Image Segmentation

Sparsh Jain
111601026

Problem Statement:
Image Segmentation based on clustering pixels.

Aproach:
I decided to evaluate a simple, and fast K-Means Clustering algorithm, against DB Scan Clustering and Agglomerative Clustering, which take a lot more time for computation. I used OpenCV library to read the images, and sklearn library to apply to clustering algorithms. Before applying any of the algorithms on the images, some pre-processing is required. I applied a median filter (avalaible as a function in preprocessing sub-library of sklearn) which is very effective in removing the salt noise. Usually, a gaussian filter is a good choice to reduce gaussian noise and to an extent, give a blur but good enough image; however, image segmentation is better performed on images with sharp edges, and hence I chose to omit that filter.

K-Means:
K-Means algorithm is applied on all the images with K = 2, 3, 4 and 5. Most of the images, qualitatively look good with K = 3.

DB Scan:
DB Scan algorithm is applied on all the images with epsilon e = 1.0, 1.5, 2.0, 2.5. Most of the images look qualitatively better with e = 1.5, however in some cases, 2.0 or even 2.5 epsilon gives a better qualitative look. This also does not perform very well in images with frequent changes of color in the image. It is also relatively slower than K-Means algorithm, and for processing and memory constraints, I had to reduce the size of the image to 50% before applying the algorithm. However, I tried running the algorithm on a full scale image, and the information loss is not affecting the ‘successful’ cases as segmentation should not take ‘minor’ details into account.

Agglomerative:
Agglomerative algorithm is applied on all the images with choice of n = 2, 3, 4, 5; same as that for K-Means algorithm. Again, results are good with K = 3 for most of the images. However, it is considerably slower algorithm, even after reducing the image to 40% of it’s original size. Running the algorithm on full scale image gives the memory error as the memory requirement exceeds the physical memory alloted to the process running the algorithm.

Conclusion:
After thoroughly examining and comparing images from all the algorithms, DB Scan seems to perform relatively poor as compared to both the other algorithms as the resulting images are still very noisy and there’s no definite number of segments. With low epsilon, it produces very large number of segments, while reducing the epsilon blends the whole image in large inappropriate segments. Agglomerative algorithm is performing nice but we had to reduce the image to 40% of the original size and still it took over 2 hours to run the script over all the 100 images. K-Means on the other hand, being considerably faster, completed within a few minutes for full scale images. In cases of separating the image into layers like background, middle layer, and foreground, K-Means looks good as Agglomerative Algorithm is not giving extremely good results in comparision, specially if you take the extra processing, memory requirements, and working on a reduced image size.

Challenges Faced:

A major chalange faced was selection of filters. There are a lot of filters available in the libraries, and applying multiple of them will result in information loss while without any filter, we risk losing to the noise. Since major noise in the images come as salt noise, applying median filter which reduces them effectively seemed a reasonable choice. Additionally, a gaussian filter is a good filter to apply as a preprocessing of the image, however, it blurs the image, causing less sharp boundaries. Image segmentation problem is based on boundaries and hence I chose to knowingly not apply this filter. Other filters, like averaging were not significant in this case.

Another challenge faced was selection of appropriate size reduction for DB Scan and Agglomerative Algorithm to keep computation time reasonable while not reducing the image size too much.

Finally, qualitative comparision between results of different algorithms was a tough choice. There is no hard and fast rule here, but I'm preferring K-Means over Agglomerative Algorithm for it's considerably fast speed and only slightly lower perfomance in comparision. However, if the goal is preciseness, Agglomerative Algorithm should be chosen, given you have enough computing power and satisfy memory requirements.

Drive Link:

I have uploaded the jupyter-notebook for running the algorithms one image at a time for comparision purposes. I have also uploaded the python scripts for running the algorithms for all the images and saving the results. Here is the drive link for the results obtained on running the script.