# UNIVERSITY OF BURGUNDY

DIGITAL IMAGE PROCESSING PROJECT

**SLIC SUPERPIXELS COMPARED TO**
**STATE-OF-THE-ART SUPERPIXEL METHODS**

Submitted by

Gopikrishna Erabati

Anirudh Puligandla

Avinash N.P.N

Supervisor:

Dr. Désiré Sidibé

# Contents

# I. Introduction

Superpixel is a polygonal part of a digital image, larger than a normal pixel, that is rendered in the same color and brightness. Superpixels are becoming increasingly popular for use in computer vision applications. They have become key building blocks of computer vision algorithms. It is not always defined what constitutes a good superpixels algorithm. However, there are few algorithms that output a desired number of regular, compact superpixels with a low computational overhead.

*Why Superpixels ?*

1. They group pixels into meaningful regions which can be used to replace rigid structure of pixel grid.
2. They capture image redundancy.
3. They provide convenient first stage to compute image features.
4. They greatly reduce complexity of subsequent image processing tasks.

Because of the above stated reasons superpixels have become significant in computer vision.
As already stated that it is difficult to define what constitutes a good algorithm for superpixel generation for a certain application, the following properties are generally desirable.

1. When used to reduce computational complexity as a earlier step, superpixel algorithm should be fast, memory efficient and simple to use.
2. It should improve the segmentation performance.
3. Superpixels should adhere well to image boundaries.

Adherence to boundaries:

Superpixel is an image region containing similar pixels. An image boundary/edge is a boundary between different parts of an image (not outside boundary). So, good superpixel will have all edges in image covered by superpixel boundaries, because otherwise, this would mean that there is edge going through a superpixel, i.e., superpixel would span two different regions, that is not compatible with superpixel definition. As they become large and large, they extend through weak edges in image, strong edges however, still be covered by superpixel boundaries.

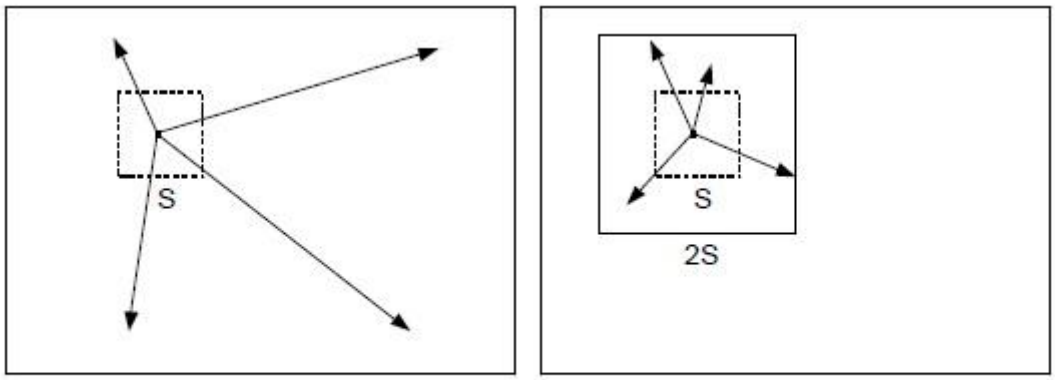
We, performed analysis of state-of-the-art superpixel methods like TP09, GS04, QS08, evaluating their speed, adhere to image boundaries. But, no existing methods were providing our desirable properties. We, implemented superpixels using simple linear iterative clustering (SLIC), which adapts k-means clustering. This method is easy to use, offers flexibility in compactness and number of superpixels it generates, and is freely available.

# II. SLIC Superpixels

SLIC is an adaptation of k-means clustering for superpixel generation, with two important distinctions:

1. The number of distance calculations is reduced by limiting the search space to a region proportional to the superpixel size.

2. The distance measure is changed to give importance to color and spatial proximity by a weighting factor, which simultaneously provide control over compactness and size of superpixel.




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

*(b)* SLIC searches a limited region

Figure 1 Reducing the superpixel search region.

## A. Algorithm

*/*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 x 3 neighborhood.
Set label *l(i)* = -1 for each pixel *i*.
Set distance *d(i)* = ∞ for each pixel *i*.

**repeat**
  */*Assignment*/*
  **for** each cluster center $C_k$ **do**
    **for** each pixel *i* in a 2S x 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 ≤ threshold

SLIC is provided with only one parameter i.e., *k*, the desired number of approximately equally sized superpixels. It is simple to use and understand. In the first step, we initialize *k* cluster centers by sampling the regular grid space S pixels apart. The grid interval is $S = \sqrt{N/k}$ is chosen to produce roughly equally sized pixels. The centers are moved to lower gradient position because lower gradient

position have less probability of having an edge and as superpixel centre cannot be on an edge for superpixel to exist.

In the next step, each pixel *i* is associated with the nearest cluster centre whose search region overlaps its location, as shown in Figure 1. By limiting the search space we significantly reduce the distance calculations required and thereby increasing the speed of algorithm. This is the main advantage over k-means clustering, as in k-means clustering each pixel is compared with all cluster centers. This is possible for SLIC by the introduction of a distance measure D, which determines nearest cluster centre for each pixel.

In the later step, as each pixel is associated with the nearest cluster centre, an update step adjusts the cluster centre to the mean vector of all the pixels belonging to the cluster. Now, we have two cluster centre and an error E ($L_2$ norm) is calculated between new cluster centre locations and previous cluster locations. The assignment and update steps can be repeated until the error converges.

## B. Distance Measure

SLIC superpixels takes into account *labxy* color-image plane space. This makes a problem while defining distance measure D. By defining D to be fice dimensional Euclidean distance in *labxy* color space will cause inconsistencies in clustering for different superpixel sizes. For large superpixels, spatial distance over weighs color, thereby giving more importance to spatial proximity than color proximity, which gives rise to more compact pixels that do not adhere well to image boundaries.

Therefore, we normalize the color and spatial proximity by their respective maximum distances within a cluster $N_s$ and $N_c$.

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

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

The maximum spatial distance in a cluster $N_s = S = \sqrt{N/k}$, whereas the maximum color distance varies from image to image and cluster to cluster. This problem is avoided by fixing maximum color distance to a constant *m.* The simplified distance measure we use in practice is

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

In the distance measure D as above, *m* allows us to weigh the relative importance between color and spatial proximity. When *m* is large, spatial is given more important and resulting superpixels are more compact. When *m* is small, the superpixels adhere more to image boundaries. *m* can be in the range [1,40].

This procedure can be adapted to grey scale images by putting both 'a' and 'b' values to zero. It can also be extended to handle 3D supervoxels, by including the depth information to spatial proximity.

About the complexity of this algorithm, by localizing the search in the clustering procedure, it avoids performing thousands of redundant distance calculations. A pixel can fall in neighborhood of less than

eight cluster centers, so SLIC is O(N) complex. SLIC is linear in the number of pixels, irrespective of *k*.

# III. Results and Discussions

Here, we tried to do a comparison of different superpixel algorithms namely SLIC,TP09, GS04, QS09. We did a parameter analysis of these algorithms and the results are given as below:

## A. SLIC

In SLIC algorithm, the parameters we provide to algorithm are an image to form the superpixels, number of superpixels and weighing factor *m* as we discussed earlier, which provides control over compactness and adherence to boundaries.

We have used authors code to generate labels of superpixels and we have developed a code to use these labels and form boundaries and thereby overlap these boundaries on original image to show superpixels on the image.

The results of SLIC for different values of *m* are as shown in Figures 2,3 and 4. We can observe from Figure 2 (*m* = 2), the superpixels which are generated are less compact and more adhere to image boundaries. In Figure 3 (*m* = 12), the superpixels are optimum compact and has good adherence to boundaries. In Figure 4 (*m* = 37), the superpixels are more compact in size but they badly adhere with the boundaries. So, by this parameter analysis ,we can infer that by varying values of *m* , we can have a control over compactness and adherence to image boundaries of the superpixels depending on our needs.

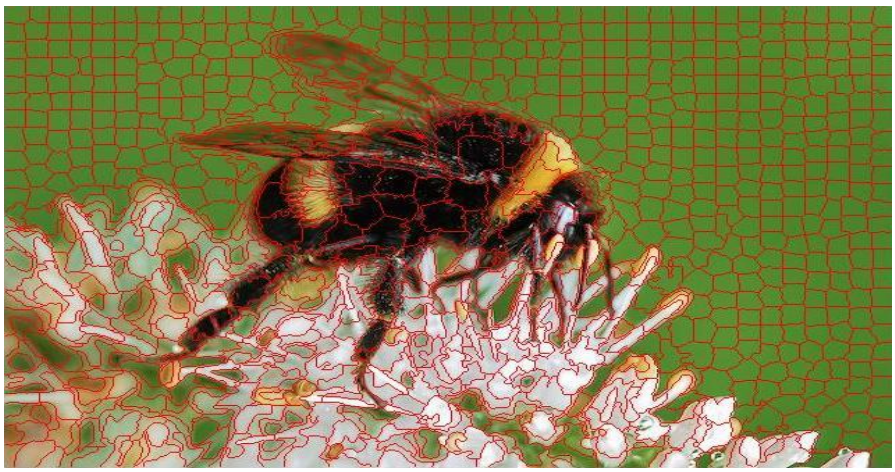


Figure 2. SLIC Superpixel (m = 2) more adherence to boundary and less compact in size.

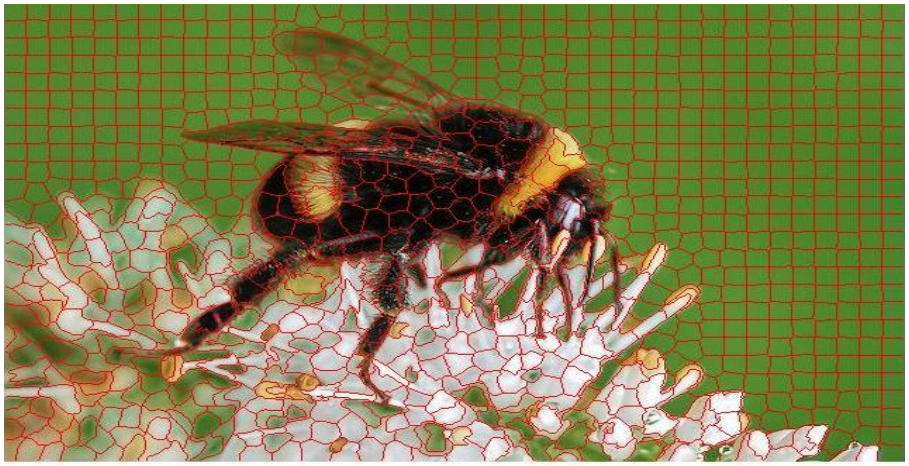


Figure 3. SLIC Superpixel (m = 12) good adherence to boundary and good compact in size.

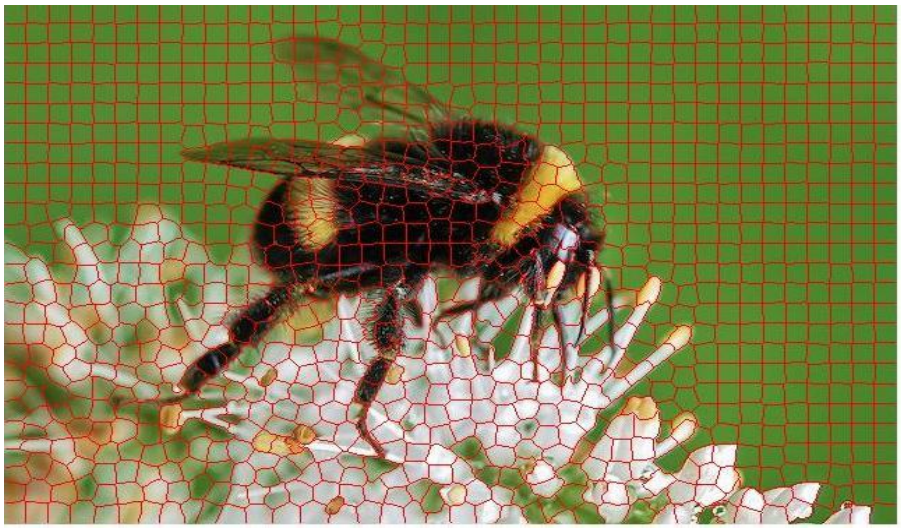


Figure 4. SLIC Superpixel (m = 40) bad adherence to boundary and good compact in size.

In SLIC algorithm, we can also control the approximate number of superpixels formed by entering the number of superpixels. Figures 5,6 and 7 show the superpixels of size 64, 256 and 1024 pixels respectively.

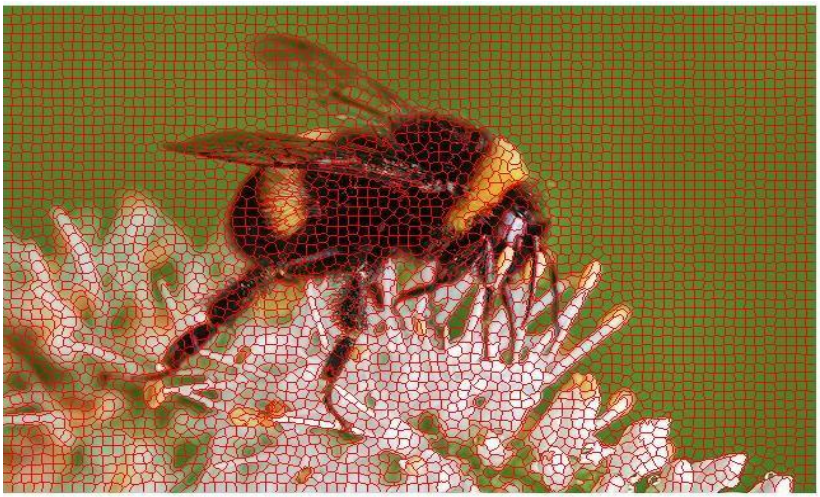


Figure 5. SLIC Superpixel (m = 12) with superpixel size of 64 pixels.

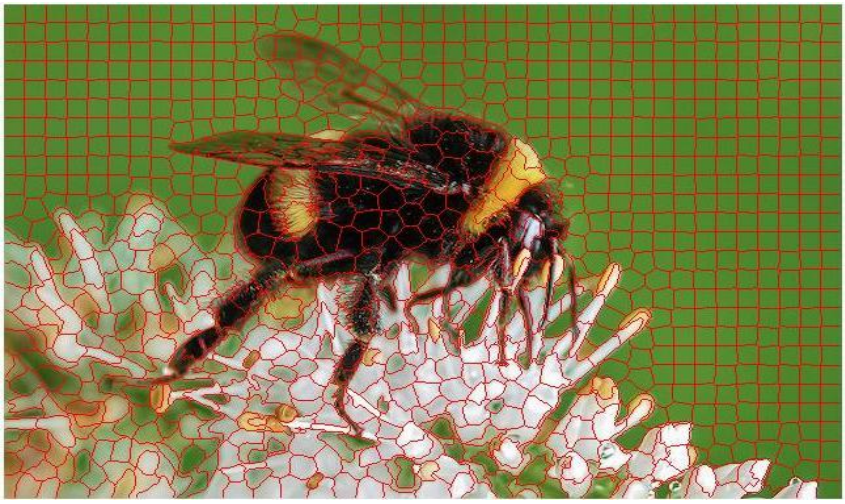


Figure 6. SLIC Superpixel (m = 12) with superpixel size of 256 pixels.

Figure 7. SLIC Superpixel (m = 12) with superpixel size of 1024 pixels.

## B. TP09

In Turbopixels algorithm, there is only one parameter to tune and that is number of superpixels to form. So, it provides direct control over number of superpisels. Although from Figures 8,9 and 10, we can infer that it provides compact and consistent superpixel but it has poor adherence with the boundary. It has slow running time and almost 50 times slower than SLIC algorithm for a 425 x 640 image.

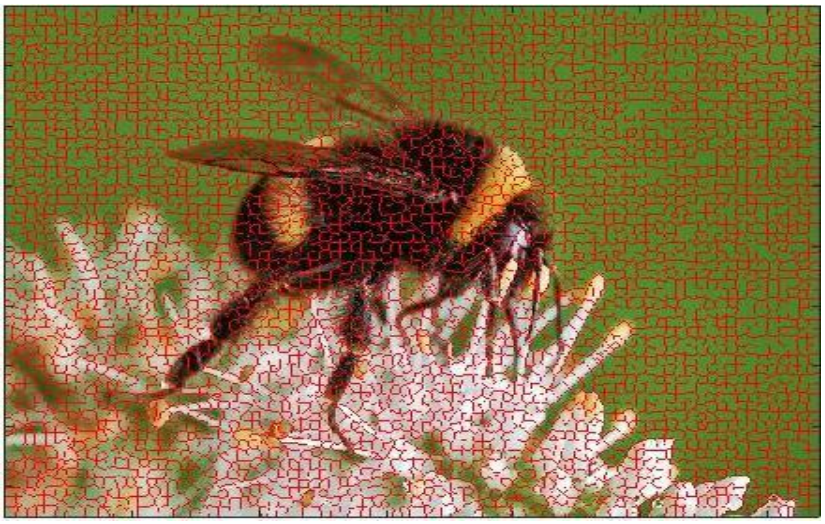


Figure 8 TP09 algorithm superpixel with 64 pixels size

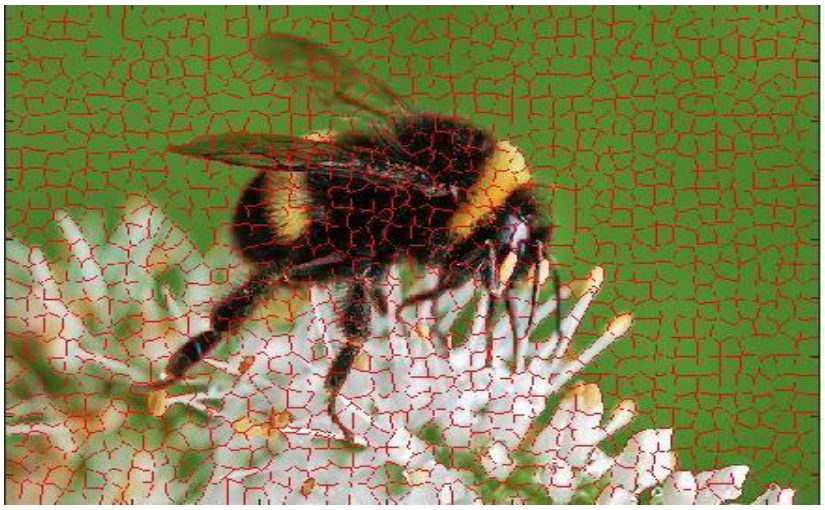


Figure 9 TP09 algorithm superpixel with 256 pixels size

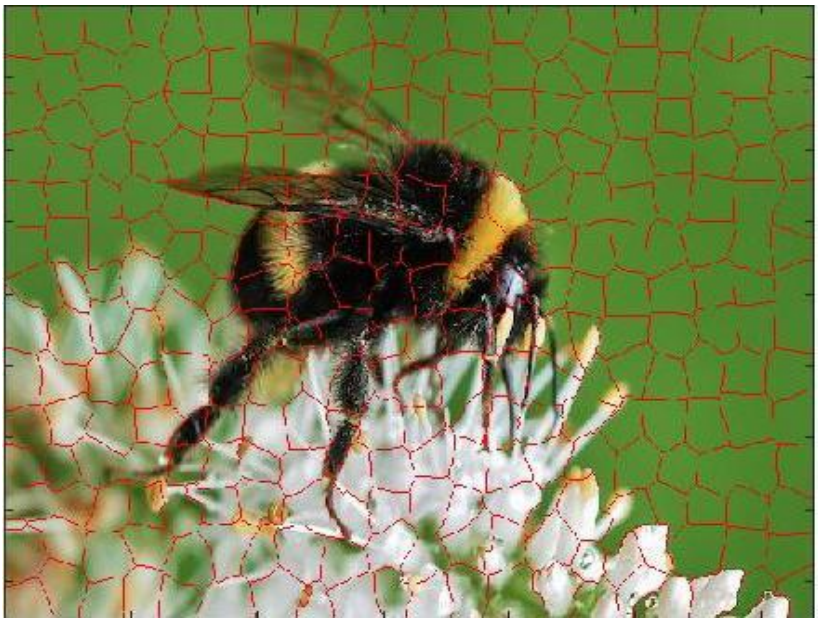


Figure 10 TP09 algorithm superpixel with 1024 pixels size

## C. GS04

The GS04 algorithm takes four parameters and input, the input image, sigma value to smooth the image before segmentation, k scalar parameter for the threshold function , min_size: parameter for minimum component size enforced by post-processing. However, it does not allow to control the number of superpixel and compactness of superpixel. From Figures 11,12 and 13 , we can see that we can control the smoothing of image so that narrow edges gets smoothed before performing segmentation.

Figure 11 GS04 algorithm with zero smoothing factor

Figure 12 GS04 algorithm with 0.5 smoothing factor

Figure 13 GS04 algorithm with 1 smoothing factor

Figure 14 GS04 algorithm with 100 threshold factor

Figure 15 GS04 algorithm with 1000 threshold factor

From Figures 14 and 15 , we can infer that by increasing the threshold factor we can super pixel of higher size but with poor compactness. However, this GS04 algorithm adheres well to image boundaries.

## D. QS09

The QS09 algorithm takes four different parameters, in which a parameter 'ratio' is used to tradeoff between color importance and spatial importance (larger values give more importance to color). From Figures 16 and 17 we can see that as we increase the 'ratio' parameter the superpixel well adhere to image boundaries. This algorithm requires several non-intuitive parameters to be tuned. It does not offer control over compactness of superpixels. This takes lot of time to run as compared to other methods.

Figure 16 QS09 algorithm with ratio = 0.1

Figure 17 QS09 algorithm with ratio = 0.9

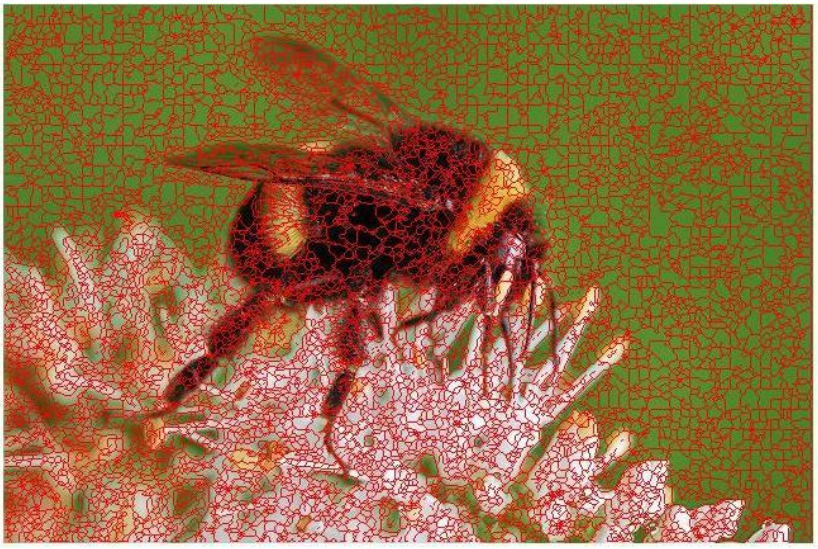


Figure 18 QS09 algorithm with kernal size = 2

Figure 19 QS09 algorithm with kernal size = 10

From figures 18 and 19, we can infer by changing the kernel size parameter which is used to estimate the density, we get more large superpixel but as with QS09 we are unable to get compact superpixel, however adherence is maintained well.

Summary of superpixel algorithms

| | TP09 | GS04 | QS09 | SLIC |
|---|---|---|---|---|
| control over amount of superpixels | Yes | No | No | Yes |
| control over compactness | No | No | No | Yes |
| Speed | 25s | 2.5s | 59s | 0.45s |
| Adherence to boundary | bad | good | average | optimum |
| Supervoxel extension | No | No | No | Yes |

# IV. GUI

We had also designed a GUI to compare all the algorithms.

The screenshot GUI is given below:

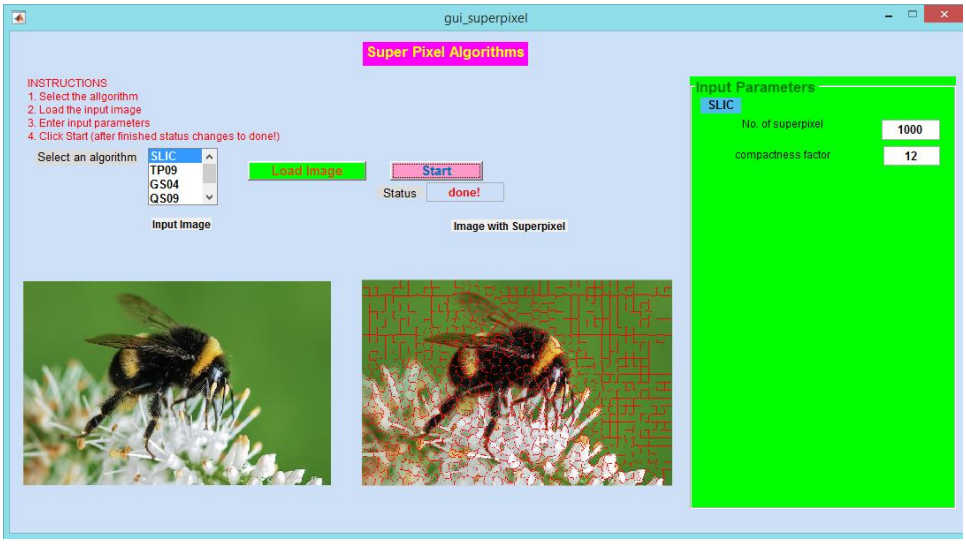



Figure 20 The GUI

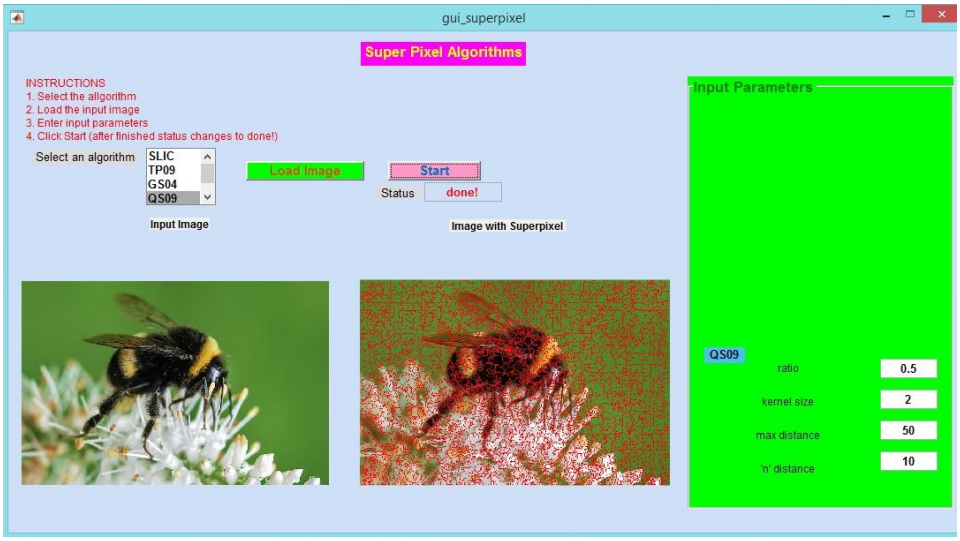



Figure 21 The GUI - 1

# V. Conclusion

SLIC algorithm based on k-means clustering, has outperformed existing superpixel methods. It is simple to use and understand. It is fast as we limit the search space. It adheres well to image boundaries. So, we can say that SLIC superpixel algorithm has all the desirable properties required for a good superpixel method.