High-quality Real-time Stereo using Adaptive Cost Aggregation and Dynamic Programming

Liang Wang, Miao Liao
University of Kentucky
Lexington, KY, USA

Minglun Gong
Laurentian University
Sudbury, ON, Canada

Ruigang Yang, David Nister
University of Kentucky
Lexington, KY, USA

Abstract

We present a stereo algorithm that achieves high quality results while maintaining real-time performance. The key idea is simple: we introduce an adaptive aggregation step in a dynamic-programming (DP) stereo framework. The per-pixel matching cost is aggregated in the vertical direction only. Compared to traditional DP, our approach reduces the typical "streaking" artifacts without the penalty of blurry object boundaries. Evaluation using the benchmark Middlebury stereo database shows that our approach is among the best (ranked first in the new evaluation system) for DP-based approaches. The performance gain mainly comes from a computationally expensive weighting scheme based on color and distance proximity. We utilize the vector processing capability and parallelism in commodity graphics hardware to speed up this process over two orders of magnitude. Over 50 million disparity evaluations per second (MDE/s)[1] are achieved in our current implementation.

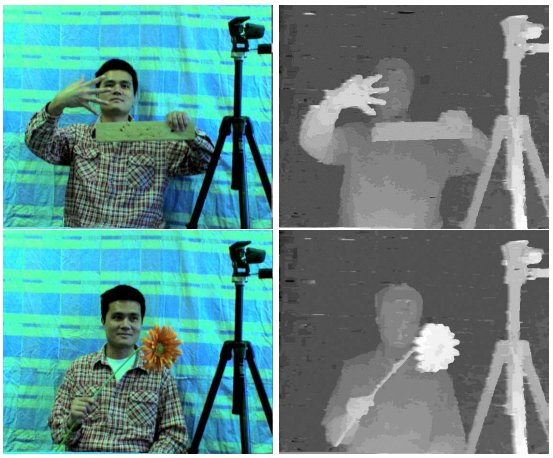

Figure 1. Two sample images and their depth maps from our live system on a 3.0GHz PC with an ATI's Radeon XL1800 graphics card. We can achieve 43 fps with 320×240 input images and 16 disparity levels.

1. Introduction

Depth from stereo has traditionally been, and continues to be one of the most actively researched topics in computer vision. Recent development in this area has significantly advanced the state of the art in terms of quality. However, in terms of speed, these best stereo algorithms typically take from several seconds to several minutes to generate a single disparity map, limiting their applications to off-line processing. There are many interesting applications, such as robot navigation and augmented reality, in which high-quality depth information at video rate is crucial.

For real-time online stereo processing, the options are rather limited; only local approaches and scanline-based optimizations are feasible. Most local approaches, while being fast, are quite fragile and prone to problems on occlusion boundaries. Scanline-based optimization utilizes dynamic programming (DP) to produce better results on occlusion boundaries, but the results are often accompanied by inconsistency between scanlines, i.e., "streaking". Recently, multi-pass DP algorithms [10, 14] were developed to reduce this unpleasant artifact, but at the cost of speed.

Our approach is inspired by a recent paper from Yoon and Kweon [29]. They presented a new cost aggregation scheme that uses a fix-sized support window with per-pixel varying weight. The weight is computed based on color similarity and geometric distance to the center pixel of interest. Very strong results were obtained without any global optimization. However, the aggregation process is computationally expensive. As reported in [29], it took about one minute to produce a small depth map, which makes real-time application impossible.

We instead introduce this adaptive scheme into a DP framework. The per-pixel matching cost is aggregated in a

[1] The number of disparity evaluations per seconds (MDE/s) corresponds to the product of the number of pixels times the disparity range times the obtained frame-rate and therefore captures the performance of a stereo algorithm in a single number.

1D vertical window. It significantly reduces the "streaking" artifacts without using multiple DP passes. Furthermore, we implemented this cost aggregation scheme on the graphics hardware to speed up the computation over 10 folds. The aggregated cost volume is transferred back to CPU for final disparity selection using DP. Thus our approach not only makes use of both CPU and GPU in parallel, but also makes each part do what it is best for: the graphics hardware performs image warping and aggregation in massive parallelism, and the CPU performs DP that requires more flexible looping and branching capability. Therefore we can produce high-quality depth map at video rate, as shown in Figure 1. More live results can be found in the accompanying video.

In addition we propose an approximation to the full adaptive window in [29]. Disparity maps with comparable quality to that in [29] can be generated in software in terms of seconds, instead of minutes. An implementation on the graphics hardware further improves the speed by one or two orders of magnitude.

We have evaluated our approach using the Middlebury stereo benchmark data set. Among all DP-based algorithms reported there, it is ranked first in the new evaluation system (the old evaluation system is no longer functional). Combined with its high speed capability, our approach is a step forward to distill recent advances in stereovision to enable real-time applications that require high quality depth data.

2. Related Work

The current state-of-the-art stereo algorithms all use some form of *global* optimization to estimate the disparity maps. These algorithms are not feasible for real-time applications. For the scope of this paper, we briefly overview several classes of algorithms that can achieve video rate. Interested readers are referred to a recent excellent survey and evaluation by Scharstein and Szeliski [21] for other techniques.

Local stereo algorithm These algorithms take the disparity decision by choosing the one with the best matching score for each pixel, i.e., a winner-takes-all (WTA) approach.

Only a few years ago even these approaches were out of reach of standard computers so that special hardware had to be used to achieve real-time performance [8, 13, 25, 15, 7]. Software-only real-time systems began to emerge at the turn of the century. For example, Mulligan and Daniilidis proposed a new trinocular stereo algorithm in software [17] to achieve 3-4 frames/second on a single multi-processor PC. Hirschmuller introduced a variable-window approach while maintaining real-time suitability [12, 11]. The algorithm achieves about 12M disparity estimation per second (MDE/s) on a 450MHz Pentium II computer. Commercial solutions are now also available. The stereo algorithm from Point Grey Research [19] yields approximately 80 MDE/s on a 2.8Ghz processor. All these methods used a number of techniques to accelerate the calculation, most importantly, assembly level instruction optimization using Intel's MMX extension.

Acceleration using graphics hardware Driven by the consumers' need for more realistic rendering, the graphics hardware has become so versatile that it can be used for applications other than rendering. Its potential to accelerate depth estimation was first explored by Yang *et al*. [28]. A plane-sweep approach [3] was adopted to effectively use the graphics hardware's capability to warp and process images (textures). Later, they improved this technique in terms of both speed and accuracy [27, 26]. Their new approach can achieve 117 MDE/s on an ATI Radeon 9800 XT card.

Zach *et al*. introduced a mesh-based stereo algorithm that lends itself well to implementation on commodity graphics hardware [30]. It iteratively refines a 3D mesh hypothesis to estimate the depths. Cornells and Van Gool combined plane-sweep and iterative refinement to compute the depths for fine 3D structures [4], all the computations are carried out on the graphics hardware. Both approaches can generate quality depth maps but require quite a few iterations to converge.

Scanline optimization using DP For high processing speed, the above approaches use a local optimization, which limits the accuracy of the disparity maps generated. To achieve better accuracy, some researchers adopted DP optimization into realtime and near-real-time stereo system as well. For example, Sun [22] proposes a DP-based algorithm, which achieves 5-7 MDE/s on an 500Mhz PC. Using a coarse-to-fine approach and MMX based assembler optimization, Forstmann *et al*. [9] are able to significantly increase DP's speed. The experimental results show that their approach achieves about 100 MDE/s on an AthlonXP 2800+ processor.

Traditional DP algorithms optimize the depth map on a scanline by scanline basis. The inter-scanline consistency is not enforced, leading to the well-known horizontal "streaking" artifacts. A number of approaches have been proposed to address this issue (e.g., [1, 2, 5, 18]). More recently, new algorithms have been proposed to perform DP on a tree constructed from the matching cost volume [24], or perform DP multiple times in both horizontal and vertical directions [10, 14]. Very strong results with virtually no streaking have been obtained. Using the old benchmark data set from [20], [14] is probably the best DP-based approach (this algorithm has not been evaluated using the new data set). But it requires a few seconds to compute a small image.

Gong and Yang [10] use the graphics hardware to accelerate the computation. They can achieve about 20 MDE/s. Instead of using multiple passes, our approach uses a vertical adaptive filter to enforce inter-scanline consistency and therefore only requires a single DP pass.

3. Our Approach

Given a pair of images, the goal of a stereo algorithm is to establish pixel correspondences between the two images. For the scope of this paper, we assume *rectified* images as input, i.e., the epipolar lines are aligned with corresponding scanlines. In this case, the correspondence can be expressed as a disparity value (a scalar), i.e., if $p(x, y)$ and $q(x', y')$ are corresponding pixels in the left and right image respectively, then the disparity d of $p(x, y)$ and $q(x', y')$ is defined as the difference of their horizontal image coordinates, i.e., $d = x - x'$. Note that we have $y \equiv y'$ since corresponding pixels must be on the same scan line in rectified image pairs. The output of a stereo algorithm is a disparity map, i.e., a map that records the disparity value for every pixel in one image (the reference image).

Following the taxonomy in [21], our algorithm contains three major steps: matching cost computation, cost aggregation, and finally disparity selection. In the first step, a matching cost for every possible disparity value of each pixel is computed. To reduce the ambiguity in matching, the cost is summed over a small neighborhood window (support region) in the second aggregation step. In the last disparity selection step, we use DP along each scanline to assign a disparity value for each pixel.

The key in our method is the cost aggregation. We will therefore first briefly describe the first and last step, and then focus on the cost aggregation step in section 3.3. In particular we will demonstrate its advantages over traditional box filters. Finally we show how to effectively accelerate the first and second step using commodity graphics hardware in section 3.4.

3.1. Matching cost computation

A widely used matching cost is the absolute difference between the left and right pixel intensities:

$$|p(x, y) - q(x + d, y)|, \tag{1}$$

where d is the hypothesized disparity value. Under the Lambertian surface assumption, a pair of corresponding pixels in the left and right view should have identical intensities, leading to a zero(optimal) matching cost.

For every pixel $p(x, y)$ in the reference image, we loop through all disparity hypotheses to calculate their matching costs using equation 1. In the end, we obtain a matching cost volume $\mathbf{C}$ – a three-dimensional array indexed by x, y, and d.

3.2. Disparity selection using DP

DP is one of the most widely used algorithms in stereo. Like several other real-time approaches (e.g, [6, 9, 10]), we choose DP because it can offer both horizontally optimized results and comparatively low computational complexity suitable for real-time computation.

The stereo correspondence problem can be treated as finding an optimal path of disparities that minimizes the global energy function (equation 2) for each scanline through a two-dimensional search plane. The search plane is a slice in the matching cost volume $\mathbf{C}$, i.e.,$\mathbf{C}(x, y, d)$ where y is a constant.

$$E(d) = E_{data}(d) + E_{smooth}(d) \tag{2}$$

In equation. 2, the data term is the matching cost (e.g. defined by equation 1) and the smoothness term E_{smooth} encodes the smooth assumption. In our implementation, E_{smooth} is formulated as equation 3, where λ is a constant used to penalize depth discontinuities.

$$E_{smooth} = \lambda \cdot \sum_{x} |d(x) - d(x + 1)| \tag{3}$$

DP can efficiently handle this class of problems that exhibit optimal substructure by combining the solutions to sub-problems. We use DP to extract the best path in the following manner: for each scanline y in the reference image we construct a matrix $\mathbf{M}$ with N rows and W columns, where N represents the disparity range and W is the image width. Each entry in the matrix is a potential place along the path. During initialization, a slice of the cost volume $\mathbf{C}$ is copied into $\mathbf{M}$, i.e., $\mathbf{M}(x, d) = \mathbf{C}(x, y, d)$. Next the matrix $\mathbf{M}$ is re-filled in a left-to-right fashion based on the ordering constraint. That is, the matrix is updated using:

$$\begin{aligned}\mathbf{M}(x, d) = \mathbf{M}(x, d) + \min(&\mathbf{M}(x - 1, d - 1) + \lambda, \\ &\mathbf{M}(x - 1, d), \mathbf{M}(x, d + 1) + \lambda)\end{aligned} \tag{4}$$

After the rightmost column is filled, the best path can be found by back-tracking. The optimal path corresponds to a disparity assignment for each pixel on the scanline. The DP process is repeated over all the scanlines to generate the full disparity map.

3.3. Matching cost aggregation

We now introduce the key component in our algorithm. In a typical stereo algorithm, the matching cost volume $\mathbf{C}$ is updated as

$$\mathbf{C'}(x, y, d) = \sum_{(x', y') \in \Omega} \mathbf{C}(x', y', d), \tag{5}$$

where Ω is a small neighborhood around $p(x, y)$. The implicit assumption made here is that the surface is locally

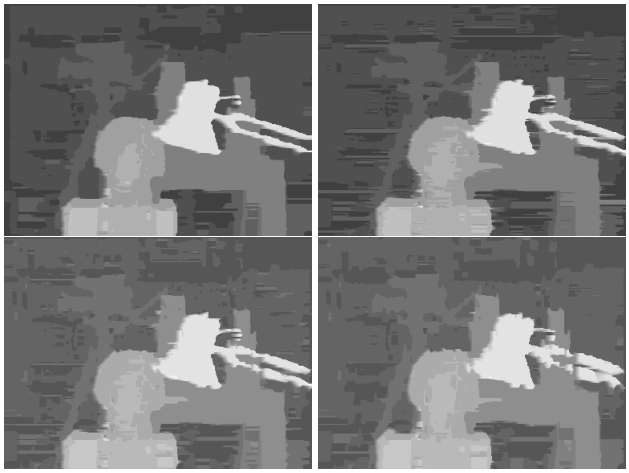

Figure 2. Comparison of adaptive weight window with fixed-weight window. The depth result was computed using dynamic program after cost aggregation. The top-left image is from a 32×1 support window with adaptive weight. The other are from a $n \times 1$ support window with constant weight, where $n = 1$(top-right), 4(bottom-left), 8(bottom-right), respectively.

smooth and fronto-parallel (facing the camera), so neighboring pixels are likely to have the same disparity value. However, this assumption is invalid on occlusion boundaries. How to effectively deal with pixels on occlusion boundaries remains an active research topic in stereo.

We adopted the adaptive weight approach from [29]. The basic idea is to aggregate the matching cost based on both color and geometric proximity. Given a pixel p (we dropped the coordinate indices for simplicity in notation), and a pixel l in its support region, the matching cost from l is weighted by the color difference (Δc_{pl})) between p and l, and the Euclidian distance (Δg_{pl}) between p and l on the image plane. The formulation for the weight $w(p, l)$ is:

$$w(p,l) = \exp\left(-(\frac{\Delta c_{pl}}{\gamma_c} + \frac{\Delta g_{pl}}{\gamma_g})\right), \qquad (6)$$

where γ_c and γ_g are weighting constants determined empirically[2].

The aggregated cost is computed as a *weighted* sum of the per-pixel cost. That is:

$$\mathbf{C'}(p,d) = \frac{\sum_{l \in \Omega_p, l' \in \Omega_q} w(p,l) w'(q,l') \mathbf{C}(p,d)}{\sum_{l \in \Omega_p, l' \in \Omega_q} w(p,l) w'(q,l')}. \qquad (7)$$

Note the slight change in notation that we index into the cost volume using pixels (i.e., p) instead of pixel coordinates (i.e., x, y). p and q are the hypothesized matching pixels, and their weight masks are denoted as $w(p, l)$ and $w'(q, l')$. l and l' are the corresponding pixels in p and q's support region, respectively. In equation 7, the weights from both images are used.

This simple approach has shown to be remarkably effective. It is similar in spirit to many segmentation-based stereo algorithms(e.g. [23]), but it avoids the difficult problem of image segmentation by using a continuous weighting function.

However, this aggregation scheme is very expensive in terms of computation. Unlike many schemes that utilize a rectangular window, which can be efficiently computed either incrementally or through separate passes, the weighting mask varies from pixel to pixel because the center pixel has different colors. As reported in [29], it took about one minute to produce a small depth map (i.e., about 0.03 Mde/second), which makes real-time application impossible.

Vertical aggregation We instead use the above scheme to aggregate over a vertical 1D window (i.e., a single column). After aggregation, pixels between scanlines are likely to have a more consistent cost if they are sharing the same color (and therefore more likely to belong to the same surface). The inter-scanline consistency can be better enforced in the final disparity selection stage.

Figure 2 illustrates the advantages of using an adaptively weighted window in the vertical direction. With a small fixed-weight window (e.g. a mean filter), the occlusion boundaries are preserved and the fine 3D structures are visible, but the noise and the "streaking" in the disparity map are also obvious. Similar results were obtained in [9] in which the cost from the previous scanline is aggregated. With a larger window, the overall result is much smoother, but the occlusion boundaries are over-smoothed and some fine structures are missing. In contrast, using a large window with adaptive weight can maintain fine structures and at the same time suppress noise and streaks in the disparity map.

Two-pass aggregation We also investigated the use of two separate 1D windows, one horizontal, and one vertical, to approximate the full rectangular weight mask in equation 6. In essence, we first aggregate the cost in the vertical direction as described above, then sum the aggregated cost again using a horizontally weighted mask.

The weighted-sum function in equation 7 is not separable in theory, but we found that our two-pass approximation worked well in practice. Figure 3 shows several masks for different pixels in the Tsukuba image. In most cases, our approximation mask is very similar to the original mask. The rightmost image has two diagonal thin lines with similar color and the weight in our approximation, while still correct, drops off much faster. In these worst-case scenarios in

[2]The formulation in the original paper includes a weighting constant k for the entire equation, which seems to be redundant after normalization. We do not use it in our experiments.

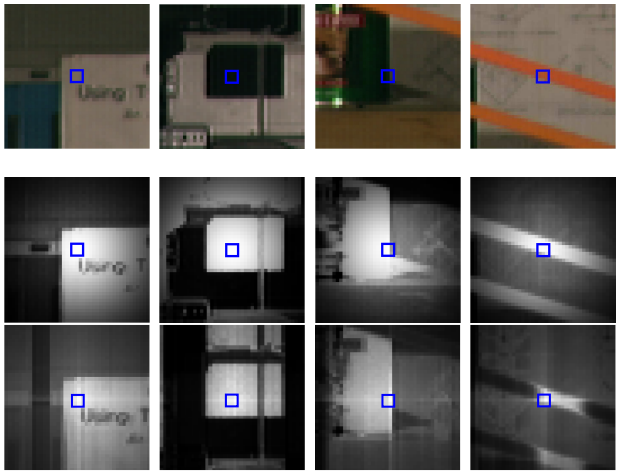

Figure 3. A comparison of our approximated weight masks with the original masks. (top row) close-up views at several pixel locations in the Tsukuba image. The blue square marks the center pixel. (second row) the corresponding masks copied from [29]. (third row) the masks using our two-pass aggregation.

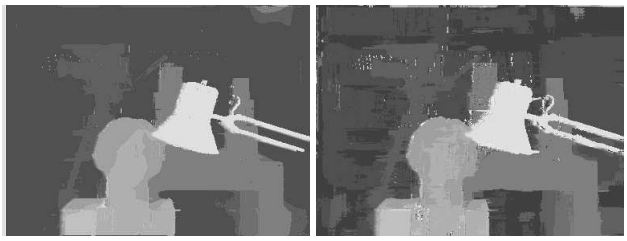

Figure 4. A comparison of our approximation with the original method [29]. (left) a sample depth map from [29]. (right) the depth map using our two-pass approximation. The parameters in equation 6 are identical to these used in the original paper. A "winner-takes-all" approach is used to select the best disparity value from the aggregated cost volume. The computation was carried out in software and finished in six seconds on a 3.0Ghz PC.

which the object boundary is diagonal, our approximation is similar to a full mask computed from a smaller support region.

Our two-pass approximation brings significant savings in computation. reducing the complexity per disparity from $O(n^2)$ to $O(n)$, where n is the size of the window. In Figure 4 we show the depth maps resulting from our approximation and compare it with the result from [29]. The result is visually comparable, but our result was produced in six seconds while the original approach took about one minute.

3.4. Acceleration using graphics hardware

We have implemented the first (cost computation) and second step (cost aggregation) on graphics hardware to facilitate real-time computation.

In the first step, we store the two input images as two textures. For each disparity hypothesis d, we draw a 2D rectangle aligned with two input textures, one of them being shifted by d pixels. We use the pixel shader, a programmable unit in the graphics hardware [16], to compute the per-pixel absolute difference, which is written to an output texture. Since the graphics hardware is most efficient at processing 4-channel (RGB + alpha) color images, we compute four disparity hypotheses at a time and store the absolute-difference images in different channels. To search over N disparity hypothesis, $\lceil N/4 \rceil$ rendering passes are needed and the matching cost volume is stored in $\lceil N/4 \rceil$ textures.

For the second aggregation step, we first need to compute the weight masks for both images. Similar to the process used in cost computation, we shift the image over itself to compute the weights according to equation 6 and store them in textures. The 1D window size is always set to a multiple of four to facilitate the 4-vector processing capability on the graphics processing unit (GPU). After weight mask computation, we can step through the cost volume to aggregate the weighted cost. A fairly complex pixel shader program is implemented to index into both the cost textures and weighting mask textures to calculate the final cost. To aggregate the cost over N disparity hypotheses with a window of $H \times 1$, $\lceil \frac{N \cdot H}{16} \rceil$ rendering passes are needed in our current implementation.

The advantage of using graphics hardware mainly comes from the parallelism inherent in today's GPU. The latest generation has up to 24 pixel shader units. Both cost computation and aggregations are very regular operations that can benefit most from GPU's parallel architecture.

The aggregated cost volume can be used for a "winner-takes-all" disparity selection on GPU (as in [26, 4]), or it can be read back to main memory for CPU processing using DP. It should be noted that it is also possible to implement the entire DP process on GPU. However, as reported in [10], a GPU-based DP program is actually slower than its CPU counterpart. This is mainly due to the significant number of rendering passes needed and the lack of true branching capability on GPU [10]. Therefore we adopted a co-operative approach, using the GPU to compute the cost volume and the CPU to carry out DP. With the new PCI-Express interface between CPU and GPU, the communication bandwidth is huge (full-duplex at 4GB/second), removing a long-existing bottleneck between GPU and CPU. Our approached not only makes use of both CPU and GPU in parallel, but also makes each part do what it is best at doing: the GPU performs image warping and aggregation in massive parallelism, and the CPU performs DP which requires more flexible looping and branching capability.

4. Results

We first evaluate the reconstruction quality of our approach using the benchmark Middlebury stereo data

set [20]. The cost computation and aggregation are computed on the graphics hardware and the DP is computed in software. The few four parameters are set to be identical across all experiments. They are: support window size (32×1), two parameters in equation 6 ($\gamma_c = 10$, $\gamma_g = 40$), and the discontinuity cost ($\lambda = 7$) in DP (if used). The results from all four test images are shown in figure 5. One may notice that the Tsukuba image in the second row is not as good as the one in figure 4 (right), which is computed in CPU. We believe it is a precision issue in the GPU implementation. While the pixel shader performs computation in 32-bit floating point numbers, we store the intermediate result as image textures (8-bit per channel). The accumulation error from the two-pass aggregation is much more significant than the one-pass vertical-only aggregation. This also explains the problem that two-pass+DP (last row) is not as good as one-pass+DP (first row). Changing the discontinuity cost could improve the results in the last row, but still not as good as those in the first row.

Using the online system at [20], we have evaluated the numerical accuracy of our best result (vertical aggregation + DP) and show part of the result in Table 1, more can be found online. Our approach is better than all DP-based approaches in the current table.

The old evaluation table at Middlebury, which contains more algorithms, is no longer functional. We have collected the results (based on real images) from all DP-based approaches and compare them to our approach in Figure 6. Our result is likely to rank second numerically, behind [14], which took several seconds to compute one depth map.

Live system We integrated our algorithm into a stereo system with live video input. **Results from this live system can be found in the accompanying video.**[3] The input images are rectified with lens distortion removed. This preprocessing is implemented on the graphics hardware using texture mapping functions.

Figure 1 shows some live images from our system. Notice the fine structures and clean object boundaries our approach is able to produce. The speed of our system is summarized in Table 2. Our GPU-accelerated version is two orders of magnitude faster than its CPU counterpart. It should be noted that our CPU implementation is not yet optimized at the assembly level, which could lead to 2-3 times speedup.

5. Conclusion and Future Work

In this paper we present a novel approach using adaptive weighting in a stereo framework based on DP. The use of a color and distance weighted cost-aggregation window in the vertical direction significantly reduces the "streaking" arifacts without using multiple DP passes. Experimental results have shown that it is among the best performing DP algorithms in terms of both quality and speed. In addition an approximation for the 2D adaptive window is developed, which leads to a fully GPU-accelerated implementation to achieve two orders of speed-ups compared to the original approach in [29].

Looking into the future, optimizing DP using MMX (as in [9]) is likely to double the frame rate. We would also like to investigate the precision issue on the graphics hardware. Current graphics hardware does provide limited support for high-precision texture maps, at the cost of significant performance degradation (the hardware is optimized to work with 8-bit textures). From an algorithmic standpoint, our DP implementation enforces the ordering constraint for speed consideration. Very fine 3D structures (such as the flower stem in Figure 1) may disappear if it is far away from any object. We plan to investigate the use of scanline optimization [21], which enforces the smoothness constraint directly without using the ordering constraint. Another interesting venue to explore is to enforce the temporal consistency in the video in order to reduce the flickering.

Acknowledgements This work is supported in part by University of Kentucky Research Foundation, US Department of Homeland Security, and US National Science Foundation grant IIS-0448185.

[3] Some of the motion artifacts were caused by the faulty cameras that sometimes failed to deliver synchronized images. This hardware failure happened three days before the submission deadline and we apologize that we cannot fix it in time.

References

[1] P. Belhumeur. A binocular stereo algorithm for reconstructing sloping, creased, and broken surfaces in the presence of half-occlusion. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 431–438, 1993. 2

[2] S. Birchfield and C. Tomasi. Depth discontinuities by pixel-to-pixel stereo. *International Journal of Computer Vision (IJCV)*, 35(3):269–293, 1999. 2

[3] R. Collins. A Space-Sweep Approach to True Multi-Image Matching. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, pages 358–363, June 1996. 2

[4] N. Cornells and L. V. Gool. Real-time connectivity constrained depth map computation using programmable graphics hardware. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1099–1104, 2005. 2, 5

[5] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs. A Maximum Likelihood Stereo Algorithm. *Computer Vision and Image Understanding (CVIU)*, 63(3):542–567, 1996. 2

[6] A. Criminisi, J. Shotton, A. Blake, C. Rother, and P. Torr. Efficient dense-stereo with occlusions and new view synthesis by four state DP for gaze correction. Technical report MSR-TR-2003-59, Microsoft Research, 2003. 3, 8

[7] A. Darabiha, J. Rose, and W. J. MacLean. Video-Rate Stereo Depth Measurement on Programmable Hardware. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 203–210, 2003. 2

[8] O. Faugeras, B. Hotz, H. Mathieu, T. Viville, Z. Zhang, P. Fua, E. Thron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy.

Figure 5. Resulting depth maps from the Middlebury stereo data set. (top row) vertical aggregation (one-pass) + DP; (middle row) two-pass aggregation + WTA; (bottom row) two-pass aggregation + DP

Algorithm	Avg. Rank	Tsukuba nonocc	Tsukuba all	Tsukuba disc	Venus nonocc	Venus all	Venus disc	Teddy nonocc	Teddy all	Teddy disc	Cones nonocc	Cones all	Cones disc
Our Algorithm	**9.2**	**2.05** 10	**4.22** 11	**10.6** 11	**1.92** 11	**2.98** 10	**20.3** 14	**7.23** 4	**14.4** 6	**17.6** 4	**6.41** 9	**13.7** 9	**16.5** 11
Reliability-DP [10]	10.3	1.36 5	3.39 8	7.25 6	2.35 12	3.48 13	12.2 12	9.82 9	16.9 8	19.5 7	12.9 15	19.9 15	19.7 13
TreeDP [24]	10.3	1.99 9	2.84 7	9.96 10	1.41 7	2.10 7	7.74 7	15.9 13	23.9 13	27.1 15	10.0 12	18.3 12	18.9 12
DP [21]	13.3	4.12 13	5.04 13	12.0 12	10.1 17	11.0 17	21.0 15	14.0 12	21.6 12	20.6 9	10.5 13	19.1 13	21.1 14

Table 1. Performance comparison of the proposed method with other high-quality DP-based approaches.

Real time sorrelation-based stereo: Algorithm, amplementations and application. Technical Report 2013, INRIA, August 1993. 2

[9] S. Forstmann, J. Ohya, Y. Kanou, A. Schmitt, and S. Thuering. Real-time stereo by using dynamic programming. In *Proc. of CVPR Workshop on Real-time 3D Sensors and Their Use*, 2004. 2, 3, 4, 6, 8

[10] M. Gong and Y.-H. Yang. Near real-time reliable stereo matching using programmable graphics hardware. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 924–931, 2005. 1, 2, 3, 5, 7, 8

[11] H. Hirschmuler. Improvements in Real-Time Correlation-Based Stereo Vision. In *Proceedings of IEEE Workshop on Stereo and Multi-Baseline Vision*, pages 141–148, Kauai, Hawaii, December 2001. 2

[12] H. Hirschmuller, P. Innocent, and J. Garibaldi. Real-Time Correlation-Based Stereo Vision with Reduced Border Errors. *International Journal of Computer Vision*, 47(1-3), April-June 2002. 2

[13] T. Kanade, A. Yoshida, K. Oda, H. Kano, and M. Tanaka. A Stereo Engine for Video-rate Dense Depth Mapping and Its New Applications. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, pages 196–202, June 1996. 2

[14] 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. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1075–1082, 2005. 1, 2, 6, 8

[15] K. Konolige. Small Vision Systems: Hardware and Implementation. In *Proceedings of the 8th International Symposium in Robotic Research*, pages 203–212. Springer-Verlag, 1997. 2

[16] Microsoft. DirectX, 2003. http://www.microsoft.com/windows/directx. 5

[17] J. Mulligan, V. Isler, and K. Daniilidis. Trinocular Stereo: A New Algorithm and its Evaluation. *International Journal of Computer Vision (IJCV), Special Issue on Stereo and Multi-baseline Vision*, 47:51–61, 2002. 2

[18] Y. Ohta and T. Kanade. Stereo by intra- and inter- scanline search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 7(2):139–154, 1985. 2

[19] Point Grey Research Inc. http://www.ptgrey.com. 2

[20] D. Scharstein and R. Szeliski. www.middlebury.edu/stereo. 2, 6, 8

[21] D. Scharstein and R. Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, 47(1):7–42, May 2002. 2, 3, 6, 7, 8

[22] C. Sun. Fast stereo matching using rectangular subregioning and 3D maximum-surface techniques. *International Journal of Computer Vision (IJCV),*, 47(1-3):99–177, 2002. 2

[23] H. Tao and H. Sawhney. Global matching criterion and color segmentation based stereo. In *Proceedings of IEEE Workshop on Applications of Computer Vision (WACV)*, pages 246–253, 2000. 4

[24] O. Veksler. Stereo correspondence by dynamic programming on a tree. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 384–390, 2005. 2, 7, 8

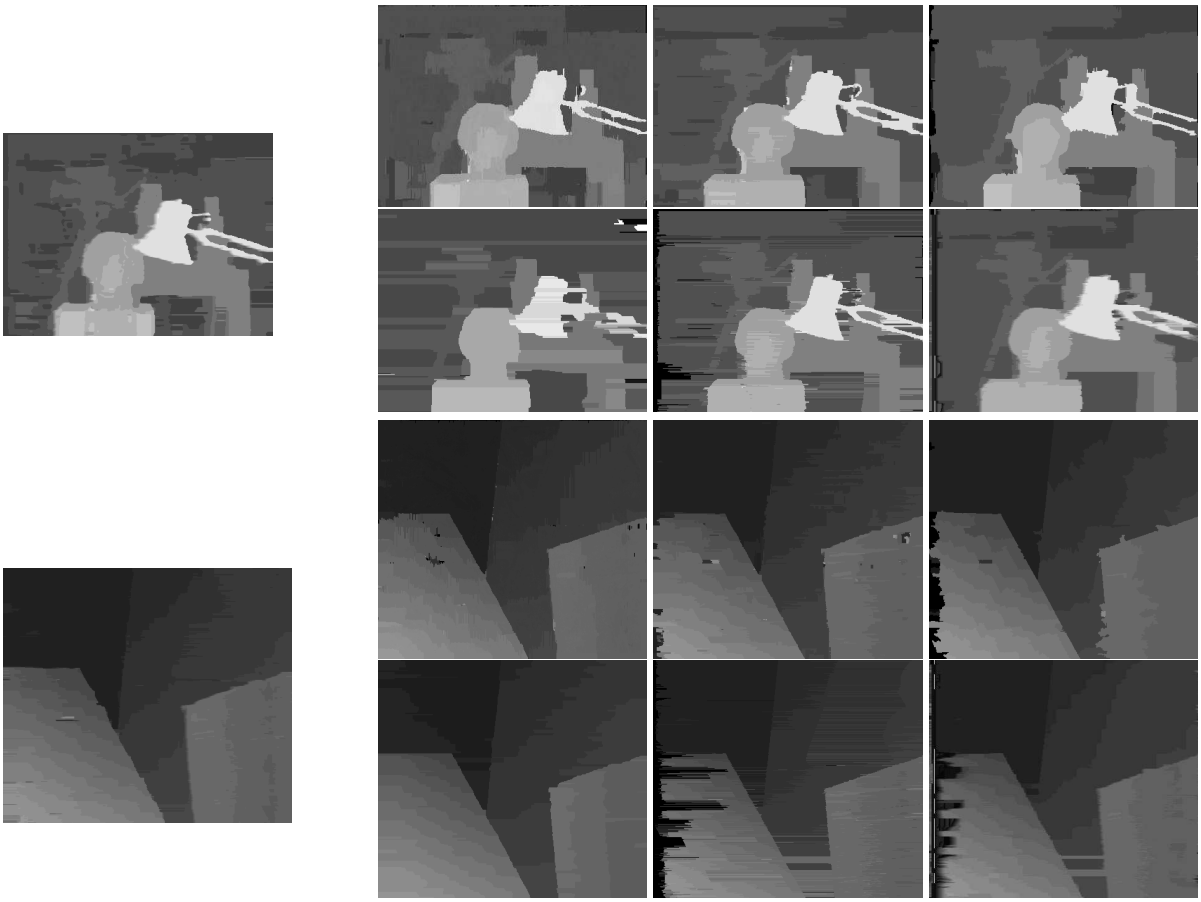

Figure 6. A visual comparison of our results with those from all other DP algorithms in the old evaluation table in [20]. The left two images are ours, and images from other DP algorithms are arranged based on their ranking (from high to low). They are 2-pass DP [14], Reliability-DP [10], Tree DP [24], 4-State DP [6], DP [21], and Realtime DP [9].

Image Size	Disparity Range	Time (second)					MDE/second			
		GPU Accelerated			CPU Only		WTA		VA+DP	
		WTA	VA+DP		WTA	VA+DP	GPU	CPU	GPU	CPU
			CPU	GPU						
640×480	16	0.127	0.101	0.079	23.2	15.2	38.6	0.21	48.2	0.32
	32	0.221	0.195	0.131	43.8	29.1	44.3	0.22	50.3	0.34
	48	0.313	0.301	0.183	64.2	42.4	47.0	0.23	48.8	0.35
320×240	16	0.039	0.023	0.023	5.50	3.61	31.2	0.22	52.7	0.34
	32	0.064	0.046	0.042	10.2	6.78	38.3	0.24	53.0	0.36
	48	0.087	0.073	0.054	14.5	9.63	42.1	0.25	50.2	0.38

Table 2. Real-time Performance. The test system is a 3.0Ghz PC with a Radeon XL1800 graphics card from ATI. VA+DP denotes vertical aggregation with DP, WTA denotes "winner-takes-all" with 2-pass aggregation (all on the GPU). Note that in the GPU-accelerated version of VA+DP, the GPU and CPU are working in parallel. We used the longer time to compute the MDE number.

[25] J. Woodfill and B. V. Herzen. Real-Time Stereo Vision on the PARTS Reconfigurable Computer. In K. L. Pocek and J. Arnold, editors, *IEEE Symposium on FPGAs for Custom Computing Machines*, pages 201–210, Los Alamitos, CA, 1997. IEEE Computer Society Press. 2

[26] R. Yang and M. Pollefeys. Multi-Resolution Real-Time Stereo on Commodity Graphics Hardware. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 211–218, 2003. 2, 5

[27] R. Yang, M. Pollefeys, and S. Li. Improved Real-Time Stereo on Commodity Graphics Hardware. In *IEEE Workshop on Real-time 3D Sensors and Their Use (in conjunction with CVPR'04)*, 2004. 2

[28] R. Yang, G. Welch, and G. Bishop. Real-Time Consensus-Based Scene Reconstruction Using Commodity Graphics Hardware. In *Proceedings of Pacific Graphics 2002*, pages 225–234, Beijing, China, October 2002. 2

[29] K.-J. Yoon and I.-S. Kweon. Locally Adaptive Support-Weight Approach for Visual Correspondence Search. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 924–931, 2005. 1, 2, 4, 5, 6

[30] C. Zach, A. Klaus, and K. Karner. Accurate Dense Stereo Reconstruction using Graphics Hardware. In *EUROGRAPHICS 2003*, pages 227–234, 2003. 2