Polygon-Based Stereo Matching Using Normalized Cross-Correlation

Bjorn Mellem & Francois Guiot
Computer Science
St. Olaf College
1500 St. Olaf Avenue
Northfield, MN 50057
mellem@stolaf.edu, guiot@stolaf.edu

Abstract

The stereo matching problem is at the heart of three dimensional computer vision research. In this paper we investigate a new algorithm for stereo matching for use in real-world datasets. Since these images suffer from variations in brightness and contrast, we chose normalized cross correlation as a basis for comparison between the two images. We present an efficient algorithm for window-based stereo matching based on normalized cross correlation. This method scales with the disparity volume in question and performs well in practical implementation. We discuss the run time of the algorithm in terms of both algorithmic complexity and issues of cache coherence. The algorithm is very fast in a CPU implementation and possesses room for improvement either by increasing cache coherence or implementing on an accelerator chip or GPU.

Introduction

In this paper we discuss an efficient stereo matching algorithm for use on real-world datasets. Our aim is to develop a robust method of stereo matching to be used for translating paired images of objects and spaces into three-dimensional models. We model our algorithm on the first two stages of Zeng-Fu Wang and Zhi-Gang Zheng's cooperative optimization algorithm, presented in "A Region Based Matching Algorithm Using Cooperative Optimization" (Wang and Zheng, 1-3).

The algorithm begins with a window-based stereo matching phase to establish disparity estimates for as many points as possible. This phase uses normalized cross correlation as a basis for comparing two windows. We present an efficient and exact method for calculating normalized cross correlation which scales with the disparity volume of the image pair in question.

The algorithm then uses segmentation of the reference image (here, the left image) into polygons to identify planar surfaces in the image. We use the RANSAC algorithm to fit a disparity plane to each polygon in the left image. During this phase, our goal is to fill gaps in data and lessen the influence of outliers on the final disparity map. We add several constraints to the processing and results of the RANSAC algorithm to handle poor data quality.

Finally, we aim to extend the algorithm to include the direct comparison of polygons. For this stage we segment both the left and right images and compare edges and paths in the left image to their counterparts in the right image.

Window-Based Normalized Cross Correlation

The first stage of our algorithm is a window-based method for calculating disparities using normalized cross correlation. Use of normalized cross correlation is motivated by its invariance under brightness and contrast variations. This allows it to accurately identify matches between real-world images suffering from common variations in data (Zhao et. al., 1).

The idea of normalized cross correlation is to find a pair of pixels p in the left image and q in the right image which maximize a correlation coefficient. The difference between the image coordinates of these two pixels gives the disparity for the pixel pair. For normalized cross correlation, the coefficient to maximize is defined by

$$\gamma(u,v) = \frac{1}{N}\frac{\sum_{x,y}\left(f(x,y)-\bar{f}\right)\left(t(x-u,y-v)-\bar{t}\right)}{\sigma_f\sigma_t}$$

where x and y range over the window around pixel p, (u,v) is difference between the location of pixel p in the left image and pixel q in the right image, N is the number of pixels in each window, and $f(x,y)$ and $t(x,y)$ are the intensities of pixel (x,y) in the left or right images, respectively. The mean values $\bar{f}$ and $\bar{t}$ are taken to range over the pixels in the left and right windows, respectively, as are the standard deviations σ_f and σ_t.

Direct computation of this coefficient can be extremely costly, especially the computation of the standard deviations σ_f and σ_t. We adopted a method for computing σ based on the formulas

$$\sigma_f = \frac{1}{N}\sum_{x,y} f(x,y)^2 - \bar{f}^2$$

and

$$\bar{f} = \frac{1}{N}\sum_{x,y} f(x,y)$$

These sums can be efficiently calculated using integral images over f and f^2, as shown by J. P. Lewis in “Fast Normalized Cross-Correlation” (Lewis, 4) and Briechle and Hanebeck in “Template Matching using Fast Normalized Cross Correlation” (Briechle and Hanebeck, 3). The same method is applied to calculating σ_t and $\bar{t}$.

Calculating the numerator of normalized cross correlation can also be computationally intensive. Several variations on a Fast Fourier Transform can be used. However, these require preprocessing using spectral filtering or careful choice of frequency cutoff parameters (Lewis, 3-4). A more recent approach involves calculating the numerator by approximating the factor $t(x-u,y-v)-\bar{t}$ using rectangular basis functions (Briechle and Hanebeck, 4).

We offer a method for computing the numerator both quickly and exactly. We begin by expanding the multiplication in the numerator:

$$\begin{aligned}&\left(f(x,y)-\bar{f}\right)\left(t(x-u,y-v)-\bar{t}\right)\\ &\qquad = f(x,y)t(x-u,y-v) - t(x-u,y-v)\bar{f} - f(x,y)\bar{t} + \bar{f}\bar{t}\end{aligned}$$

We wish to calculate the composite sum

$$\sum_{x,y} f(x,y)t(x-u,y-v) - \sum_{x,y} t(x-u,y-v)\bar{f} - \sum_{x,y} f(x,y)\bar{t} + \sum_{x,y} \bar{f}\bar{t}$$

Since $\bar{f}$ and $\bar{t}$ are constant over x, y in the summations, the last sum can be calculated as $N\bar{f}\bar{t}$ using the values of $\bar{f}$ and $\bar{t}$ which were computed along with standard deviations. Furthermore, the second and third sums can be computed from the integral images of f and t, simply multiplying by the constant mean values. The first term, $\sum_{x,y} f(x,y)t(x-u,y-v)$, is the only issue.

This can be solved by calculating integral images of the product ft over the range of possible disparity values $d \in [d_{min}, d_{max}]$. One integral image is needed for each disparity d. Using integral images in this way ensures that the intensity of any pixel p on the left image is multiplied by the intensity of a given pixel q on the right image at most once.

Efficiency of the NCC Matching Algorithm

We now show that our algorithm for matching runs in $O(M_x M_y D)$ time, where M_x and M_y are the dimensions of the image pair in pixels and $D = d_{max} - d_{min}$ is the maximum allowed disparity value. We assume the image pair is properly rectified, so that D is the number of possible windows in the right image to which a reference window in the left image must be compared.

Our algorithm first calculates integral images for f, f^2, t, and t^2. Each integral image takes $O(M_x M_y)$ time to compute. This is accomplished through the standard dynamic programming rule $s(x,y) = g(x,y) + s(x-1,y) + s(x,y-1) - s(x-1,y-1)$, where g is f, f^2, t, or t^2, and $s(x,y)$ is the value of the integral image at position (x,y) (Briechle and Hanebeck, 3).

Next, we pre-compute the sums $\sum_{x,y} f(x,y)$, $\sum_{x,y} f(x,y)^2$, $\sum_{x,y} t(x-u,y-v)$, and $\sum_{x,y} t(x-u,y-v)^2$, means $\bar{f}$ and $\bar{t}$, and standard deviations σ_f and σ_t. These must be computed for each window to be considered in either image. There will be approximately $M_x M_y$ such windows, one for each pixel in the image. These can be computed in constant time using the previously calculated integral images (Briechle and Hanebeck, 3). Thus these computations add only $O(M_x M_y)$ to the algorithm's run time.

The algorithm proceeds through each of the D disparity values. At each value, it computes an integral image of ft. It then computes the correlation coefficient for each of the $M_x M_y$ windows. It computes $\sum_{x,y} f(x,y)t(x-u,y-v)$ in three addition/subtraction operations per window using the integral image of ft. The rest of the quantities involved in the correlation coefficient can be drawn from pre-calculated values and assembled into the NCC coefficient in nine more arithmetic operations. Thus computing the correlation coefficient for a single value of (x, y, d) occurs in constant time. The total operations for computing these values is then $O(M_x M_y D)$ for the correlation coefficients plus $O(M_x M_y D)$ for the integral images of ft.

Therefore, the algorithm as a whole scales with the disparity volume $M_x M_y D$.

Cache Coherence in the NCC Algorithm

The normalized cross correlation algorithm owes its speed to pre-computing numerous means, standard deviations, and integral images. Storing this data can be memory intensive and cache coherence has a powerful impact on practical running time. The naïve approach of pre-computing all the integral images involved in the algorithm causes run time to suffer tremendously.

We approached this problem by filling the disparity volume one disparity value d at a time. This allowed us to keep one integral image of ft in memory along with pre-computed means, sums, and integral images of f, f^2, t, and t^2.

Nonetheless, cache coherence has an impact on larger images. Instrumenting our implementation revealed a 20% slowdown for images of 582 by 377 pixels and larger. Computation of the time per operation resulted in a at most of $1.3 * 10^{-8}$ seconds per operation for larger images. A small, 113 by 94 pixel image took only $1.1 * 10^{-8}$ seconds per operation. Computing the disparity volume by more localized chunks instead of dividing it solely by disparity value may improve cache coherence and reclaim this lost performance.

Polygon Segmentation

By using the color variations which occur in an image, it is possible to divide each image into polygons by gathering pixel of similar colors. We will assume that these polygons represent flat surfaces in three dimensional space. A separate process first divides images into polygons, placing edges along strong color changes. It then merges polygons which are part of the same surface. We use the output of this process to identify regions of each image which are planes in three-dimensional space.

RANSAC Plane-Fitting

The normalized cross correlation method creates good disparity maps but is neither error-free nor complete. Fitting a disparity plane to each available polygon can improve initial results. Since these polygons are supposed to represent planar surfaces within the image, disparity should vary smoothly over all points in a polygon. We fit a plane of the form $d(x, y) = ax + by + c$ to each polygon based on the computed disparities. Any three disparity values are sufficient to compute such a plane. However, some points may be outliers with incorrect disparity values. In order to reduce the influence of these outliers we use the RANSAC algorithm (Fischler and Bolles).

This algorithm fits an arbitrary number of plane generated from randomly picked points and checks how many other points fit those planes. The plane which receives the highest number of votes is considered to be the best plane for that polygon. Disparity values for all points in the polygon are then recomputed using this disparity plane (Fischler and Bolles).

We added checks to the RANSAC method to ensure that generated disparities are reasonable. If the number of data points in a polygon is below a threshold value, we discard the polygon. Once a plane has been computed for a polygon, we compute the new disparity of each vertex. If the new disparity for any vertex is above the maximum or below the minimum disparity obtained from the window based method, the computed plane is rejected and the polygon is discarded. Finally, points are weighted by their distance from the nearest edge of the polygon they inhabit. Points farther from the edges of a polygon are given higher weight than those near edges. This reduces the impact of foreground fattening in the window-based method.

Path-Based Matching

We then generally assume that edges of the created polygons will matches actual edges within the image. We also know that real edges in the image should appear in both images with an offset corresponding to the disparities of pixels along these edges. Thus we can create a disparity map from two images by matching edges in the two images and calculating the offset at each point along an edge.

Differences in luminosity or occlusions can lead to variations in the number of polygons or edges between the left and right images. A single surface in one image may be split into several polygons in the other, complicating the matter of matching edges. However, strong edges between separate surfaces should still appear as edges of polygons in both images. Thus it is possible to find those strong edges in both images and to compare them, while ignoring edges from which relevant information cannot be retrieved.

One approach is to compare edges situated on fast color changes within the image. However, these edges can be short and vary highly between the two images. Instead, a notion of an extended path is necessary. A path represents a set of continuous edges. We can compare two paths by overlaying their endpoints and computing the average distance between them.

A path can, for example, represent one edge of a table. The surface of the table has a varying disparity as all the table is not at the same depth. However, the table is a plane, so disparities will very smoothly across the table. Variations in disparity will shift and warp the edges of the table from one image to the other. However, we can find a linear transformation that will map the path to its corresponding path on the other image. In order to compute how much an edge shifts, we fit a disparity line $d = ax + b$ to the edge. If this transformation is accurately applied, the average distance between corresponding paths should be quite small. The main problem is therefore to find relevant paths among the data we can retrieve from the polygons and determine which paths to compare. Any set of continuous edges between the same two polygons should separate two surfaces. These are the paths which we wish to compare. In order to minimize statistical variations among edges, we wish to choose paths which are as long and clear as possible.

In order to aggregate edges into relevant paths that appear in both images of a matching pair, we introduce a notion of continuity between edges. Edges in an image can be represented with a graph where each edge is a node and where its neighbors are nodes sharing its vertices. For manufactured environments, most edges between planes are straight. The continuity between two neighboring edges is simply the angle between those edges. If the continuity between two neighboring edges is close to 180°, those edges should be merged into the same path for comparison between images.

Conclusion

We used our process to generate disparity maps for real-world image pairs taken from Regents Hall of Natural Sciences at St. Olaf College and on the “teddy” image from the Middlebury benchmark site (“Middlebury Stereo Evaluation”). The normalized cross correlation algorithm ran quickly when implemented on the CPU only. The “teddy” image took about 9.5 seconds to run, while full-sized, 2326 by 1508 real-world data took 8 minutes to process. RANSAC also executed quickly, taking about 1.5 seconds per image. The algorithm still has room for improving the implementation either by increasing cache coherence or implementing it on a GPU or equivalent accelerator chip.

Rectification problems present in the full-sized data prevented disparity maps from reaching the highest qualities.

The path based comparison method is still a work in progress. We have implemented a method of combining edges into paths, but have yet to see results from that algorithm either in terms of speed or output data quality.

References

Briechle, K., & Hanebeck, U. D. (2001). Template Matching using Fast Normalized Cross Correlation. *Proceedings of SPIE*, 1-8.

Fishcler, M., & Bolles, R. (1981). Random Sample Consensus: A Paradigm for Model Fitting with Application to Image Analysis and Automated Cartography. *Communications of the ACM*, 381-395.

Lewis, J. P. (1995). Fast Normalized Cross-Correlation. *Vision Interface*, 1-7.

Wang, Z.-F., & Zheng, Z.-G. (2008). A Region Based Stereo Mathing Algorithm Using Cooperative Optimization. *IEEE Conference on Computer Vision and Pattern Recognition*, 1-8.

Zhao, F., Huang, Q., & Gao, W. (2006). Image matching by normalized cross-correlation. *IEEE International Conference on Acoustics, Speech, and Signal Processing Proceedings*.