Digital Image Processing

Paper Presentation

SLIC Superpixels Compared to State-of-the-Art Superpixel Methods

Name: Deeksha Aggarwal

Roll No. - MS2019006

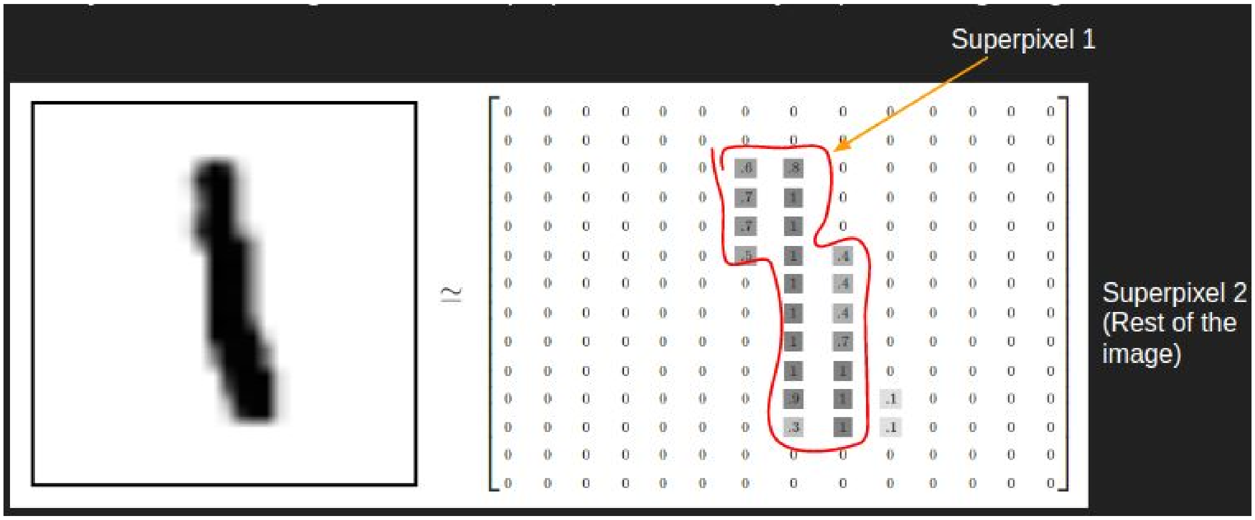
Contents

Objectives:

1. To introduce a new superpixel algorithm, simple linear iterative clustering (SLIC), which adapts a k-means clustering approach to efficiently generate superpixels.
2. To compare SLIC with the state-of-the-art superpixel algorithms, the paper has discussed the following parameters:
   - Ability to adhere to image boundaries,
   - speed,
   - memory efficiency,
   - segmentation performance.

Introduction

A superpixel can be defined as a group of pixels which have similar characteristics. It is generally color based segmentation. Superpixels can be very helpful in image segmentation.They capture image redundancy, provide a convenient primitive from which to compute image features, and greatly reduce the complexity of subsequent image processing tasks. They have become key building blocks of many computer vision algorithms, such as top scoring multiclass object segmentation entries to the PASCAL VOC Challenge, depth estimation, segmenentation.



Motivation:

Advantages of Superpixels

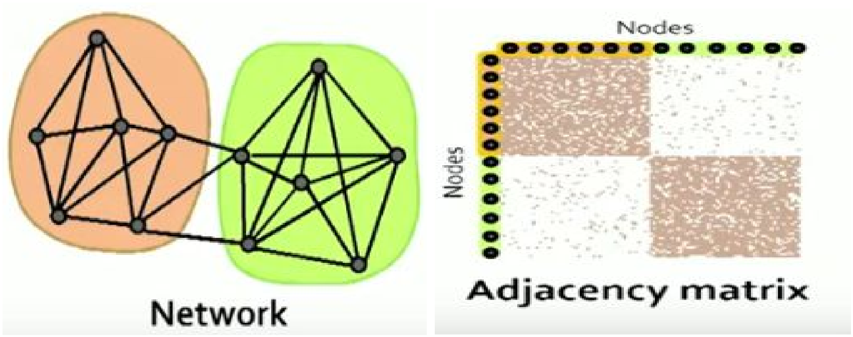
1. Computationally efficient.
2. Efficient Preprocessing step to many image segmentation algorithm.
3. Reduction in the number of hypothesis.
4. Some algorithms provide control over the amount of superpixels and their compactness or shape which helps in efficient/similar superpixel formation.

Existing Superpixel Methods:

Algorithms for generating superpixels can be broadly categorized as either graph-based or gradient ascent methods. Below, we review popular superpixel methods for each of these categories, including some that were not originally designed specifically to generate superpixels. Table 1 provides a qualitative and quantitative summary of the reviewed methods, including their relative Performance.

Graph Based Algorithms
Gradient Ascent Based Algorithms

1. **Graph Based Algorithms:**
   Treat each pixel as a node in a graph.Adjacency Matrix is made up of Edge weights between two nodes are proportional to the similarity between neighboring pixels.Superpixels are created by minimizing a cost function defined over the graph.



Some existing graph based algorithms:

The Normalized cuts algorithm or NC05 recursively partitions a graph of all pixels in the image using contour and texture cues, globally minimizing a cost function defined on the edges at the partition boundaries. It produces very regular, visually pleasing superpixels. However, the boundary adherence of NC05 is relatively poor and it is the slowest among the methods (particularly for large images), although attempts to speed up the 3 algorithms exist. NC05 has a complexity of O$N^{3/2}$, where N is the number of pixels.

GS04 or **Felzenszwalb and Huttenlocher** propose an alternative graph-based approach that has been applied to generate superpixels. It performs an agglomerative clustering of pixels as nodes on a graph such that each superpixel is the minimum spanning tree of the constituent pixels. GS04 adheres well to image boundaries in practice, but produces superpixels with very irregular sizes and shapes. It is ON log N complex and fast in practice. However, it does not offer an explicit control over the amount of superpixels or their compactness.

SL08. Moore et al. propose a method to generate superpixels that conform to a grid by finding optimal paths, or seams, that split the image into smaller vertical or horizontal regions. Optimal paths are found using a graph cuts method similar to Seam Carving 3. While the complexity of SL08 is O$N^{3/2}$ logN according to the authors, this does not account for the precomputed boundary maps, which strongly influence the quality and speed of the output.

GCa10 and GCb10 Veksler et al. use a global optimization approach similar to the texture synthesis work of [14]. Superpixels are obtained by stitching together overlapping image patches such that each pixel belongs to only one of the overlapping regions. They suggest two variants of their method, one for generating compact superpixels (GCa10) and one for constant-intensity superpixels (GCb10).

2. **Gradient-Ascent-Based Algorithms**
   Starting from a rough initial clustering of pixels, gradient ascent methods iteratively refine the clusters until some convergence criterion is met to form superpixels.

Some existing gradient ascent based algorithms:

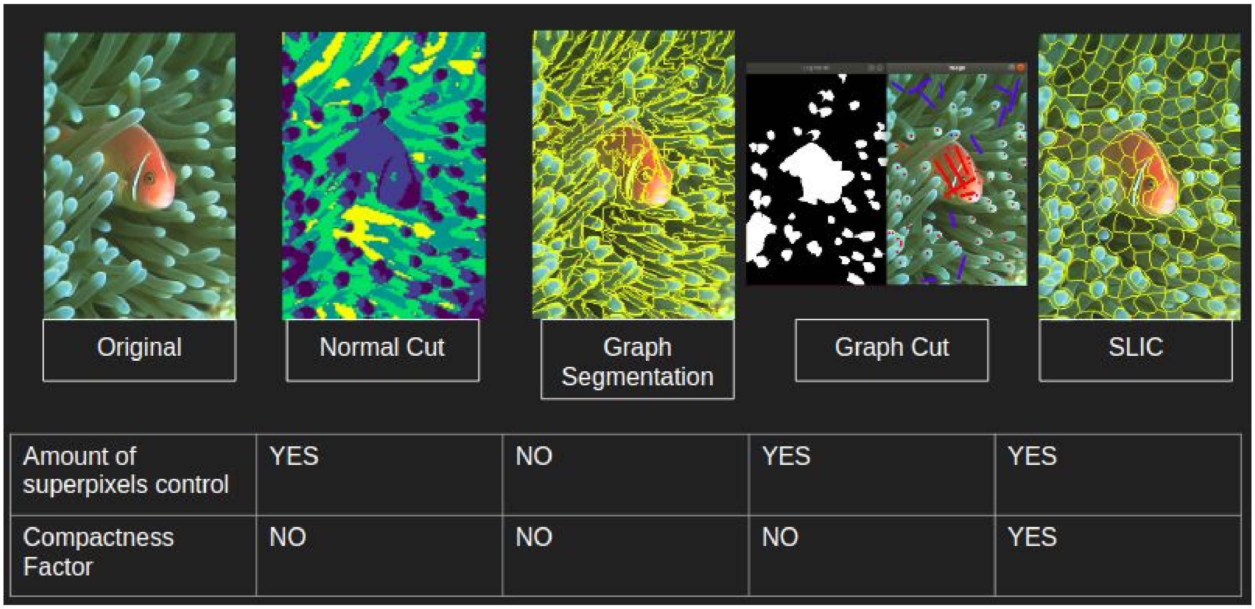
MS02., mean shift, an iterative mode-seeking procedure for locating local maxima of a density function, is applied to find modes in the color or intensity feature space of an image. Pixels that converge to the same mode define the superpixels. MS02 is an older approach, producing irregularly shaped superpixels of nonuniform size. It is OðN 2 Þ complex, making it relatively slow, and does not offer direct control over the amount, size, or compactness of superpixels.

QS08. Quick shift [25] also uses a mode-seeking segmentation scheme. It initializes the segmentation using a medoid shift procedure. It then moves each point in the feature space to the nearest neighbor that increases the Parzen density estimate. While it has relatively good boundary adherence, QS08 is quite slow, with an OðdN 2 Þ complexity (d is a small constant [25]). QS08 does not allow for explicit control over the size or number of superpixels. Previous works have used QS08 for object localization [9] and motion segmentation [2].

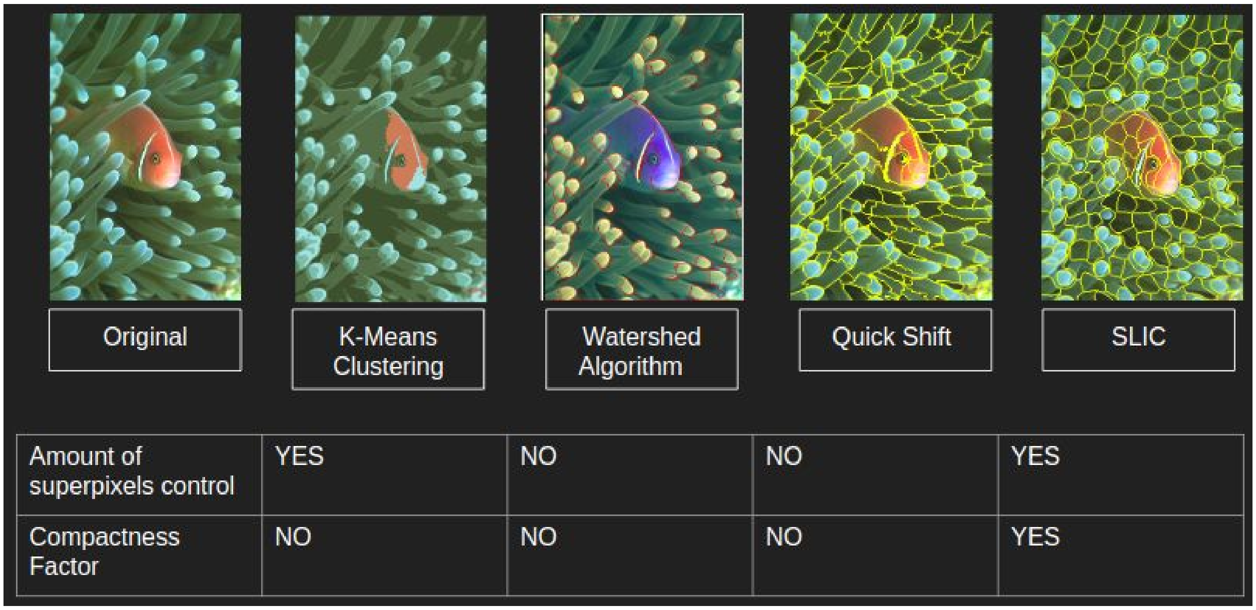
WS91. The watershed approach [28] performs a gradient ascent starting from local minima to produce watersheds, lines that separate catchment basins. The resulting superpixels are often highly irregular in size and shape, and do not exhibit good boundary adherence. The approach of [28] is relatively fast (OðN log NÞ complexity), but does not offer control over the amount of superpixels or their compactness.

TP09. The Turbopixel method progressively dilates a set of seed locations using level-set-based geometric flow [15]. The geometricflow relies on local image gradients, aiming to regularly distribute superpixels on the image plane. Unlike WS91, TP09 superpixels are constrained to have uniform size, compactness, and boundary adherence. TP09 relies on algorithms of varying complexity, but in practice, as the authors claim, has approximately OðNÞ behavior [15]. However, it is among the slowest algorithms examined and exhibits relatively poor boundary adherence.

Implementation of existing superpixel pixel algorithms:



Amount of superpixels control	YES	NO	YES	YES
Compactness Factor	NO	NO	NO	YES



Amount of superpixels control	YES	NO	NO	YES
Compactness Factor	NO	NO	NO	YES

Properties of a Good SuperPixel:

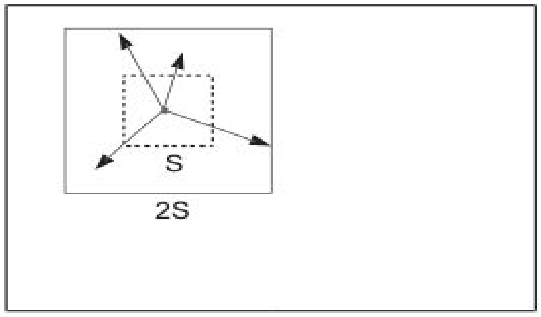
1. Faster formation.
2. Memory efficient.
3. Exhibits boundary adherence.
4. Should offer control over the amount of superpixels and its compactness.
5. Should Captures global aspects of the image.
6. Segmentation accuracy and speed.

SLIC Algorithm:

Cluster in 5-D space that is *[l_i, a_i, b_i, x_i, y_i]*, 3 for color and 2 for location of a pixel.

1. Initialise superpixel centers by sampling **k** locations on the image plane consisting of **N** pixels and **spaced S pixels apart**. The centers are moved to seed locations corresponding to the lowest gradient position in a 3X3 neighborhood.
2. For each cluster center **C_i** , compute distance **D** between **C_i** and each pixel in the neighbourhood of **C_i** within **2SX2S**.

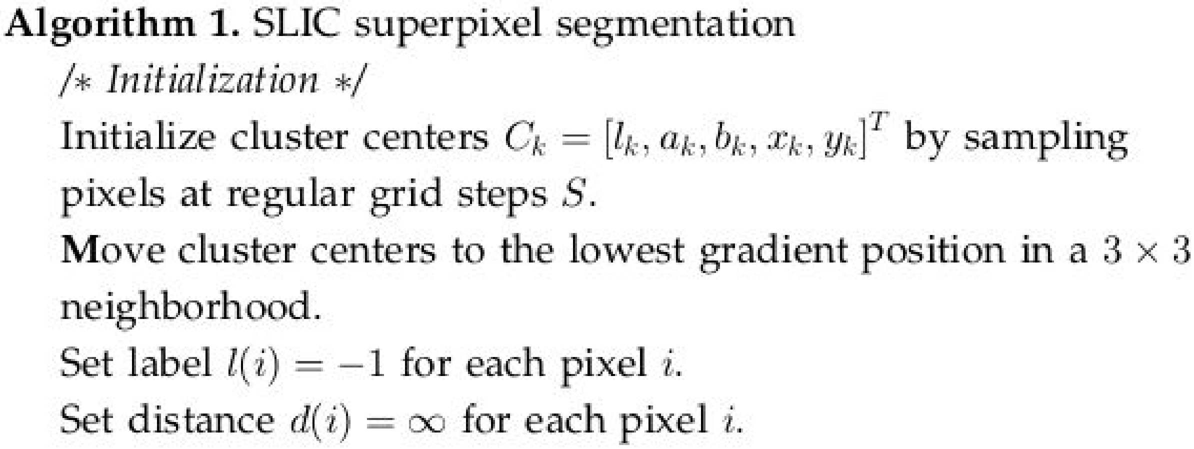
3. Assign pixel to **C$_i$** based on its current distance. Do for all **C$_i$**
4. Update the cluster centers like in k-means.
5. Repeat until convergence.



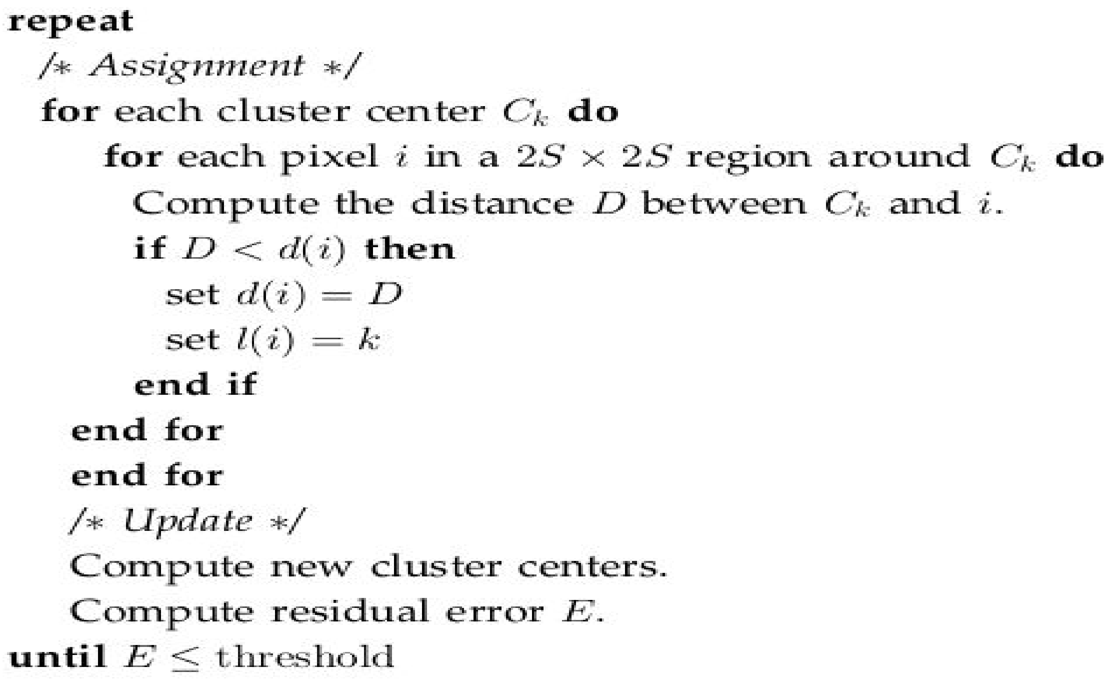
(b) SLIC searches a limited region

Parameters of SLIC ALgorithm:

1. **K** or Number of Labels in the segmented output
2. **M or compactness factor.** It balances color and space proximity. Higher values give more weight to space proximity, making superpixel shapes more square/cubic.
3. **Threshold:** Maximum number of iterations to loop.



```
Algorithm 1. SLIC superpixel segmentation
    /* Initialization */
    Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling
    pixels at regular grid steps $S$.
    Move cluster centers to the lowest gradient position in a $3 \times 3$
    neighborhood.
    Set label $l(i) = -1$ for each pixel $i$.
    Set distance $d(i) = \infty$ for each pixel $i$.
```



```
repeat
  /* Assignment */
  for each cluster center $C_k$ do
      for each pixel $i$ in a $2S \times 2S$ region around $C_k$ do
        Compute the distance $D$ between $C_k$ and $i$.
        if $D < d(i)$ then
          set $d(i) = D$
          set $l(i) = k$
        end if
    end for
    end for
    /* Update */
    Compute new cluster centers.
    Compute residual error $E$.
until $E \leq$ threshold
```

Distance Measure:

SLIC superpixels correspond to clusters in the labxy color-image plane space. This presents a problem in defining the distance measure D, which may not be immediately obvious. D computes the distance between a pixel i and cluster center C k in Algorithm 1. A pixel’s color is represented in the CIELAB color space $[l\ a\ b]^T$, whose range of possible values is known. The pixel’s position position $[l\ a\ b]^T$, on the other hand, may take a range of values that varies according to the size of the image. Simply defining D to be the 5D euclidean distance in labxy space will cause inconsistencies in clustering behavior for different superpixel sizes. For large superpixels, spatial distances outweigh color proximity, giving more relative importance to spatial proximity than color. This produces compact superpixels that do not adhere well to image boundaries. For smaller superpixels, the converse is true. To combine the two distances into a single measure, it is necessary to normalize color proximity and spatial proximity by their respective maximum distances within a cluster, N_s and N_c .
Doing so, D’ is written

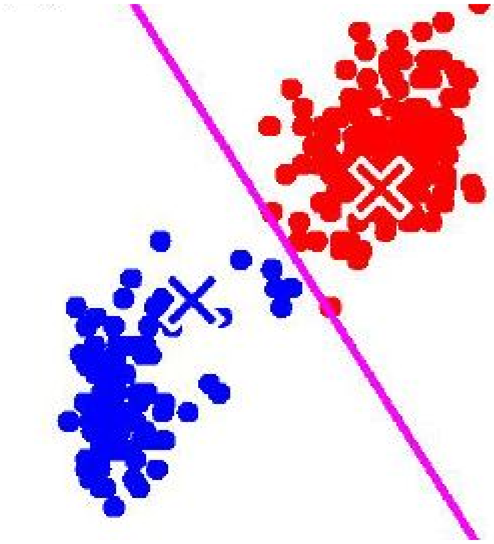
$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2},$$
$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2},$$

$$D' = \sqrt{\left(\frac{d_c}{N_c}\right)^2 + \left(\frac{d_s}{N_s}\right)^2}.$$

$$D' = \sqrt{\left(\frac{d_c}{m}\right)^2 + \left(\frac{d_s}{S}\right)^2},$$

$$D = \sqrt{{d_c}^2 + \left(\frac{d_s}{S}\right)^2 m^2}.$$

Error Measure or stopping Criteria:
Once each pixel has been associated to the nearest cluster center, the L 2 norm is used to compute a residual error E between the new cluster center locations and previous cluster center locations. The assignment and update steps can be repeated iteratively until the error converges.

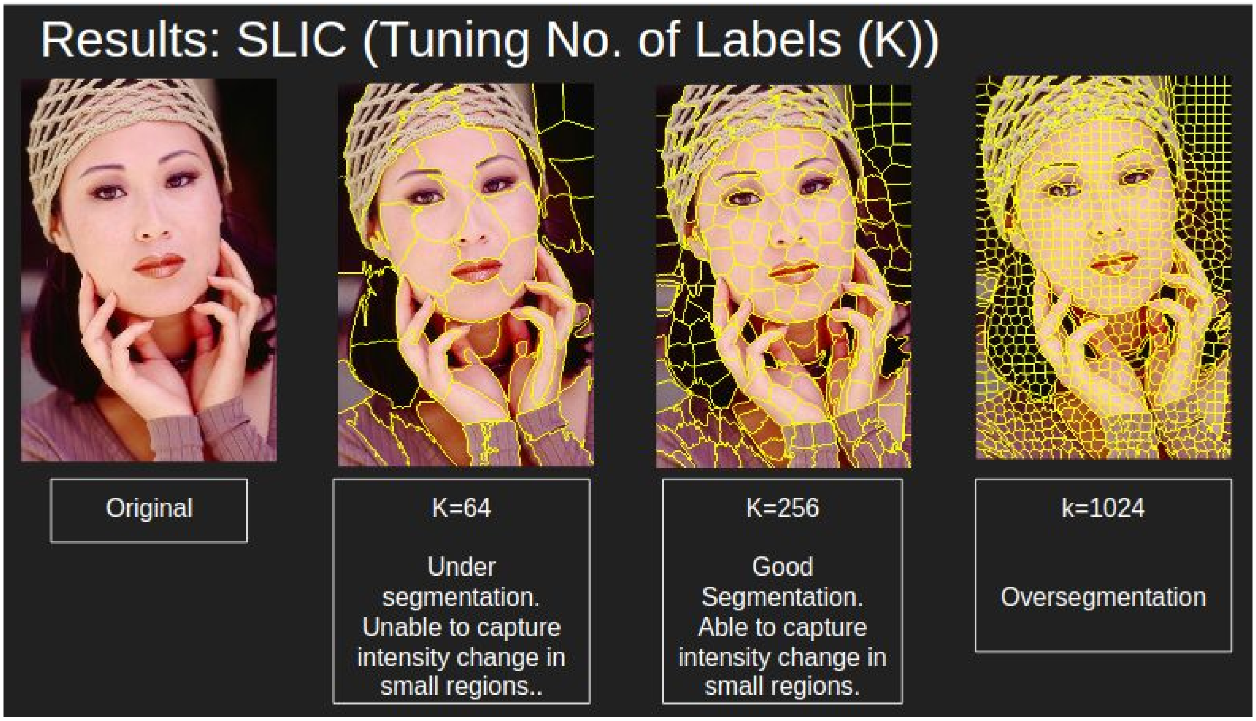


Inferences:

1. In SLIC the local regions with similar color and texture distributions are part of the same superpixel group.
2. Both Spatial and color proximity are taken into account while

assigning the labels to each pixel. This can be controlled by compactness parameter.

3. SLIC do not compute distances between each pixel but this calculation of distance is limited to a window of 2S where S is √N/k.



Results: SLIC (Tuning No. of Labels (K))

Original

K=64

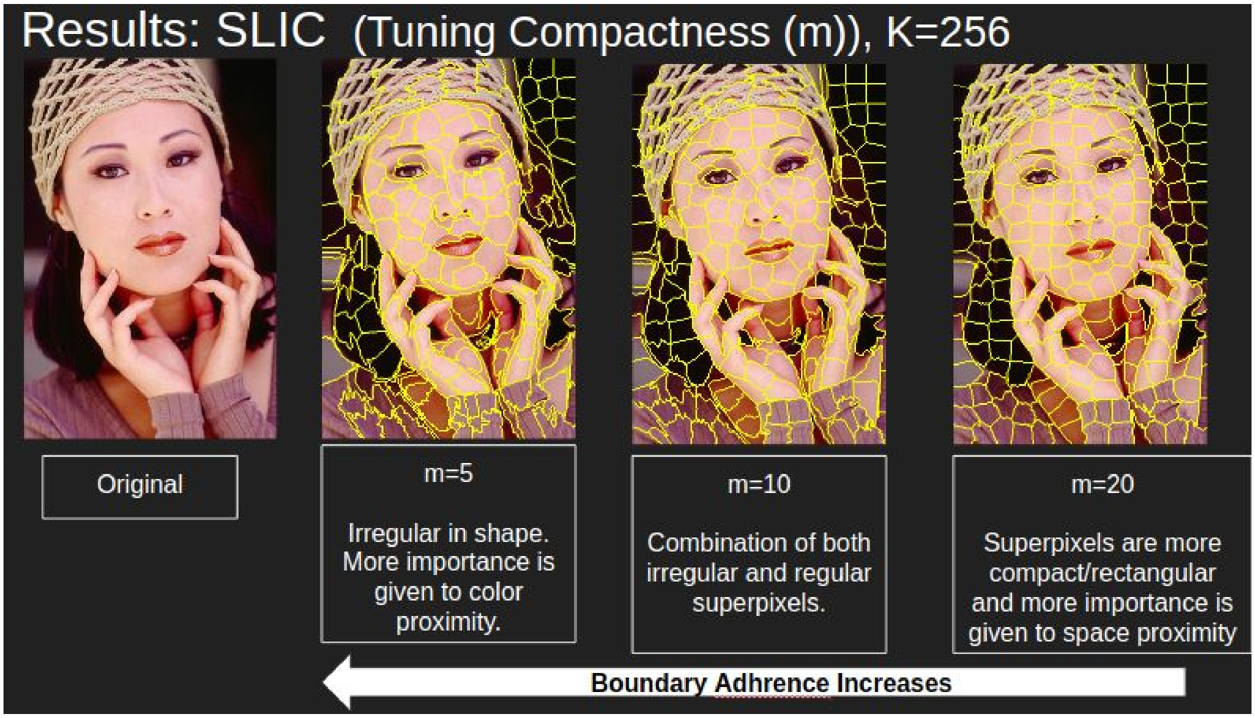
Under segmentation. Unable to capture intensity change in small regions..

K=256

Good Segmentation. Able to capture intensity change in small regions.

k=1024

Oversegmentation



Results: SLIC (Tuning Compactness (m)), K=256

Original

m=5

Irregular in shape. More importance is given to color proximity.

m=10

Combination of both irregular and regular superpixels.

m=20

Superpixels are more compact/rectangular and more importance is given to space proximity

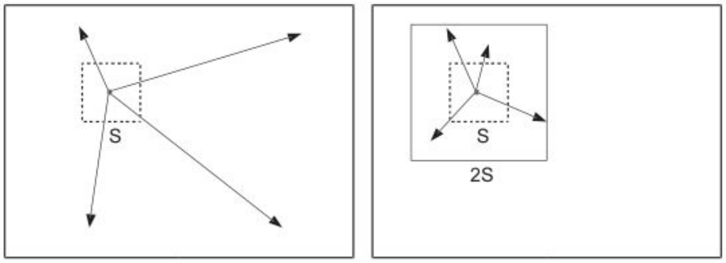
Boundary Adhrence Increases

Trade-OFFs Noticed in the results:

By increasing the compactness factor ‘m’, the superpixels become more regular or rectangular and more importance is given to spatial proximity than the color proximity in the formation of superpixels. Hence boundary adherence decreases and one superpixel might contain different color regions.

Comparison with K-Means

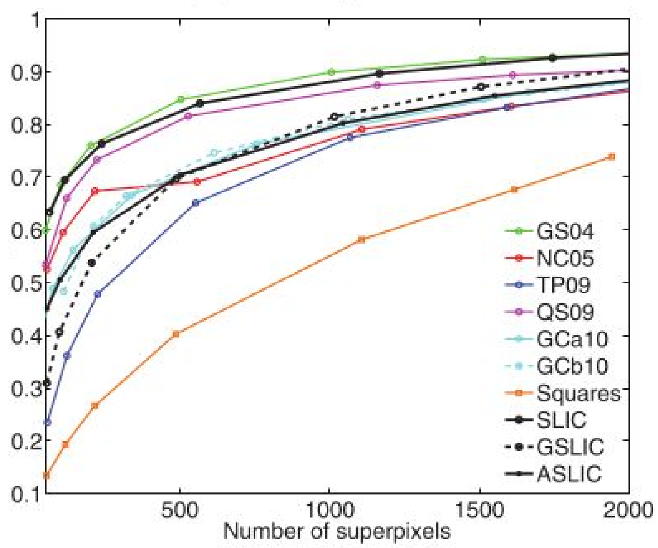
1. The number of distance calculations in the optimization is dramatically reduced by limiting the search space to a region proportional to the superpixel size. This reduces the complexity to be linear in the number of pixels N i.e **O(N)** and independent of the number of superpixels k. Complexity of K-means is **O(NkI)**
2. A weighted distance measure **combines color and spatial proximity** while simultaneously providing control over the size and compactness of the superpixels whereas k-means only deals with color/intensity proximity.



(a) standard *k*-means searches the entire image

(b) SLIC searches a limited region

Results: Segmentation Performance



Results as given in the paper i.e felzenszwalb or GS04 performed best and then slic performed the second best.

Mimicking the above results:

Case (i) Image Segmentation without machine learning and feature extraction:

Step 1: Original image is taken and thresholding is done to generate the binary ground truth masked image.

Step 2: Then SLIC segmentation is performed by tuning the parameters.

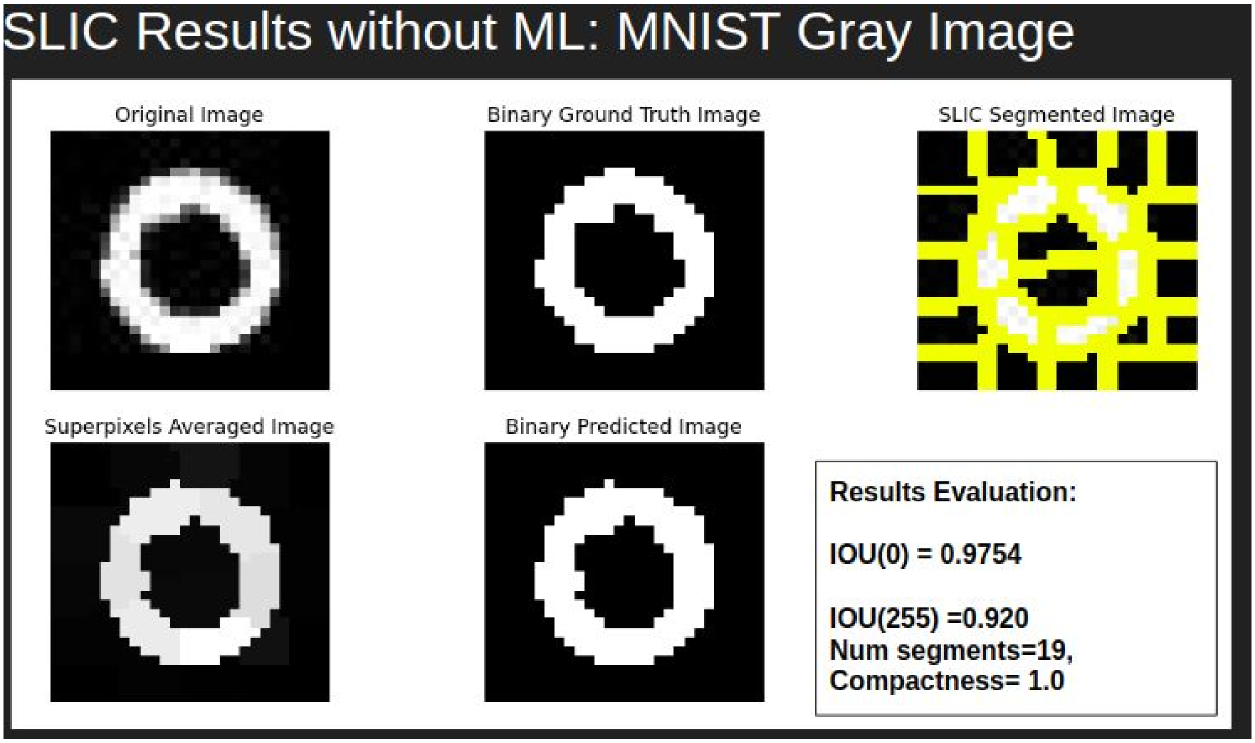
Step 3: For each segment, color averaging is performed.

Step 4: Then again thresholding was done to convert the image into binary image.

Step 5: Accuracy and IOU metric was calculated between the ground truth image generated in step 1 and the predicted image generated in step 4.

Note: Same steps are performed for Felzenszwalb algorithm and RGB images.

SLIC Results without ML: MNIST Gray Image

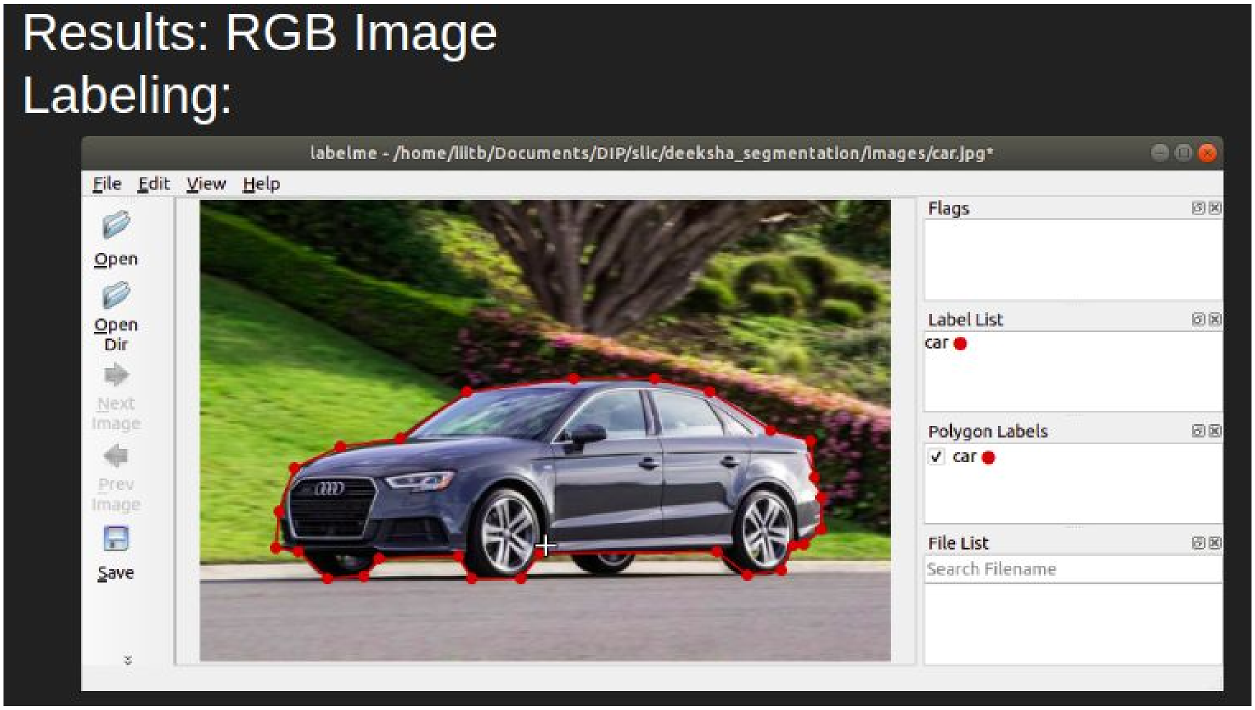


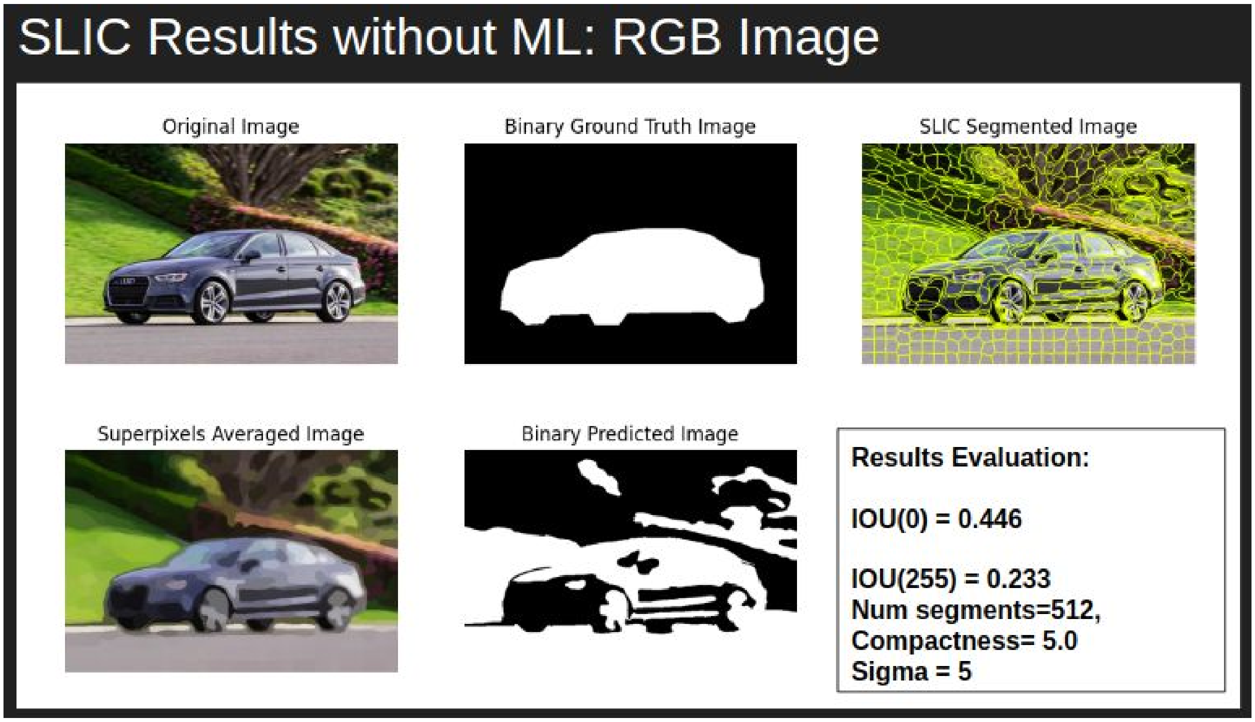
Results Evaluation:

IOU(0) = 0.9754

IOU(255) =0.920
Num segments=19,
Compactness= 1.0

Results: RGB Image Labeling:



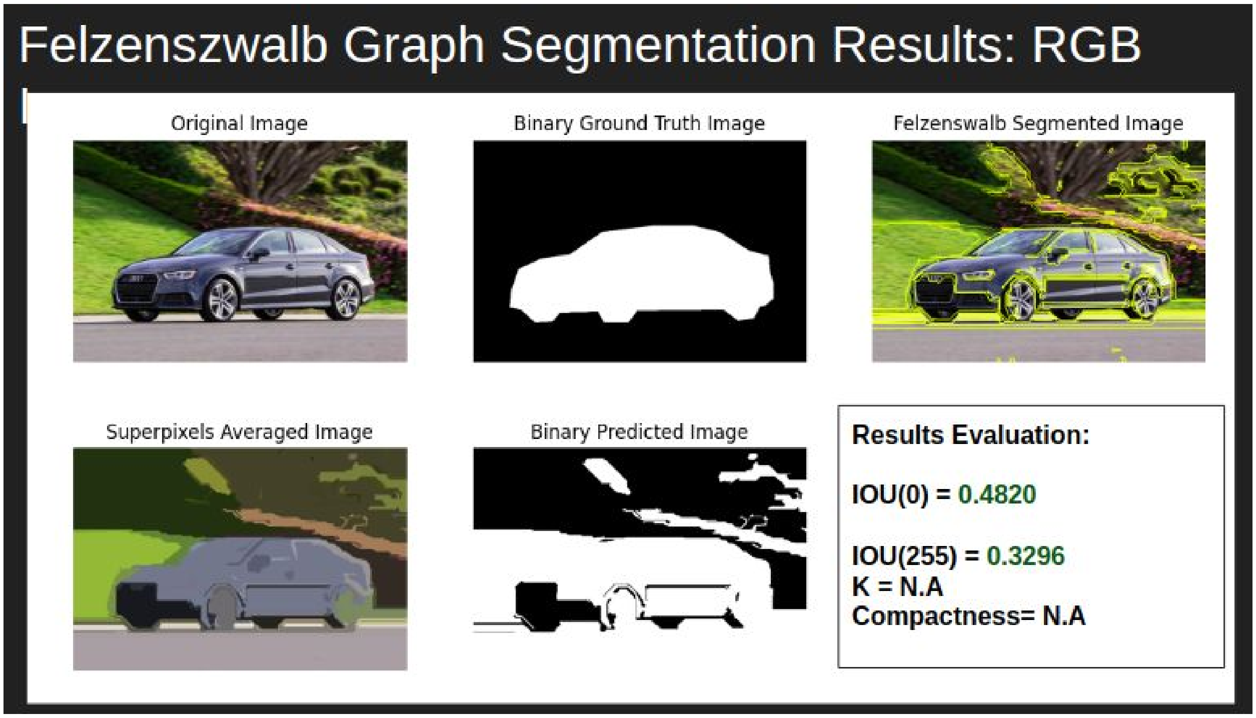


SLIC Results without ML: RGB Image

Results Evaluation:

IOU(0) = 0.446

IOU(255) = 0.233
Num segments=512,
Compactness= 5.0
Sigma = 5



Felzenszwalb Graph Segmentation Results: RGB I

Original Image
Binary Ground Truth Image
Felzenswalb Segmented Image
Superpixels Averaged Image
Binary Predicted Image

Results Evaluation:

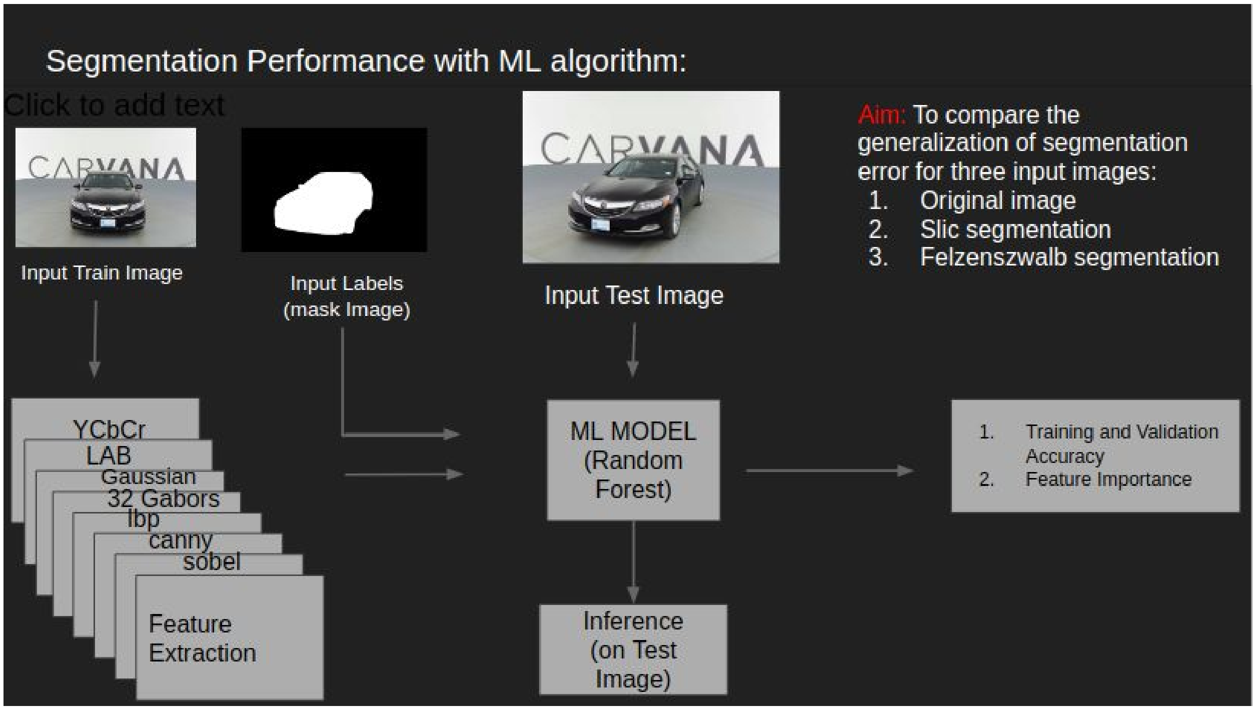
IOU(0) = 0.4820

IOU(255) = 0.3296
K = N.A
Compactness= N.A

Case (ii) Image Segmentation with machine learning and feature extraction:

Step 1: Original image is taken and its corresponding masked ground truth

image was given.
Step 2: Then 44 features were extracted for each pixel and stored in a dataframe. These features include sobel, 32 gabor filters, cielab and Ycrcb color modes, lbp, gaussians filters, and canny edge detectors.
Step 3: Then the random forest model was trained on these features.The training and testing accuracy was compared for two superpixel algorithms and original images as input.
Step 4: Then the unseen test image was taken and prediction was made by using the trained model in step 4.
Step 5: Accuracy and IOU metric was calculated between the ground truth image and the predicted image generated in step 4.



Segmentation Performance with ML algorithm:

Click to add text

Aim: To compare the generalization of segmentation error for three input images:

1. Original image
2. Slic segmentation
3. Felzenszwalb segmentation

Input Train Image

Input Labels (mask Image)

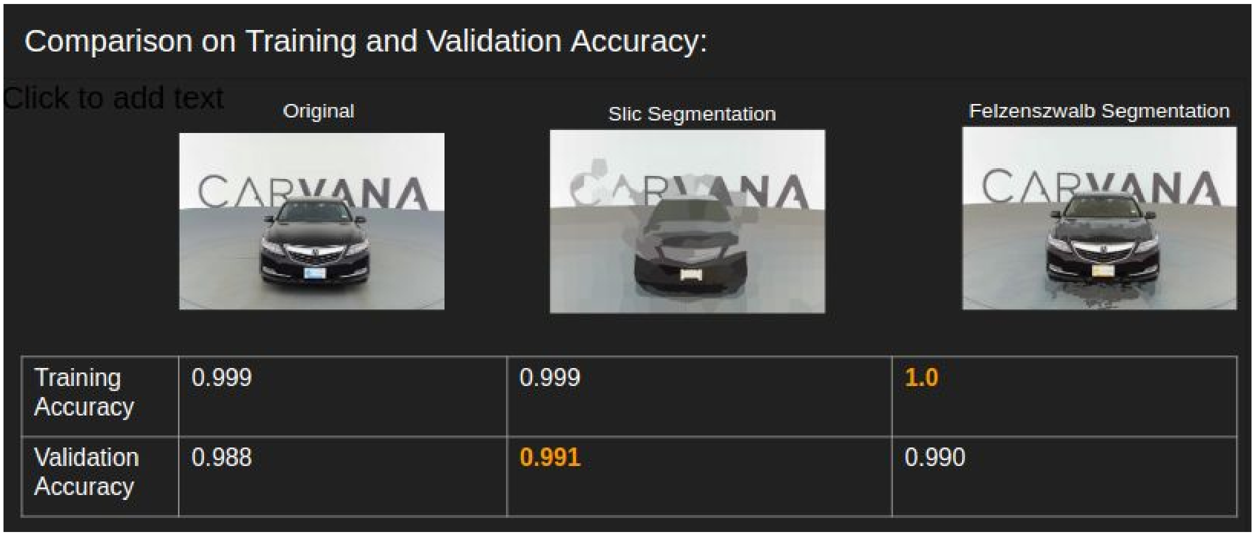
Input Test Image

YCbCr
LAB
Gaussian
32 Gabors
lbp
canny
sobel
Feature Extraction

ML MODEL (Random Forest)

1. Training and Validation Accuracy
2. Feature Importance

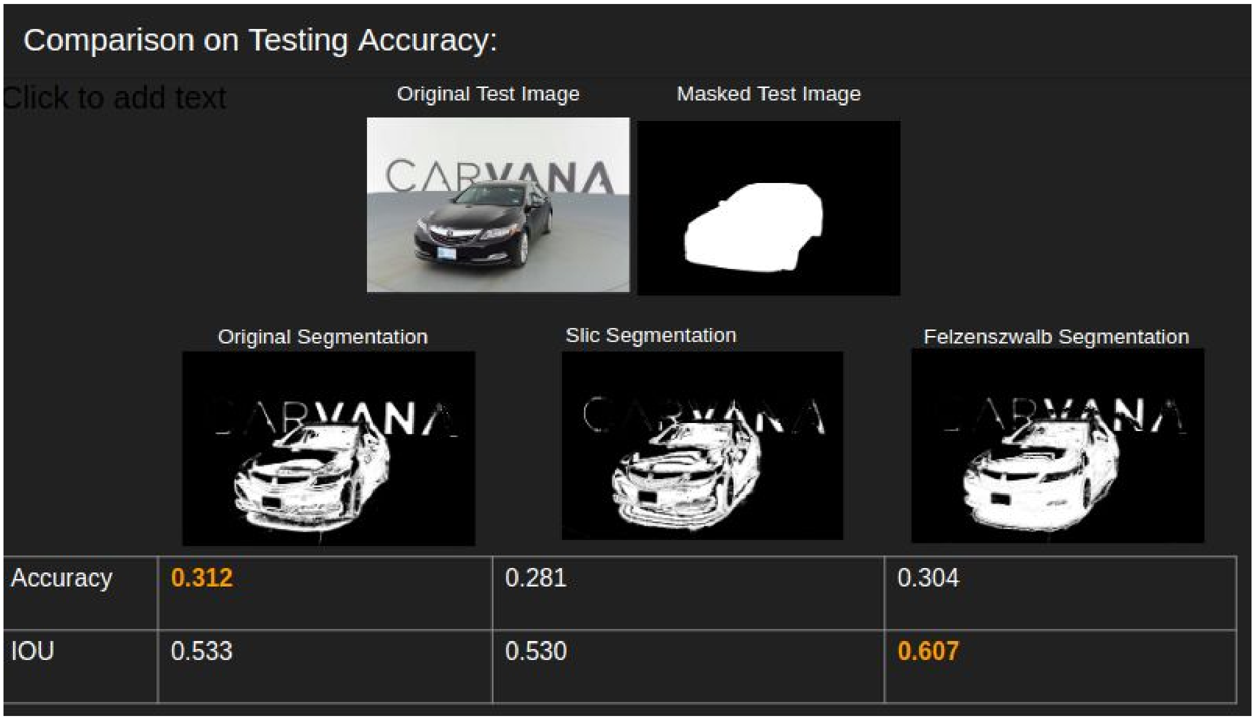
Inference (on Test Image)



Comparison on Training and Validation Accuracy:

Click to add text

	Original	Slic Segmentation	Felzenszwalb Segmentation
Training Accuracy	0.999	0.999	**1.0**
Validation Accuracy	0.988	**0.991**	0.990



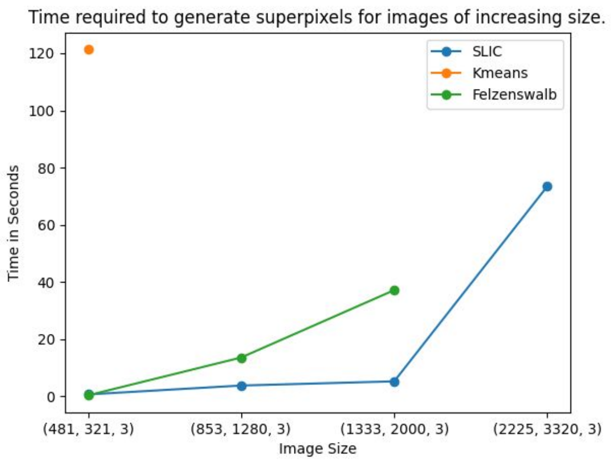
Comparison on Testing Accuracy:

Click to add text

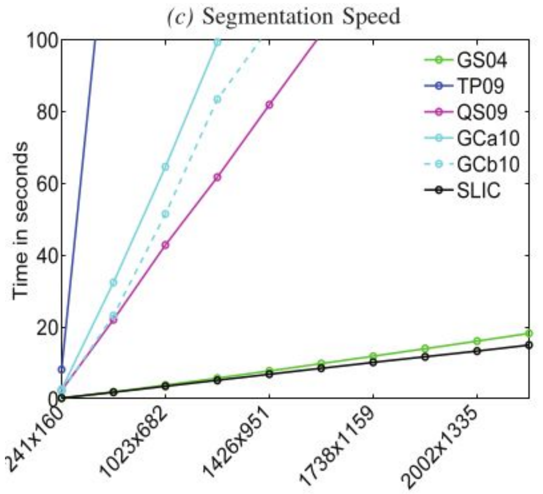
	Original Segmentation	Slic Segmentation	Felzenszwalb Segmentation
Accuracy	**0.312**	0.281	0.304
IOU	0.533	0.530	**0.607**

Results: Computational and Memory Efficiency

Superpixels are often used to replace the pixel-grid to help speed up other algorithms. Thus, it is important that superpixels can be generated efficiently in the first place. A comparison is shown in the figure below between the time required for the various superpixel methods to segment images of increasing size.



Results generated by me



Results given in the paper

Time required to generate superpixels for images of increasing size. SLIC is the fastest superpixel method, followed closely by GS04, and then a significant gap. NC05 is not plotted due to its particularly slow speed.

Possible Limitation of SLIC

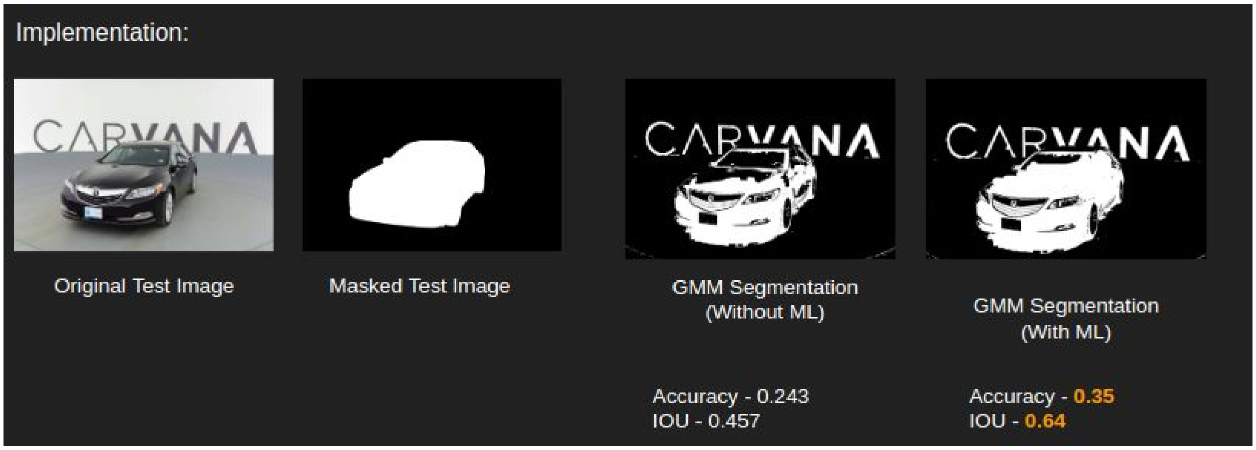
1. SLIC uses a fixed search region of 2SX2S to reduce computation complexity, but what about the boundary pixels.
2. The assignment and update steps are performed separately which can lead to low convergence rate because separately performing the assignment step and the update step also leads to a delayed feedback of pixel label change.
3. Hard Clustering which means each pixel belongs to either one of the clusters.

Implementation Possible Solutions of SLIC

1. Probabilistic clustering algorithms or soft clustering techniques in which each pixel belongs to every cluster with a certain probability. In

this way the clustering updates are done by expectation maximization algorithm.

2. **Solution: Gaussian Mixture Model**



Implementation:

Original Test Image

Masked Test Image

GMM Segmentation (Without ML)

Accuracy - 0.243
IOU - 0.457

GMM Segmentation (With ML)

Accuracy - **0.35**
IOU - **0.64**

The same set of experiments are performed with GMM algorithm and it produces prominent results in both with and without ML methods.

Conclusion:

Superpixels have become an essential tool to the vision community, and in this paper they provide the reader with an in-depth performance analysis of modern superpixel techniques. They performed an empirical comparison of five state-of-the-art algorithms, concentrating on their boundary adherence, segmentation speed, and performance when used as a preprocessing step in a segmentation framework. In addition, They proposed a new method for generating superpixels based on k-means clustering, SLIC, which has been shown to outperform existing superpixel methods in nearly every respect.
Although their experiments are thorough, they come with a caveat. Certain superpixel methods, specifically, GC10 and TP09, do not consider color information, while the other methods do. This may adversely impact their performance.

I have also pointed out some of the limitations of the SLIC algorithm and further tried to implement some of the solutions of those limitations for example implementing the GMM algorithm which considers the drawback of hard clustering of the SLIC algorithm.

Thank You