Manvi Goel
2019472

Computer Vision
Assignment 2

Language: Matlab

Question 1

Filename: m_2019472_q1.m
Write to execute: m_2019472_q1(*Image Name*)
Input: Name of the image
Sample Input: m_2019472_q1('0002.jpg')
Output: A saliency map is saved and displayed.

Assumptions.

1. The numbers of superpixels tried are

```
% The number of superpixels
numPixels = [10, 20, 50, 100, 200, 500, 1000, 1500, 2000];
```

2. The name of the output image is *2019472_Q1.jpg*
3. The path needs to be set as required.

```
% Making the features dataset using the image dataset.
% Loading the images. Specify the variable values.
path = 'C:\Users\hp\Desktop\Manvi\Semesters\6th_WinterSemester\ComputerVision\ComputerVision22\2019472_Assignment2\';
```

Working.

1. Read the image and calculate all (500) the superpixels in the image.
2. Calculate the mean color (RGB) and the mean coordinates (X-Y) for each superpixel.
3. For each superpixel, calculate the saliency value using the above values.
4. Use the saliency values to make a saliency map and normalize it using *mat2gray*.
5. Save and display the saliency map.

Analysis.

To evaluate the saliency maps generated using different numbers of superpixels and contrast the results, I calculate separation measures concentration measures and use visual inspection.

From the original image

As we generate saliency maps using superpixels, we notice that as we increase the number of superpixels from 10 to 10,000, the image starts resembling the original image.

Super pixels = 10 *to* *Superpixels = 2,000*

This happens because the size of superpixels becomes smaller as the number of pixels is increases. Thus it becomes easier for the saliency map to replicate the original image.

To make the superpixels extremely small, I attempt to increase the number of superpixels to 10,000.

Superixels = 10,000

Here, it can be seen that even the small parts of the background are prominent. Thus this is not a good example of the saliency map.

Next, I use the values of separation measure to judge the quality of the saliency maps. We know that separation measure quantifies the separation between the foreground and background distributions.

In this case, the value of the separation measure is comparable to 1 for the number of superpixels 10 to 200, but then the value goes down rapidly. It can be said that the foreground is completely separated in the former images, but the same can not be seen for the later ones. This can be confirmed from visual inspection also, where the saliency map for 20 superpixels,

Superpixels = 20

Shows a stark contrast between the foreground airplane and the background. While for a higher number of superpixels, other parts of the image can also be seen clearly, thus disturbing the distinction.

Similarly, as the background becomes clearer for concentration measures, the concentration measure goes down. This again can be seen from visual inspection, where more parts become clearer in the saliency maps.

The results from this would thus imply that the saliency maps with the least number of superpixels are mostly accurate. But we can not conclude this from the above results, as the measures are useful for evaluating the correctness of multiple saliency maps for one image. But, here, we are effectively getting saliency maps for very different images.

In the saliency with only 10 superpixels, it is hard to ascertain the original region of interest. The purpose of the saliency map is to highlight the region of interest. Thus, a tradeoff between measures and visibility of the region of interest needs to be made, keeping in mind the use of the saliency map.

Finally, considering the above factors, I consider the saliency map for 100 superpixels as most suitable for the task, which has high separation and concentration measures and highlights the region of interest while keeping the computation time reasonably low.

The final saliency map is shown below.

Final: *100 superpixels*.

References.

1. Matlab implementation of SLIC superpixels

Question 2

Filename: m_2019472_q2.m
Write to execute: m_2019472_q2(*Image Name*)
Input: Name of the image
Sample Input: m_2019472_q2('0002.jpg')
Output: 154 grayscale images with varying epsilon and min points values for analysis.

Assumptions.

1. The features taken for *dbscan* clustering are RGB colors and XY coordinates.
2. The names of the output images are of the form *(epsilon)-(min points).jpg* and are stored in the *2019472_Q2* folder in the output folder.
3. The path needs to be set as required.

```
% Making the features dataset using the image dataset.
% Loading the images. Specify the variable values.
path = 'C:\Users\hp\Desktop\Manvi\Semesters\6th_WinterSemester\ComputerVision\ComputerVision22\2019472_Assignment2\';
```

4. The following values for epsilons are tried.

```
[0.01, 0.02, 0.05, 0.08, 0.1, 0.12, 0.15, 0.2, 0.25, 0.3, 0.5, 1, 2, 10];
```

5. The following values for minimum points are tried.

```
[2, 5, 10, 15, 20, 25, 30, 40, 50, 80, 100]
```

Working.

1. Read the image and calculate the five features for each pixel.
2. Normalize all the features (column-wise).
3. Implement *dbscan* clustering for sample values of epsilons and min points.
4. Display the clusters as a grayscale image and save them.

Analysis.

To evaluate the impact of epsilon and min points on the DBSCAN clustering algorithm, I calculate region-based segmentation for the given image using various combinations of epsilons and minimum points.

An overview of epsilon and min_samples,
Epsilon is the minimum threshold distance below which two points are considered neighbors.

Minimum_samples are the minimum number of neighbors a given point should be classified as a core point.

The number of minimum points or minimum samples depends on the size of the dataset. The larger the number of samples in the dataset, the higher the value of minimum points. This is in line with the reasoning that the number of neighbors should be higher for a large dataset to be classified as a core point.

The value of minimum points thus depends on the domain of the dataset, but a rough estimate should be twice the number of features in the dataset. The number of features in the dataset is 5. Hence minimum samples should be above 10. But with our noisy dataset, larger values of minimum samples are better as it would yield significant clusters.

The same can be seen from the output images, where all the points in one cluster are given one grayscale value, while all noise points are given value 0 (needs to be interpreted as noise and not a single cluster)

Minimum points 2

to

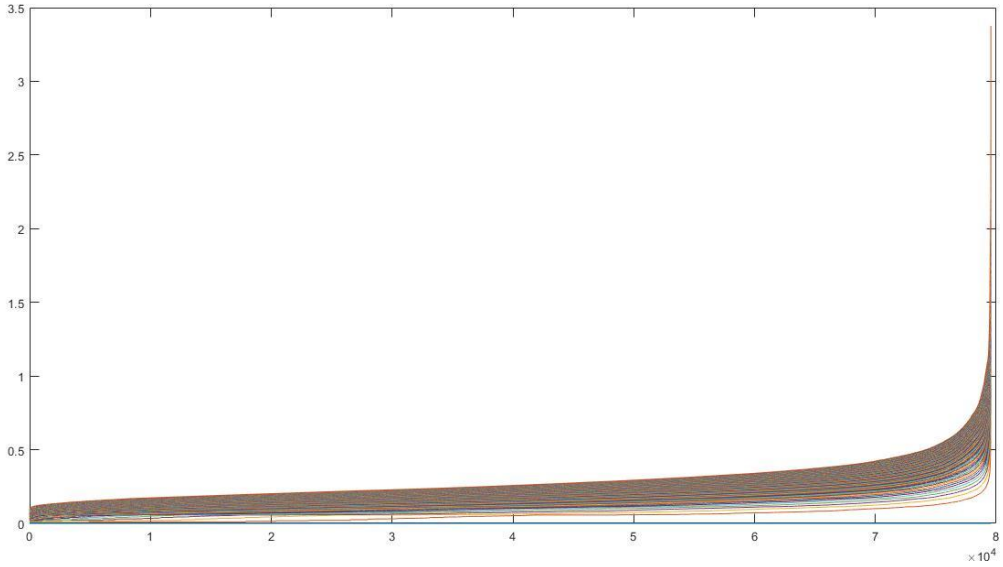
Minimum points 50

Here, it can be seen that a small value of minimum samples results in a number of samples, but since it does not reflect the region of interest. There are also many clusters, but no pattern can be noticed. This is due to the very small value of minimum samples, which does not aid in a good selection of core points.

A contrast to this would be the image to its rights, where 50 neighbors are required. In this case, most of our samples are classified as noise as the image is not dense enough to get such a high number of neighbors to classify a core point. Thus, a noisy output is expected.

Similarly, for epsilon, I conduct two evaluations, first is using the elbow method from a k nearest neighbors search.

The plot for the range of minimum points is given.



We can see that the epsilon value increases as we increase the number of minimum points for the dataset. This is in line with the algorithm, where more points can be found when considering a larger distance.

For 10 or 20 minimum points, we notice that 0.2 to 0.25 can be considered good epsilon values. We can see the same from our output images.

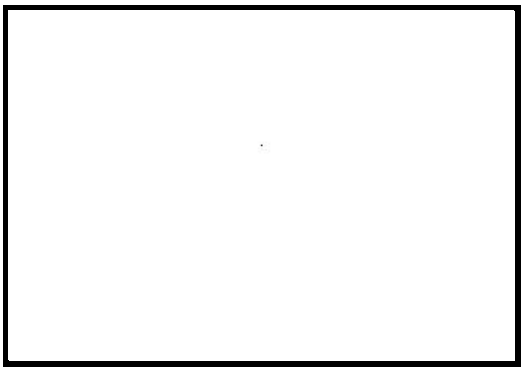
Epsilon: 0.01 and Minimum points 10

to

Epsilon: 0.2 and Minimum points 10

As we increase the epsilon value from 0.01 to 0.2, we can notice that our algorithm can better classify the samples for the same number of minimum points. The pattern in images is in the range of epsilon (euclidean distance is used).

As we continue to increase the value of epsilon to 2



Minimum points 10 and epsilon 2 (black borders are added to make the image visible)

Here, we see that all the samples are classified in one cluster due to the large epsilon value, thus defeating the purpose of clustering. We can also see that for no values of minimum points, the value of epsilon is this large even in the graph.

Another noticeable combination would be for a very low epsilon value, in this case, 0.01 since this value of epsilon is too small for the samples.

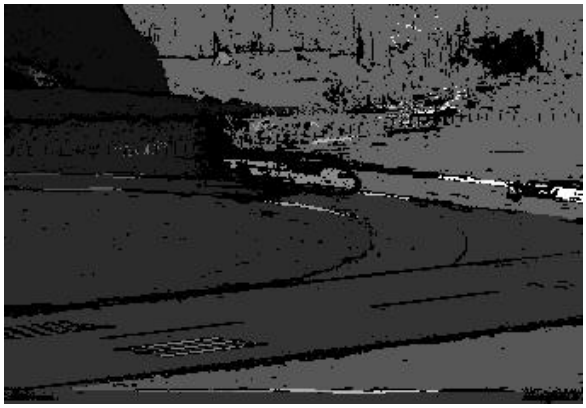
Epsilon: 0.01 and Minimum points 5

to

Epsilon: 0.01 and Minimum points 100

All the points are classified as noisy, no core points can be found, and no samples are thus clustered.

Going by this analysis, the classification that performs the most suitable clustering would be minimum points = 25 and epsilon = 0.2. We can see in this classification that mostly edge points are classified as noisy, and resemblance can be seen from the original image

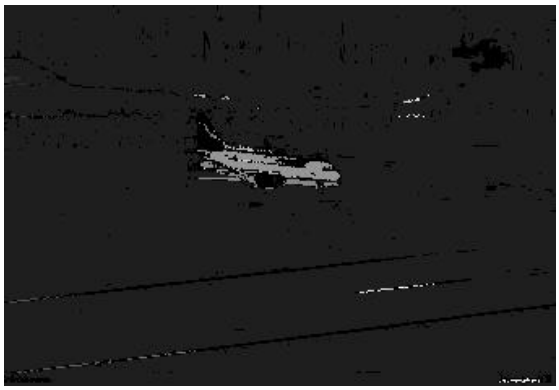


Epsilon: 0.2 and Minimum points 25

Original Image

We should also note that the given image is 'denser' (closer but has fewer neighboring points) (for the selected features) for the region of the airplane but is 'sparse' (further but more neighboring points) for the background.

We can see that in the following images,



Epsilon: 0.25 and Minimum points 15

to

Epsilon: 0.25 and Minimum points 100

The first image shows us the airplane clearly, while the second image shows the background clearly. A balance is achieved as we change the epsilon value to 0.2 and minimum points to 25.

References.

1. Matlab implementation of DBSCAN clustering.
2. https://towardsdatascience.com/machine-learning-clustering-dbscan-determine-the-optimal-value-for-epsilon-eps-python-example-3100091cfbc
3. https://medium.com/@tarammullin/dbscan-parameter-estimation-ff8330e3a3bd
4. https://link.springer.com/content/pdf/10.1023/A:1009745219419.pdf
5. https://www.ccs.neu.edu/home/vip/teach/DMcourse/2_cluster_EM_mixt/notes_slides/revisitofrevisitDBSCAN.pdf

6. https://www.datanovia.com/en/lessons/dbscan-density-based-clustering-essentials/#algorithm
7. http://www.sefidian.com/2020/12/18/how-to-determine-epsilon-and-minpts-parameters-of-dbscan-clustering/

Question 3

Filename: m_2019472_q3.m
Write to execute: m_2019472_q3(*Image Name*)
Input: Name of the image
Sample Input: m_2019472_q3('0002.jpg')
Output: 2 Images with the top 10 strongest features and the time elapsed to calculate them.

Assumptions.

1. The number of times the loop is repeated while timing is 100.
2. The names of the output images are of the form *(Name of algorithm).jpg* and are stored in the *2019472_Q3* folder in the output folder.
3. The images are already saved in the folder and are not called every time the algorithm is run.
4. The path needs to be set as required.

```
% Making the features dataset using the image dataset.
% Loading the images. Specify the variable values.
path = 'C:\Users\hp\Desktop\Manvi\Semesters\6th_WinterSemester\ComputerVision\ComputerVision22\2019472_Assignment2\';
```

Working.

1. Read the image and turn to grayscale.
2. Calculate the features (*SURF/ SIFT)* multiple times (*nLoops = 100)*.
3. Use *tic-toc* to measure the time.

Analysis.
Sample Output

```
Time taken to calculate SIFT features
Elapsed time is 3.278597 seconds.

Time taken to calculate SURF features
Elapsed time is 1.112032 seconds.
```

The major difference between SURF and SIFT is how the algorithms compute the image pyramid. While SIFT convolves the image multiple times with large Gaussian kernels to compute an image pyramid, SURF calculates an approximation of the image pyramid using integral images.

Comparing all the steps of the two algorithms, SIFT uses the difference of Gaussians convolved with different sizes of the image with the same size of the filter. At the same time, SURF approximates it by using different sizes of box filters convolved with the integral image to

generate the scale space. For keypoint detection, SURF uses the Hessian matrix to calculate the maxima instead of local extrema detection. For orientation, SIFT uses orientation of histogram while SURF uses Haar wavelet responses at every sample point around the interest points.

In conclusion, the increase in speed can be attributed to box filters. Instead of Gaussian averaging the images, SURF makes use of squares as an approximation since convolution with square is faster if the integral images are used. It also uses a Hessian matrix to obtain the key points (maxima is calculated), and Haar wavelets responses are computed to calculate the orientation.
Thus the calculation of Gaussian images and the comparison process to calculate maxima are thus optimized in SURF and thus is more computationally efficient than SIFT.

References.

1. Matlab implementation of Tic-Toc.
2. Matlab implementation of SURF Features.
3. Matlab implementation of SIFT Features.
4. SURF: Speeded Up Robust Features (Original Paper)