EE5175 : Image Signal Processsing

Vignesh S
EE16B127

Project

Image Segmentation

19/04/2020

Contents

1 Abstract

This report is a summary of the results obtained upon reproducing the paper *"Normalized Cuts and Image Segmentation"* by Jianbo Shi and Jitendra Malik. The approach used is to try to use low-level similarities (like brightness, coherence) to understand the overall scene of the image. A graph-theoretic formulation is employed to execute this. In the next few sections we explain in brief the theory behind this approach and also show some examples where this code works and where it fails. The implementation is done in python.

2 Approach used

2.1 Theory

The main 2 questions that are posed by the problem are first, what criterion to optimize? and second, is there an efficient algorithm for the optimization? These questions are answered by the graph theoretic grouping which is explained. Each pixel in the image is taken as a vertex and based on their intensities and proximity a similarity score is assigned to each of the weights (w_{ij} defined below). We partition the graph into 2 sets by removing all the common edges between the 2 sets. Define cut(A,B) as in 1. Another parameter called assoc(A,B) as defined in 1 gives us an estimate of the similarity within the partitions A and B upon separation. Our goal is maximize assoc(A,B) but simultaneously decrease cut(A,B).

But these 2 quantities depend on the number of edges considered during their computation. For example while minimizing cut(A,B) since isolated points have less weights going out of them (compared to bigger partitions) they will returned as segments. To counter this problem we define 2 new parameters called Ncut and Nassoc as shown in 1. Here we normalize the original values with assoc(.,V) which indicates all the edges from a particular partition to all the other vertices. Hence isolated point have Ncut value as 1. From 1 we see that Ncut and Nassoc sum upto a constant. Hence maximizing Nassoc is same as simultaneously minimizing Ncut.

We can write the Ncut (or 4 times Ncut) in a form as shown in 2. This form

indicates difference between sum total of the weights from A to the entire graph and the sum of all weights from A to itself. This is nothing but sum of all the weights from A to B (since only 2 partitions), which is nothing but Ncut (factor of 4 is due $(1+x)/2$ term). Upon some skilled assumptions and manipulations 3 can be obtained from 2. But 3 is nothing but the Rayleigh coefficient of the matrix A where : $A^TA = D^{-1/2}(D-W)D^{-1/2}$. From matrix inequality it can be shown that minimum value of Ncut is the smallest eigenvalue of $D^{-1/2}(D-W)D^{-1/2}$. This can be obtained solving the generalized linear system in 4. But due to the additional constraint from 3 we need to consider the 2nd minimum eigenvalue not the minimum eigenvalue. The eigenvalue gives Ncut and eigen vector partitions the graph.

$$w_{ij} = \begin{cases} \exp\left(-\frac{||F_i - F_j||^2}{\sigma_I^2}\right)\exp\left(-\frac{||X_i - X_j||^2}{\sigma_X^2}\right) & ||X_i - X_j|| \leq r \\ 0 & else \end{cases}$$

$$\begin{aligned} cut(A,B) &= \sum_{u\in A, v\in B} w(u,v) \\ assoc(A,V) &= \sum_{u\in A, t\in V} w(u,t) \\ Ncut(A,B) &= \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \\ Nassoc(A,B) &= \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)} \\ Ncut(A,B) &+ Nassoc(A,B) = 2 \\ k &= \frac{\sum_{x_i>0} d_i}{\sum_i d_i} \end{aligned} \tag{1}$$

$$4[Ncut] = \frac{(1+x)^T(D-W)(1+x)}{k1^TD1} + \frac{(1-x)^T(D-W)(1-x)}{(1-k)1^TD1} \tag{2}$$

$$\begin{aligned} \min_x Ncut(x) &= \min_y \frac{y^T(D-W)y}{y^TDy} \\ y^TD1 &= 0 \\ y(i) &\in \{1, -b\} \end{aligned} \tag{3}$$

$$\begin{aligned} \min Ncut &= \lambda_{min}(D^{-1/2}(D-W)D^{-1/2}) \\ (D-W)y &= \lambda Dy \end{aligned} \tag{4}$$

2.2 Algorithm

The algorithm is explained below

- Create a graph G(V,E) using all the pixels of the image and obtain the weight from w_{ij} as mentioned above.
- Obtain the matrices W and D using the graph and then solve the system 4.Since we need only 2 smallest eigenvalues/vectors we can use Lanczos eigensolver.
- Upon obtaining the y (2nd smallest eigenvector) threshold it using median or 0. But we do is try to find that threshold that minimizes Ncut and the threshold y using that.
- This (thresholding) divides y into 2 parts and divide the entire image correspondingly according to this.
- Take these 2 partitions and then run the same algorithm on them recursively by having an area and Ncut threshold to ensure reasonable segments.

3 Sample outputs

3.1 Gray-scale images

We consider 3 kinds of images based on the scene and analyze how the algorithm works against them.

- When object is in focus in front of the background
- When object is merged or similar to the background
- When there is no object (only background)

Consider the examples shown for the first case 1 (the stag and crow). Here we see that it identifies the objects and separates it from the surroundings. But the crow is better identified when compared to the stag. This is because there is a higher contrast between the crow and the background (the sky, you can see the branch comes with the crow due to lack of contrast between them) when compared to the contrast between stag and its surroundings (mainly lacking between the body and the upper background).
Now coming to the merged object image 2 where the bug is present on the leaf. Here we see that the segmentation is more based due to the blurring (reducing the intensity values slightly) than the object detection. We see different parts of the plant are separated, whereas the bug is not isolated from the leaf. This can be corrected by reducing the radius used for constructing the weight matrix and increasing sigX.
In the case with no object we see that it distinguishes the snowy mountains from the rocky terrain and the sandy shore. The river is not identified separately due the area constraint applied.

Figure 1: Object in focus

Figure 2: Merged object

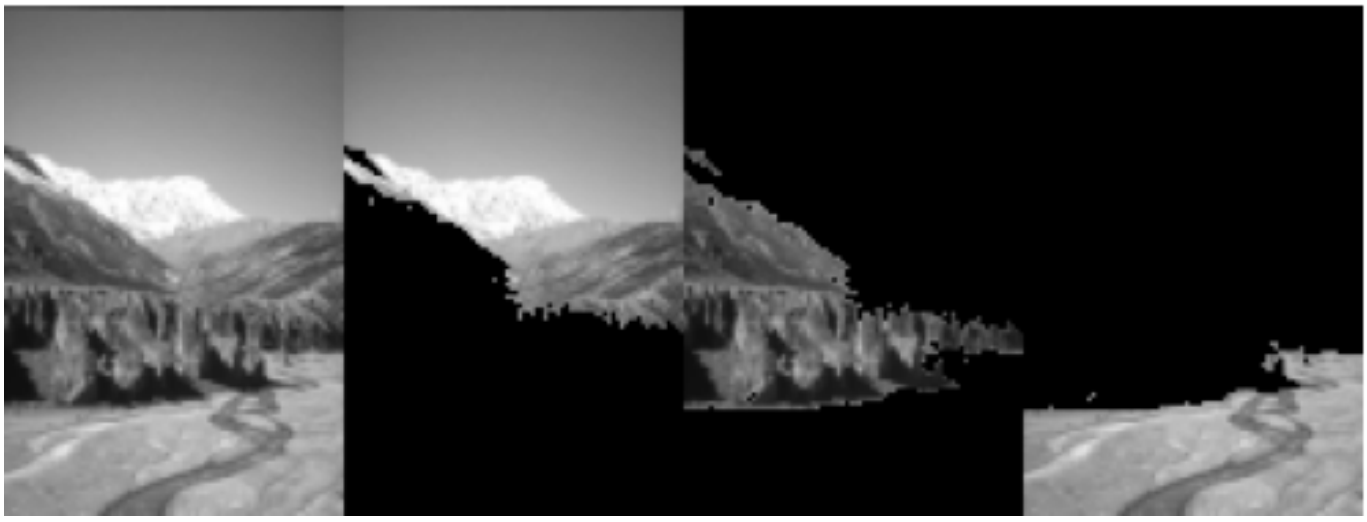

Figure 3: No object

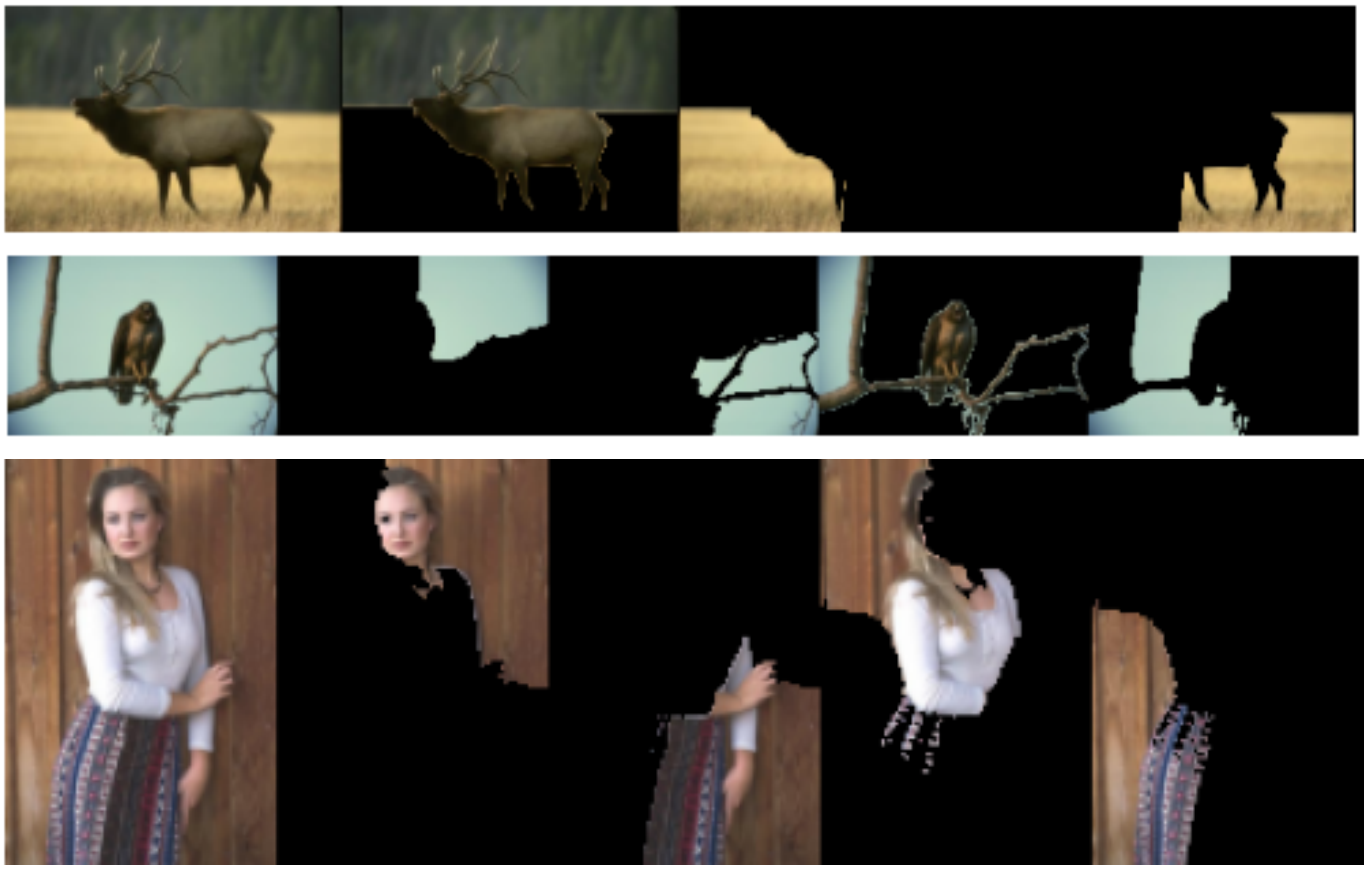

Figure 4: Object in focus

3.2 Colored Images

Here also we consider the 3 types of images. Also we take the RGB counterparts of the above gray scale images and analyse them. The main difference between this and the gray scale is that we have 3 values for each pixel and we define weight matrix by taking the norm -squared of these values (in gray-scale it is just square of the pixel intensities). Hence we make the sigI(RGB) = sigI(gray-scale)$\sqrt{3}$. The partitions obtained for these (RGB) images is very similar to the gray-scale ones. This would be expected since loosely speaking the norm-squared of the RGB image intensity (is addition of the 3 intensities) is 3 times the intensity-squared of the gray-scale image. Here in 4 the lady's image we would've expected the background wall to be separated from the lady. Instead the lady is separated into different parts along with the background. This is because the intensity and contrast along her body is varying much differently compared to the background.

Figure 5: Merged object

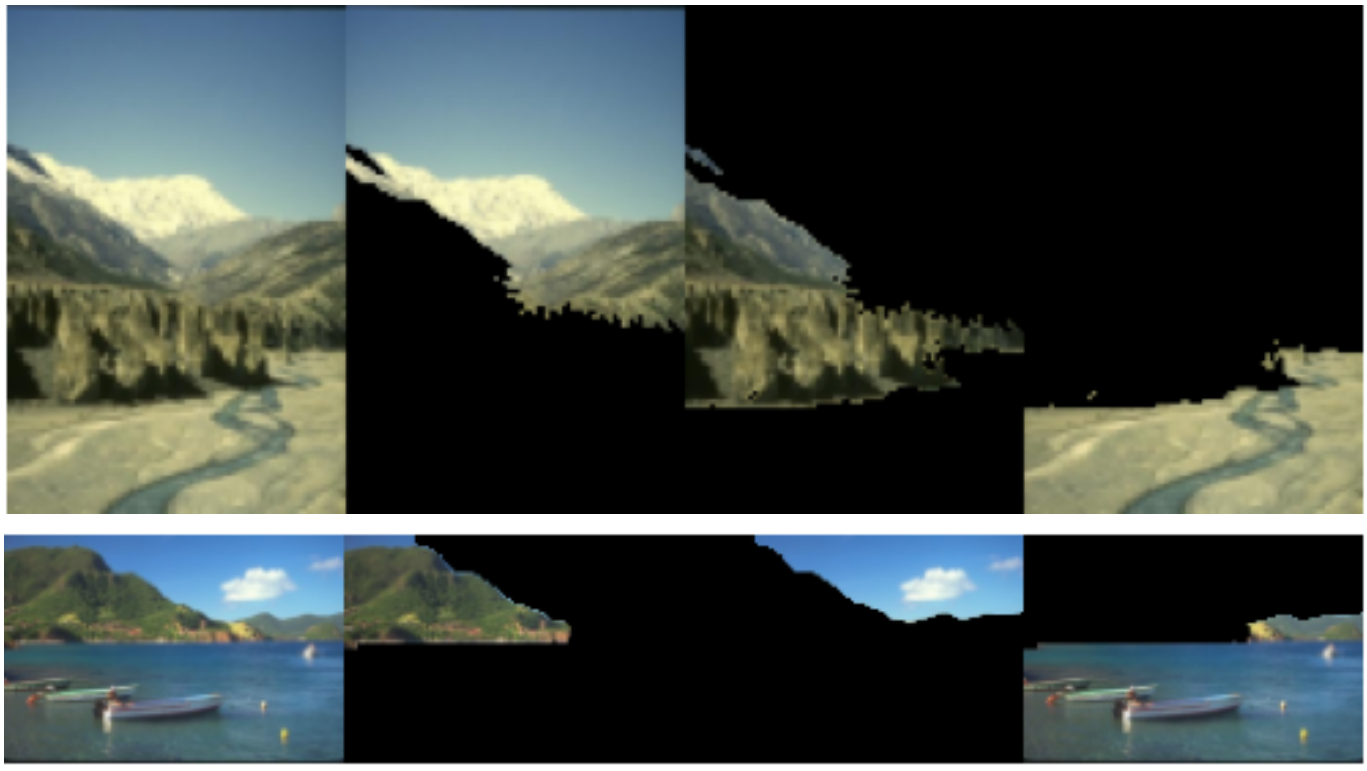

Figure 6: No object

3.3 Effect of hyperparameters

The hyperparameters play a very important role in obtaining the required segmentation. The hyperparameters that we are allowed to tweak for this algorithm are,

- sigI : pixel intensity variation (taken to be $0.1/\sqrt{3}$ for gray-scale and 0.1 for RGB)
- sigX : pixel spatial variation (taken to be 4 for gray-scale and RGB)
- r : radius around central pixel considered for weight matrix(taken to be 7 for gray-scale and RGB)
- Arthres : Minimum area threshold for segments (taken to be 1500 for gray-scale and RGB)
- NCthres : Maximum Ncut threshold (taken to be 0.03 for gray-scale and for RGB). Only used to limit absurd segments from being created.

We vary different hyperparameters and see their effect.

- Consider the image 7 shown, here the sigI value is varied from $0.1/\sqrt{3}$ to 1. The segments are changed from 3 to 4 and they are segmented differently. We can understand the change by seeing that upon increasing sigI you are reducing the weightage to pixel intensity and relatively increasing the weightage to spatial component. Hence the segments tend to form paritions only with closer pixels. Therefore it is divided into 4 quadrants (closest possible partitions considering the area threshold).

- Consider the image 8 shown where sigX changes from 4 to 2. This is increasing the emphasis on the spatial weightage keeping the same intensity weightage. As a result we see the ones with similar intensity and those which are closer are segregated. This is clearly visible since the stag and upper background are now separated into multiple partitions.

Figure 7: SigI = 0.058, SigI = 1

- Consider the image 9 where the Arthres is varied from 1500 to 3500. As a result the smaller segments are constrained to become a single partition as seen.

- The effect of r for r=5 and r=7 is same for the above considered images. But to see a considerable difference due to r we need higher computational power for calculating the 2nd eigen vector.

- The effect of resizing the image as seen in 10 is better with lesser downsizing. All the images considered except the second image (downsize by 2) of 10 is downsized by 4 times the original image. We see 1st partition of 2nd image of 10 is better than the partitions of the 1st image (where the sky is divide into 2 at the centre). But due to lack of computational power (caused while calculating 2nd eigen vector) we had to downsize all images 4 before processing them.

  Hence we see the impact of hyperparameters on the segments obtained.

Figure 8: SigX = 4, SigX = 2

Figure 9: Arthes = 1500, Arthres = 3500

Figure 10: downsize : 4 times, downsize - 2 times

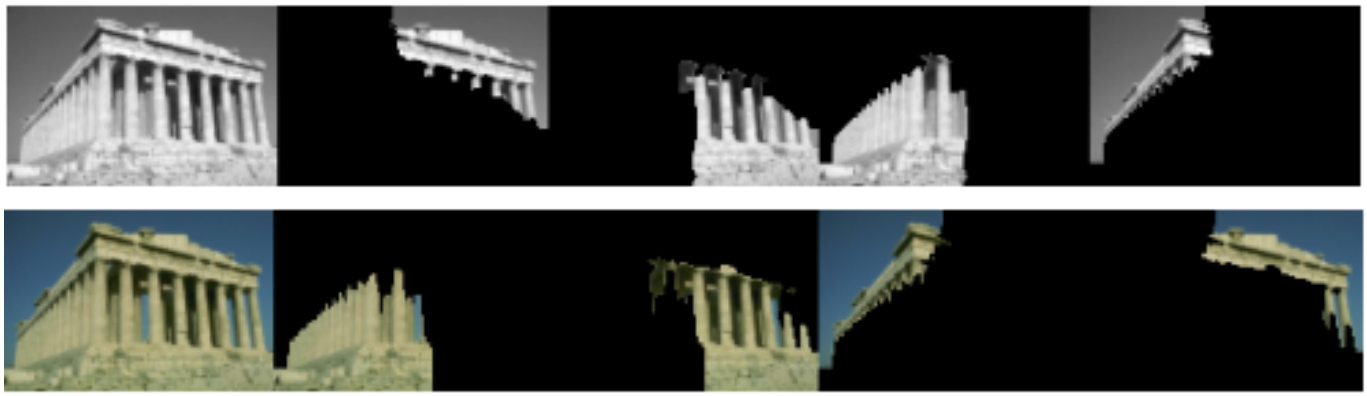

Figure 11: Close variation in contrast

4 Failed output samples

4.1 Analysis

There are several cases where the image nature and the hyperparameters are such that we do not obtain the eigenvector required to partition the image (due to ill-conditioned matrices etc.). Apart from these cases, there are some cases where the algorithm does not partition the image into required (clearly visible) partitions. One such example is shown in image 11 where the segments should be the should be the sky and the Parthenon (building). But instead there is a mix up of some part of the segments. This can be attributed to the fact that due to quick variation of contrasts near the pillars, the algorithm does get have a fair idea of the image through the weight matrix (this is similar to the zebra image in the original paper). For such images, we can add an additional term in the weight matrix to accommodate for the rate of variation of intensities. This might help in the segmentation.

Another example that can be improved is 12 where we see a significant color difference between the bug and the leaf, we see that it not segmented properly. This can be corrected by have difference in the R,G,B intensities ($\exp[-(F_{Ri} - F_{Rj})]$) separately (or along with the norm) instead of just taking the norm of all 3 color intensities.

4.2 Image samples

Figure 12: Segment based on color

5 Takeaways

Based on the above examples we see that the Normalized cut algorithm works pretty well for segmenting images. However there are some cases where it fails to give the required output. Some modifications can be made to this to improve, but we cannot guarantee a significant improvement. With Deep Learning models (U-nets) there can be better partitions which can be obtained. This can be largely attributed to the fact that there is no hyperparameter choice to be made in DL models (all is accounted for in the layers of neural networks) unlike here where everything is heavily dependent on them. If we make this algorithm choose hyperparameters so as to obtain the best possible segments, it will improve significantly.

-End of report-