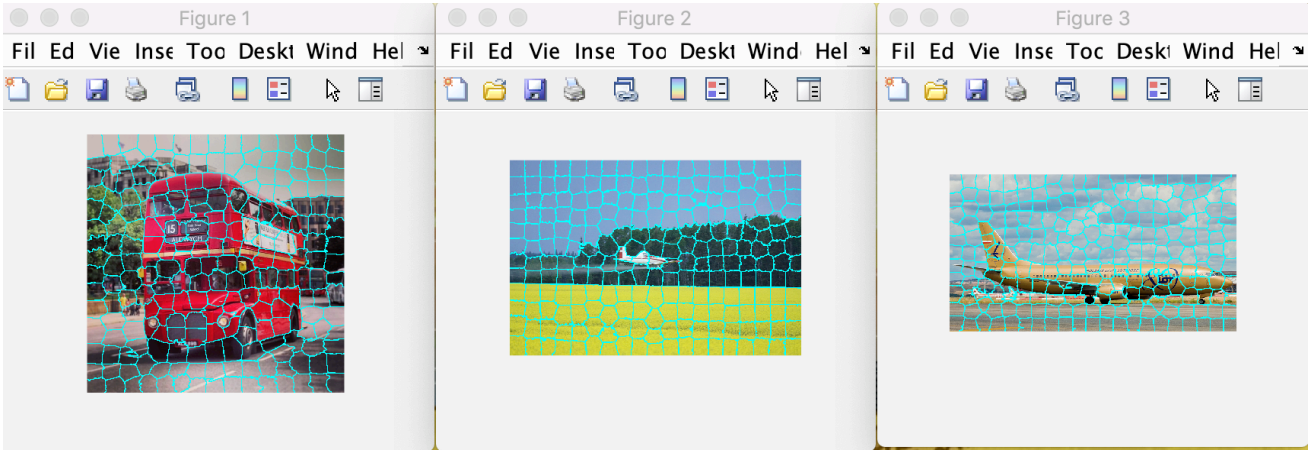
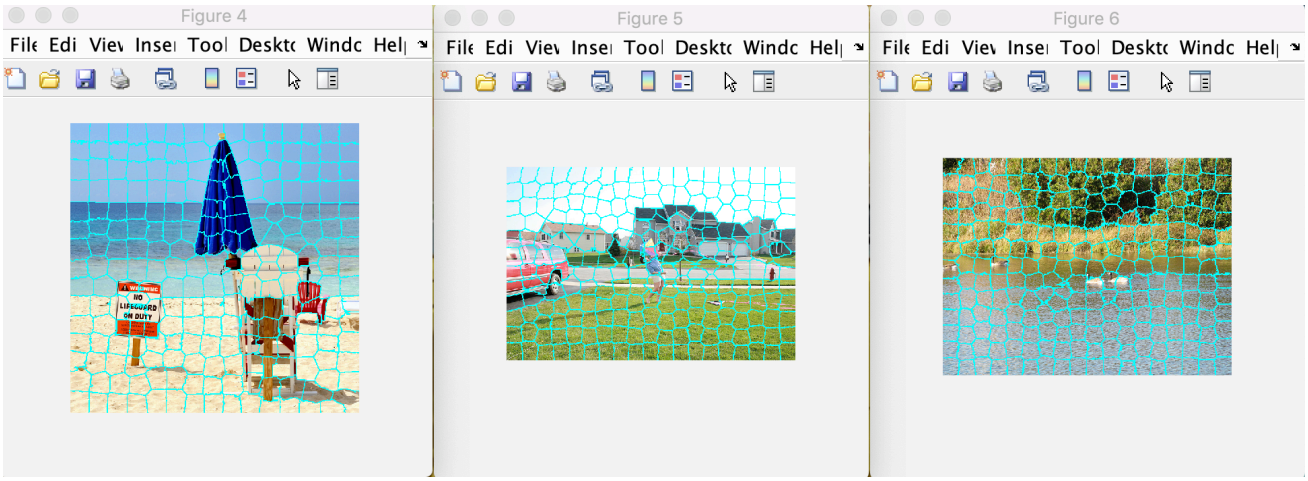
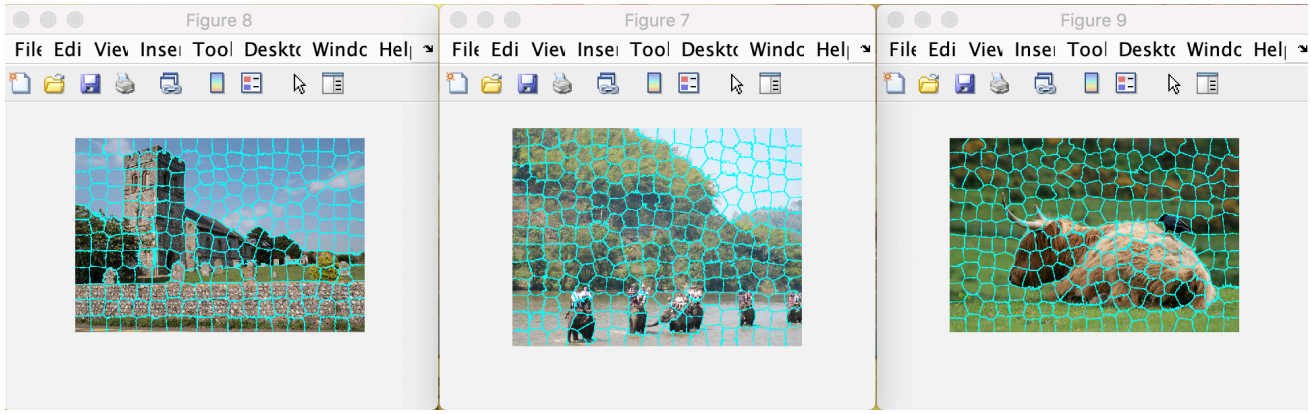
Bayram Muradov
21503664
CS 484 Image Analysis
16 December 2019

## Part 1: Superpixel segmentation

- *Obtained Segmentation Results (Parameters used: n=248):*



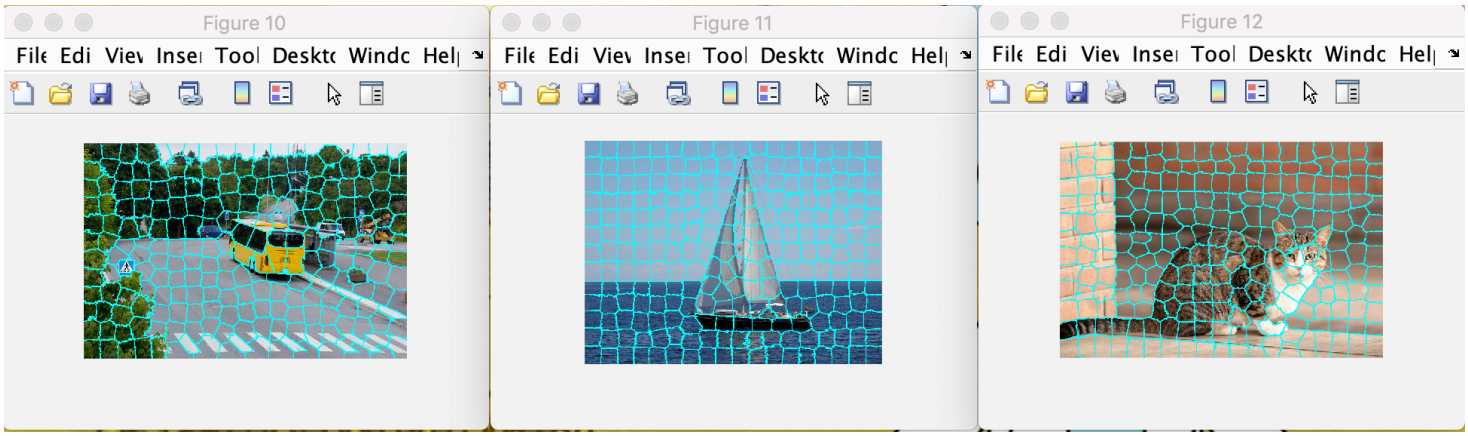


Figure 10
Figure 11
Figure 12

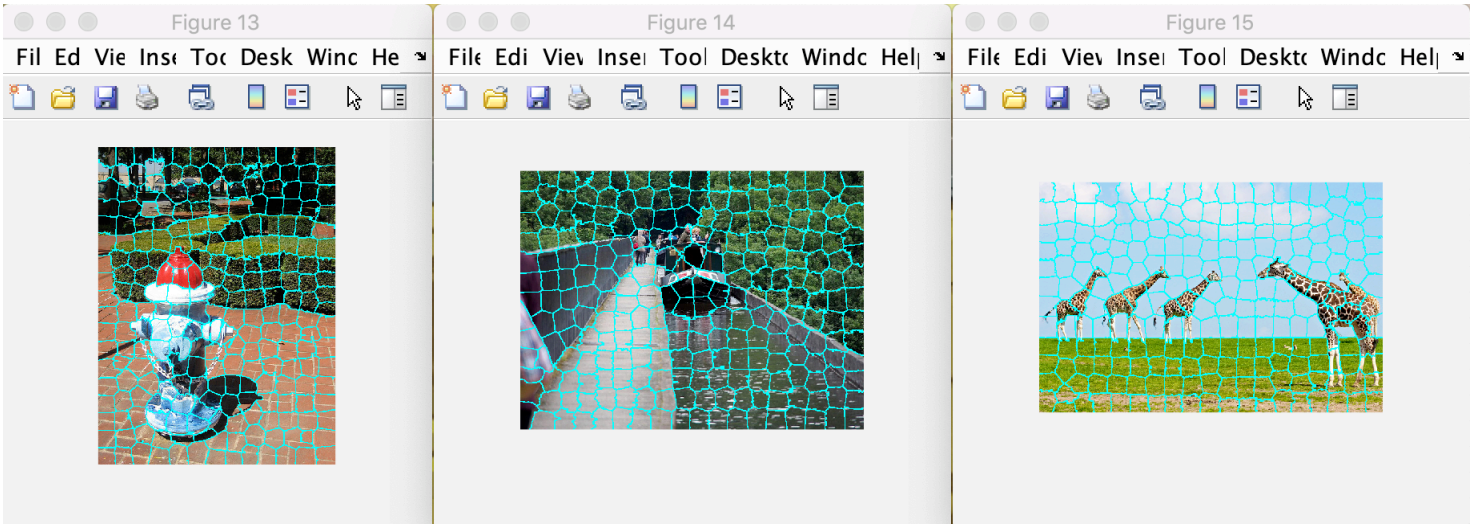


Figure 13
Figure 14
Figure 15

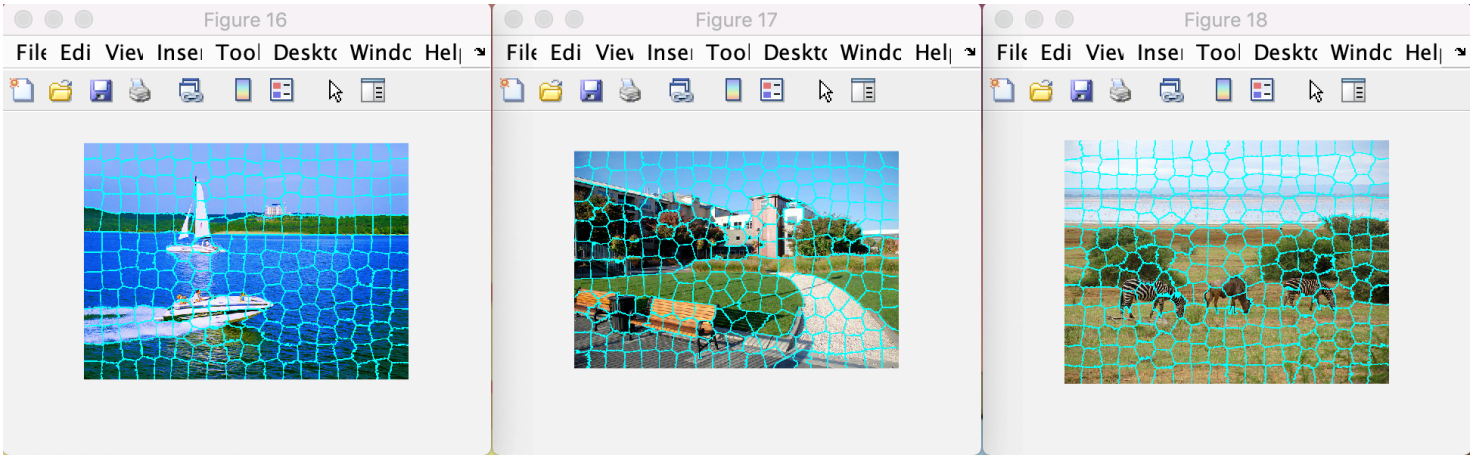


Figure 16
Figure 17
Figure 18

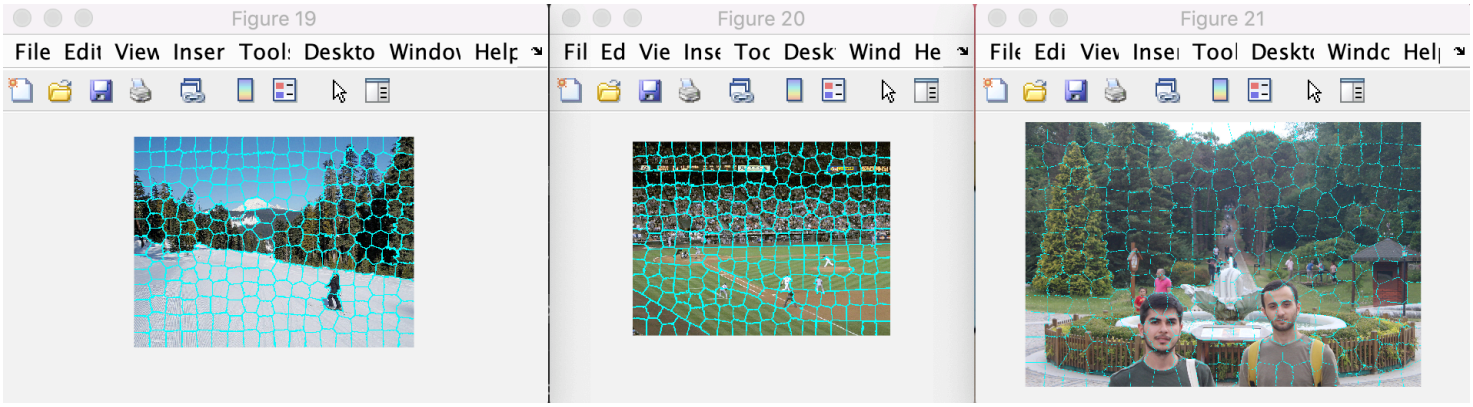


Figure 19
Figure 20
Figure 21

• *Label Counts for each image:*

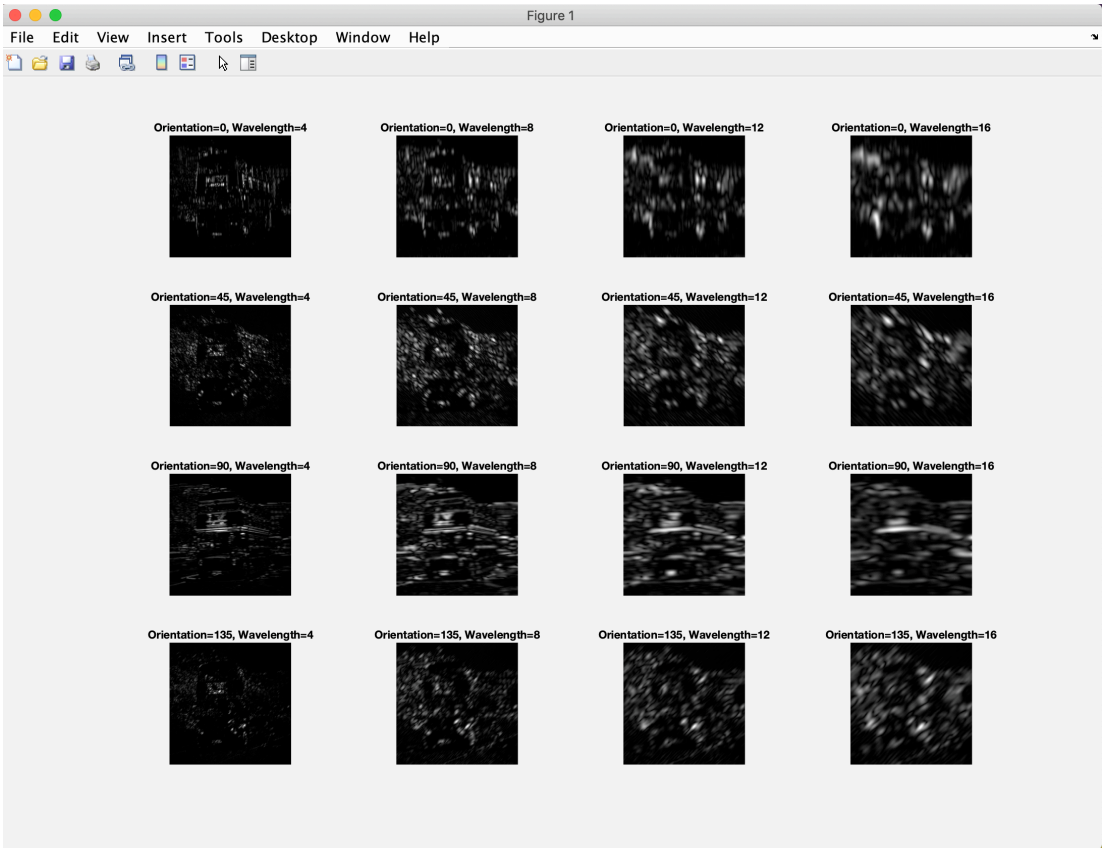
```
MEX completed successfully.
** reading image : 000000001584.jpg **
== label count:225 ==
** reading image : 000000002680.jpg **
== label count:228 ==
** reading image : 000000005477.jpg **
== label count:231 ==
** reading image : 000000006609.jpg **
== label count:225 ==
** reading image : 000000010061.jpg **
== label count:228 ==
** reading image : 000000023230.jpg **
== label count:234 ==
** reading image : 000000119641.jpg **
== label count:234 ==
** reading image : 000000120572.jpg **
== label count:228 ==
** reading image : 000000124108.jpg **
== label count:228 ==
** reading image : 000000179265.jpg **
== label count:228 ==
** reading image : 000000234656.jpg **
== label count:234 ==
** reading image : 000000255965.jpg **
== label count:228 ==
** reading image : 000000256770.jpg **
== label count:234 ==
** reading image : 000000257566.jpg **
== label count:234 ==
** reading image : 000000297356.jpg **
== label count:228 ==
** reading image : 000000384481.jpg **
== label count:234 ==
** reading image : 000000424312.jpg **
== label count:228 ==
** reading image : 000000516316.jpg **
== label count:234 ==
** reading image : 000000548358.jpg **
== label count:234 ==
** reading image : 000000551360.jpg **
== label count:234 ==
** reading image : arbo.JPG **
== label count:228 ==
```

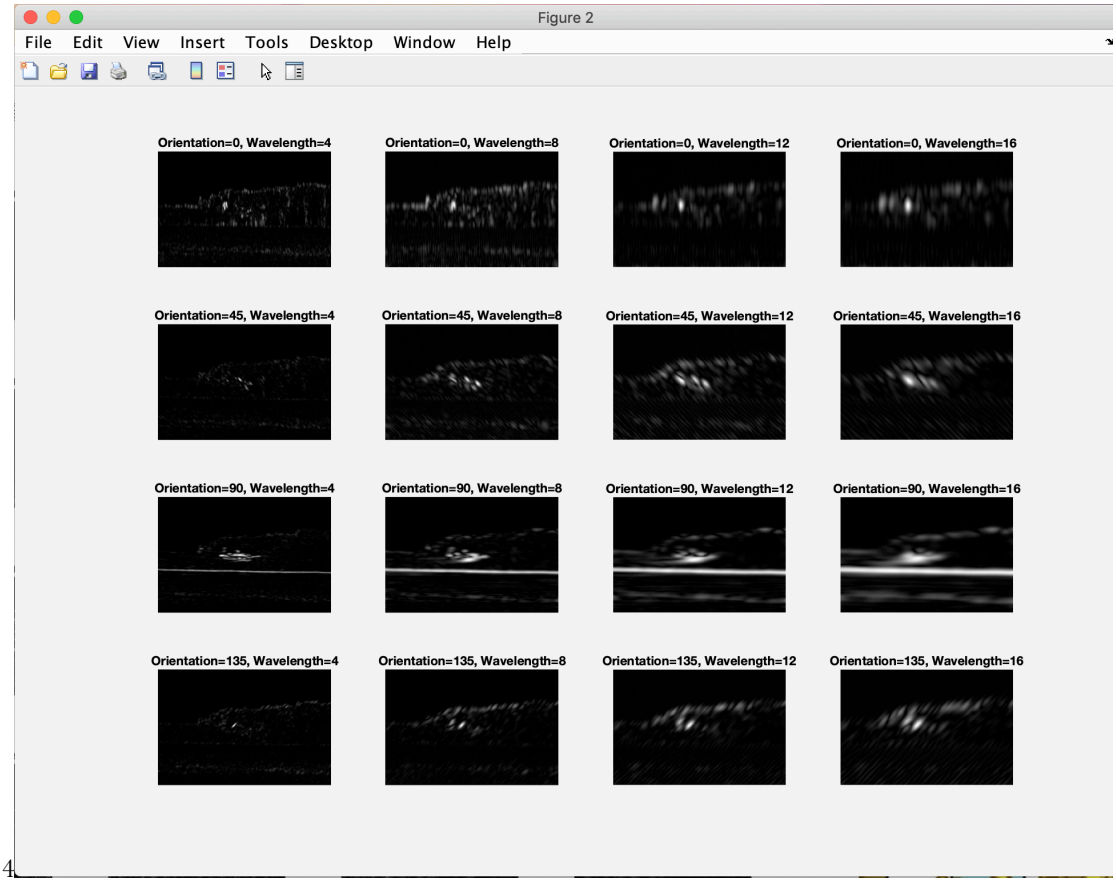
• *Followed Procedure & Discussion:*

After trying several parameters, each image in the given dataset has been over segmented by parameter n=248 in '*oversegment*' function provided in the .m file. '*oversegment*' function uses SLIC0 version of SLIC implementation since it is simpler in terms of arguments it takes as input and performs well in most of the cases. In order to use .cpp code given in the homework description( https://ivrl.epfl.ch/research-2/research-current/research-superpixels/), .cpp code was added to the working directory and complied inside of .m file.

## Part 2: Gabor Texture Feature

- *Obtained Results for 1st 5 images: (Parameters used: scale: [4 8 12 16], orientation: [0 45 90 135])*



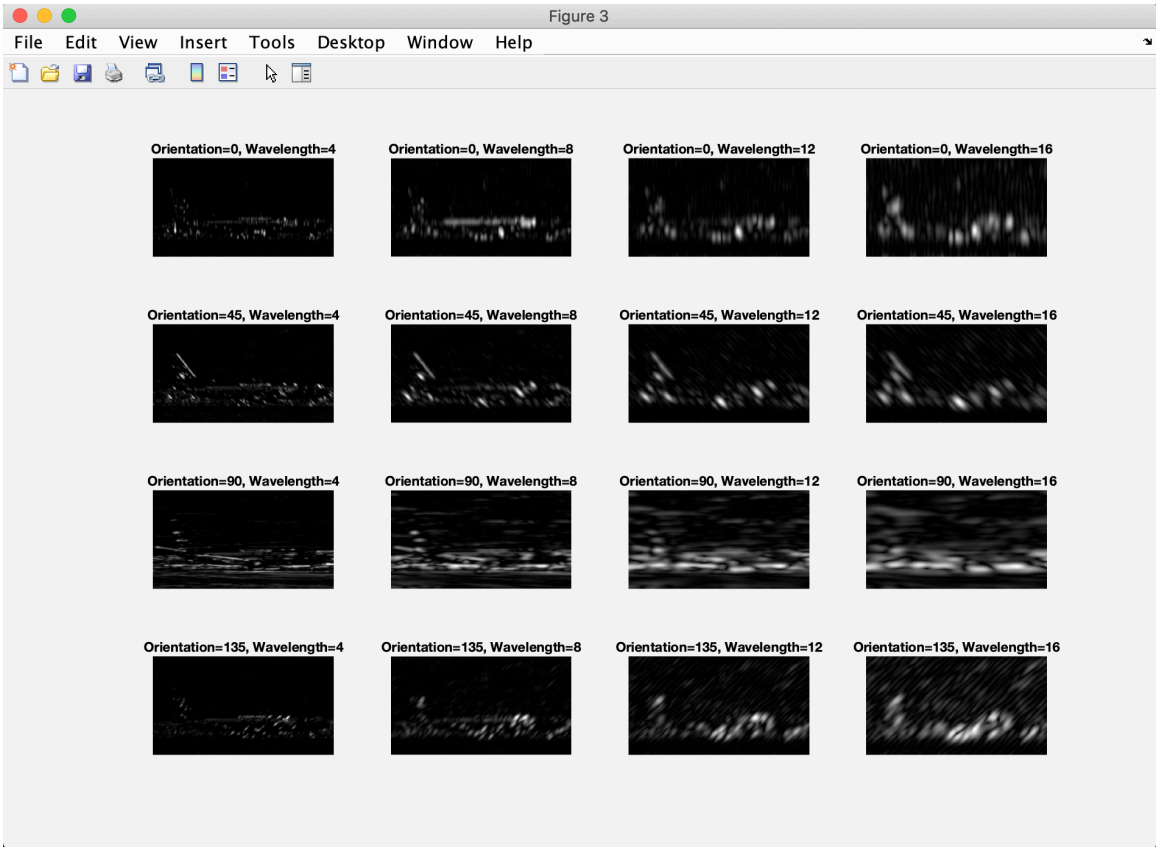


Figure 3
File Edit View Insert Tools Desktop Window Help
Orientation=0, Wavelength=4
Orientation=0, Wavelength=8
Orientation=0, Wavelength=12
Orientation=0, Wavelength=16
Orientation=45, Wavelength=4
Orientation=45, Wavelength=8
Orientation=45, Wavelength=12
Orientation=45, Wavelength=16
Orientation=90, Wavelength=4
Orientation=90, Wavelength=8
Orientation=90, Wavelength=12
Orientation=90, Wavelength=16
Orientation=135, Wavelength=4
Orientation=135, Wavelength=8
Orientation=135, Wavelength=12
Orientation=135, Wavelength=16

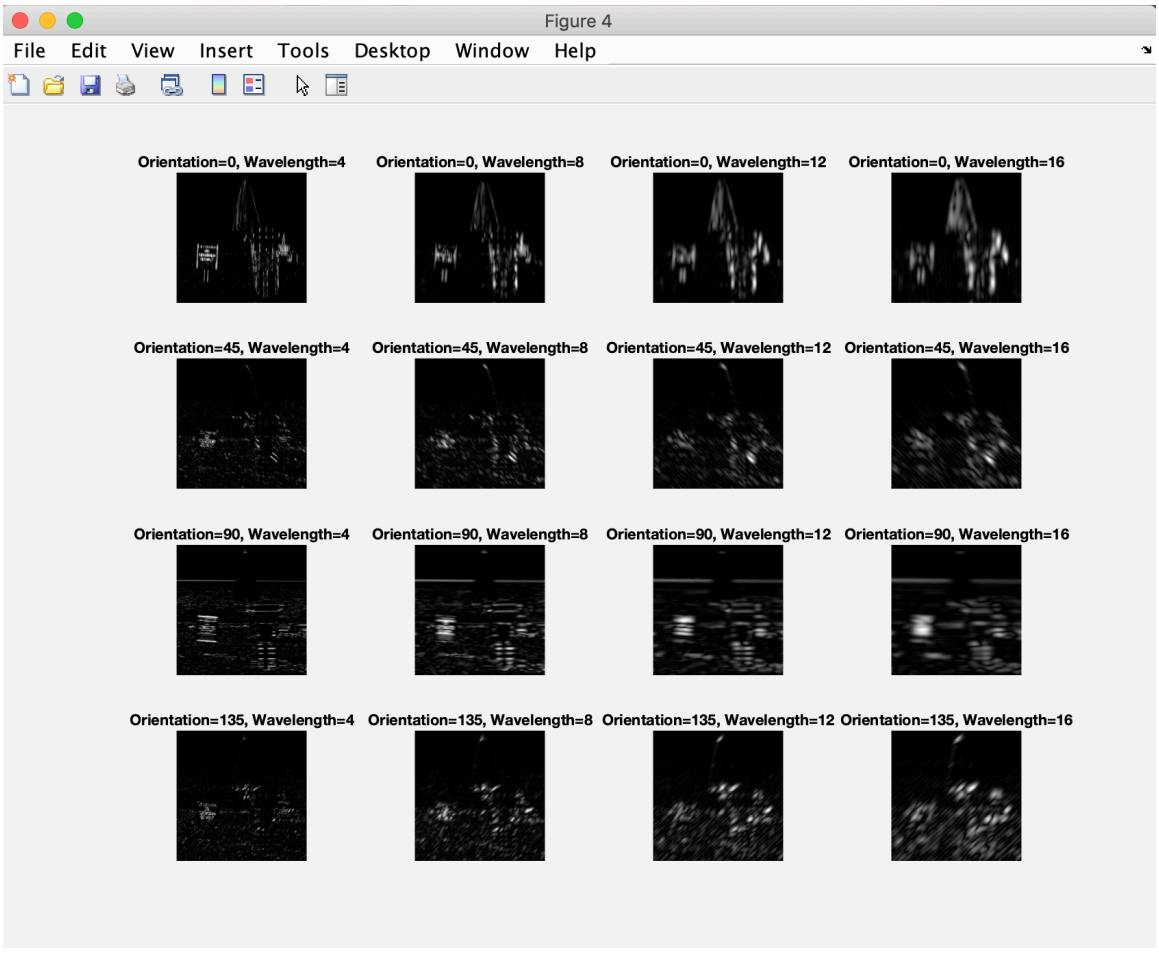
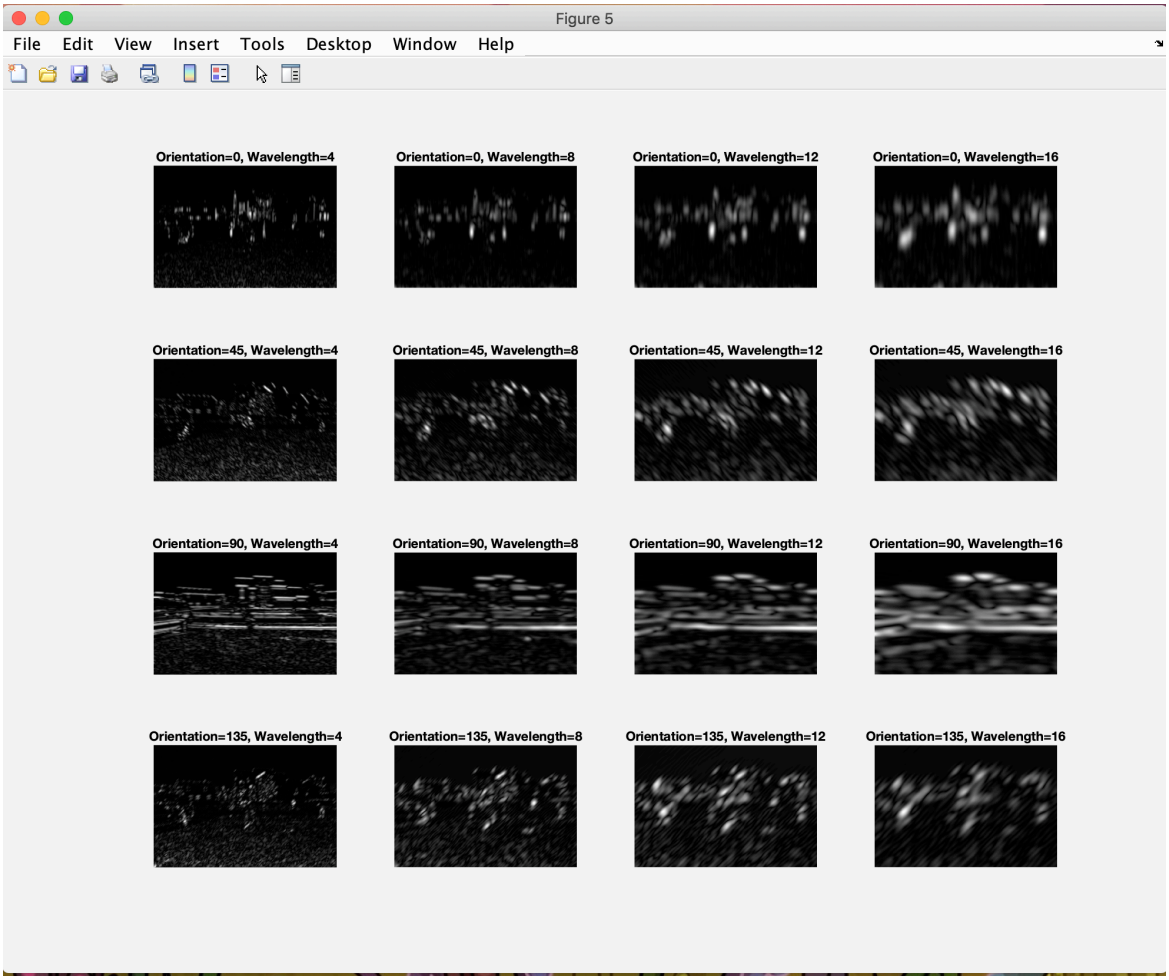


Figure 4
File Edit View Insert Tools Desktop Window Help
Orientation=0, Wavelength=4
Orientation=0, Wavelength=8
Orientation=0, Wavelength=12
Orientation=0, Wavelength=16
Orientation=45, Wavelength=4
Orientation=45, Wavelength=8
Orientation=45, Wavelength=12
Orientation=45, Wavelength=16
Orientation=90, Wavelength=4
Orientation=90, Wavelength=8
Orientation=90, Wavelength=12
Orientation=90, Wavelength=16
Orientation=135, Wavelength=4
Orientation=135, Wavelength=8
Orientation=135, Wavelength=12
Orientation=135, Wavelength=16



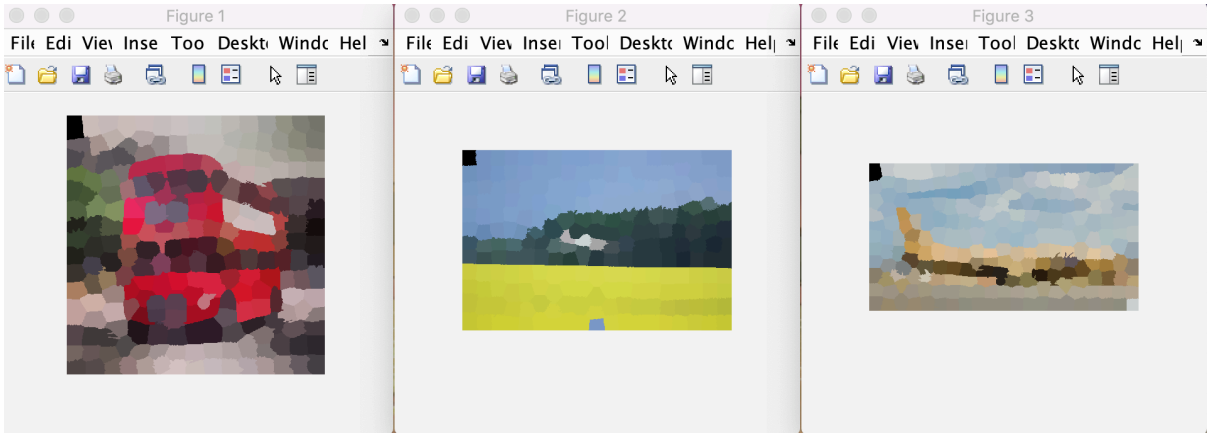

• *Followed Procedure & Discussion:*

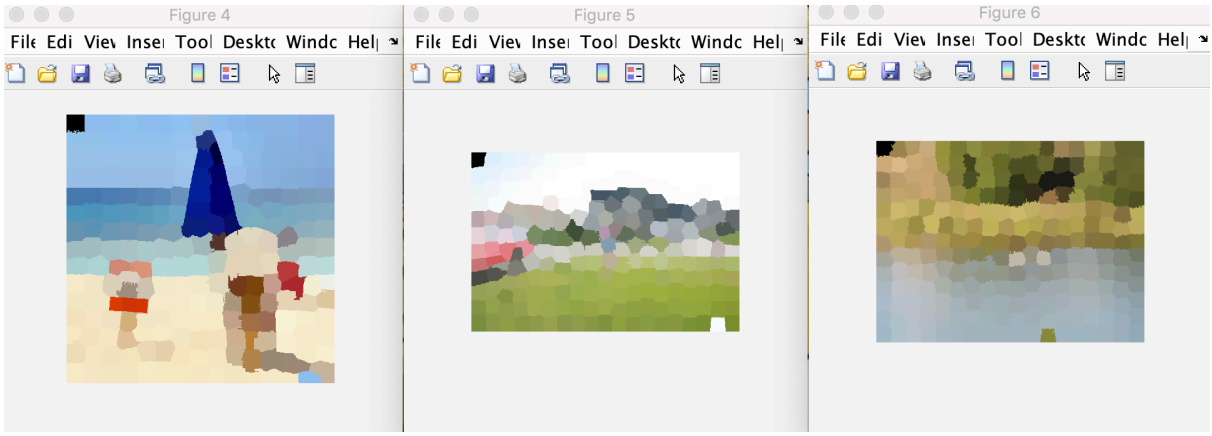
Each image in the dataset was converted to grayscale before calculation of Gabor texture features. After conversion, *'imgaborfilt'* method was used in order to compute Gabor filter responses with scale: [4 8 12 16] and orientation: [0 45 90 135]. These values have been chosen due to their default nature.

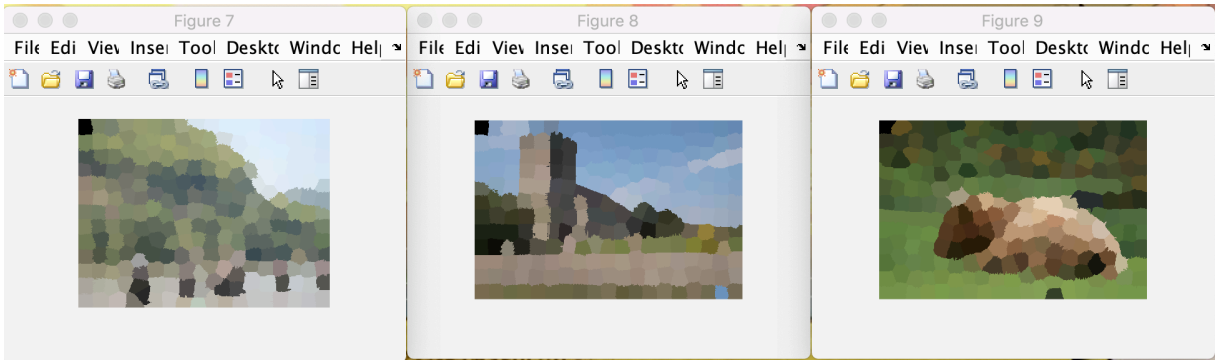
## Part 3: Region Merging Algorithm

- *Obtained Results: (Parameters used: threshold=5)*

1. *Using only Color Feature Vector*



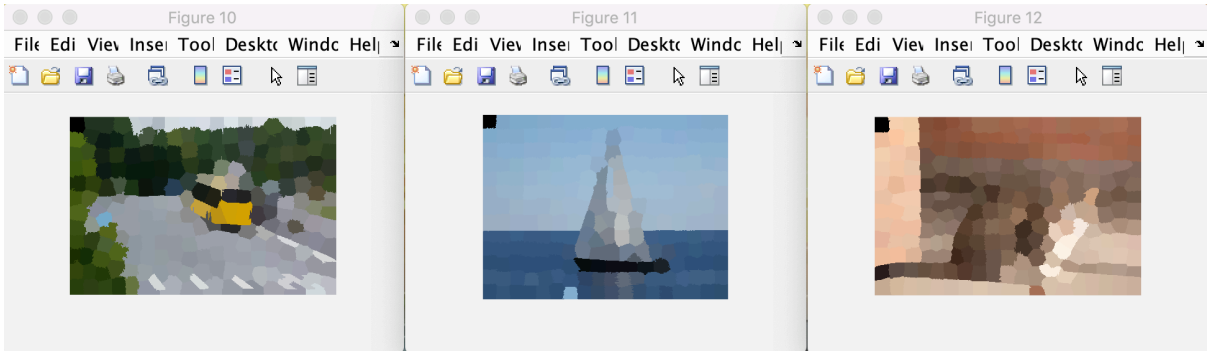


Figure 10
Figure 11
Figure 12

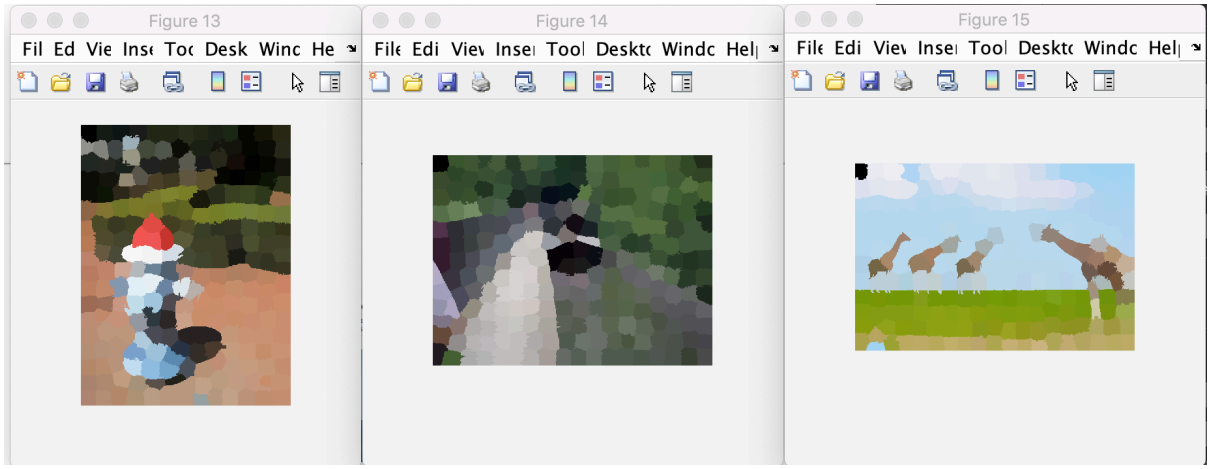


Figure 13
Figure 14
Figure 15

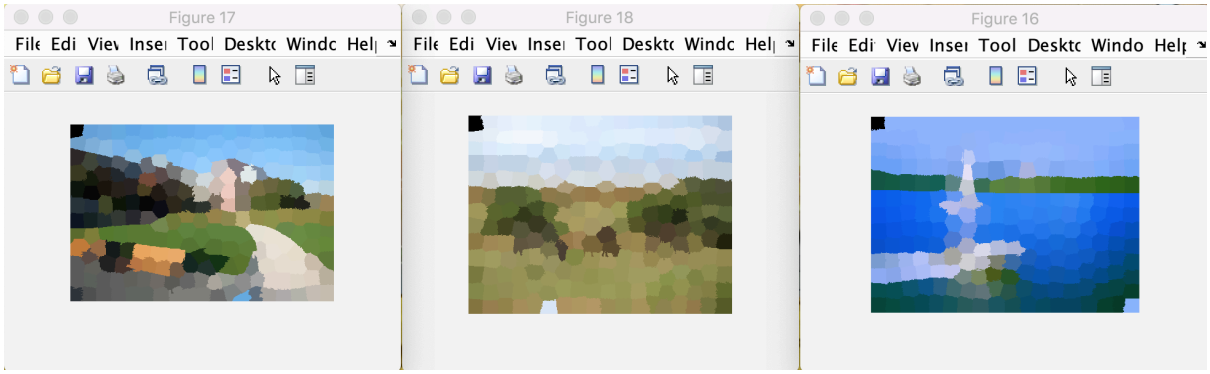


Figure 17
Figure 18
Figure 16

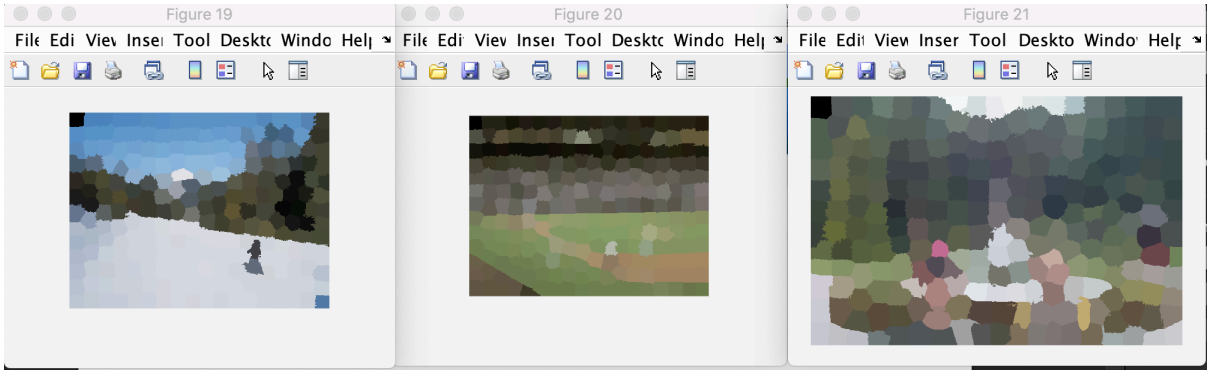
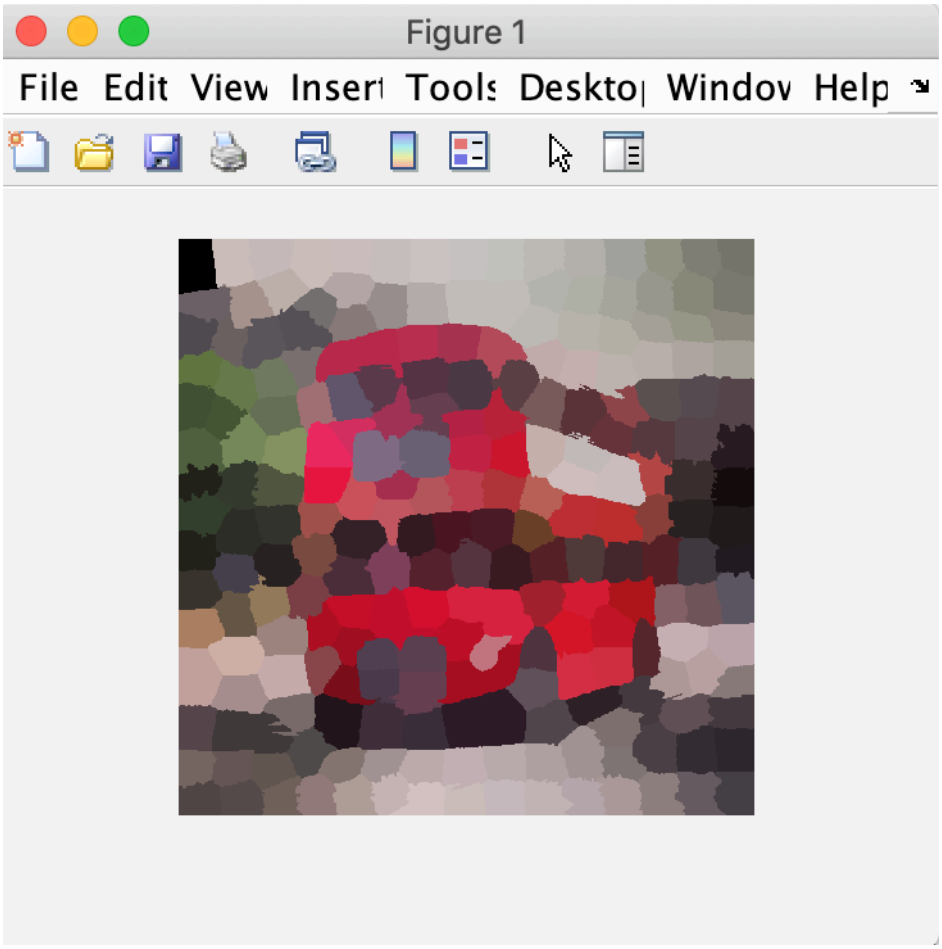


Figure 19
Figure 20
Figure 21

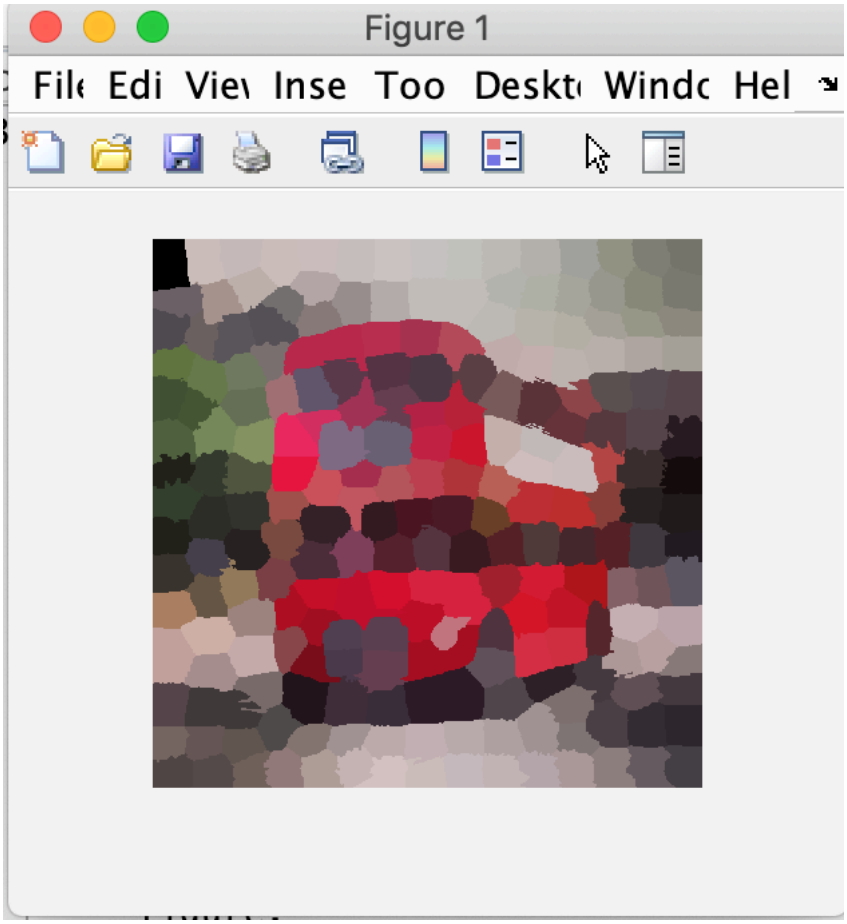
2. *Using only Texture Feature Vector*

*Important Note: Since this part of the algorithm is computationally heavy (takes approximately 4 minutes to compute texture fv. for all super pixels in a single image) only 1 example is attached to this report. In order to get merge of only 1 picture, values in the code have been hardcoded.*



*3. Using Color& Texture Feature Vector*

*Important Note: Since this texture part of the algorithm is computationally heavy (takes approximately 4 minutes to compute texture fv. for all super pixels in the image) only 1 example is attached to this report. In order to get merge of only 1 picture, values in the code have been hardcoded.*




• *Followed Procedure & Discussion:*

In the algorithm, first color feature (R, G, B) / texture feature vectors were calculated for each superpixel in the given images. After calculation, for each image, feature vectors have been iterated several times in order to merge the regions that have less Euclidian distance than the threshold value. Last step has been repeated until there was no region remained to be merged. In order to merge two adjacent regions, their average RGB values have been calculated and color vector has been updated with these average values in appropriate positions of the cell array. Black spots on top left corner of the images appear due to the boundary condition since my implemented algorithm assigns value of 0 to the pixels with no neighbors to achieve an appropriate stopping condition.