Energy Minimization for Stereo Correspondence

Travis Latzke

December 2, 2020

Abstract

The use of graph cuts to reduce energy functions has been proven to produce remarkable results in a number of early vision tasks. One of these tasks is known as the stereo correspondence problem, which involves the derivation of a disparity map, given multiple images. This paper explores a prominent graph cut technique developed by Kolmorogov and Zanih. Their alpha-expansion algorithm is compared to area-based and scanline matching algorithms on images obtained from the Middlebury stereo dataset collection. The results are presented quantitatively, as well as visually.

1 Introduction

Stereo vision is a large and active area of research in computer vision that involves capturing images of the same scene at multiple vantage points to extract 3D information from the scene being captured. The 3D information is extracted by taking advantage of how each camera is set up relative to one another. If each camera is able to capture the same point (the point is not occluded from any of the cameras), then the point may reside at different pixel locations, p_1, p_2, ..., p_n for n images taken by n cameras. If we restrict the cameras to be collinear, we can take advantage of epipolar geometry properties that place restrictions on the space for pixels to correspond to one another. For example, if we place the cameras, so that they are collinear along the y-axis, then a point $p_i = \langle p_i^x, p_i^y \rangle$ from image i will correspond to a point p_j in image j through the following relationship:

$$p_i + d = \langle p_i^x, p_i^y \rangle + \langle d^x, d^y \rangle = \langle p_j^x, p_j^y \rangle = p_j$$

where $d^y = 0$, so the disparity, d, can be thought of as one dimensional, because it's y component will always be zero. Therefore, the search space for a d where two points correspond is reduced from two dimensions to one dimension. If the camera's are not collinear, the images may undergo an image rectification process that projects the images onto the same plane. The problem of finding a matching correspondence for all pixels is known as dense stereo correspondence and will be the main topic throughout the rest of this paper. There will also be an assumption that n = 2; each scene will consist of a pair of images, L and R. We will also assume that L and R have been rectified, thereby sharing a common image plane.

1.1 Related Work

There are numerous techniques to estimate a mapping f, where f_p is the disparity of pixel p, which is inversely related to the depth of the object captured at pixel p. One naive approach to solving for f, would be to find a pixel $q \in R$ that is part of the same epipolar line as some $p \in L$ and find search for a disparity, d, such that $d = \min_{0 \leq l \leq k} (L(p + l) - R(q))^2$ for some max disparity constraint set at k. This

naive approach would work well if the lighting was constant between the two images and each pixel had a unique intensity. However, this is hardly the case because real world images often lack regions that provide enough texture and consistent lighting for this method to work well. One way to address the problem of texture-less regions is to use area-based matching.

$$s_{tx}(r,c) = \sum_{i}^{m}\sum_{j}^{n}\Big[t(i,j) - x(r+i-\frac{m}{2}, c+j+\frac{n}{2})\Big]^2$$

Instead of comparing one pixel directly to another pixel, the area-based function compares a square patch to another square patch, where the square patches are respectively centered around p and q, where p in L and q in R, and $q = p+d$. Another useful aspect of the area-based approach is that it is useful in real time stereo systems because it can be accelerated by hardware implementations as noted in [8]. However, the algorithm will still produce undesirable results if there are texture-less patches smaller than the window size. Another pitfall to this algorithm is that it does not treat occluded pixels gracefully.

In addition to finding patches that correspond to one another, one can also incorporate a feature-based matching technique. This technique has proven to be robust to changes in lighting conditions because the underlying feature structure can be extracted for various lighting differences. One disadvantage of using feature-based matching is that unique features are often sparse, hence, the resulting disparity map will be sparse as well. If the depth is known to change smoothly throughout the scene, then filtering techniques could be applied to the resulting sparse map to produce a smoother result. However, if the depth in the scene is not smooth, then the filtering will likely cause a large number of depth estimation errors.

Another technique breaks the images into scanlines, where $L_{scanline}$ and $R_{scanline}$ represent the pixel intensity values of an entire row of pixels from their corresponding images. Dynamic Programming (DP) may be used to calculate how much one scanline should be shifted to match the other scanline, to minimize the difference between the respective scan lines.

More recent techniques have defined energy functions for f, $E(f)$, where the goal is to construct a disparity mapping that minimizes f. The energy function is typically defined as:

$$E(f) = E_{data}(f) + E_{smooth}(f)$$

where $E_{data}(f)$ measures how well the pixels in L correspond to the pixels in R under the mapping f and $E_{smooth}(f)$ usually applies penalties to pixels whose neighbors do not have the same label. The precise definition of these terms usually vary depending on the algorithm chosen to reduce the energy function. Some initial techniques to minimize f under E involve the use of simulated annealing. Although simulated annealing provided reasonable results, the algorithm does not run in a reasonable amount of time. This is largely because simulated annealing performs small, random, and local search steps during each iteration of the algorithm.

Interestingly, Greig et al. [6] discovered how to incorporate graph cut techniques into binary vision applications that required exactly two labels. Roy et al. [7] was able to extend the graph cut technique to include multiple labels, so that the technique could be applied stereo vision. Boykov et al. [2], built upon this discovery and developed a technique using graph cuts that involved the notion of alpha expansions, and alpha beta expansions. In related work, Kolmogorov et al. [1] used a similar technique to [2], but constructed their Graphs using different edge weights to include the energy of occluded pixels in their energy function. In fact, the rest of this paper is dedicated to Kolmogorov's algorithm [1] and presents the results obtained from

implementing their algorithm into a Python program. The following section will provide an overview of definitions that will be useful throughout the rest of the paper.

2 Graph Cut Method Overview

Let A be an unordered set of assignments $A = \{(p,q) \,|\, p_y = q_y \text{ and } 0 \leq q_x - p_x \leq k\}$ where k is the max disparity length and $p \in L$ and $q \in R$. Let $d(a)$ be the disparity of an assignment. Let $A^\alpha = \{(a) \,|\, d(a) = \alpha\}$ be the set of all assignments with disparity α. Let $f(a) = 1$ if a is an active assignment and 0 otherwise. Let $A(f)$ be the set of all active assignments. A configuration f' is within a single α expansion if $A(f') \subset A(f) \cup A^\alpha$. Let $N_p(a)$ be the number of assignments that p corresponds to.

The energy function to be minimized in [1] for stereo correspondence is defined as follows:

$$E(f) = E_{data}(f) + E_{smooth}(f) + E_{occ}(f)$$

$$E_{data}(f) = \sum_{a \in A(f)} D(a)$$

$$D(a) = (I_L(p) - I_R(q))^2$$

$$E_{occ}(f) = \sum_{a \in A, N_p(a) = 0} K$$

$K = 10\lambda$ and λ is some specified parameter.

In order to define the smoothness penalty, the notion of a neighborhood system on assignments as $\mathcal{N} \subset A \times A$ must be developed. $\mathcal{N}$ is restricted to only consist of assignments that have the same disparity.

The smoothness term can now be defined as:

$$E_{smooth}(f) \sum_{a_1, a_2 \in \mathcal{N}} V(a_1, a_2)$$

$$V(a_1, a_2) = \begin{cases} 3\lambda & \text{if } \max(|I(p) - I(r)|, |I(r) - I(s)|) < 8 \,|a_1, a_2 = (p,q), (r,s) \\ \lambda & \text{otherwise} \end{cases}$$

A directed graph, G, in [1] is constructed from assignments, so that each vertex, v, will correspond to an assignment from the set of assignments in an alpha expansion. $A(f') \subset A(f) \cup A^\alpha$. The edge directions and weights are defined in the table below. A min s-t cut on G will partition the vertices in a way that minimizes the energy function $E(f')$. A min s-t cut is the minimum set of all edges that need to be removed from G, so that there is no path from the node s to the node t and vice versa. A min s-t cut is the dual of the max flow problem and can be solved by calculating a max flow[3] in G.

edge	weight	for
(s, a)	$D_{occ}(a)$	$a \in A(f)$
(a, t)	$D_{occ}(a)$	$a \in A^{\alpha}$
(a, t)	$D(a) + D_{smooth(a)}$	$a \in A(f)$
(s, a)	$D(a)$	$a \in A^{\alpha}$
(a_1, a_2) (a_2, a_1)	$V(a_1, a_2)$	$\{a_1, a_2\} \in \mathcal{N}$ $a_1, a_1 \in A(f) \cup A^{\alpha}$
(a_1, a_2)	∞	$a_1 \in A(f), a_2 \in A^{\alpha})$ $a_1, a_2 \in N_p(f')$
(a_2, a_1)	K	$a_1 \in A(f), a_2 \in A^{\alpha})$ $a_1, a_2 \in N_p(f')$

3 Implementation Details

To analyze the algorithm's performance, a Python program was created to run the algorithm on various image sets from the Middleberry stereo dataset. Of course, there were advantages and disadvantages in choosing Python as the programming language. The primary advantage being that the syntax of the program was closer to a mathematical language then a lower level language like c++. However, the prototyping efficiency of Python comes at a cost of longer runtimes on the datasets compared to lower level language. The initial graphing library used to perform the min st-cut step was Networkx. This was originally chosen because the library had much better API documentation and had more API's to build a graph with. However, one drawback to Networkx, is that most of their API's are built entirely in Python because the library maintainers built the library with the goal of trying to minimize external dependencies. This pure Python approach resulted in very long run times while computing the min st-cut on a graph. Fortunately, the PyMaxflow library has a flexible API to build a directed graph, perform min st-cuts, and it's API is linked to underlying c++ function calls. The PyMaxflow library was able to perform the min st-cut in a matter of microseconds while the Networkx library took longer than ten seconds to perform the same cut.

Aside from using Networkx to perform the min-st cut, the other major bottleneck in the program came from constructing the graph needed for the alpha expansion step. Unfortunately, the only way to build the graph is to iterate over all assignments that are involved in an alpha expansion step and then add all edges required for that assignment. Note that each assignment is either in A^0 or A^{α}. To keep track of the active assignments, the arrays p_A and q_A were constructed, so that $p_A[p]$ holds some disparity, d, at pixel p. Suppose that $p + d = q$. Then $q_A[q]$ will also be d, so that the reverse assignment can quickly be computed, $q - d = p$. The arrays are a convenient data structure when considering all of the neighbors of an assignment $\langle p, q \rangle$. There are a total of eight potential neighbors to $\langle p, q \rangle$, according to the neighbor function N_1 (four assignments that are neighbors to p, and four assignments that are neighbors to q). The q_A is particularly convenient when needing to add the edges that correspond to the assignment neighbors of q. Note that each potential neighborhood of $\langle p, q \rangle$ was only considered if the neighboring assignment a' had the same disparity as $\langle p, q \rangle$. This was because [4] described the neighborhood system of this algorithm to consist only of assignments with matching disparities.

4 Results

The graph cut algorithm was analyzed by using a pair of datasets obtained from the Middleberry stereo data set site [4] and [5]. Each dataset consists of a rectified

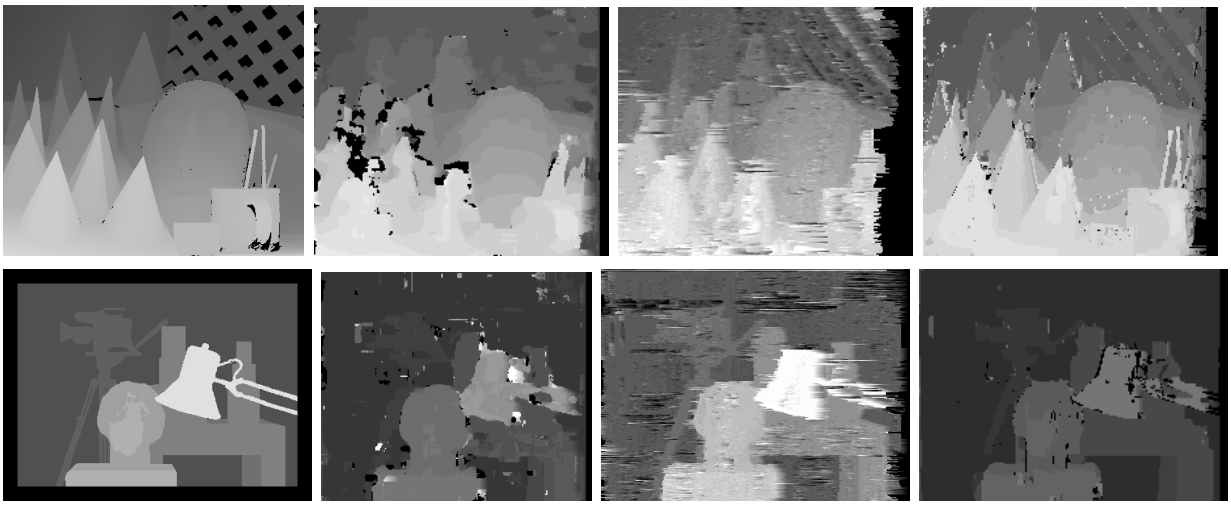


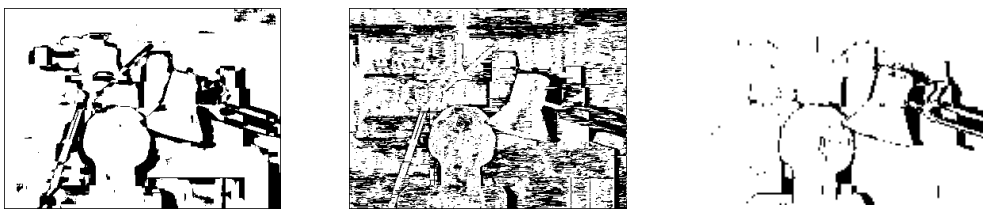
Table 1: **Top Left:** Cones ground truth [5]. **Top Middle Left:** SSD with window size equal to 7. **Top Middle Right:** Dynamic Programming (DP). **Top Right:** Alpha expansion, $\lambda = 4$. **Bottom Left:** Tsukuba ground truth [6]. **Bottom Middle Left:** SSD with window size equal to 5. **Bottom Middle Right:** Dynamic Programming (DP). **Bottom Right:** Alpha expansion, $\lambda = 2.5$.

left and right view, along with an image that encodes the ground truth disparity map. The first dataset will be referred to as the cones dataset [5], because it contains several cones placed irregularly throughout the image scene. The second dataset will be referred to as the Tsukuba dataset [4], which has been a classic dataset to use for measuring a stereo matching algorithm's performance.

The results were compared to the ground truth images by calculating the percentage of pixels that were within a one pixel range of the ground truth disparity. In addition to the graph cut method, two other algorithms (area-based SSD and DP scanline) were tested to compare the efficacy of the graph cut algorithm. Table 1 compares the results of the three approaches; it is clear that the disparity map from the graph cut approach resembles the ground truth image more closely, than the SSD and DP approaches. Figures 1 and 2 illustrate the results quantitatively.

(a) SSD: 16.52% (b) DP: 23.66% (c) Graph Cut: 11.76%

Figure 1: Error percentage for the Cones dataset.



(a) SSD: 19.85% (b) DP: 32.41% (c) Graph Cut: 8.23%

Figure 2: Error percentages for the Tsukuba dataset.

Through observing the results, the alpha expansion algorithm appears to make more mistakes near object boundaries that are likely to have been occluded from one of the views. This could be a likely reason why the algorithm classified more pixels incorrectly in the disparity map from the Cones dataset versus the disparity map from the Tsukubu dataset. The Cones dataset has more objects that are distributed in a way that makes them occluded from one of the views.

5 Conclusion/Acknowledgments

This project would not have been possible without the datasets from the Middlebury stereo dataset website. The ideas and algorithms that were implemented in this project came from the original work of [1] and [2] and would not have been possible without referencing their work. The work done by Georgia Tech Instructors and TAs to set up this project and provide guidance along the way is also very appreciated.

References

[1] Kolmogorov, Vladimir. Zabih, Ramin. (2019). Computing Visual Correspondence with Occlusions via Graph Cuts. https://www.cs.cornell.edu/rdz/Papers/KZ-ICCV01-tr.pdf. 11/20/2020.

[2] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. In International Conference on Computer Vision, pages 377–384, September 1999.

[3] L. Ford and D. Fulkerson. Flows in Networks. Princeton University Press, 1962.

[4] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision, 47(1/2/3):7-42, April-June 2002.

[5] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), volume 1, pages 195-202, Madison, WI, June 2003.

[6] Sebastien Roy and Ingemar Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In IEEE Proc. of Int. Conference on Computer Vision, pages 492–499, 1998

[7] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. Journal of the Royal Statistical Society, Series B, 51(2):271–279, 1989.

[8] Georgoulas C., Sirakoulis G. and Andreadis I. Real-time Stereo Vision Applications. http://www.br2-publication.com/download/publications/books/InTech-Real_time_stereo_vision_applications.pdf