Reducing Search Space for Stereo Correspondence with Graph Cuts

Olga Veksler
University of Western Ontario
Middlesex College, London ON Canada
olga@csd.uwo.ca

Abstract

In recent years, stereo correspondence algorithms based on graph cuts have gained popularity due to the significant improvement in accuracy over the local methods. Even though there has been a noticeable progress in efficient max-flow algorithms, the computational cost for graph cut stereo is still quite heavy, especially if the disparity search range is large. In this paper, we investigate and compare several ways of limiting the disparity search range. We show that the immediately obvious ideas based on thresholding or the hierarchical approach do not work reasonably well. We do, however, find that we can utilise the results of fast local correspondence methods for disparity range reduction of the more expensive graph cuts method. The idea is to understand and exploit the ways in which the local stereo correspondence methods fail. We are able to achieve 2.8 times average speed-up with only a modest degradation in performance, 1.7% average energy increase.

1 Introduction

In recent years, there has been a significant progress in addressing stereo correspondence. This progress can be objectively measured using to the Middlebury stereo database with ground truth developed by Scharstein and Szeliski [13]. Algorithms based on graph cuts are among the most successful according to the Middlebury database. The majority of these methods are based on the approach that was introduced by Boykov, Veksler, and Zabih in [4]. They formulate the problem in the global optimisation framework, with the energy function defined on the disparity map. The energy function penalises intensity differences between the corresponding pixels and discontinuities in the disparity map.

Since the original work, there were many improvements and extensions. Kolmogorov and Zabih [9] extended [4] by treating the left and right images symmetrically and modelling occlusions. This results in a better performance, but most importantly, proper occlusion handling is the key to the multi-view scene reconstruction [10].

Birchfield and Tomasi [2] developed an algorithm which instead of discretizing, finds disparity estimates in a continuous range. The algorithm iteratively segments the image into non-overlapping regions using graph cuts, and then fits an affine disparity model into each region. Lin and Tomasi [11] extended [2] by modelling each region with a spline and also modelling occlusions. Both of these methods in general work very well, but are very computationally intensive. In addition, since the two iterative steps are not tied

by a common objective function, these algorithms can produce poor results if a region segmented by graph cuts does not contain reliable information to estimate its disparity.

Another direction of research is to use segmentation in graph cuts stereo [8, 15, 6]. One or both images of the stereo pair are segmented and graph cuts optimisation is applied to segments, not to individual pixels. Disparity within each segment is assumed to follow one of a few specific models, for example a specific planar model. This approach saves computational time, however it fails when the segmentation results are misleading, that is when pixels not obeying any of the assumed models do fall into the same segment.

Computational efficiency of graph cut optimisation in computer vision has been greatly improved with the min-cut/max-flow algorithm of Boykov and Kolmogorov [5]. We used this algorithm for our implementation.

The running time of the algorithms above ranges from rather expensive (more than a minute on the standard images) to moderately expensive (several seconds on the standard images). Experimentally, the complexity of graph cut stereo is linear [5]. However, the constant overhead is quite large compared to the less accurate but more efficient methods such as the window matching or dynamic programming. Since the complexity grows linearly with the number of disparities, a natural way of speeding the algorithm up is to reduce the disparity search range. In this paper, we explore different ways of reducing the disparity search range. While our results can be applied to any stereo algorithm, we chose the graph cuts framework due to its relative efficiency and proven accuracy. Another close competitor would be belief propagation based stereo correspondence [14], but the straightforward implementation of belief propagation has a higher computational cost than graph cuts (although see [7] for an efficient implementation).

We have explored three types of approaches for limiting the disparity range. First approach is to retain only the best candidate disparities for each pixel. This simple technique works surprisingly poorly. The second approach is to perform stereo correspondence hierarchically, and use the lower resolution results to limit the disparity range at higher resolutions. This approach is also far from successful. The third approach is to use the results of a cheap local stereo algorithm to limit the disparity range for the more expensive graph cut algorithm. The idea is to analyse and exploit the failures of local correspondence algorithms. This third scheme is not only superior to the first two, but it also results in almost no loss in accuracy with a significant efficiency improvement, see Sec. 3.

For images with ground truth, we measure the actual accuracy of stereo correspondence. For the third reduction strategy based on local correspondence, we observed that if the accuracy goes down, it does so by an insignificant amount. A surprising result is that the accuracy occasionally goes up, we discuss the possible reasons in Sec. 3.

For images without ground truth, we use the percent of energy increase as a measure of performance. Although decrease in energy does not strictly corresponds to increase in accuracy, for a well-designed energy function, smaller energy values roughly correspond to better stereo results. This point is supported by the success of the energy optimisation approach to stereo, where it has been demonstrated that better optimisation methods (like graph cuts) lead to more accurate stereo results. We show that using our reduction strategy based on local correspondence, we can reduce the running time by an average of 2.8 times while the resulting energy is worse only by an average of 1.7%, where the analysis is done on a large dataset of 32 stereo pairs. This is a considerable improvement in efficiency gained for a small price in accuracy, and it moves the graph-cuts based algorithms closer to real-time implementation.

1.1 Stereo Correspondence with Graph Cuts

In this section we review the original graph-cut stereo correspondence of [4]. Let $\mathscr{P}$ be the set of all pixels in the left image, and $\mathscr{L}$ be the finite set of discretized disparities. For each pixel p in the left image, we need to find its disparity d_p. Let us use d to denote the set of all pixel-disparity assignments, that is d is a mapping from pixels $\mathscr{P}$ to labels $\mathscr{L}$. The desired disparity map d should minimise the energy

$$E(d) = \sum_{p \in \mathscr{P}} D_p(d_p) + \sum_{(p,q) \in \mathscr{N}} V_{pq}(d_p, d_q). \quad (1)$$

The set $\mathscr{N}$ includes all pairs of neighbouring pixels, which in our implementation is the standard 4-connected neighbourhood system of the 2D grid. The terms $D_p(d_p)$ measure how much pixel p likes the disparity d_p. This can be the absolute or squared difference between the pixel p in the left image and pixel p shifted by d_p in the right image. Following the previous work, we use the sampling insensitive similarity measure of Birchfield and Tomasi [1]. The smaller the values of $D_p(d_p)$ correspond to the more likely disparities d_p for pixel p. The pairwise penalty $V_{pq}(d_p, d_q)$ expresses our prior knowledge about the smoothness of disparity map d. It is used to encourage nearby pixels to have similar or the same disparities. In this paper, we use a simple discontinuity-preserving Potts model [4], which simply penalises nearby pixels if they have different labels. That is $V_{pq}(d_p, d_q) = \lambda \mathscr{I}(d_p \neq d_q)$, where $\mathscr{I}(\cdot)$ is 1 if its argument is true and 0 otherwise. Optimising the energy in equation (1) is NP-hard, and we use the α-expansion algorithm of [4] to find an approximate solution, which is within a factor of 2 of the optimum. However, in practice, the solution is much closer to the optimum, see [12]. We found it is sufficient to run the α-expansion algorithm for two cycles, since there is no significant energy reduction after the second cycle.

2 Reducing the Disparity Search Range

The running time of stereo with the α-expansion depends linearly on the number of disparity labels searched. Thus reducing the number of disparities explored per pixel by half reduces the running time by half. In this section we evaluate three different approaches to reducing the disparity space: best-candidate based, hierarchical, and a local-method based. We demonstrate experimentally and explain why the first two approaches do not work well, and why the last approach is able to achieve impressive speedup without much loss in performance. The third approach is evaluated extensively in Sec. 3.

2.1 Filtering

Before describing the reduction approaches, we explain the filtering step which is common to all of them. For each pixel p, let $\mathscr{L}_p \subset \mathscr{L}$ denote the set of all disparity candidates chosen for pixel p by a reduction algorithm. Let $\mathscr{P}_l = \{p \in \mathscr{P} | l \in \mathscr{L}_p\}$, that is $\mathscr{P}_l$ is the set of pixels for which the disparity l is chosen as a candidate disparity. For each $l \in \mathscr{L}$ we will enlarge $\mathscr{P}_l$ as follows: $\mathscr{P}_l^{enlarged} = \{p \in \mathscr{P} | |p - p'| < h \text{ for some } p' \in \mathscr{P}_l\}$, where $|p - p'|$ is the Manhattan distance between pixels p and p'. For some of the disparity reduction algorithms, the parameter h can be chosen intelligently, as will be described below. For other reduction algorithms, enlarging $\mathscr{P}_l$ serves as a filtering step to fill in

any holes which may exist due to noise or other artifacts. The final algorithm performs α-expansions only for pixels $p \in \mathscr{P}_{\alpha}^{enlarged}$, for each $\alpha \in \mathscr{L}$. Given $\mathscr{P}_l$, the set $p \in \mathscr{P}_{l}^{enlarged}$ can be computed very efficiently using distance transforms [3].

2.2 Reduction Approach 1: Best Candidate Based

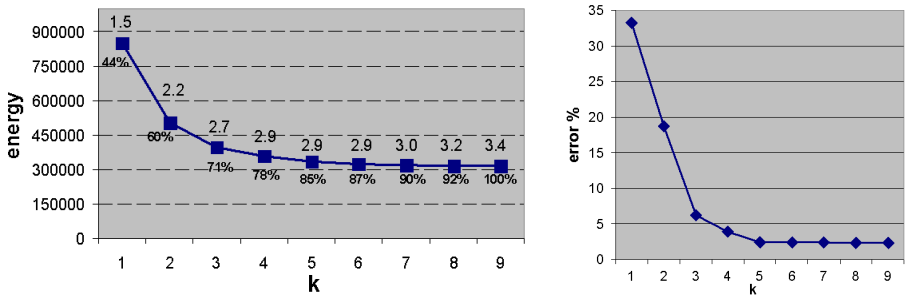
The most obvious way to reduce the search space is to select only the "best" disparities for each pixel based on the individual matching scores $D_p()$. There are two simple ways to select the "best" disparities. First way is for each pixel p, threshold the values of $D_p()$, that is select all $l \in \mathscr{L}$ such that $D_p(l) < t$ for some threshold t. The second way is to simply select the k best disparity labels, i.e. the labels l corresponding to the k smallest $D_p(l)$'s. In both cases, we have to make a choice for a nuisance parameter, namely the threshold t in the first case, and k in the second case. It turns out that in general, there is no good value for either parameter t or k, if the image has large low texture areas.

Consider the "tsukuba" scene[1] in Fig. 2(a). The chart of energy vs. parameter k is in Fig. 1(a). For each value of k, above the square we show the running time in seconds (which is the time to compute the disparity reduction and the time to do optimisation with two cycles of α-expansion algorithm). Below the square we show the percentage of all possible disparity labels for all pixels evaluated (that is $100 * \frac{\sum_{l \in \mathscr{L}} |\mathscr{P}_l|}{|\mathscr{L}| * |\mathscr{P}|}$, where $|S|$ denotes the size of a set S). To get a reasonable approximation to the energy which can be achieved with the full disparity range (which corresponds to k=15, since $|\mathscr{L}|$ = 15), we have to take k to be at least 3, in which case we have to evaluate already 71% of all possible disparity/pixel pairs, which means there is almost no savings in time, 2.7 seconds vs. 3.4 seconds if no disparity reduction is performed[2]. The chart of percentage error vs. parameter k is shown in Fig. 1(b). Here we compute the error as the percent of pixels which are more than 1 disparity away from the ground truth, the same measure as the one used in [13]. Again, to get a reasonable accuracy in stereo matching, we have to set $k = 3$, and there is almost no computational savings for this value of k.

We conclude from Fig. 1 that for no value of k we get a significant speed up in computational time without a significant gain in the energy. The same is true if look at the other way of choosing the best candidates, by choosing only the disparity labels l such that $D_p(l) < t$ for some threshold t. If we construct the graph of energy vs. t (omitted here to save space), we will see a chart similar to that in Fig. 2. For example, the percentage error corresponding to $t = 2$ is 4.63, which is significantly worse than the error of 2.09 which we get with no disparity space reduction. However the running time for $t = 2$ is already 2.4 seconds, which is not much better than the full disparity range graph-cuts time of 3.4 seconds. Notice that the degradation in energy directly corresponds to the quality of the disparity map. For both methods discussed in this section, we set the filtering parameter h, discussed in section 2.1 to 3. This filtering step fills in any gaps in $\mathscr{P}_l$ due to noise. If we choose a smaller value for h, the running time will improve but at the cost of significantly worse energy value. If we choose a larger value for h the energy values will improve at the cost of significantly larger running time. For no value of h we found the results on the scene in Fig. 2(a) to be satisfactory, that is either the energy value is too far from the

[1] This stereo pair is taken from the Middlebury database, but it is originally from the University of Tsukuba.

[2] The running time shown in the graph in case $k = 15$ (3.4 seconds) does not include the time to perform the reduction step, since it's redundant for $k = 15$. That is why the savings in time for $k = 3$ is slightly worse than 29%, because the running time for $k = 3$ is the total running time which does include the reduction step.



(a) (b)

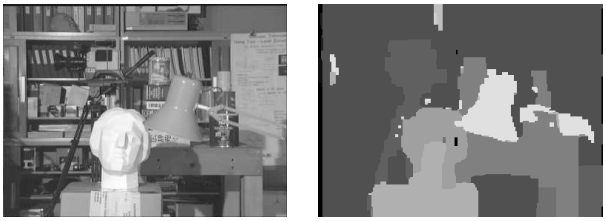
Figure 1: Results on stereo pair in Fig. 2(a). In (a) we show energy vs. k; running times are in seconds above squares; percent of disparity range reduction is below the squares. In (b) we show error percent vs. k, where error is counted as in [13]

desired one, or the running time is not significantly better compared to the running time without the disparity space reduction.

The reasons for such a poor performance of the best candidate strategy reduction are easy to understand. When the scene has significant areas of low texture, the pixels in the low texture areas have many good disparity candidates. Choosing k best candidates selects some disparities essentially at random, and for many pixels the correct disparity is not included among the k best candidates unless k is large. However if k is large, then many disparity/pixel pairs will need to be evaluated, thus the resulting energy will be good (not far from the energy with no disparity space reduction), but the there will be very little savings in the running time. This is exactly the picture we observe in the graph in figure 1. Similar analysis holds with the best candidate selection based on thresholding. With the appropriate choice of t (that is t large enough to include the correct disparity in the selected disparity candidates set) the pixels in the texture-less areas will have many disparity candidates selected, and thus there will be no significant savings in speed.

2.3 Reduction Approach 2: Hierarchical

In this section, we explore the hierarchical approach for disparity range reduction, which is a standard approach to reducing search space. We construct a Gaussian pyramid and compute disparities at each level using graph cuts. We use disparity results at coarser levels to restrict the disparity range at higher levels. More specifically, suppose the disparity of pixel p computed at a course level is l. Then we allow the disparity of p at the next level of the pyramid to be between $l-\delta$ and $l+\delta$. To get reasonable results, we found that it is also necessary to reduce λ in the energy function at coarser levels, we divided λ by 2 for each level of the pyramid. After experimenting with a range of δ, we came to the conclusion similar to that in section 2.2. Choosing a small value of δ results in good efficiency but a significant loss in accuracy. Larger values of δ result in good accuracy but no gain in efficiency. Here is a summary of performance on the scene in Fig. 2(a): if $\delta = 1$, the error is 9.7% and the running time is 0.87 seconds; if $\delta = 2$, the error is 6.88% and the running time is 1.46 seconds; if $\delta = 3$, the error is 4.17% and the running time is 1.98 seconds. The results for $\delta = 3$ are shown in Fig. 2. As can been seen from this



(a) (b)

Figure 2: (a) "Tsukuba" stereo scene; (b) results of hierarchical reduction for $\delta = 3$.

figure, if there is a mistake made at a coarser level, it gets propagated to the finer and final levels.

2.4 Reduction Approach 3: Local Algorithm Based

Stereo correspondence is an inherently ambiguous problem, where each pixel needs to cooperate with neighbours to reduce the matching ambiguity. Suppose a pixel p has ten disparity candidates with almost equally low matching score D_p. Without cooperation from its neighbours, we cannot know which disparity candidate is the correct one. Thus we must include all ten disparity candidates in $\mathscr{L}_p$, which does not help to reduce the disparity search space much. In other words, the problem with the best candidate reduction strategy is that no steps are taken to reduce the matching ambiguity when constructing the reduced disparity space. In this section, we will show how to use fast but not very accurate stereo algorithms to reduce the ambiguity in matching, and how this will help to construct a much smaller disparity search space without significant degradation in the results. We explore two fast stereo algorithms, window matching and dynamic programming.

Reduction Strategy Based on Window Matching

In this section, we show how to use the results of a window matching algorithm to reduce matching ambiguities and to produce a smaller disparity search space which is still quite likely to contain the correct match. We use a variation of the standard window matching algorithm, where each pixel p with coordinates (x,y) gets assigned disparity d_p such that

$$d_p = \text{argmin}_{l \in \mathscr{L}} \sum_{-s \le i \le s, -s \le j \le s} D_{(x+i,y+j)}(l),$$

where s is the window size and $D_{(x+i,y+j)}(l)$ is the matching cost for pixel $(x+i,y+j)$ at disparity l.

It is well known that when performing window matching with a window of radius h, the object boundaries either shrink or fatten by a margin of up to h, depending on the image texture. This gives a simple idea for producing an appropriate reduced disparity space from the results of the window matching: a label l should be in the set of candidate disparities for pixel p if there is a pixel q within Manhattan distance of h which got assigned disparity l by the window matching algorithm. Visually this corresponds to taking the results of window matching, expanding each region assigned to disparity l by

a margin of h, and taking the resulting region to be $\mathscr{P}_l$, that is the set of pixels for which disparity l is a candidate disparity in the reduced search space. This can be done very efficiently using the technique described in section 2.1. Notice that using notation of section 2.1, $\mathscr{P}_l = \{p | d_p^{window} = l\}$, where d_p^{window} is the disparity of pixel p computed by the window matching algorithm. $\mathscr{P}^{enlarged}$ as defined in section 2.1 is then exactly the set we use for the reduced disparity space as described in the previous paragraph.

Another way of looking at the proposed reduction algorithm is as follows. When we aggregate the matching costs in a window centred at pixel p and find the disparity corresponding to the best aggregated cost, what we are really doing is finding the correct disparity for the majority of pixels in a window centred at p, but we do not know for which pixels in the window the computed disparity is the correct one. Thus when we assign the winning disparity l to p, what we are really saying is that disparity l is likely to be the true disparity for the majority of pixels in the window centred at p. Thus to collect the likely disparity candidates for p, we must find all disparities assigned to any pixel q that may include p in the window centred at q. Such pixels q are exactly those which lie at Manhattan distance of less than h from p, if the window radius is h, and thus we have an intelligent choice of h to be used in the "filtering" step of section 2.1.

The only question that remains is how to set the radius h. It should not be too large so that the thin objects do not completely disappear from the disparity map and then there is no chance to recover their disparities in our reduced disparity space. And, of course, h should not be too small so that there is enough disambiguation. In our implementation, we actually use 2 window sizes, a smaller window and a larger window. This way, the time to perform window matching is still small, while the smaller objects are not "erased" and the disambiguities are significantly reduced. We used window radiuses of $h = 2$ and $h = 8$. Thus in the final algorithm, we perform window matching with $h = 2$, then perform the filtering step of section 2.1 to get the reduced disparity range $\{P_l^2 | l \in \mathscr{L}\}$, where superscript 2 denotes the fact that we performed matching with window of size 2. Then we do the same steps for window of radius $h = 8$, to get the second reduced disparity space $\{P_l^8 | l \in \mathscr{L}\}$. Finally we combine both disparity spaces, and the final disparity search space is $\{P_l = P_l^2 \cup P_l^8 | l \in \mathscr{L}\}$.

Reduction Strategy Based on Dynamic Programming

Another efficient stereo algorithm is based on dynamic programming. Here we use the idea of [13] that stereo correspondence can be performed by optimising the energy in Eq. (1) on each scanline individually using dynamic programming, ignoring the vertical penalty terms. In our implementation, we use the speed-up in [7] for efficiency.

After we perform dynamic programming, we need to extract a reliable reduced disparity space. The correct disparities for each scanline may correspond not to the best solution for that scanline, but to a solution which is not too far in terms of energy form the best solution. Extracting the first several best solutions from the matrix computed during DP is not very efficient, and thus we have opted for a heuristic that extracts several solutions with energies which are not too far from the best solution.

Here is how our heuristic works. Suppose we are performing dynamic programming on a scanline corresponding to some fixed y. During dynamic programming, we compute cost matrix $C(x,l)$, where $C(x,l)$ is the cost of the best possible assignment to pixels with coordinate $x' \leq x$ if pixel with coordinate $x+1$ is assigned disparity l. We also store the

Figure 3: Some of the 32 stereo scenes we used in evaluation.

label of pixel x corresponding to such best cost assignment in matrix $L(x,l)$. In addition to this information, we compute the cost of the second best assignment to pixels with coordinates $x' \leq x$ if pixel with coordinate $x+1$ is assigned disparity l, as well as the label of pixel x in this second best cost assignment. Having computed these cost and label matrices, we can now find k paths with costs close to the cost of the best path. When computing the best path, we have to trace from the end of the scanline to the beginning tracing the best label assignments. Suppose that at some pixel with coordinate x, the cost of the best path is C_1 and the cost of the second best path is C_2, and C_2 is not much larger than C_1. Let P be the path that follows the best path until pixel with coordinate x and then switches to the second choice at pixel x and tracing that new path to the beginning of the scanline. Then the cost of path P is larger than the best path by only $C2-C1$. Thus in order to find k paths with costs not too far from the best path cost, we find k pixels with the smallest difference between the second and the first choice path costs.

After the k low cost solutions are produced, a label l is put into L_p if pixel p is assigned disparity l by at least one of the k solutions. Finally, we perform the filtering described in section 2.1. Unlike the window matching approach, there is no clear intuition on how to set parameter h for filtering. For all our experiments, we set $h = 6$ and $k = 10$.

3 Experimental Results

In this section we present the experimental results. For the energy in eq. (1) we chose $\lambda = 40$. All the other parameters are as discussed in sections 2.4 and 2.4. We only give the results for the reduction strategies in Sec. 2.4, since the reduction strategies in Sec. 2.2 and Sec. 2.3 are too poor to be useful, as discussed earlier. All experiments were performed on Pentium 4CPU 2.6GHz.

We have evaluated the reduction algorithms on a dataset of 32 real stereo pairs. Some of these images are from the Middlebury database and have ground truth. Other images were found on the Internet and include images from CMU, Microsoft, Tsukuba and Stanford universities. A few examples of the images we used are shown in figure 3.

The results are summarised in figure 4. The second line has results of window based reduction, and the third line has the results of DP based reduction. The second column gives the average (over the dataset of 32 stereo pairs) increase of energy as compared

	Mean±Std. Dev. % Energ. Incr.	Mean±Std. Dev. Speedup (factors)	Eng. Incr. Range	Speedup Range
Window Based	1.65±1.03	2.81±0.82	[0.0947,4.32]	[1.53,4.43]
DP Based	3.44±2.23	3.22±0.89	[0.076,9.27]	[1.63,4.9]

Figure 4: Summary of Results on the 32 stereo pairs. See text for explanation.

to the full disparity search algorithm. The energy increase is measured as percentage increase from the non-reduced disparity version. The same column also show the standard deviation of the percentage energy increase. The third column shows the average and the standard deviation of the speed-up factors achieved with the reduction over the full disparity search space. Notice that for the reduced disparity range versions, we count the total time, i.e. the time to do disparity reduction and time to do graph cuts. The last 2 columns show the ranges of percent energy increase and the range of speedup factors, to give the idea about the best and worst case performance of the reduction algorithms.

The window-based scheme works better, on average. The average increase in energy is only 1.65% and the average running time speed-up is a factor of 2.88. One reason why window-based scheme may work better is that we have a better understanding the types of errors made by the window-based algorithm and how to construct the reduced disparity range not effected by these errors.

For the Middlebury images with ground truth, we can compare the percentage errors made by the full search and the reduced search algorithms[3].These errors are summarised in figure 5. In most cases, the errors for the window reduction method are not significantly worse, and in some cases, like the Venus scene, the results are significantly better than the non-reduced graph cuts stereo. This suggests that when the disparity at some pixel p differs between the full search algorithm and window based reduction algorithm, neither algorithm finds the correct disparity for p, which implies that establishing the correct disparity at p is difficult. The results on the Venus scene suggest that occasionally the window based reduction discards certain disparities that the full search algorithm erroneously assigns. Due to unfortunate choice of parameter λ, the full search algorithm might have oversmoothed the Venus scene, whereas the window reduction algorithm does not have the opportunity to oversmooth because the disparities needed for oversmoothing are recognised as erroneous and are not included in the reduced disparity space. Running times in seconds for the images in Fig. 5 for the full disparity search were: 3.57, 9.29, 20.9, 21.2, 7.6, 4.35, listed in the same order as the images appear in the table in Fig. 5. For the window based disparity reduction the running times, in the same order, were: 1.29, 2.6, 6.09, 5.8, 1.87, 0.96, thus the speedup factor, on average, for these images is 3.7. Maximum speed up of 4.5 is on the *Map*, which is the most textured one, and minimum speedup of 2.7 is on the *Tsukuba*, which is most textureless one.

References

[1] S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE TPAMI*, 20(4):401–406, April 1998.

[3] We computed these errors in the same way as in [13].

	Tsukuba	Venus	Teddy	Cones	Sawtooth	Map
full range disparity search	2.09	2.55	13.8	9.06	1.05	0.5
DP reduction	2.96	2.23	13.9	9.07	1.17	0.46
Window reduction	2.22	1.39	12.8	8.87	1.18	0.51

Figure 5: Error percentage on Middlebury images with ground truth

[2] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *ICCV99*, pages 489–495, 1999.

[3] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *PAMI*, 10(6):849–865, November 1988.

[4] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *IEEE CVPR*, pages 648–655, 1998.

[5] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE TPAMI*, 26(9):1124–1137, September 2004.

[6] Y. Deng, Q. Yang, X. Lin, and X. Tang. A symmetric patch-based correspondence model for occlusion handling. In *ICCV 2005*, pages II: 1316–1322, 2005.

[7] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient belief propagation for early vision. In *CVPR 2004*, pages I: 261–268, 2004.

[8] L. Hong and G. Chen. Segment-based stereo matching using graph cuts. In *CVPR04*, pages I: 74–81, 2004.

[9] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions via graph cuts. In *ICCV*, July 2001.

[10] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *7th ECCV*, volume III of *LNCS 2352*, pages 82–96, Copenhagen, Denmark, May 2002. Springer-Verlag.

[11] M.H. Lin and C. Tomasi. Surfaces with occlusions from layered stereo. *PAMI*, 26(8):1073–1078, August 2004.

[12] T. Meltzer, C. Yanover, and Y. Weiss. Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *ICCV05*, pages I: 428–435, 2005.

[13] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1/3):7–42, April–June 2002.

[14] J. Sun, N. Zheng, and H. Shum. Stereo matching using belief propagation. *IEEE TPAMI*, 25(7):787–800, September 2004.

[15] Y. Wei, M. Lhuillier, and L. Quan. Fast segmentation-based dense stereo from quasi-dense matching. In *ACCV 2004*, 2004.