A REAL-TIME STEREO MATCHING ALGORITHM USING GRAPHIC HARDWARE AND HIERARCHICAL METHOD

Sang Hwa Lee and Siddharth Sharma

Dept. of Electrical Eng., BK21 Information and Technology, Seoul National Univ., S. Korea
email: lsh529@snu.ac.kr, sidxavier1@gmail.com

ABSTRACT

This paper proposes a real-time stereo matching algorithm implemented in the graphic hardware. The likelihood model is parallelized and implemented using GPU programming for real-time operation. And the prior energy model is proposed to improve the accuracy of disparity estimation. First, the likelihood matching based on rank transform is implemented in GPU programming. The shared memory handling in graphic hardware is introduced in calculating the matching errors. Once an initial disparity map is determined based on the likelihood model, then the disparity map is iteratively updated by the prior model of disparity field. The prior model reflects the smoothness of disparity map and is implemented by a pixel-wise energy function. The disparity is determined by minimizing the joint energy function which combines the likelihood model with the prior model. These processes are performed in the hierarchical successive approximation approach. The disparity map is interpolated using color-based similarity. This paper evaluates the proposed approach with the Middlebury stereo images. According to the experiments, the proposed method shows good estimation accuracy with more than 30 frame/second for 640x480 images and 60 disparity range. The proposed method is expected real-time stereo camera systems to be popular in the usual PC environments.

1. INTRODUCTION

Stereo matching is one of the most active research topics to estimate disparity information from slightly different views. As the interest and need of 3-D systems are recently increased, the stereo camera and stereo matching become more important. Excellent overview of the various issues involved in stereo matching is presented by Scharstein and Szeliski [1], and Redert [2]. The algorithms for the stereo matching can be broadly classified into two categories, local and global algorithms. The local algorithms estimate disparity at a pixel using only image observations available in a finite window. Many effective local algorithms for stereo matching have been reported in the literature [3, 4, 5, 6, 7]. Unlike the local algorithms, the global methods consider the correlation of disparities in the neighborhood. The correlation is usually modeled as smoothness prior function. And the joint energy function is globally minimized to find the disparity. These approaches are related to Markov random field (MRF) models and energy minimization methods. The MRF model based algorithms rely on energy minimization methods such as graph cuts [8], belief propagation [9], nonlinear diffusion [10, 11], and dynamic programming [12]. In those methods, the 3-D energy fields are generated using likelihood and prior models for every disparity at each pixel, and the disparity fields are estimated in the iterative process of energy minimization.

The algorithms mentioned before concentrate on improving the accuracy in disparity estimation. The MRF modeling and energy minimization processes need too much computational load to be implemented in real-time stereo systems. As the interest in 3-D visual systems is increased, the real-time stereo matching is also required in many applications. This paper deals with real-time stereo matching using graphic hardware and GPU programming. Some researchers used the graphic hardware and GPU programming to improve the speed [13, 14, 15]. They usually implemented the likelihood matching function as SSD or SAD in GPU programming, and optimized the disparity field using dynamic programming. These approaches showed very fast (near real-time) operation [18, 19, 20]. Even though their works showed good estimation accuracy with near real-time speed, it is not suitable for real-time applications. This paper implements fast likelihood matching in GPU programming, and proposes a global optimization scheme to improve the accuracy within a small number of iterations. Memory allocation problem and correlation of disparity field are proposed for fast and accurate disparity estimation.

The rest of paper is organized as follows. Section 2 describes the likelihood matching based on the rank transform and GPU programming. Shared memory handling technique is also proposed. The proposed prior modeling is explained in Section 3. The color-based interpolation and successive estimation in the hierarchical scheme are described in Section 4. We show the experimental results in Section 5, and finally conclude this paper in Section 6.

2. LIKELIHOOD MATCHING

2.1 Rank Transform

Likelihood matching is the basic process to find correspondences. As we described in Section 1, SSD and SAD show fast but inaccurate performances. Adaptive windows and support weights show accurate but slow performances. This paper exploits rank transform as the likelihood matching function. The rank transform has shown good matching results compared with the various matching schemes [21, 22]. We implements the rank transform in GPU programming, and propose some implementation techniques for real-time operation. Sliding window and fast memory access are implemented for the rank transform. The rank transform at a pixel (i, j) is expressed as

$$r(i,j) = M - \sum_{(x,y)\in W} U\left(I(i+x, j+y) - I(i,j)\right), \qquad (1)$$

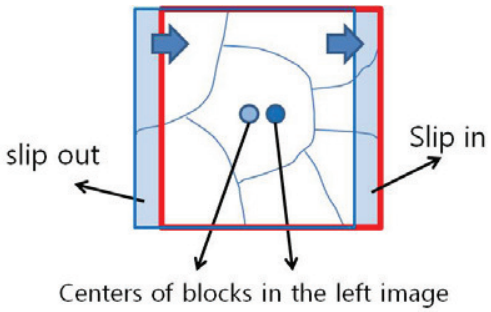


Figure 1: Sliding window method.

where $U(\cdot)$ is the unit step function, M is the number of pixels in the window W. The matching is performed by the sum of absolute differences (SAD) for rank transformed images,

$$SAD(d_{ij}) = \sum_{(x',y')\in B_{ij}} \left| r_r(x',y') - r_l(x'+d_{ij},y') \right|, \qquad (2)$$

where B_{ij} is a matching window centered at (i,j), and $r_r(i,j)$ and $r_l(i,j)$ are the rank transformed values of right and left images respectively. In (2), d_{ij} is the disparity value on the horizontal epipolar line.

We implement sliding window method in calculating (2). When we consider the SAD values for the adjacent blocks in the left image, the block locations in the left image are shifted by one pixel in the horizontal direction as shown in Fig. 1. Thus, the leftmost column of pixels in the block centered at (i,j) slips out of the block, and the rightmost column of pixels slips into the block centered at $(i+1,j)$ to calculate (2). When the matching block in the right image is the same for two adjacent blocks in the left image, the whole calculation of (2) is to subtract the SAD value of leftmost pixels and to add the SAD value of rightmost pixels to the previous total SAD value. We exploit this observation called sliding window method. In this case, the order of calculating $SAD(d_{ij})$ is changed. We usually calculate the SAD values of all disparities for a block in the left image. On the other hand, we calculate first $SAD(d_{ij})$ and then calculate $SAD(d_{i+1,j})$ of adjacent block using the sliding window method. We should note that the disparities d_{ij} at (i,j) and $d_{i+1,j}$ at $(i+1,j)$ have the following relation in the sliding window method, $|d_{ij} - d_{i+1,j}| = 1$.

Furthermore, if the SAD value of one-column pixels is implemented by GPU parallel processing, the calculation of (2) becomes much faster. This paper first aggregate the likelihood matching cost for each column of pixels in GPU programming, and exploits the sliding window concept to calculate the whole SAD value for each disparity. This reduces the calculation time in the SAD aggregation which is usually main part of time consumption.

2.2 GPU programming

For fast operation, we implement the rank transform and SAD processes in GPU programming. The GPU implementation consists of 3 parts: 1) Data exchange between host and GPU, 2) Rank transform computation, and 3) Block based cost aggregation. We implement sliding window method in cost aggregation, and exploit the sharing memory to reduce memory overlapping. This implementation reduces the data flow between CPU and GPU, which improves the speed.

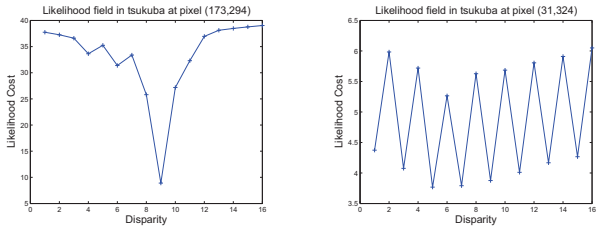
For the first part the pinned memory is used for fast data exchange between host and device as it efficiently uses memory cache. Rank transform is computed by separate kernels for left and right images. Each thread of the kernel calculates rank transform on one pixel. For this each thread block first copies the data required by its threads for rank transform computation using texture cache onto its shared memory. Each thread then reads rank window pixels from low latency shared memory to do the calculation. Using shared memory greatly reduces the additional common or overlapping data reading from the device memory.

Next, for the cost aggregation kernel, we propose a computation technique which greatly improves the implementation speed and we name it as *disparity sweep method*. Implementation of sliding window based aggregation is not new on GPU but we observed that the conventional GPU based sliding window techniques suffer greatly from inefficient memory management. Device memory access is one of the primary bottlenecks of any data intensive application on GPU. It was observed that the conventional implementations had two drawbacks. One is that the image data is directly read from the device memory for aggregation. Thus, the threads continuously and alternatively read data from left and right images. This temporal inconsistency forces the texture cache to flush after each reading, which makes it ineffective. The other drawback is on memory overlapping. To make best use of sliding window technique, the entire image is evaluated for each disparity one by one. This results in numerous overlapping memory reading for images spread across disparity range.

To address these bottlenecks we designed a new implementation in which we combine disparity range with groups of *disp_sweep* (16 in our case). That is to say, for each read data we find the required SADs for *disp_sweep* number of range. Basically, each block is responsible for calculating disparities of N number of rows, and thus each thread for N pixels facilitates the sliding window approach. First WIN_SIZE number of rows from left and right images are read one by one and stored onto the shared memory. We read in extra data from right image as *APRON* with the width equal to *disp_sweep*. This contiguous data access pattern is better suited for using texture reading. Since box filtering is separable, first, threads sum and store the vertical column SADs for *disp_sweep* number of range in different arrays, then each thread collaboratively uses column sums by other threads and performs horizontal aggregation. The disparity sweep technique reduces device memory reading, which boosts speed. A shared memory array is continuously updated with current minimum SAD cost value and corresponding disparity value. In order to calculate disparity for next image row, threads flush the unwanted first row of previous aggregation window and read in the new bottom row for the current aggregation window into the shared memory. The same aggregation procedure is then repeated.

3. PRIOR MODELING

This paper exploits the smoothness prior of disparity field to improve the accuracy. Since the processing of prior modeling should be fast and parallelized for real-time operation, we introduce the prior modeling as a kind of postprocess-



(a) (b)

Figure 2: Examples of likelihood matching. (a) reliable likelihood matching with large ratio between the first and second minimum errors, (b) unreliable likelihood matching with small ratio.

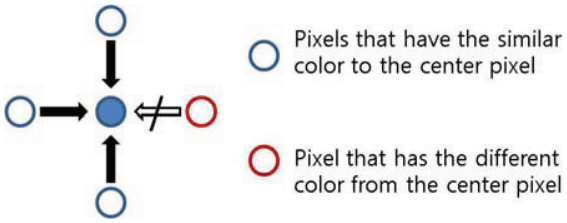


Figure 3: The prior modeling based on color similarity.

ing after finding the initial disparity map by the fast likelihood matching. Dynamic programming is popular for fast prior modeling, but it suffers from some artifacts on the scanlines [14, 15, 16, 17]. Adaptive dynamic programming and belief propagation do not show the real-time performance [12, 19, 20].

The proposed prior modeling is similar to the MRF modeling, and the joint energy function is defined as

$$E(d_{ij}) = SAD(d_{ij}) + \alpha_{ij} \sum_{n \in N} w_c(d_n) \left| d_{ij} - d_n \right|, \quad (3)$$

where $SAD(d_{ij})$ is the likelihood matching error of rank transform at a pixel (i, j) and disparity d_{ij}. d_n is the disparity value in the neighborhood N, which is the first-order MRF window with four elements. A parameter α_{ij} is a weighting factor of smoothness prior. When α_{ij} is large, the disparity map becomes smoother and the effect of likelihood matching is reduced. On the other hand, the smaller values of α_{ij} reduce the effect of smoothness prior. In the paper, α_{ij} is adaptively changed with respect to the reliability of likelihood matching. In the case of bad likelihood matching and occlusion, α_{ij} is increased so that the effect of unreliable likelihood matching is reduced in determining the disparity. The reliability $R(i, j)$ is determined by the ratio between the first and second minimum matching error,

$$\frac{1}{\alpha_{ij}} \propto R(i,j) = \frac{SAD(d_{ij}^2)}{SAD(d_{ij}^1)}, \quad (4)$$

where $SAD(d_{ij}^1)$ and $SAD(d_{ij}^2)$ are the first and second minimum matching errors, respectively. α_{ij} is inversely varied by $R(i, j)$. Figure 2 shows the examples of well matched and ill matched pixels, respectively. As we can observe, the pixels that have large ratios between the first and second minimum matching errors show reliable disparity estimation using the likelihood matching.

And $w_c(d_n)$ in (3) is a weight based on color information between (i, j) and neighborhood n. Figure 3 shows the neighborhood of prior modeling and color-based interaction. When the color of neighboring pixel n is similar to that of current pixel (i, j), the weight is increased so that the effect of neighborhood with similar colors is emphasized. On the other hand, the weight is decreased to stop the interaction of neighborhood when the colors are different. The color-based weight preserves the disparity discontinuity in processing the smoothness prior modeling. From the initial disparity map by the likelihood matching, the disparity map is recursively updated as below,

$$d_{ij}^{(k+1)} = \min_{d_{ij}} \left[SAD(d_{ij}) + \alpha_{ij} \sum_{n \in N} w_c(d_n) \left| d_{ij} - d_n^{(k)} \right| \right], \quad (5)$$

where $d_{ij}^{(k)}$ is the disparity value estimated in the k^{th} iteration. This process makes the disparity map smoother especially in the occlusion and textureless regions, since the likelihood matching is unreliable and unstable.

4. HIERARCHICAL APPROACH

The processes mentioned in Section 2 and 3 are performed in the hierarchical scheme. That is to say, the stereo images are decomposed into the Gaussian pyramid structure and disparity map is estimated by successive refinement from the lowest resolution to the highest one. From the lowest resolution, the disparity is estimated by the algorithm in Section 2 and 3. Then, the disparity map is doubled and interpolated to upper level using color similarity. The disparity map in the lower resolution is zero-order interpolated by the neighboring disparity with the most similar color. This color-based interpolation preserves the discontinuity of disparity map. The estimated disparity map in the lower level is refined with small search range. For the whole disparity range, $0 \le d \le S$, and the number of pyramid layers, L, the search range for the lowest resolution is set as

$$0 \le d \le \left\lceil \frac{S}{2^L} \right\rceil, \quad (6)$$

where $\lceil \cdot \rceil$ is the ceiling function. For the next layers, we set the search ranges to prevent estimation errors in the lower resolutions from propagating into the next layers. We define search ranges with negative and positive directions, which are not usual in stereoscopic ordering constraints,

$$-\gamma \left\lceil \frac{S}{2^L} \right\rceil \le d \le \gamma \left\lceil \frac{S}{2^L} \right\rceil, \quad (7)$$

where γ adjusts the search ranges. The reduced search range improves the speed of estimation process. This hierarchical approach improves both estimation accuracy and speed.

5. EXPERIMENTAL RESULTS

The proposed algorithm has been implemented with graphic hardware (Nvidia GTX-285). We evaluated the proposed algorithm using the Middlebury stereo images [23]. The parameters in the proposed algorithm are summarized in Table

Table 1: Parameters in the experiments.

Parameters	Value
RT window size	7 x 7
Block aggregation size	9 x 9
number of pyramid layers	4
Gaussian filter	5 tabs, variance=1.0
γ in (7)	1.0

Table 2: Time consumption for detailed processes.

Process	time (ms)	fps
RT (7x7, left/right images)	1.8	313.4
SAD (9x9 block aggregation)	6.4	88.1
prior modeling	7.2	78.3
Total	15.4	36.6

1. We counted time consumption from image loading to disparity map generation. The accuracy of disparity maps are evaluated in [23], and the speed is evaluated by DES (disparity estimation per second),

$$DES = \frac{\text{image size } \times \text{ disparity range}}{\text{time consumption}}. \quad (8)$$

For real-time applications of VGA (640×480) image and 60 pixels of disparity range, we need 553 MDES or 30 frames/second (fps). We restrict the number iterations in processing the prior modeling to keep 30 fps. We evaluate the proposed algorithms in the aspect of both accuracy and speed. However, it is a bit difficult to compare the accuracy since most reports of fast algorithms are focused on the operation speed. There is no enough benchmark to compare the accuracy of fast algorithms.

First, we decompose the time consumption of proposed method in the original resolution. Table 2 shows the time construction of each process. The cost aggregation consumes much time in GPU programming since it needs much calculation. The prior modeling is usually processed in CPU-based operation, thus it takes much time, too. When we implement the proposed method in the multi-resolution approach, the time consumption for cost aggregation and prior modeling is decreased since the image size and disparity range are also reduced. However, we need another computation for Gaussian pyramid construction and disparity interpolation. We limit our final results above 30fps.

Table 3 summarizes some results of the proposed algorithm and fast methods. As we expect, the algorithms for real-time operation is not so good in accuracy. However, the proposed algorithm is competitive with the algorithms for real-time operation. Figure 4 shows the disparity maps by the proposed algorithm. As we can see, the erroneous regions in the disparity map become smooth, and discontinuity is well preserved in processing the prior modeling and color-based interpolation. The unreliable disparities of occlusion and textureless regions are corrected by the prior modeling. This result means that the prior modeling and color-based interpolation in the hierarchical approach improve the accuracy of disparity estimation.

According to the experiments, the proposed algorithm estimates the disparity map in real-time with reasonable accuracy. Since there is room for further processing within the time limitation, it is possible to improve the accuracy in the proposed algorithm. Occlusion handling and adaptive weights optimization should be further performed.

6. CONCLUSION

This paper has proposed a real-time stereo matching algorithm using GPU programming. The proposed approach first calculates the likelihood matching error based on rank transform. The likelihood matching error is parallelized and implemented in GPU programming. The adaptive memory handling in graphic hardware is introduced in aggregating the matching errors, which improves the speed. Once an initial disparity map is determined based on the likelihood matching, then the disparity map is recursively updated by the prior model of disparity field. The prior model reflects the smoothness of disparity field and is implemented by a pixel-wise energy function. The disparity at each pixel is finally determined by minimizing the joint energy function which combines the likelihood matching error with the prior energy model. This processing is performed in the hierarchical approach, and color-based disparity interpolation is proposed to preserve discontinuity. According to the experiments with Middlebury stereo images, the proposed method shows good estimation accuracy with more than 30 frames per second for 640x480 images. The proposed method is suitable for real-time stereo system in the usual PC environment.

7. ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0007200).

REFERENCES

[1] D. Scharstein and R. Szeleiski, "A taxonomy and evaluation of dense two frame stereo correspondence algorithm," *IJCV*, vol. 47, no. 1/2/3, pp. 7-42, 2002.

[2] A. Redert, E. Hendricks, and J. Biemond, "Correspondence estimation in image pairs," *IEEE Signal Proc. Magazine*, vol. 16, no. 3, pp. 29-46, 1999.

[3] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: Theory and experiments," *IEEE Trans. on PAMI*, vol. 16, no. 9, pp. 920-932, 1994.

[4] Y. Boykov, O. Veksler, and R. Zabih, "A Variable Window Approach to Early Vision," *IEEE Trans. PAMI*, vol. 20, no. 12, pp. 1283-1294, 1998.

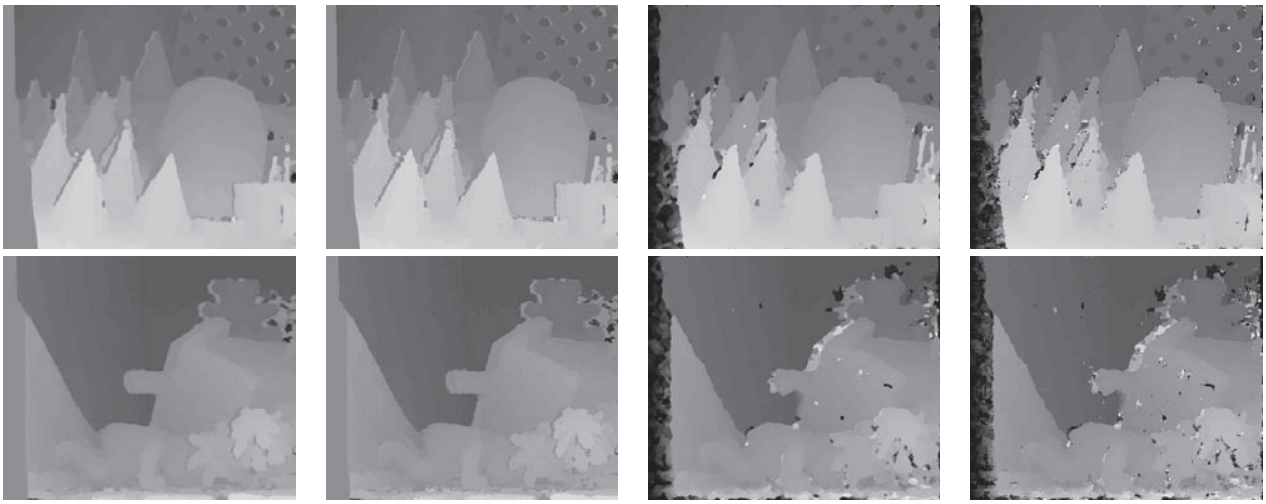
[5] O. Veksler, "Fast Variable Window for Stereo Correspondence using Integral Images," *Proc. of CVPR*, vol. 1, pp. 556-561, 2003.

[6] Yong Seok Heo, Kyong Mu Lee, Sang Uk Lee, "Robust Stereo Matching Using Adaptive Normalized Cross Correlation," *IEEE Trans. PAMI*, vol. 32, 2010.

[7] K. J. Yoon and I. S. Kwon, "Adaptive support weight approach for correspondence search," *IEEE Trans. PAMI*,vol. 28, no.4, pp. 650-656, 2006.

Table 3: Comparison of accuracy and speed.

Algorithm	*fps*	*DES*	*Teddy*			*Cones*		
			nonoc	*all*	*disc*	*nonoc*	*all*	*disc*
Rank Transform (RT)	68.7	1268 M	14.4	23.2	32.1	7.01	17.1	18.7
Rank Transform + prior model	36.2	675 M	13.3	22.2	30.3	6.29	16.4	17.2
Hierarchical RT	50.3	928 M	9.2	18.2	21.3	5.87	15.1	14.9
Hierarchical RT + prior model	**31.2**	**583 M**	**7.8**	**17.1**	**18.9**	**5.01**	**14.3**	**13.8**
SAD (11x11)	67.9	1252 M	19.6	27.2	34.0	12.1	21.0	22.0
LSTDP [16]	0.73	13.5 M	11.1	16.4	23.4	6.39	11.8	13.5
GORDP [17]	3.26	60 M	9.03	16.8	18.4	13.1	20.1	20.1



(a) (b) (c) (d)

Figure 4: Disparity maps by the proposed algorithm for *Cones* and *Teddy* images. (a) Proposed method (RT+hierarchical+prior model), (b) RT+hierarchical, (c) RT+prior model, (d) RT only.

[8] Y. Boykov, O. Veksler, and R.Zabih, "Fast approximate energy minimization with graph cuts," *IEEE Trans. on PAMI*, vol. 23, no. 11, pp. 1222-1239, 2001.

[9] J. Sun, N. N. Zheng, and H. Y. Shum, "Stereo matching using belief propagation," *IEEE Trans. on PAMI*, vol. 25, no.7, pp. 399-406, 2003.

[10] D. Scharstein and R. Szeliski, "Stereo matching with non-linear diffusion," *IJCV*, vol. 28, no. 2, pp. 155-174, 1998.

[11] S. H. Lee, Y. Kanatsugu, and J.-I. Park, "Stochastic diffusion for stereo matching and line fields estimation," *IJCV*, vol. 47, no. 1/2/3, pp. 195-218, 2002.

[12] O. Veksler, "Stereo correspondence by dynamic programming on a tree," *Proc. of CVPR*, vol. 2, pp. 384-390, 2005.

[13] Z. Wang and Z. Zheng, "Near real-time reliable stereo matching using programmable graphics hardware," *Proc. of CVPR*, vol. 1, pp. 924-931, 2005.

[14] J. Congote, J. Barandiaran, I. Barandiaran, and O. Ruiz, "Real-time dense stereo matching with dynamic programming," *Proc. of CEIG*, 2009.

[15] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister, "High quality real-time stereo using adaptive cost aggregation and dynamic programming," *Proc. of 3DPVT*, pp. 798-805, 2006.

[16] Yi Deng and Xueyin Lin, "A fast line segment based dense stereo algorithm using tree dynamic programming," *Proc. ECCV*, LNCS 3953, pp. 201-212, 2006.

[17] M. Gong and Y.-H. Yang, "Real-time stereo matching using orthogonal reliability dynamic programming," *IEEE Trans. IP*, vol. 16, no. 3, pp. 879-885, 2007.

[18] A. Hosni, M. Bleyer, and M. Gelautz, "Near real-time stereo with adaptive support weight approaches," *Proc. of 3DPVT*, 2010.

[19] C. K. Liang, *et. al*, "Hardware-efficient belief propagation," *Proc. of CVPR*, pp. 80-87, 2009.

[20] Q. Yang, K.-H. Tan, and N. Ahuja, "A constant-space belief propagation algorithm for stereo matching," *Proc. of CVPR*, 2010.

[21] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," *Proc. of ECCV*, pp. 151-158, 1994.

[22] J. Banks and M. Bennamoun, "Reliability analysis of the rank transform for stereo matching," *IEEE Trans. on Sys., Man, and Cyber., Part B:*, vol. 31, no. 6, 2001.

[23] http://vision.middlebury.edu/stereo.