# CSE527 Final Report

# Fast Energy Minimization for Stereo Via Graph Cut

## Yubin Lai

### 1 ABSTRACT

In two-view stereo, we need to find the disparity map for the input images. This problem can be formulated in terms of energy minimization. In my project, I’m using the energy

[1], where is the disparity map. Here, measures the extent to which the disparity map is not piecewise smooth,

, where is the set of neighboring pixels(von Neumann neighborhood) and , where K is a constant. measures the dissimilarities between the corresponding pixels given the disparity map [1]. The calculation of is given in [2]. This report is not going into its detail.

There are generally two types of methods to minimize this energy. First one is to find the global minimum, but it’s very slow. Second one is to find the local minimum, but the local minimum might be very far away from the global minimum. In my project, I’m using graph-cut-based -expansion algorithm to find the local minimum. Not only this method is fast, but also the local minimum it find is proved to be within a known factor of the global optimum.

Input :

Two images taken from different viewpoints for stereo

Output:

A new image containing the disparity information for the first image

### 2 -EXPANSION ALGORITHM

A disparity map contains many possible disparity values for its pixels. An α-expansion move allows any set of pixels to change its disparity to α. An example of α-expansion move is shown in Fig 1 [1].



Fig. 1. Example of an α-expansion move

The α-expansion algorithm for two-view stereo goes like this [1]:

1. *Start with an arbitrary disparity map*
2. *Set*
3. *For each possible disparity value*

*3.1. Find ’’ = arg min among within one -expansion of*

*3.2. If , set and set*

1. *If goto 2*
2. *Return f*

In step 3.1, we need to find the best -expansion move given which is proved to be equivalent to solving the minimum cut problem on an appropriately defined two-terminal graph . The structure of the graph is illustrated in Fig. 2 [1].

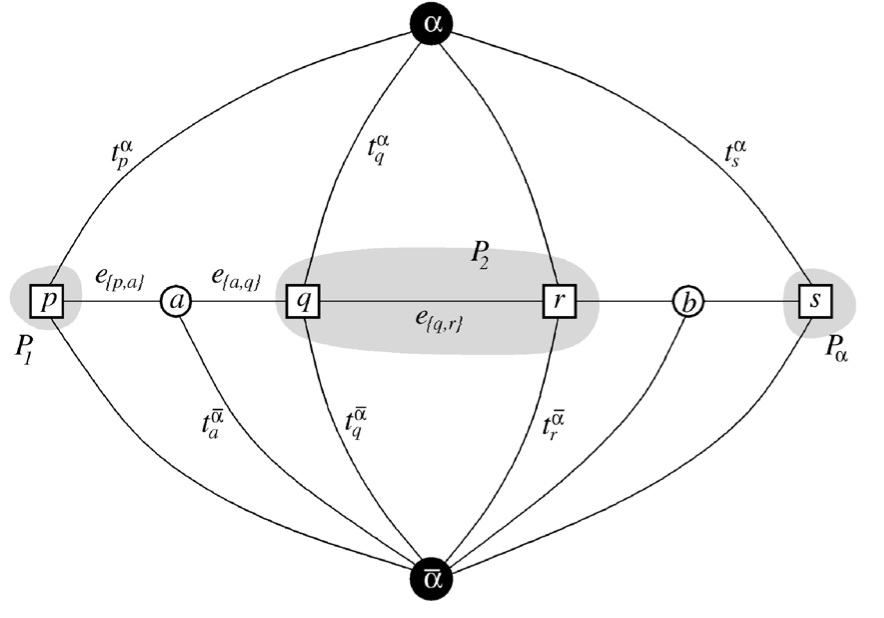
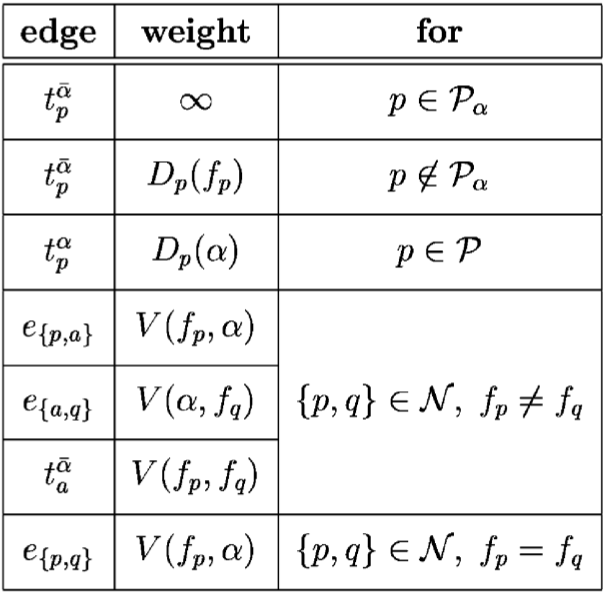


Fig. 2. An example of for a 1D image. The set of pixels in the image is and the current partition is where . and are auxiliary nodes.

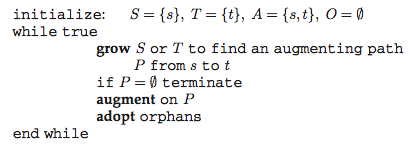
The weights assigned to edges are [1]



The proof of the equivalence between solving the minimum cut of this graph and the step 3.1 of the algorithm can be find in [1].

### 3. MINIMUM CUT

In the α-expansion algorithm described above, we need to compute the minimum cut of the equivalent graph . Standard min-cut/max-flow algorithms to solve this problem are Dinic’s algorithm [3] and push-relabel algorithm [4]. In this project, I’m using an empirically faster algorithm [5] to compute the minimum cut. The general structure of this algorithm is [5]:



The detail of the three stages: growth, augmentation and adoption, can be found in [5]. This algorithm has worst-case complexity , where is the minimum cut. For a 2D image of pixels, the worst-case running time is way too slow. However, this algorithm has very good empirical performance. In my implementation, the running time is almost linear.

### 4. RESULTS

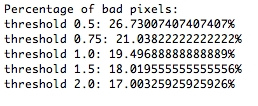
Two datasets, *Teddy* and *Cones* from Middlebury Stereo Datasets, are used to evaluate the -Expansion algorithm (including the minimum cut algorithm [5]).

 (a) (b)

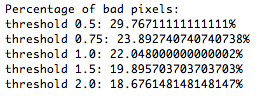
 (c) (d)

Fig. 3. Results of Cones (a, b) and Teddy (c, d) from Middlebury Stereo Datasets

The disparity maps generated are illustrated in Fig. 3. In addition to the resulting disparity map, the percentage of bad pixels compare to the group truth is also computed. The result for Cones is:

****

. And the result for Teddy is:



The threshold is the max difference of disparity between the result and the ground truth for a good pixel. The errors are evaluated in all pixels including occluded regions and unknown regions (such as the leftmost parts). The initial disparity map in the -expansion algorithm is all 1.

### 5. CONCLUSIONS

In this project, I implemented the -expansion algorithm in [1], the mini-cut/max-flow algorithm described in [5] and the pixel dissimilarity measure described in [2]. The dissimilarity measure takes least work. The -expansion algorithm requires understanding of the paper and some coding. The mini-cut/max-flow, called by -expansion algorithm, surprisingly takes the most work and the main reason is this algorithm has worst-case complexity , a lot of work is spent to improve the speed of this algorithm. All codes are written in Java, no lib except Java standard libs is used.

### REFERENCES

[1] Y.Boykov, O. Veksler, and R. Zabih, “Fast Approximate Energy Minimization via Graph Cuts”, *IEEE Trans, Pattern Analysis and Machine Intelligence,* vol. 23, no. 11, pp. 1222-1239, Nov. 2001.

[2] S. Birchfield and C. Tomasi, “A Pixel Dissimilarity Measure that Is Insensitive to Image Sampling,” *IEEE Trans, Pattern Analysis and Machine Intelligence*, vol. 20, no. 4, pp. 401-406, Apr. 1998.

[3] E.A. Dinic, “Algorithm for Solution of a Problem of Maximum Flow in Newworks with Power Estimation,” *Soviet Math. Dokl.,* vol. 11, pp. 1277-1280, 1970.

[4] A.V. Goldberg and R.E. Tarjan, “A New Approach to the Maximum-Flow Problem,” *J. ACM*, vol. 35, no. 4, pp. 921-940, Oct. 1998.

[5] Y. Boykov and V. Kolmogorov, “An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision,” *Proc. Int’l Workshop Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 359-374, Sept. 2001.