FAST VARIABLE CENTER-BIASED WINDOWING FOR HIGH-SPEED STEREO ON PROGRAMMABLE GRAPHICS HARDWARE

Jiangbo Lu [*,†], *Gauthier Lafruit* [†], *and Francky Catthoor* [*,†]

[*] Department of Electrical Engineering, University of Leuven, Belgium

[†] Multimedia Group, IMEC, Kapeldreef 75, B-3001, Leuven, Belgium

ABSTRACT

We present a high-speed dense stereo algorithm that achieves both good quality results and very high disparity estimation throughput on the graphics processing unit (GPU). The key idea is a variable center-biased windowing approach, enabling an adaptive selection of the most suitable support patterns with varying sizes and shapes. As the fundamental construct for variable windows, a truncated separable Laplacian kernel approximation is proposed for the efficient pixel-wise weighted cost aggregation. We also present a number of critical optimization schemes to boost the real-time speed on GPUs. Our method outperforms previous GPU-based local stereo methods and even some methods using global optimization on the Middlebury stereo database. Our optimized implementation completely running on an Nvidia GeForce 7900 graphics card achieves over 605 million disparity estimations per second (Mde/s) including all the overhead, about 2.1 to 12.1 times faster than the existing GPU-based solutions.

Index Terms— Stereo vision, real-time dense stereo, GPGPU

1. INTRODUCTION

Depth from stereo is an important computer vision topic that has attracted intensive research interests for decades. A substantial amount of work has been done on stereo, which is systematically surveyed and evaluated by Scharstein and Szeliski [1]. In general, casting a stereo problem as a global optimization problem usually leads to high quality disparity estimation results, but most of these global techniques are too computationally expensive for online processing. Real-time stereo applications today still largely rely on some local methods together with a winner-takes-all (WTA) decision strategy.

Typically, local window-based approaches choose to aggregate the matching cost over a given support window to increase the robustness to noise and texture variation. However, to obtain accurate results at depth discontinuities as well as on homogeneous regions, an appropriate support window for each pixel should be decided adaptively. To this end, several local methods have been proposed. For instance, Fusiello et al. [2] performed the correlation with nine windows anchored at different points and retained the disparity with the smallest matching cost. However, this method and its generalized technique, i.e., shiftable windows [1] usually require a relatively large number of candidate support windows to achieve good estimation results, and moreover their box-filters cannot adequately differentiate the impact of support pixels with different spatial locations. Recently, Yoon and Kweon [3] proposed a state-of-the-art local window method yet at a very demanding computational cost, where pixel-wise support-weights are defined using a Laplacian kernel, and they modeled the grouping strength for each support pixel.

Nonetheless, solely resorting to local methods is not a cure-all for achieving dense stereo at high video rate. In fact, until recently software-only real-time stereo systems begin to emerge, which exploit assembly level instruction optimization using Intel's MMX extension, but few CPU cycles are left to perform other tasks including high-level interpretation of the stereo results. Harnessing some powerful built-in features of the modern graphics processing unit (GPU), Yang et al. first proposed a pyramid-shaped correlation kernel [4] and small-scale adaptive support windows [5]. Though very impressive disparity estimation throughput is obtained on GPUs, these techniques cannot strike an optimal quality balance between homogeneous and heterogeneous regions. Later on, Gong and Yang [6] proposed an image-gradient-guided correlation method with improved accuracy, while still maintaining real-time speed on GPUs. Inspired by [3], Wang et al. [7] recently introduced an adaptive aggregation step in a dynamic-programming stereo framework. The high-quality results are obtained by their complicated cost aggregation and global optimization strategy, and a real-time speed is enabled by utilizing the unique processing capabilities of both the CPU and the GPU.

This paper presents a novel stereo algorithm that is specially designed to achieve the competitive disparity quality and the high-speed execution on GPUs. At the heart of the proposed algorithm is a variable center-biased windowing approach, enabling an adaptive selection of the most suitable support patterns for different regions. Our method is in spirit similar to the variable window approach [8], but it is much faster by avoiding the costly dynamic programming.

Concerning the real-time speed, the proposed method is by far the fastest among all these GPU-based approaches. The major contributing factors are three-folds: 1) our highly efficient core stereo processing, 2) a number of special implementation optimizations on the GPU, and 3) upgrading to the advanced graphics hardware. Completely running on an Nvidia GeForce 7900 graphics card, our optimized implementation achieves over 605 million disparity estimations per second (Mde/s), compared to a maximum speed of 289 Mde/s in [5], 117 Mde/s in [6], and 50 Mde/s on CPU+GPU in [7].

2. THE PROPOSED STEREO MATCHING ALGORITHM

Following the taxonomy in [1], our stereo 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 suppress the influence of mismatches during the subsequent cost aggregation step, we adopt the truncated absolute difference (TAD) as the matching cost measure. Similar to most local approaches, the proposed algorithm places a key emphasis on the cost aggregation step to reduce the ambiguity in matching, and we will therefore focus on this core part for the remaining of this Section. In the last disparity selection step, a local WTA optimization is performed at each pixel, simply choosing the disparity associated with the minimum cost value. The entire framework of our stereo algorithm is illustrated in Fig. 1.

The proposed cost aggregation step is composed of two parts: 1)

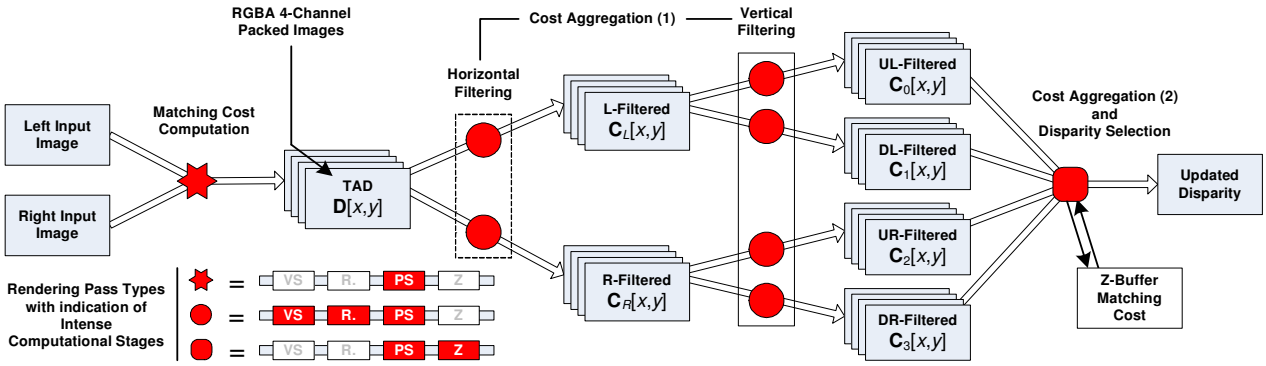


Fig. 1. The framework of our proposed stereo algorithm and its optimized implementation (processing elements and data flow) on the GPU.

efficient separable quadrant-based cost aggregation, and 2) adaptive selection of variable center-biased windows.

2.1. Efficient Separable Quadrant-Based Cost Aggregation

The goal of this stage is to construct an elementary set of window costs, so that a useful range of variable center-biased support windows can be easily built upon this basic set in the subsequent stage. Instead of using box-filters commonly adopted for the window cost aggregation [2], we propose a truncated separable approximation to an isotropic 2D Laplacian kernel for this task. The latter proves its effectiveness in a state-of-the-art local stereo method [3] by appropriately deciding the strength of grouping for each pixel. To achieve a good trade-off between the quality and the speed, we only attempt at computing the adaptive support-weight for each pixel based on the geometric proximity in [3], while leveraging the variable center-biased windows to ensure competitive quality results in both homogeneous regions and depth discontinuities.

Strictly speaking, the classical Laplacian kernel is non-separable because of its isotropic Euclidean distance term. This actually renders its exact implementation costly. To ease the computational cost, we propose a separable approximated variant to the 2D isotropic Laplacian kernel (see Fig. 2(a)), as follows,

$$f_p\left(\Delta\widetilde{g}_{pq}\right) = exp\left(-\frac{\Delta\widetilde{g}_{pq}}{\gamma_p}\right) = exp\left(-\frac{\left(|\Delta x_{pq}| + |\Delta y_{pq}|\right)}{\gamma_p}\right) = exp\left(-\frac{|\Delta x_{pq}|}{\gamma_p}\right) \cdot exp\left(-\frac{|\Delta y_{pq}|}{\gamma_p}\right) , \tag{1}$$

where $\Delta\widetilde{g}_{pq}$ is an approximated Euclidean distance between p and q in the 2D image domain and γ_p determines the fall-off rate of the kernel. From (1), it is apparent that a separable approximation to the 2D Laplacian kernel (SA-LAP) is obtained. As a result, SA-LAP can be very efficiently implemented by two cascaded 1D filterings along the x and y axes. It reduces the computational load significantly from $O(M\times N)$ to $O(M+N)$, when an $M\times N$ kernel is considered.

To tackle the "foreground-fattening" problem near depth discontinuities due to a large centralized window, we propose to partition it into four slightly overlapping regions (or quadrants), each containing the central pixel as shown in Fig. 2(b). With these four elementary support windows, a set of quadrant-based matching costs (i.e., $\mathbf{C}_i[x,y], i = 0,1,2,3$) can be computed for the central pixel to avoid filtering across strong edges. These four costs can be more rigorously derived by 2D convolutions as given in (2), where $\mathbf{D}[x,y]$ denotes the intensity-truncated absolute difference image, and $\mathbf{f}[x,y]$ is the finite-length impulse response of the proposed SA-LAP defined over an $N\times N$ square window. By truncating $\mathbf{f}[x,y]$ properly to each $(W+1)\times(W+1)$ $(W=(N-1)/2)$ region spanned by the shifted support window, four different truncated SA-LAP filter kernels are reached (i.e., $\mathbf{f}_i[x,y], i = 0,1,2,3$ in (2)), and a sample support-weight map of $\mathbf{f}_0[x,y]$ is shown in Fig. 2(c).

$$\begin{cases} \mathbf{C}_0[x,y] = \mathbf{D}[x,y] * *\mathbf{f}_0[x,y] = \sum_{k=0}^{W}\sum_{l=-W}^{0} \mathbf{D}[x-k,y-l]\mathbf{f}[k,l] \\ \mathbf{C}_1[x,y] = \mathbf{D}[x,y] * *\mathbf{f}_1[x,y] = \sum_{k=0}^{W}\sum_{l=0}^{W} \mathbf{D}[x-k,y-l]\mathbf{f}[k,l] \\ \mathbf{C}_2[x,y] = \mathbf{D}[x,y] * *\mathbf{f}_2[x,y] = \sum_{k=-W}^{0}\sum_{l=-W}^{0} \mathbf{D}[x-k,y-l]\mathbf{f}[k,l] \\ \mathbf{C}_3[x,y] = \mathbf{D}[x,y] * *\mathbf{f}_3[x,y] = \sum_{k=-W}^{0}\sum_{l=0}^{W} \mathbf{D}[x-k,y-l]\mathbf{f}[k,l] \end{cases} , \tag{2}$$

In fact, we also find that the truncated SA-LAP idea is conceptually close to the non-linear Kuwahara filter [9], which is an edge-preserving noise-reduction filter, and the configuration of four regions in both methods are even similar. However, our method does not need the variance computation, and moreover, we stress the contribution of pixels closer to the central one by a center-biased filter.

2.2. Adaptive Selection of Variable Center-Biased Windows

After the elementary set of quadrant-based costs are computed using (2), we propose to construct three categories of center-biased support windows with variable sizes and shapes (unlike the fixed-sized windows in [2]). These three categories are devised to approximate the ideal support configuration for homogeneous areas (Fig. 2(d)), depth edges (Fig. 2(e-h)), and depth corners (Fig. 2(i-l)), respectively. Subsequently, best patterns of minimum cost values are selected from each configuration category, and their respective size-penalized average window costs (i.e., $\bar{\mathbf{C}}_a$, $\bar{\mathbf{C}}_b$, and $\bar{\mathbf{C}}_c$) are defined in (3). For conciseness, we omit $[x,y]$ from notations whenever appropriate.

$$\begin{cases} \bar{\mathbf{C}}_a[x,y] = \left(\mathbf{C}_0 + \mathbf{C}_1 + \mathbf{C}_2 + \mathbf{C}_3\right) / \left(4 \cdot F\right) \\ \bar{\mathbf{C}}_b[x,y] = \dfrac{min\left(\mathbf{C}_0+\mathbf{C}_1, \mathbf{C}_0+\mathbf{C}_2, \mathbf{C}_2+\mathbf{C}_3, \mathbf{C}_1+\mathbf{C}_3\right)}{2 \cdot F} + \alpha \\ \bar{\mathbf{C}}_c[x,y] = min\left(\mathbf{C}_0, \mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3\right) / F + \beta \end{cases} , \tag{3}$$

where $F \equiv \sum_{k=0}^{W}\sum_{l=0}^{W} \mathbf{f}[k,l]$, and $0 < \alpha < \beta$.

In (3), we normalize the window costs by the aggregated support-weights, since we will be comparing windows of different sizes. This normalization factor also helps to scale up the data precision range when implemented on GPUs. An additive penalty term α (β) for $\bar{\mathbf{C}}_b$ ($\bar{\mathbf{C}}_c$) is also included in (3) to explicitly implement bias to larger

windows. As discussed in [8], this term is crucial in untextured regions, where the normalized window costs are approximately equal for variable support patterns, and larger windows should be preferred for a reliable performance. The empirical penalty parameters α and β are set constant for all our experiments.

Once the three best representative patterns are decided from (3), we choose the minimum value of them as the cost for the central pixel, i.e., $\bar{\mathbf{C}}_{min}[x,y]$ in (4), enabling the adaptive selection of the most suitable support window for the pixel at $[x,y]$.

$$\bar{\mathbf{C}}_{min}[x,y] = min\left(\bar{\mathbf{C}}_a, \bar{\mathbf{C}}_b, \bar{\mathbf{C}}_c\right) \ . \tag{4}$$

2.3. Accelerating Cost Aggregation by 1D Filtered Data Reuse

As the fundamental construct of our stereo algorithm, the truncated SA-LAP is also very fast to compute, because $\mathbf{f}_0[x,y]$ and $\mathbf{f}_1[x,y]$ in (2) are not only separable, but also they share the same 1D horizontal convolution kernel in the x-direction, i.e.,

$$\begin{cases} \mathbf{C}_0[x,y] = \mathbf{D}[x,y] * *\mathbf{f}_0[x,y] = \mathbf{D}[x,y] * \mathbf{f}_0[x] * \mathbf{f}_0[y] \\ \mathbf{C}_1[x,y] = \mathbf{D}[x,y] * *\mathbf{f}_1[x,y] = \mathbf{D}[x,y] * \mathbf{f}_1[x] * \mathbf{f}_1[y] \ , \end{cases} \tag{5}$$

let $\mathbf{f}_L[x] = \mathbf{f}_0[x] \equiv \mathbf{f}_1[x]$ and $\mathbf{C}_L[x,y] = \mathbf{D}[x,y] * \mathbf{f}_L[x]$, then

$$\begin{cases} \mathbf{C}_0[x,y] = \mathbf{C}_L[x,y] * \mathbf{f}_0[y] \\ \mathbf{C}_1[x,y] = \mathbf{C}_L[x,y] * \mathbf{f}_1[y] \ , \end{cases} \tag{6}$$

so only three 1D convolutions are needed to obtain $\mathbf{C}_0[x,y]$ and $\mathbf{C}_1[x,y]$. As shown in Fig. 1, the same is valid for $\mathbf{C}_2[x,y]$ and $\mathbf{C}_3[x,y]$. Hence, the major complexity of our stereo algorithm only consists of 6-times 1D image filtering, and this regular pixel-wise processing can be greatly accelerated by harnessing the powerful parallel processing capability inherent in today's GPUs.

3. AN OPTIMIZED IMPLEMENTATION ON THE GPU

To greatly boost the real-time computation speed, we have carefully implemented the proposed stereo algorithm on the GPU, by utilizing its powerful vector processing capability and speed-optimized internal data representation. The entire algorithm is implemented with Microsoft Direct3D API and the high-level shader language (HLSL). HLSL is used to program parallel vertex processors and fragment processors, with the latter being the cornerstones of the high-speed stereo on GPUs nowadays. Due to the limited space, interested readers are referred to [5] for the review of the GPU rendering pipeline.

To maximize data-level parallelism, we use the GPU's efficient 4-channel processing to compute four disparity hypotheses at a time and store the TAD costs into different color channels (refer to Fig. 1), as in [5]. Consequently, to search over L disparity hypotheses, only $\lceil L/4 \rceil$ rendering passes are needed. Moreover, exploiting the built-in bilinear texture lookup capability in the GPU, we approximate the 1D truncated SA-LAP filtering (with a kernel size of $W+1$) by $W/2$ bilinear texture lookups and one nearest texture sampling, plus the weighted summations and normalization within one rendering pass. Considering the fact that the rasterizer contains eight dedicated texture coordinate interpolators, we shift the computation of support pixel coordinates from pixel shaders (PS) to vertex shaders (VS) and the rasterizer (R.) for hardware-accelerated coordinate interpolation.

For the high-speed streaming data storage and access, we store the intermediate results in a 8-bit per-channel texture format, though fragment processors perform internal computation in 32-bit floating point numbers. Since the TAD is adopted as our matching cost measure, the associated threshold τ can be used to adequately scale the dynamic range of the aggregated cost. Hence, the quality degradation due to the limited data precision noticed in [5] is significantly

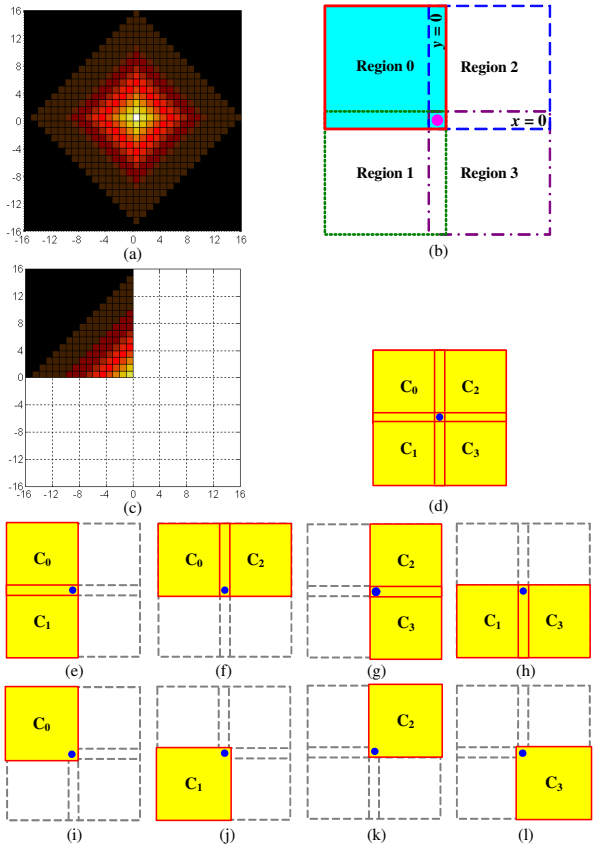


Fig. 2. Illustration of the proposed variable center-biased windowing approach. (a) The 2D support-weight map of our SA-LAP with $\gamma_p = 5.0$, $N = 33$ (b) The proposed edge-preserving quadrant-based support windows (Region 0-3) (c) A sample support-weight map of Region 0 (d) The support configuration category a (e-h) The support configuration category b (i-l) The support configuration category c.

reduced. As another important scheme for the acceleration, the advanced *Render-to-Texture* technique in Direct3D is adopted, eliminating the overhead of copying intermediate results from the frame buffer to the textures. The built-in depth test (or Z-test) is enabled in our implementation as well to accelerate the cost-sorting based disparity selection, similar to [5]. In Fig. 1, all the deployed rendering passes in our optimized implementation are highlighted.

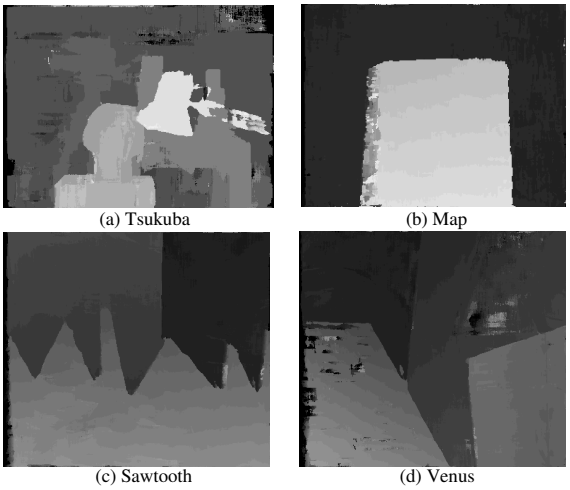
4. EXPERIMENTAL RESULTS

We first evaluate the disparity estimation quality of the proposed approach using the benchmark Middlebury stereo database [10]. The parameters in our stereo algorithm are set constant across all experiments. Specifically, we set γ_p of the proposed SA-LAP to 5.0 and N to 33, so each truncated support window has an identical size of 17×17. The threshold τ used for TAD is set to 20 for the test stereo data, but it can be adapted to cope with different image noise intensity and non-diffuse surfaces in other stereo images. The penalty parameters α and β are set to 0.02τ and 0.04τ, respectively.

Using the online evaluation service [10], we list the quantitative results of our approach (9-window) and those of some other methods roughly in descending order of performance in Table 1 (as obtained from [10]), where the numbers represent error rates (the smaller the better). Table 1 shows that our local area-based approach even outperforms some stereo methods using global optimization (e.g., Scan-

Table 1. Quantitative comparison of the proposed method with other approaches using the benchmark Middlebury stereo database.

Algorithm	Tsukuba			Sawtooth			Venus			Map	
	nonocc.	untex.	disc.	nonocc.	untex.	disc.	nonocc.	untex.	disc.	nonocc.	disc.
Relax+occl.	6.33	6.63	22.93	1.51	0.29	15.06	1.44	1.24	19.11	0.43	5.99
Ours (9-window)	**3.95**	**4.77**	**14.42**	**1.73**	**0.82**	**7.83**	**5.04**	**10.23**	**10.12**	**0.65**	**8.51**
Stochastic diffusion	3.95	4.08	15.49	2.45	0.90	10.58	2.45	2.41	21.84	1.31	7.79
Genetic	2.96	2.66	14.97	2.21	2.76	13.96	2.49	2.89	23.04	1.04	10.91
SSD+MF [1]	5.23	3.80	24.66	2.21	0.72	13.97	3.74	6.82	12.94	0.66	9.35
Ours (4-window)	**7.63**	**13.00**	**14.30**	**2.22**	**1.79**	**8.30**	**9.04**	**19.42**	**9.20**	**0.66**	**8.60**
Gradient-guided [6]	**4.91**	**5.86**	**12.60**	**2.38**	**2.82**	**7.92**	**9.43**	**19.39**	**20.71**	**1.24**	**9.96**
Scanline Opt. [1]	5.08	6.78	11.94	4.06	2.64	11.90	9.44	14.59	18.20	1.84	10.22
Dynamic Prog. [1]	4.12	4.63	12.34	4.84	3.71	13.26	10.10	15.01	17.12	3.33	14.04
MIP [5]	**7.07**			**10.4**			**13.3**			**2.33**	
AW4 [5]	**9.68**			**5.79**			**15.7**			**0.91**	



(a) Tsukuba (b) Map (c) Sawtooth (d) Venus

Fig. 3. Our estimated disparity maps for Middlebury test data set.

line optimization [1]) and the iterative stochastic diffusion algorithm. We have also collected the results of local stereo methods implemented on GPUs as the bold items in Table 1 (blank fields are not available in these papers), and our approach compares favorably with them for all sorts of regions (nonocc., untex., and disc.). Particularly, it has a good performance near depth discontinuities (see Fig. 3), because of our adaptive selection of variable center-biased windows.

As a comparison, we report the results of a simplified variant (4-window) to our proposed 9-window method, relying on the four corner patterns in Fig. 2(i-l). This 4-window variant preserves the accuracy at depth discontinuities, but the error rates for the untextured regions are doubled, lacking of a versatile set of variable windows.

To examine the execution speed on the GPU, we follow the same approach in [5] by varying the size of stereo images and the disparity search range. Our optimized implementation runs on an Nvidia GeForce 7900 graphics card with 512 MB video memory, housed in a 3.2 GHz PC with 1 GB main memory. The test results in Table 2 show that our approach can reach 605 Mde/s including the overhead to download images and read-back the disparity map, which is several times faster than today available stereo methods on GPUs.

5. CONCLUSION AND FUTURE WORK

We propose a real-time stereo algorithm with variable center-biased windows, which is built upon a truncated separable Laplacian kernel.

Table 2. Speed test on an Nvidia GeForce 7900 graphics card.

Size	Disp. Range	Specific Execution Time (ms)			Overall Performance		
		Algorithm	Download	Read-back	Total Time	fps	Mde/s
512^2	16	6.62	0.72 x 2	0.81	8.88	113	472
	32	13.23	0.72 x 2	0.81	15.49	65	542
	64	26.27	0.72 x 2	0.81	28.53	35	588
	96	39.30	0.72 x 2	0.81	41.56	24	**605**
256^2	16	2.02	0.17 x 2	0.25	2.61	**383**	402
	32	3.92	0.17 x 2	0.25	4.52	221	464
	64	7.60	0.17 x 2	0.25	8.20	122	512
	96	10.89	0.17 x 2	0.25	11.48	87	548

Our method achieves quality results for both homogeneous regions and depth discontinuities, while its optimized implementation on the GPU is significantly faster than the existing GPU-based approaches.

Future work will focus on further improving the quality and the speed trade-off of our method on programmable graphics hardware.

6. REFERENCES

[1] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. of Computer Vision (IJCV)*, vol. 47(1), pp. 7–42, May 2002.

[2] A. Fusiello, V. Roberto, and E. Trucco, "Efficient stereo with multiple windowing," in *Proc. of CVPR*, 1997, pp. 858–863.

[3] K. J. Yoo and So Kweon, "Adaptive support-weight approach for correspondence search," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 28, no. 4, pp. 650–656, April 2006.

[4] R. Yang and M. Pollefeys, "Multi-resolution real-time stereo on commodity graphics hardware," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2003, pp. 211–218.

[5] R. Yang, M. Pollefeys, and S. Li, "Improved real-time stereo on commodity graphics hardware," in *Proc. of CVPRW*, 2004.

[6] M. Gong and R. Yang, "Image-gradient-guided real-time stereo on graphics hardware," in *Proc. Int. Conf. 3D Digital Imaging and Modeling (3DIM)*, 2005, pp. 548–555.

[7] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister, "High-quality real-time stereo using adaptive cost aggregation and dynamic programming," in *Proc. of 3DPVT*, 2006.

[8] O. Veksler, "Fast variable window for stereo correspondence using integral images," in *Proc. of CVPR*, 2003, pp. 556–561.

[9] P. Bakker, L. J. van Vliet, and P. W. Verbeek, "Edge preserving orientation adaptive filtering," in *CVPR*, 1999, pp. 535–540.

[10] D. Scharstein and R. Szeliski, www.middlebury.edu/stereo.