouping Algorithm as discussed in paper:
  - Given an image or image sequence, set up a weighted graph G = (V, E) and set the weight on the edge connecting two nodes to be a measure of the similarity between the two nodes.
  - Solve (D – W)x = λDx for eigenvectors with the smallest eigenvalues.
  - Use the eigenvector with the second smallest eigenvalue to bipartition the graph.
  - Decide if the current partition should be subdivided and recursively repartition the segmented parts if necessary.

Fig. 2. A gray level image of a baseball game.

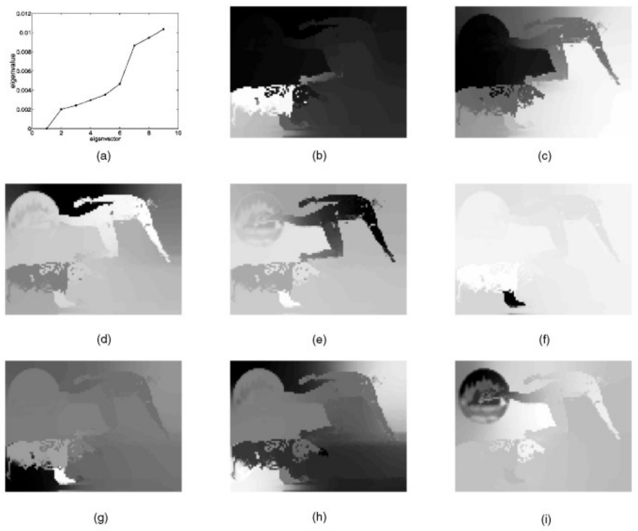


Fig. 3. Subplot (a) plots the smallest eigenvectors of the generalized eigenvalue system (11). Subplots (b)-(i) show the eigenvectors corresponding the second smallest to the ninth smallest eigenvalues of the system. The eigenvectors are reshaped to be the size of the image.

Sklearn.cluster:

- MiniBatchKMeans:
  - Here I have used MiniBatchKmeans of 200 batch size each to save time of computation
- SpectralClustering:
  - Apply clustering to a projection of the normalized Laplacian.
  - In practice Spectral Clustering is very useful when the structure of the individual clusters is highly non-convex or more generally when a measure of the center and spread of the cluster is not a suitable description of the complete cluster. For instance when clusters are nested circles on the 2D plane.
  - If affinity is the adjacency matrix of a graph, this method can be used to find normalized graph cuts.
  - When calling `fit`, an affinity matrix is constructed using either kernel function such the Gaussian (aka RBF) kernel of the euclidean distanced `d(X, X)`:
    - `np.exp(-gamma * d(X,X) ** 2)`
  - or a k-nearest neighbors connectivity matrix.
  - Parameter “affinity” :
    - used to construct affinity matrix, here used two methods :
      - ‘nearest_neighbors’ : construct the affinity matrix by computing a graph of nearest neighbors.
      - ‘rbf’ : construct the affinity matrix using a radial basis function (RBF) kernel.
    - We are going to construct different affinity matrix by varying our parameters and see how it effect clustering

Goal:

Image segmentation on Berkelay Segmentation Benchmark train set

Dataset:

Worked on 200 images with there ground truth.

Libraries imported:

- os: for directory operations
- skimage.io.imread: for reading images
- scipy.io.loadmat: for loading matlab images
- numpy: for matrix operations
- sklearn.cluster : MinibatchKmeans, SpectralClustering
- cv2: OpenCv
- sklearn.metrics.cluster.v_measure_score:
  - The V-measure is the harmonic mean between homogeneity and completeness, between our clustures formed and ground truths
  - A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class.
  - A clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster.

- Matplotlib.pyplot : for graph plotting

Self made libraries involved:

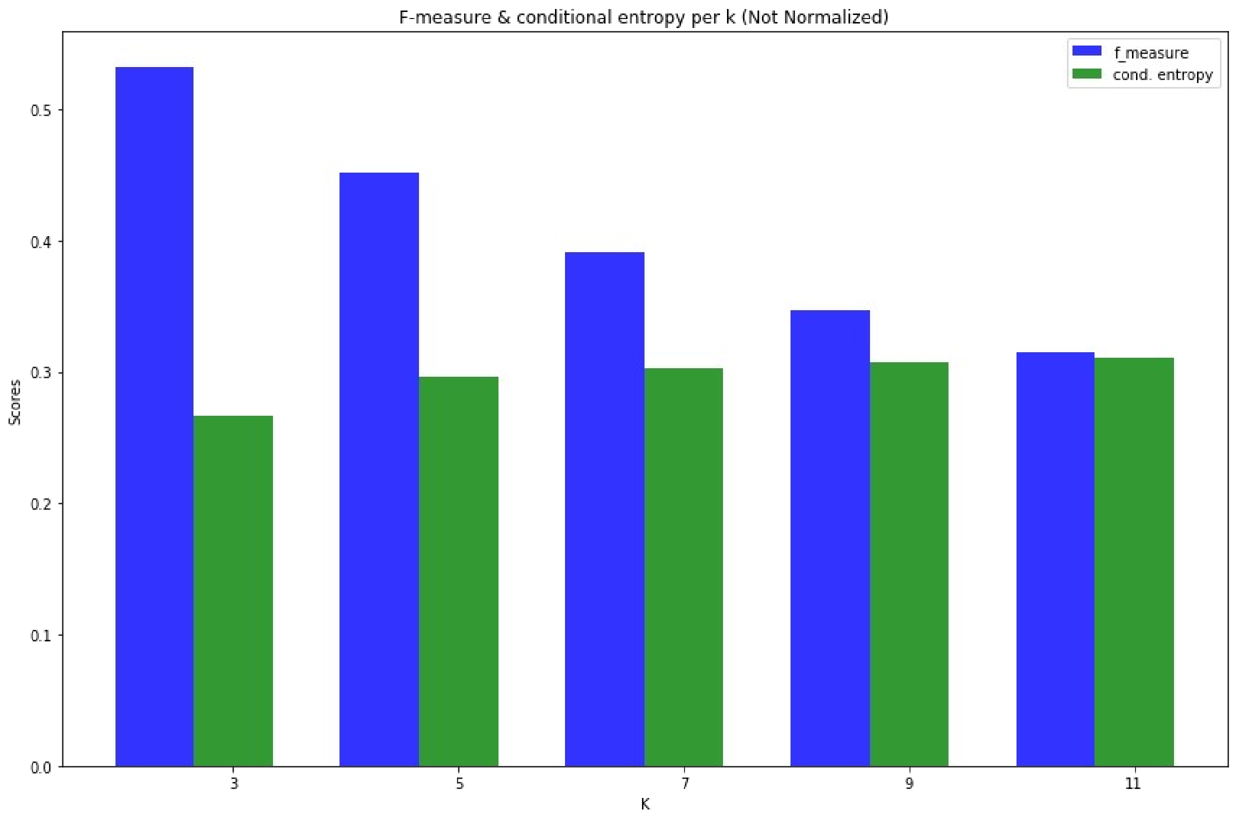
- Data_utils.py:
  - load_images: for loading images from dataset
  - load_ground_truth: for loading human handmade clusters
  - load_segmentations: for loading the segments made by the algorithm
- Metrics.py:
  - f_measure : for measuring f1_score between the formed cluster and ground truth
- Cluster_algorithm.py:
  - train: for training and visualise our predictions
  - calc_performance: calling f_measure and conditional entropy(v_measure)
- Normalized_cut.py:
  - train: for training and visualise our predictions
  - get_segmentations: extraction of segments formed by normalized cut
  - calc_performance: performance calculation
- K_means.py:
  - get_segmentations: extraction of segments formed by k-means clustering
- Visuals.py:
  - plot_img_with_gnd: for plotting the ground truths
  - plot_images_truths: for plotting images
  - plot_k_performance: for plotting performance

Note:

- number of clusters are kept fixed in every case
- n = [3, 5, 7, 9, 11]

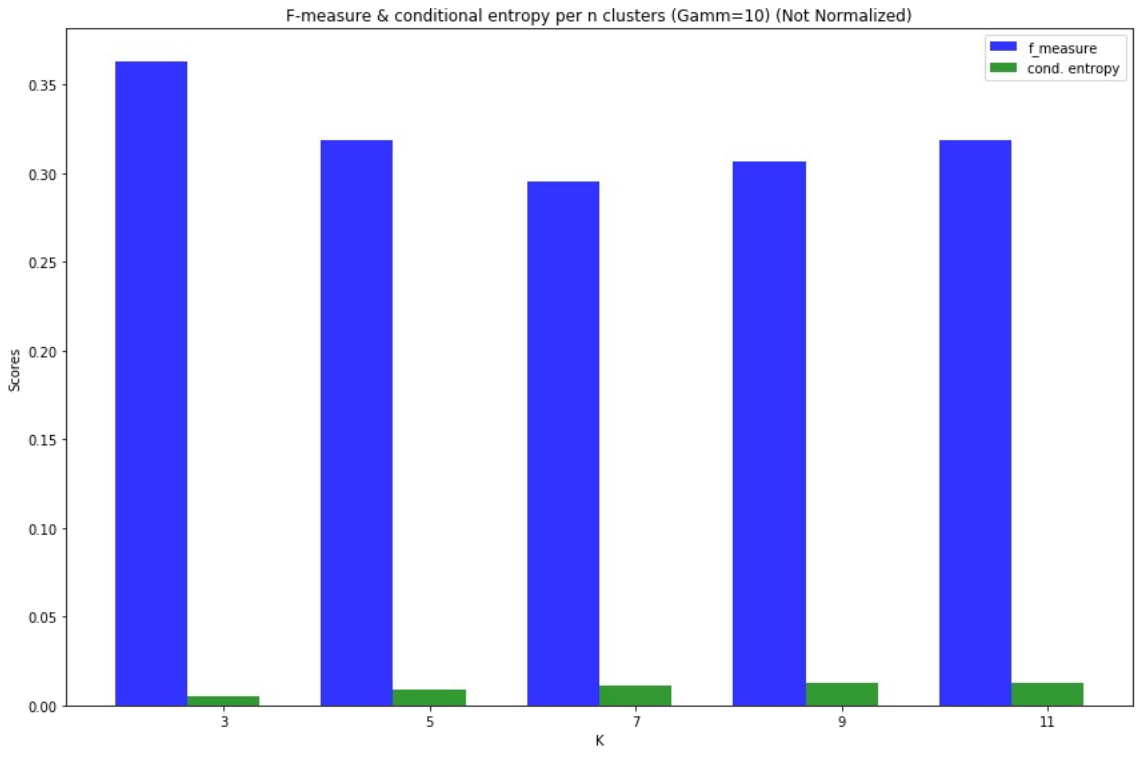
K-Means Operation:

- Time of execution = 1073.98 seconds
- K=3: Total average f-measure=0.5327, Total average cond_entropies=0.2670
- K=5: Total average f-measure=0.4514, Total average cond_entropies=0.2967
- K=7: Total average f-measure=0.3907, Total average cond_entropies=0.3029
- K=9: Total average f-measure=0.3474, Total average cond_entropies=0.3070
- K=11: Total average f-measure=0.3149, Total average cond_entropies=0.3102

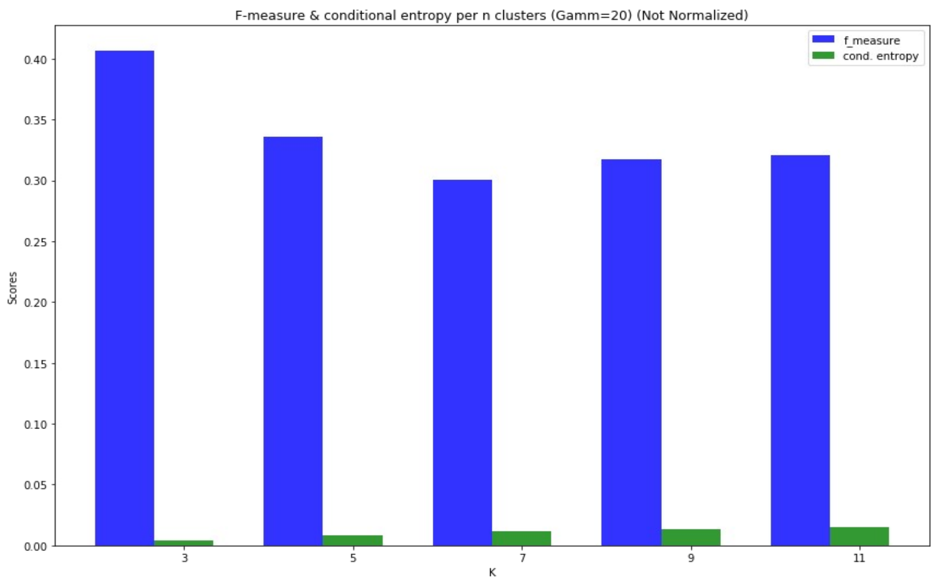


Normalized Cut:

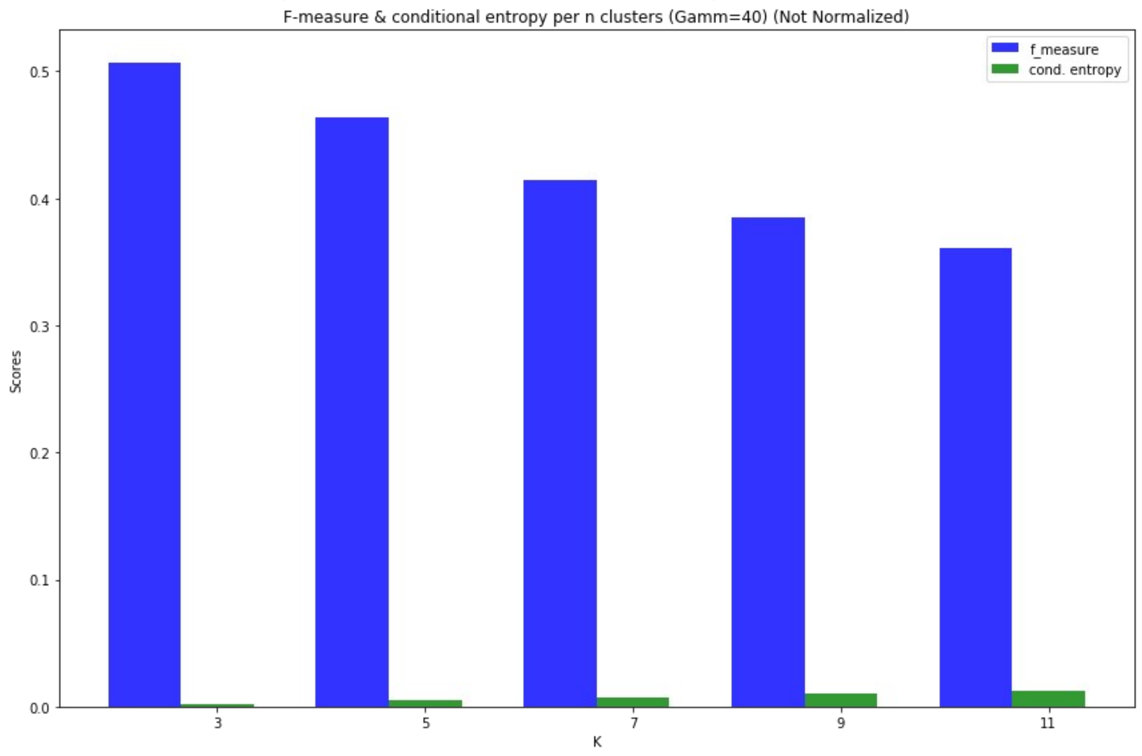
- rbf method:
    - gamma 10 :
        - Time of execution = 212.36 seconds
        - K=3: Total average f-measure=0.3635, Total average cond_entropies=0.0055
        - K=5: Total average f-measure=0.3191, Total average cond_entropies=0.0090
        - K=7: Total average f-measure=0.2951, Total average cond_entropies=0.0114
        - K=9: Total average f-measure=0.3070, Total average cond_entropies=0.0124
        - K=11: Total average f-measure=0.3188, Total average cond_entropies=0.0124



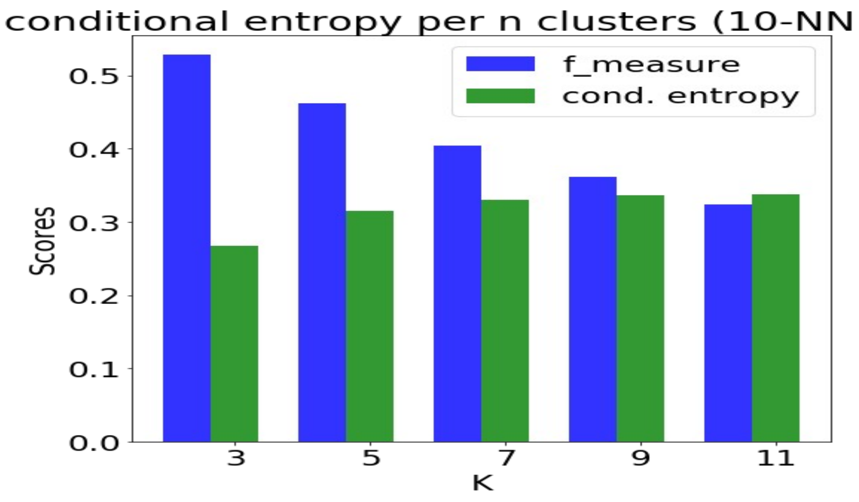
    - gamma 20:
        - Time of execution = 184.97 seconds
        - K=3: Total average f-measure=0.4275, Total average cond_entropies=0.0036
        - K=5: Total average f-measure=0.3103, Total average cond_entropies=0.0083
        - K=7: Total average f-measure=0.2792, Total average cond_entropies=0.0107
        - K=9: Total average f-measure=0.2692, Total average cond_entropies=0.0141
        - K=11: Total average f-measure=0.2874, Total average cond_entropies=0.0149



- gamma 40:
  - Time of execution = 183.83 seconds
  - K=3: Total average f-measure=0.5073, Total average cond_entropies=0.0021
  - K=5: Total average f-measure=0.4641, Total average cond_entropies=0.0046
  - K=7: Total average f-measure=0.4145, Total average cond_entropies=0.0071
  - K=9: Total average f-measure=0.3850, Total average cond_entropies=0.0098
  - K=11: Total average f-measure=0.3612, Total average cond_entropies=0.0120



- Nearest Neighbour Method:
  - number of neighbours 10:
    - Time of execution = 246.50 seconds
    - K=3: Total average f-measure=0.5280, Total average cond_entropies=0.2669
    - K=5: Total average f-measure=0.4625, Total average cond_entropies=0.3147
    - K=7: Total average f-measure=0.4046, Total average cond_entropies=0.3297
    - K=9: Total average f-measure=0.3610, Total average cond_entropies=0.3363
    - K=11: Total average f-measure=0.3244, Total average cond_entropies=0.3377



- number of neighbours 20
  - Time of execution = 255.38 seconds
  - K=3: Total average f-measure=0.5430, Total average cond_entropies=0.2795
  - K=5: Total average f-measure=0.4668, Total average cond_entropies=0.3194
  - K=7: Total average f-measure=0.4074, Total average cond_entropies=0.3299
  - K=9: Total average f-measure=0.3604, Total average cond_entropies=0.3338
  - K=11: Total average f-measure=0.3229, Total average cond_entropies=0.3346

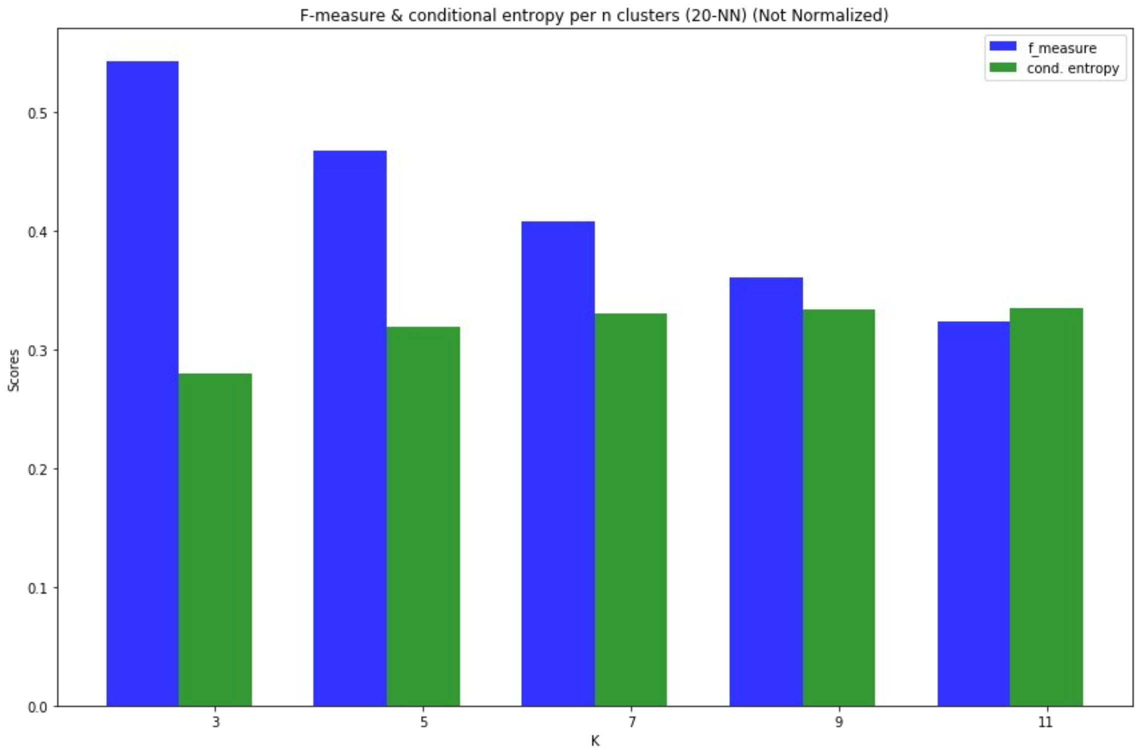
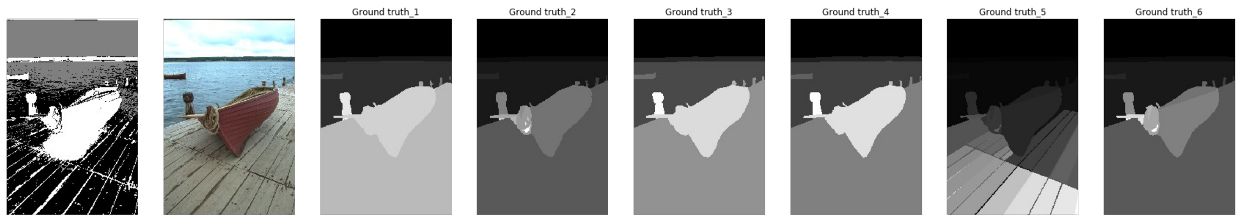
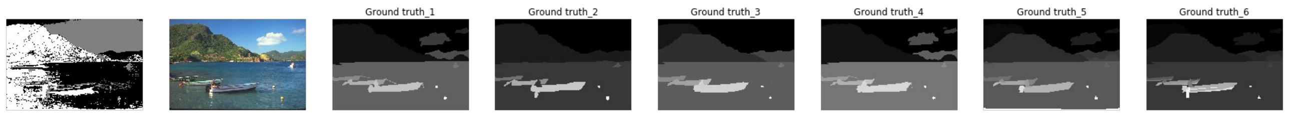
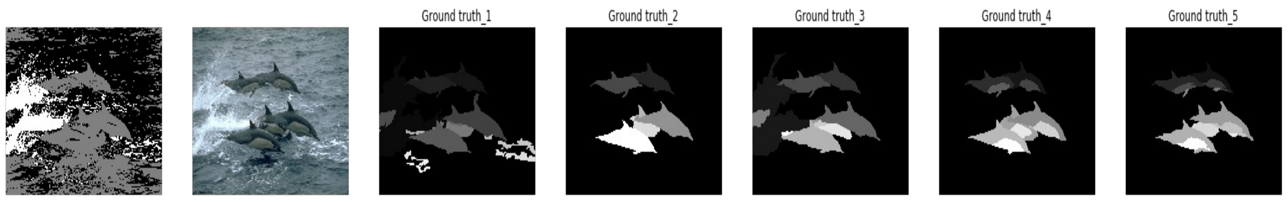


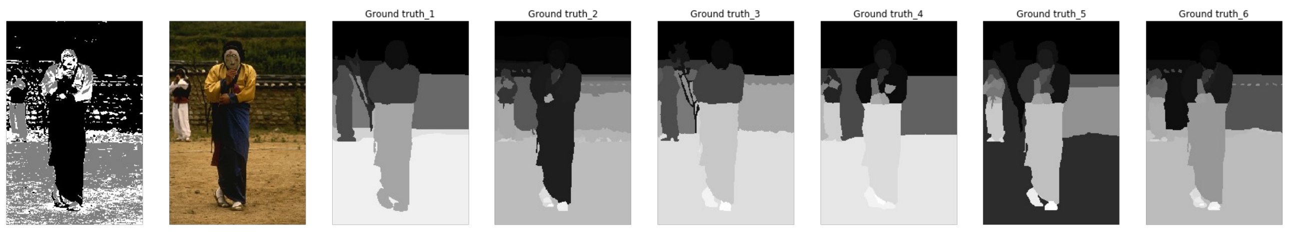
Image results of k-mean and Normalized Cut:

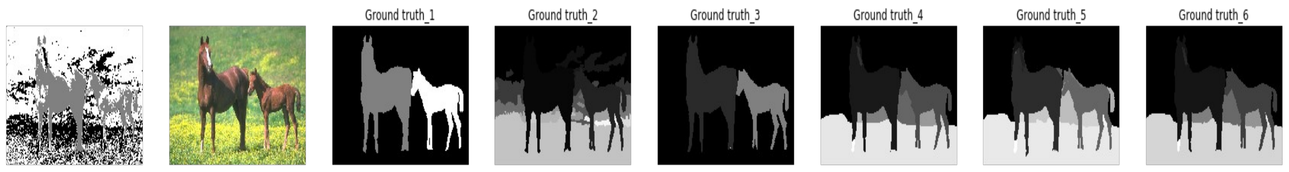
- number of clusters = 3
- k-means results:



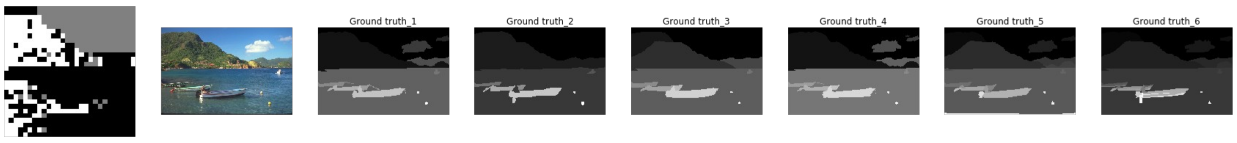


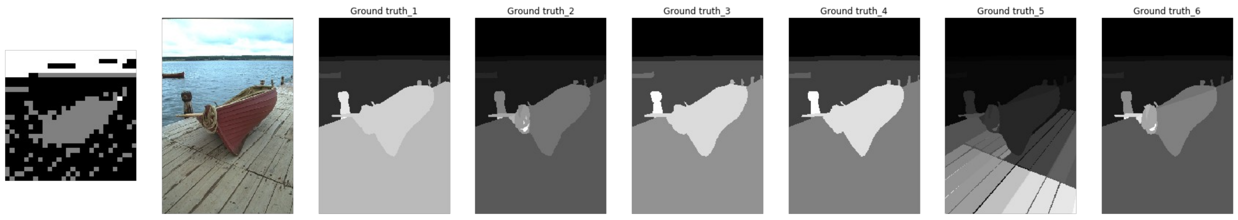


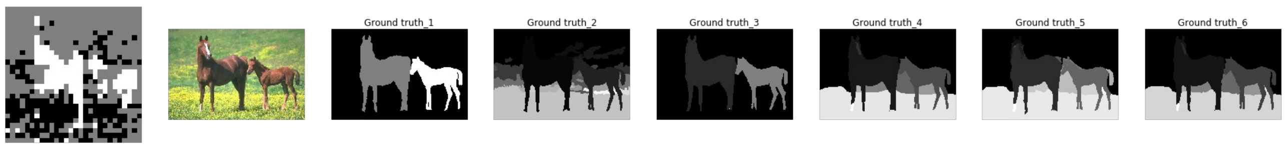


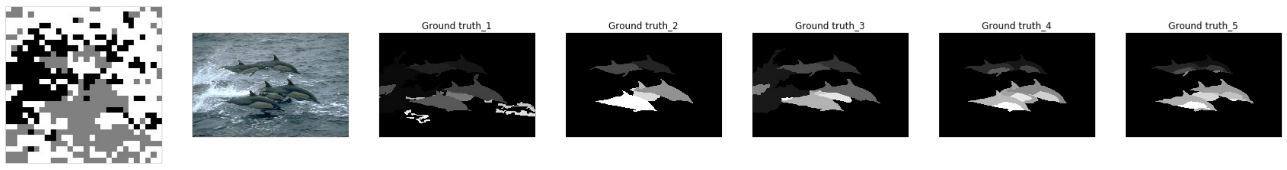


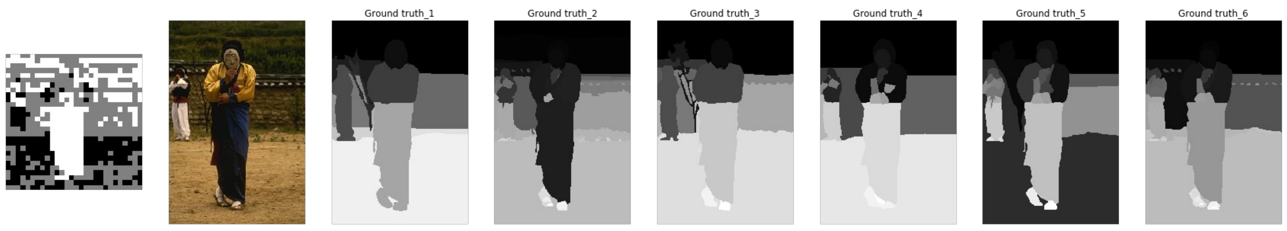
- Normalized Cut:



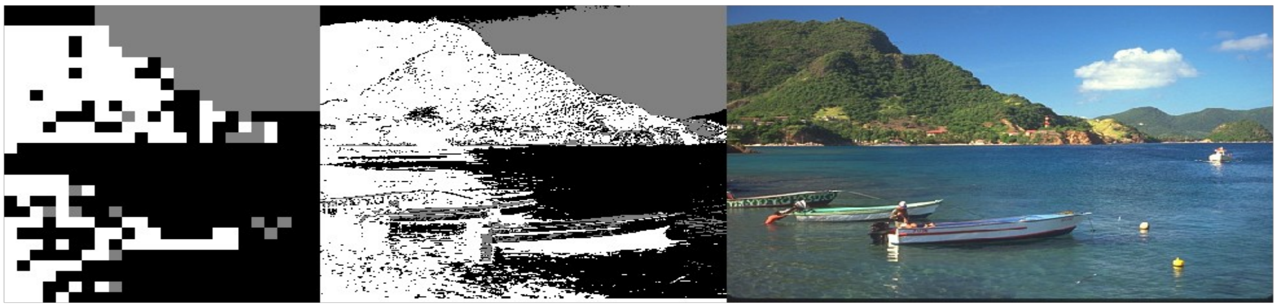


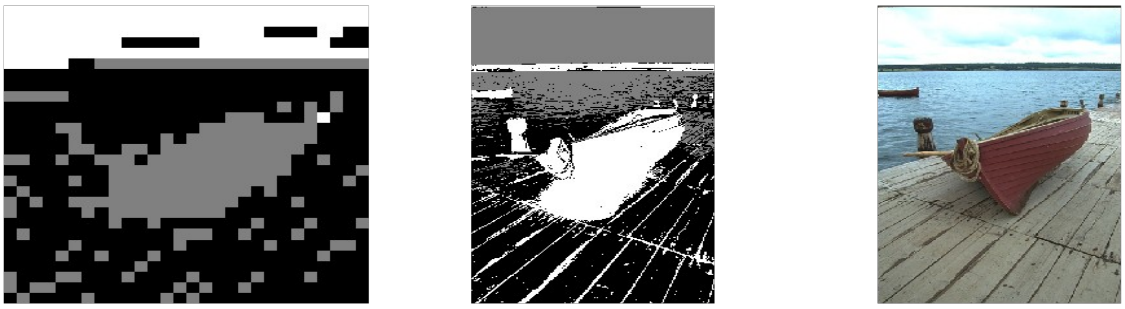


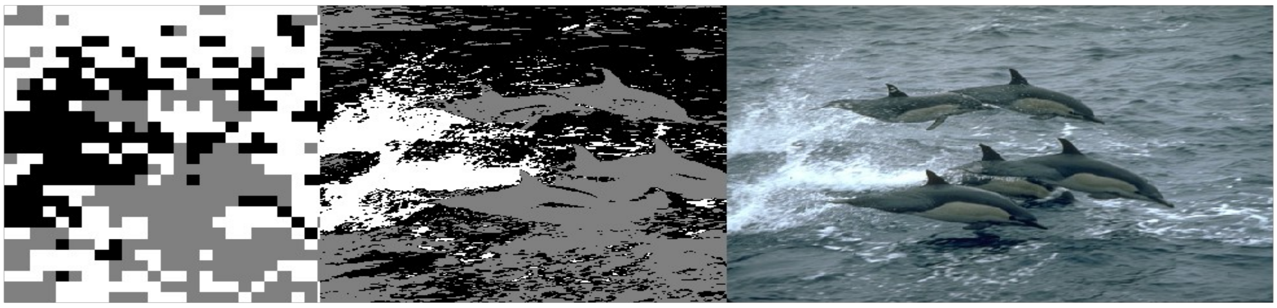


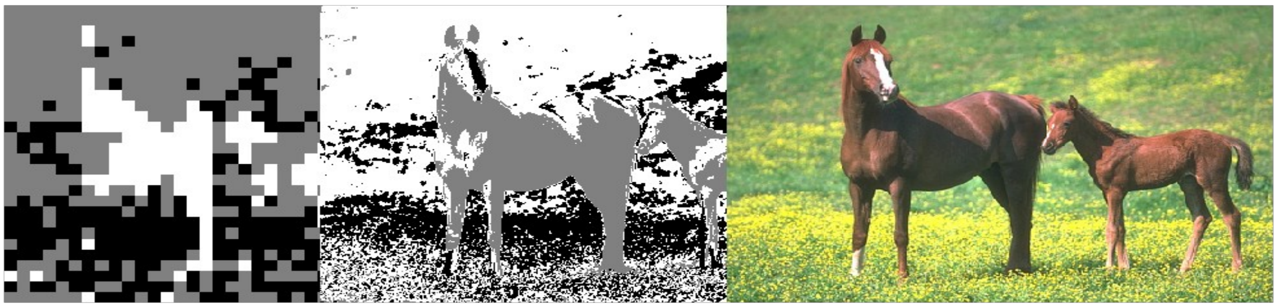


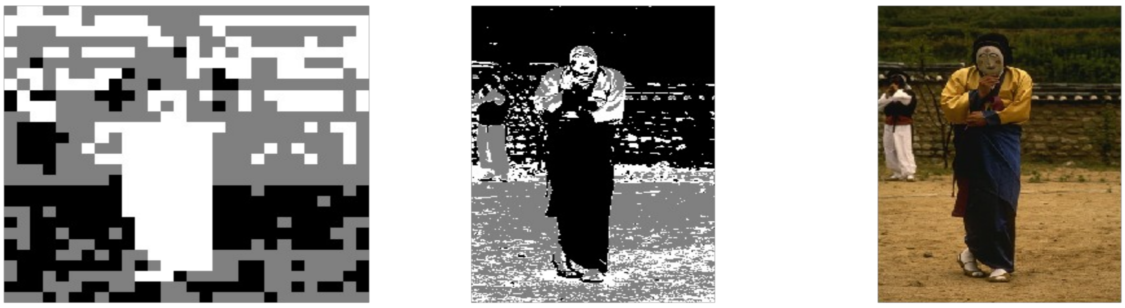
Comparision of Normalized Cut, k-means and original image:











Conclusion:

- Time of execution taken by Normalized Cut method is very less
- F-measure and v_measure values have also ovetaken by Normalized cut, so we may use it for feature extraction in ml models.
- Normalized Cut method had given us the liberty to tune our parameters which is not in case of Kmeans
- But the segmentaion as can be seen by normalized cut is not as the ground truth, which is not the case with k-means