ARBITRARILY-SHAPED WINDOW BASED STEREO MATCHING USING THE GO-LIGHT OPTIMIZATION ALGORITHM

Xiaoyuan Su *Taghi M. Khoshgoftaar*
Computer Science and Engineering
Florida Atlantic University
xsu@fau.edu, taghi@cse.fau.edu

ABSTRACT

In this paper, we present a stereo matching algorithm using arbitrarily-shaped windows and a local optimization method called Go-light. The disparity map comes from a five-pixel arbitrarily-shaped window matching and a regular window based matching. It is then optimized by the Go-light optimization method, in which an outlier disparity value is replaced by the average of its surrounding ones when certain constraints are met. Experiments show that the accuracy of our algorithm is comparable to some of the state-of-the-art stereo correspondence algorithms on the Middlebury stereo data.
.

Keywords— stereo correspondence, arbitrarily-shaped window, Go-light optimization, stereo constraints, local stereo matching

1. INTRODUCTION

Stereo correspondence is one of the most active research areas in computer vision. The main task of stereo correspondence is to find the disparity map between a pair of images taken from two different viewpoints on the same scene. As stereo matching is an ill-posed problem with inherent ambiguities, it remains a difficult vision problem for the reasons of noise, textureless regions, depth discontinuity and occlusions [1].

Local methods (window-based) of stereo correspondence capture disparity only using intensity values within a finite neighboring window. Global methods of stereo correspondence such as graph cut [2][3] and belief propagation [1] are used to optimize the disparity map through various minimization techniques of energy that considers matching cost, depth discontinuities and occlusion.

For local stereo matching, small-window methods can accurately capture disparity in highly textured regions, but produce noisy disparities in textureless regions; while big-window methods produce smooth disparities in textureless regions, but are difficult to get accurate disparities for densely textured regions. Veksler uses variable windows [4] to avoid fixed windows size and take advantages of different window sizes. Kim et al. proposed a rod-shaped shiftable windows [5] to produce accurate disparity values for certain texture intensive regions. Rod-shaped shiftable windows typically use 36 orientations and their shapes are a short straight line (so-called rod-shaped), however, these windows are not flexible enough.

In this paper, we propose an arbitrarily-shaped window stereo matching method, which uses a five-pixel window that has arbitrary shapes and orientations. The arbitrarily-shaped windows can accurately capture the disparity for densely textured regions, and work together with a regular window that is good at matching textureless regions.

Instead of using energy minimization based optimization, we propose the Go-light optimization method for the disparity map. The idea was inspired from the game of Go, in which a white piece will be eliminated and claimed as opponent's territory when it's surrounded by black pieces. When a disparity is surrounded by different disparities, it can be replaced by its neighbors' average when certain conditions are met. We vary the distance values between the active point and its neighbors in the iterations of optimization and use threshold values to avoid over-pruning.

Go-light optimization is similar to a diffusion-based technique [6] with respect to its ability to smooth outliers. Its advantage is it is easy-to-implement and highly effective.

We work on the Middlebury stereo data and evaluate the performance of our algorithm in terms of the accuracy for all regions, non-occluded regions and depth discontinuity regions against the ground-true disparity maps according to the Middlebury test bed [7].

We describe the framework of our algorithm in Section 2. The experimental design and result are in Section 3. Our conclusions and future work are in Section 4.

2. FRAMEWORK

2.1. Local Stereo Matching

Local stereo correspondence methods only use the intensity values of pixels to make stereo matching.

Instead of using sum of squared differences (*SSD*) or normalized cross correlation (*NCC*), we use root mean squared error (*RMSE*) as our matching metric and use a universal threshold value for different window sizes to determine a match or non-match.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} | L_i(l,y) - R_i(r,y) |^2}$$

where N is the total number of pixels in a window, $L(l, y)$ and $R(r, y)$ are intensity values of pixels in the left window and right window.

For each pixel in each scanline in the reference image, we seek the most similar pixel in the same scanline of the other image, in terms of the smallest *RMSE*. If this *RMSE* value is smaller than a threshold value, we conclude that there is a match between the pixels and then calculate their difference along the horizontal axis as the disparity value. Otherwise, we report there is an occlusion here, which means there is no match for this pixel. If there is a match between pixels (l, y) and (r, y), the disparity value at (l, y) is $| l - r |$. A disparity map has the disparity values for every pixel in the reference image.

2.2. Arbitrarily-shaped Windows

We propose an arbitrarily-shaped window, which is adaptive for most kinds of shapes. As illustrated in Figure 1, where a regular square window can not find matches for certain regions in the pair of images (Figure 1(a)), an arbitrarily-shaped window can do it (Figure 1(b)). Our arbitrarily-shaped-window strategy is to try out all kinds of shapes and orientations and pick the winning shape that has the minimum similarity value in terms of *RMSE*.

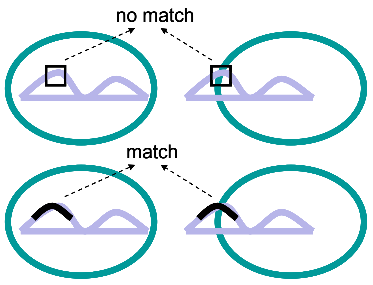

Figure 1, an illustration of arbitrarily-shaped windows (a) a regular square window can not find a match (b) an arbitrarily-shaped window can

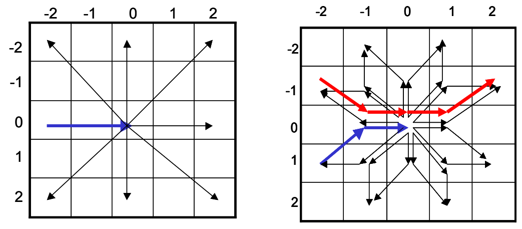

Figure 2, examples of arbitrarily-shaped windows (a) scenario A, and (b) scenario B

Within a 5*5 neighboring square (Figure 2), we have the point (0, 0) always located in the middle of five pixels. In scenario A, when the first three points are (0, -2), (0, -1) and (0, 0), and our search route for other pixels to form a unique 5-pixel combination ends at one of the other peripheral points, we will have seven different shapes/orientations. Starting from another peripheral point of the square and ending at a different peripheral point, we will have six more unique shapes/orientations (a duplicated one has been removed). Continue starting from any other peripheral point and ending at a different peripheral one, we will totally have $\sum_{i=1}^{7}(i)=28$ shapes/orientations for scenario A. In scenario B (Figure 2(b)), we use the remaining peripheral points of the square from scenario A. When our first three points are (1, -2), (0, -1) and (0, 0), we will have 15 different shapes/orientations (including the one with the same point as starting and ending points, but with different starting and returning routes). Taking other routes and making sure that the central point is (0, 0) and start point and end point are peripheral points of the square, we will get totally $\sum_{i=1}^{15}(i)=120$ unique shapes/orientation. Summed from these two scenarios, we will have a total of 148 different shapes/orientations to pick a five-pixel arbitrarily-shaped window.

When doing stereo matching, we seek the shape with the smallest *RMSE* value for the pair of pixels in both images, and if this value is smaller than the threshold value, we regard there is a match between these two pixels.

The five-pixel arbitrarily-shaped window is one of the smallest windows for stereo matching. Considering we need to try all shapes/orientations for each pair of pixels, the computation time is the five-pixel matching time multiplied by 148, which is equivalent to matching with a square window of size 27.

2.3. The Go-Light Optimization Method

As optimization through energy minimization of disparity and occlusion is difficult because exact inference is basically intractable, we propose a novel Go-light optimization method instead of using traditional belief propagation or graph cut methods.

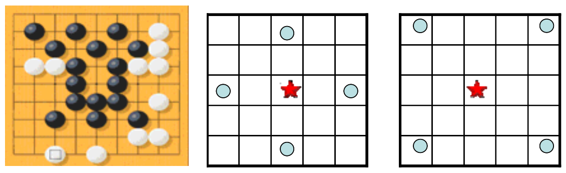

Figure 3, the Go-light optimization (a) the game of Go (b)(c) scenario A and B of the Go-light optimization method

Our Go-light optimization method was inspired from the game of Go, in which a black (or white) piece will be eliminated and claimed as its opponent's territory if it's surrounded by white (or black) pieces (Figure 3(a)). The underlying principle here is the disparity continuity assumption: in a small region, when a disparity value is greatly different from its surroundings, it is deemed as an outlier and should be replaced or optimized.

Instead of strictly enforcing the actual Go game rules, we use a loose-principled scenario, a Go-light optimization method. The Go-light optimization compares the disparity value of the central point with two groups of four certain-distanced neighbors (Figure 3(b)(c)). If the disparity of the central pixel is not equal to any of its neighbors' disparities, and its difference from the average of the neighbors' disparities is bigger than a threshold, it will be replaced by the average of the neighbors' disparities. Starting from one, we vary the distances incrementally in each iteration of the optimization, in which the central point has the same vertical and horizontal distances from any of its neighbors. To prevent over-pruning, we use threshold values, which are multiples of the product of the distance and the standard deviation of the neighboring disparities. The algorithm is illustrated in Figure 4.

```
Algorithm: Go-light (M, N, R, K, dis(disparity map), $\varepsilon$)
For Round r=1 : M (M<=5)
For iteration n=1 : N (N<=20)
For repeat_times=1: R (R<=5, constant for each iteration)
    distance d=n,
    threshold $T_i$=K*d*$std_i$ (std: standard deviation of neighbors, K=½ , 1, or 2, i=A, B scenarios (Figure 3))
    For each point dis(x, y) in the disparity map,
        If: dis(x,y) $\neq \forall\in$ {dis(x-d, y), dis(x+d, y), dis(x, y-d), dis(x, y+d)} && abs(dis(x,y)-$average_A$)>$T_A$ (here '$\forall\in$' means 'any of ')
        Then: dis(x,y)=$average_A$(dis(x-d, y), dis(x+d, y), dis(x, y-d), dis(x, y+d))
        If: dis(x,y) $\neq \forall\in$ {dis(x-d, y-d), dis(x+d, y+d), dis(x+d, y-d), dis(x-d, y+d)} && abs(dis(x,y)-$average_B$)>$T_B$
        Then: dis(x,y)=$average_B$(dis(x-d, y-d), dis(x+d, y+d), dis(x+d, y-d), dis(x-d, y+d))
        If $\Delta dis_n < \varepsilon$, $\Delta dis_r < \varepsilon$, Then: Exit (when difference of the disparity map between iterations/rounds is decreasing and is smaller than a limited small value, break the iteration/round)
```

Figure 4, The Go-Light Optimization Algorithm

3. EXPERIMENTAL DESIGN AND RESULTS

We work on Middlebury's four new evaluation data, Tsukuba, Venus, Cones and Teddy for quantitative evaluation [8]. We evaluate our algorithm in terms of the percentage of *bad* pixels, i.e., pixels whose absolute disparity error is greater than 1. Specifically, we calculate percentages for (1) pixels in non-occluded regions, (2) all pixels and (3) pixels near depth discontinuities. We ignore a border of 10 pixels for Venus, and 18 for Tsukuba when computing statistics [7].

For the local stereo correspondence, we use the disparity map from the five-pixel arbitrarily-shaped windows and a square window. When optimizing the disparity map using the Go-light algorithm, we use different parameter settings for different data. For example, for the data Cones, we use iteration numbers of 15, 15 and 2 and the parameter *K* of 1, 0.5 and 1 for the three rounds. For each round of optimization, a bigger iteration number means a bigger neighboring range, and a bigger *K* value means stricter thresholds.

In our experiments, we work on different extents of using the arbitrarily-shaped windows: a pure 3*3 window based matching (no arbitrarily-shaped windows, we call this case 1), a 90%-square-window-matching plus a 10%-arbitrarily-shaped-window-matching (case 2), and a 10%-square-window-matching plus a 90%-arbitrarily-shaped-window-matching (case 3). For case 2, we use the arbitrarily-shaped windows for locations where a regular 3*3 window can not find matches. For case 3, we use the regular square window with size of 9, 11 or 13 for the regions that the arbitrarily-shaped windows can not find matches, considering that arbitrarily-shaped windows are good at matching highly textured areas and big windows are good at matching textureless ones. Experimental result shows that except for the data *Teddy*, using of arbitrarily-shaped windows produce more accurate disparities (Table 1), and overall, case 2 performs the best of the three different usages of arbitrarily-shaped windows. Our arbitrarily-shaped windows matching does not perform well for the data *Teddy* as it is suitable for highly textured images, while *Teddy* has a big textureless region. It takes case 2 about 8 minutes to match the data Tsukuba, working on an Intel Pentium 4, 1G memory computer (a longer running time than global methods, but reasonable for a local one).

all %	Tsukuba	Venus	Teddy	Cones
case 1	5.21	4.07	**22.39**	18.36
case 2	**4.92**	3.57	22.62	**17.51**
case 3	4.99	**3.48**	22.66	19.07
nonocc %	Tsukuba	Venus	Teddy	Cones
case 1	3.54	2.98	**14.48**	10.11
case 2	**2.98**	2.52	16.49	**9.78**
case 3	3.22	**2.47**	16.44	11.68

Table 1, performance of using different extents of arbitrarily-shaped windows (in terms of percentage bad pixels for non-occluded regions and all regions)

The overall evaluation of our algorithm is in Table 2 and Figure 5. By the time of submission, our average rank on the Middlebury stereo evaluation webpage [8] is No. 21 for error threshold =1, and No. 16 for error threshold = 0.5. Compared with other disparity optimization methods, our algorithm is better than scanline optimization [7], and comparable with graph cuts using alpha-beta swaps [9] on the new version of Middlebury evaluation. Compared with other window-based stereo correspondence algorithms, our algorithm is better than the pixel-to-pixel algorithm [10] and comparable with the discontinuity preserving algorithm [11] and the variable window algorithm [4] on the previous version of Middlebury evaluation data. Due to limited space, we do not list all the evaluation results here.

The parameter settings for Go-light optimization can be unified for different data by analyzing the distributions of highly textured and textureless regions. We plan to investigate this in the future.

Tsukuba			Venus		
nonocc	all	disc	nonocc	all	disc
2.98	4.92	15.1	2.47	3.48	27.5
Cones			Teddy		
nonocc	all	disc	nonocc	all	disc
9.78	17.5	21.3	14.5	22.4	33.0

Table 2, evaluation of our algorithm on the Middlebury data (in terms of percentage of bad pixels for non-occluded, all and disparity discontinuity regions)

4. CONCLUSIONS

In this paper, we present an optimization method called Go-light for stereo correspondence, which replaces an outlier disparity value with the average of its surrounding ones when certain constraints are met, and effectively removes noises and enforces disparity continuity. To further improve the stereo matching performance, we combine the disparity values from an arbitrarily-shaped window based matching into a regular window based matching. Experiments show that using of arbitrarily-shaped windows generally produces more accurate disparities for stereo matching. Working on the standard Middlebury stereo data, the performance of our algorithm is comparable with some state-of-the-art algorithms.

It will be interesting to use our Go-light algorithm to optimize other stereo matching methods, and get our arbitrarily-shaped window based stereo matching optimized by a global optimization method.

REFERENCES

[1] J. Sun, N-N. Zheng, and H-Y. Shum, “Stereo Matching Using Belief Propagation”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 25, No. 7, 2003.

[2] V. Kolmogorov and R. Zabih, “Computing Visual Correspondence with Occlusions using Graph Cuts”, Proceedings of International Conference of Computer Vision, 2001.

[3] V. Kolmogorov and R. Zabih, “What Energy Functions can be Minimized via Graph Cuts?”, Proceedings of European Conference of Computer Vision, 2002.

[4] O. Veksler, “Fast Variable Window for stereo correspondence Using Integral Images”, IEEE Computer Vision and Pattern Recognition (CVPR’03), 2003.

[5] J.C. Kim, K.M. Lee, B.T. Choi, and S.U. Lee, “A Dense Stereo Matching Using Two-Pass Dynamic Programming with Generalized Ground Control Points”, IEEE Computer Vision and Pattern Recognition (CVPR’05), 2005.

[6] D. Scharstein and R. Szeliski, “Stero Matching with Non-linear Diffusion”, *IJCV*, 28(2), pp. 155-174, 1998.

[7] D. Scharstein and R. Szeliski, “A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms”, International Journal of Computer Vision, Vol. 47, No.1, 2002.

[8] Middlebury stereo, http://www.middlebury.edu/stereo

[9] Y. Boykov, O. Veksler, and R. Zabih, “Fast Approximate Energy Minimization via Graph Cuts”, PAMI, 23(11), 2001.

[10] S. Birchfield, and C. Tomasi, “Depth Discontinuities by Pixel-to-Pixel Stereo”, ICCV 1998.

[11] M. Agrawal, and L. Davis, “Window-Based Discontinuity Preserving Stereo”, CVPR 2004.

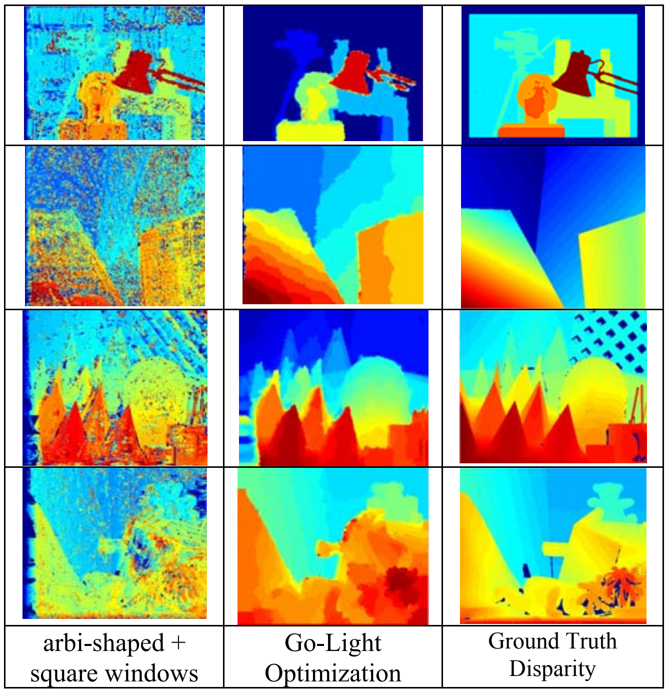

Figure 5, the result of our stereo matching algorithm (from top to down: Tsukuba, Venus, Cones, Teddy. Same color on different maps does not necessarily represent a same disparity)