Implementing and experimenting with the counting min cuts algorithm for planar graphs devised by Dr. Ivona Bězáková and Adam J. Friedlander: Questions of prime interest were: how different are the min cuts from each other? How many min cuts are there in an image? Can we use some pre-processing stage to improve the segmentation?

Hypotheses

- All min-cuts in an image, represented as a grid graph, are similar to each other.
- There is a single min cut or a small number of min cuts.
- Pre-processing stage can be used without affecting the quality of the segmentation.

Background

Planar graphs are simple graphs which can be drawn on a plane without any of its edges crossing.

Dual of a planar graph is a graph created using vertices as faces of the planar graph, and then connecting vertices represented by adjacent faces.

We use the Max-flow min-cut theorem to find out a segmentation of an image. Images can be represented as grid graphs with each pixel as a node and adding bidirectional edges between neighboring pixels. Sink is selected by the user inside an object, and source can be selected anywhere outside. Edge weights are decided using function 1. This function is a slightly modified version of the function used in ref [3].

EdgeWeight = \frac{1}{\exp(-\sqrt{\text{difference}})} \tag{1}

where difference = difference between luminance values of pixels. In our case \( \epsilon = 1.43 \times 10^{-4} \) (for difference = 20).

Step 4: The output will be number of paths between d-c, which is 4. These paths can be counted with a simple data structure programming algorithm. A random cut can be sampled out of these 4 cuts by tracing any of these paths in such a way that a vertex appearing in more number of min cuts is picked with a higher probability.

Multiple possible segmentations

For faster running times, we ignore larger differences and treat them all as equal to a chosen \( \epsilon \) value. Equation 1 accounts for luminance difference of 20. If we want to account for larger differences smaller \( \epsilon \) should be chosen. But it increases running time of the code. Fig. 13 supports this. Experimentation with pre-processing step leads to an approach, measuring median, Gaussian filter etc and histogram equalisation yielded a trade off between finer segmentation vs faster running time.

Counting cuts algorithm

Counting all the min-cuts in a graph is a \#P-Complete problem. But with planar graphs it can be done in polynomial time. This algorithm [1] has a time complexity of \( O(n^4 + n \log n) \), where \( n \) is number of vertices in the input planar graph, \( d \) is the length of a sink to source path in a graph obtained after contracting strongly connected components in the residual graph. In following visual representation of the algorithm d=1, from fig. 5. \( O(n \log n) \) part of the time complexity is for finding the max flow. For planar graphs as shown by ref[2], max flow can be found out in \( O(n \log n) \). Following is the visual representation of the algorithm.

Results

We use some pre-processing stage to improve the segmentation. Depending on the image, pre-processing step may improve running time, as it removes noise which gives lesser min cut count.

Conclusions

- Counting cuts algorithm can be used to find out multiple segmentations of an object in an image.
- All the min cuts are visually similar to each other.
- There can be an exponential number of min cuts in an image, though visually similar.
- Depending on the image, pre-processing stage may improve running time, as it removes noise which gives lesser min cut count.
- Min cut count grows exponentially with image size. Hence compression can be used to improve running time and still get a decent segmentation.
- Epsilon value chosen makes a huge impact on the final outcome. For a faster running time larger values should be chosen. But this means we might lose on finer segmentation, as larger epsilon values mean accounting for only a small difference between luminance values of neighboring pixels. It’s essentially a trade off between finer segmentation vs faster running time.

References

- Efficient Planar Graph Cut with Applications in Computer Vision. F. R. Schmidt, E. Töppe, D. Cremers, IEEE CVPR, Miami, FL, June 2009
- http://radiopaedia.org/articles/pagc35670c
- The Berkeley Segmentation Dataset and Benchmark (accessed 1/5/2016)
- https://www.eecs.berkeley.edu/Research/Projects/CS/vision/index.html

Image segmentation using minimum (s,t)-cuts

Student: Gaurav Joshi (gpj5552@rit.edu), Advisor: Dr. Ivona Bězáková (ib@cs.rit.edu)

Department of Computer Science, Rochester Institute of Technology