Partial Demosaicing for stereo matching of CFA images on GPU and CPU

Naiyu Zhang, Jean-Charles Creput, Hongjian Wang, Cyril Meurie, Yassine Ruichek

Partial Demosaicing for Stereo Matching of CFA Images on GPU and CPU

Naiyu ZHANG, Jean-Charles CREPUT, Hongjian WANG, Cyril MEURIE and Yassine RUICHEK
IRTES-SET, University of Technology of Belfort-Monbeliard, Belfort, France
Email: {naiyu.zhang, jean-charles.creput, hongjian.wang, cyril.meurie, yassine.ruichek}@utbm.fr

***Abstract*—This paper presents a GPU implementation of a partial demosaicing scheme that is specially designed for stereo matching of CFA image. This method consists of three main techniques keys: the adapted matching cost for CFA image, the estimated Second color component based on Hamilton's estimate method and a robust cost aggregation window. Experiments are carried out to explore the performance for this method on GPU both at matching quality and matching efficiency, with comparison with version on CPU. The experiments on different size image pairs from Middlebury dataset show that this method can be substantially accelerated on GPU when the image size is large and has still space for improvements in performance.**

Keywords: ***GPU, Demaosaicing Stereovision, CFA Image, CUDA***

I. INTRODUCTION

In most implementation of stereo matching, such as in intelligent cars and integrated robot systems, color image pairs can be acquired by two types of cameras: the one equipped with three sensors associated with beam splitters and color filters providing the so-called full color images, of which each pixel is characterized in Red, Green and Blue levels, and the one equipped only with a single-sensor.

In the second case, the single-sensor can not provide a full color image directly but actually deliver a color filter array (CFA) image, of which every pixel is characterized by a single color component that can be one of the three color components: Red, Green and Blue. So, the missing color components has to be estimated at each pixel. This process of estimating the missing color components is usually referred to as CFA demosaicing and produces a demosaicing color image where every pixel is represented by an estimated color point [1]. This estimation step brings some artifacts in color density values, so, it is interesting to find one stereo matching directly based on CFA image.

As the demosaicing methods intend to produce demosaiced color images, they attempt to reduce the presence of color artifacts, such as the false colors ot zipper effects, by filtering the images [2]. So, some useful color information for stereo matching may be lost in the color demosaiced images. As a result, the demosaiced color image pairs stereo matching quality usually suffer either from color artifacts or from the alteration of color texture caused by demosaicing schemes.

The method that is used in this paper is an alternative solution to match pixels by analyzing directly the CFA images without reconstructing the full color image by demosaicing processing, but only estimating a second color component. The method is a local dense stereo matching method that was first proposed by Halawana [3]. Here, we study its potential for Graphics Processing Unit (GPU) implementation.

The following contents are organized in three parts. The Section II presents the partial demosaicing matching method including the estimation of the second color component and the adapted matching cost. The Section III describes the experiment platform and CUDA implantation. The Section IV details the experimental results. At the end is a conclusion.

II. PARTIAL DEMOSAICING MATCHING METHOD

A. Second Color Component

The partial demosaicing matching method starts with the mosaiced CFA images. Here, the CFA images are those images obtained according to the Bayer color filter, each two-by-two submosaic contains 2 green, 1 blue and 1 red filter, each covering one pixel sensor and the mosaiced CFA image is the one whose pixel contains only one color component according to the Bayer color filter.

Different from the classic methods, which estimate all the missing color components for every pixel in the CFA images, the partial demosaicing method estimates only one color component, the Second Color Component (SCC), for every pixel. Here, the SCC is defined as the color component that is available in the same line. This means that SCC is the green color for all the red and blue pixels while for the green pixels the SCC is the red color component for even lines and the blue color component for odd lines. Summarized as (1).

$$SCC(x,y) = \begin{cases} \hat{G}(x,y) & \text{for red and blue pixels} \\ \hat{R}(x,y) & \text{for green pixels in even lines} \\ \hat{B}(x,y) & \text{for green pixels in odd lines} \end{cases} \quad (1)$$

B. Hamilton's Estimate Method

This method is an edge-adapted demosaicing method presented by Hamilton and Adams [4]. To select the interpolation direction, this method takes into account both gradient and Laplacian second-order values by using the green levels available at nearby pixels and red (or blue) samples located two apart.

We take the case GRG, as illustrated in Fig. 1, as example, to estimate the missing green level at the red pixels, this method uses the following algorithm:

a) Approximate the horizontal Δ^x and vertical Δ^y gradients thanks to absolute differences as (2).

 ISBN: 978-1-61208-310-0

b) Interpolate the green level as (3).

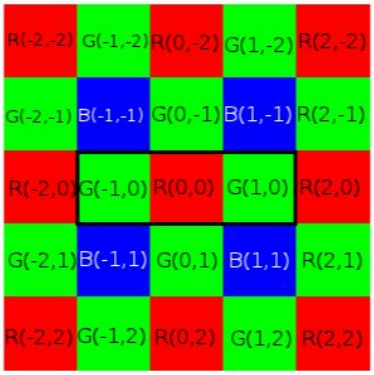


Fig. 1: Estimation of SCC in case of GRG.

$$\begin{cases} \Delta^x = |G_{-1,0} - G_{1,0}| + |2R - R_{-2,0} - R_{2,0}| \\ \Delta^y = |G_{0,-1} - G_{0,1}| + |2R - R_{0,-2} - R_{0,2}| \end{cases} \quad (2)$$

$$\hat{G} = \begin{cases} \frac{G_{-1,0}+G_{1,0}}{2} + \frac{2R-R_{-2,0}-R_{2,0}}{4} & \text{if } \Delta^x < \Delta^y \\ \frac{G_{0,-1}+G_{0,1}}{2} + \frac{2R-R_{0,-2}-R_{0,2}}{4} & \text{if } \Delta^x > \Delta^y \\ \frac{G_{-1,0}+G_{1,0}+G_{0,-1}+G_{0,1}}{4} & \\ \quad + \frac{4R-R_{-2,0}-R_{2,0}-R_{0,-2}-R_{0,2}}{8} & \text{if } \Delta^x = \Delta^y \end{cases} \quad (3)$$

Since this method well combines two color component data in partial derivate approximations by exploiting spectral correlation in the green plane estimation, the precision is well guaranteed.

Each pixel with coordinates (x, y) in the partially demosaiced color images is characterized by a two-dimensional partial color denoted $\hat{I}_{PA}$. As shown in (4), this partial color point is composed of the available color component and the estimated second color component.

$$\hat{I}_{PA}(x,y) = \begin{cases} (R(x,y), \hat{G}(x,y))^T & \text{if x is odd and y is even} \\ (\hat{R}(x,y), G(x,y))^T & \text{if x is even and y is even} \\ (\hat{G}(x,y), B(x,y))^T & \text{if x is even and y is odd} \\ (G(x,y), \hat{B}(x,y))^T & \text{if x is odd and y is odd} \end{cases} \quad (4)$$

C. Adapted Matching Cost

The method is a local dense stereo matching method also called window-based approach. It respects the very assumption that the color information of neighbors of a left pixel is close to those of the same neighbors of its homologous right pixel in the right image. So, the matching costs are defined between the window around the left pixel and the window around the candidate right pixels in the corresponding line (epipolar line) in the right image. The window is shifted over all possible pixels so that a matching cost between the left pixel and each candidate in the right image is obtained. By the Winner-Takes-All method, the final disparity estimation is realized by selecting the window with the lowest matching cost.

The matching cost, SSD (Sum of Squared Differences cost), is adapted as (5).

$$SSD^{\omega}(x_l, y, s) = \sum_{i=-\omega}^{\omega} \sum_{j=-\omega}^{\omega} \| \hat{I}_{lPA}(x_l + i, y + j) - \hat{I}_{rPA}(x_l + i - s, y + j)\|^2 \quad (5)$$

Where the $\| \bullet \|$ is the Euclidean norm. While s is the spatial shift along the horizontal epipolar line and ω the half-width of the $(2\omega + 1) \times (2\omega + 1)$ aggregation window.

Since these pixels of horizontal lines with the same parity in the left and right partially demosaiced color images are characterized by the same two color components, we can reasonably assume that the partial color points of two homologous pixels are similar. Because the partial costs compare the partial color points of left and right pixels located on the same horizontal lines, they reach an extremum when the shift is equal to the disparity.

III. Experiment

A. Experiment Platform and CUDA Implementation

We implement on GPU the partial demosaiced matching method of H. Halawana [3]. We use the well known Middlebury databases [5] to evaluate the method on both accuracy and computation time, as the image size grows. It is worth noting that the method was only implemented in a sequential computer (CPU) with no systematic evaluation of the trade-off between quality and computation time. The ten datasets used in our experiments are entitled *'Aloe', 'Bowling1', 'Cloth1', 'Flowerpots', 'Lampshade1', 'Midd1', 'Monopoly', 'Plastic', 'Rocks1'* and *'Wood1'*. All the datasets of 2 views are used here (full-size (width: 1240...1396, height: 1110), half-size (width: 620...698, height: 555), and third-size (width: 413...465, height: 370)) . In these datasets, the color stereo images are acquired by high resolution cameras equipped with one-single-sensor [6]. That's to say, the full color images are in fact color images that have been demosaiced by a specific chip integrated in the camera. They could contain artifacts caused by the demosaiced step. What's more, the work of applying a demosaicing step on CFA images, which have been generated by sampling color components from these previously demosaiced color images involves applying two successive demosaicing steps on the CFA images acquired by the camera. However, since Middlebury is the most frequently utilized database, an evaluation on this database will allow future comparative evaluation with new methods.

Our experiments are carried out both on CPU and GPU. For the CPU, we use a Intel(R) Core(TM)2 Duo CPU E8400 of 3.00GHz, with two cores having a cache of 6144KB. The GPU used in the experiments is a GPU GeForce GTX 570 of NVIDIA. It has 15 multiprocessors of 32 cores, GPU clock speed 1.54GHz, Memory clock rate 2000MHz, Memory Bus width 320 bits while the total amount of global memory is 1280 Mb (constant memory 65536 Kb, shared memory per

 ISBN: 978-1-61208-310-0

block 49152 Kb). The system, on which the experiment is evaluated is Ubuntu 11.04 32 bits.

The programming interface we used for parallel computation on GPU is the Compute Unified Device Architecture (CUDA). The parallel computation work is realized by a *kernel* function, which is executed concurrently by multiple threads on data elements. All these threads are organized into a two level concepts: *grid* and *block*. A *kernel* has one *grid*, which contains multiple *blocks*. Every *block* is formed of multiple threads. The dimension of *grid* and of *blocks* can be one-dimension, two-dimension or three-dimension. The performance of GPU with CUDA is closely related to thread organization and memory accesses, which should attract much attention according to various computation works and GPU platform. Based on our experiment platform, given an image of size $W \times H$ as example and briefly lay out the CUDA settings and the parameter values mentioned in our algorithm.

In our experiments, we use the two-dimension *blocks* of 16×16 size. Every thread takes care of one pixel and the size of *grid* is obtained by $\frac{W+16-1}{16} \times \frac{H+16-1}{16}$ for a given $W \times H$ image. In the SCC estimation step, a grid of $W \times H$ threads is created and each thread takes care of one pixel in the estimation, and the shared memory is employed to for fast memory access. For the matching cost computation, a grid of $W \times H$ threads is employed to compute the matching cost for every pixel at a set of given disparity and then pick out the best homologous candidate pixel by the Winner-Takes-All method, here, the main data is stored on Global memory space while the shared memory space is used to support the computation of matching cost and the comparison in WTA processing.

B. Experiment Process

The procedures on CPU and on GPU are almost the same.

As the datasets are all full color images, at the very beginning of the experiments, a simulation step for every pair of images is realized to obtain their CFA images needed by keeping only one of the three color components at every pixel. This work is done by GPU and by CPU separately . The whole evaluation is performed according to the spatial arrangement of the Bayers CFA. Then, for every CFA image, we do partial demosaicing step. Here, the partial demosaicing estimates only the missed Second Color Component (SCC) for every pixel by Hamilton's estimate method in the left image and in the right image. So, the left demosaiced color image and the right one are produced. The estimation method is the edge-adaptive demosaicing method proposed by Hamilton as presented in the precedent section.

The original full color image is shown in Fig. 2(a) and the left demosaiced color image using Hamiltons method is shown in Fig. 2(c). They look somewhat similar. However, zooming on the square areas outlined in these images as presented in Fig. 2(b) and Fig. 2(d), shows that textured areas are locally different.

Then the local stereo matching algorithm and the Winner-Takes-All method are taken into actions. In this step, as shown in Fig. 3 for every pixel in the left image, the method finds out its homologous pixel from a group of candidate pixels in the right image and compute their disparity s by computing the matching cost SSD, which is modified to adapt to the partial demosaiced color images.

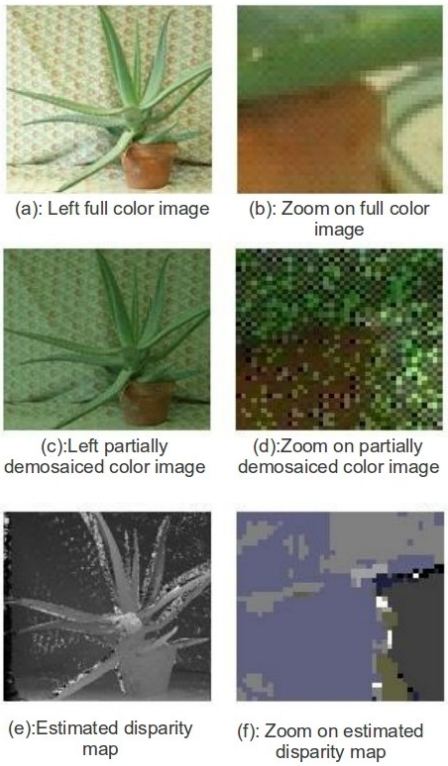


Fig. 2: 'Aloe' left image.

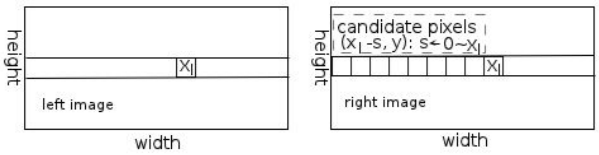


Fig. 3: matched pixel and its candidate ones in the right image.

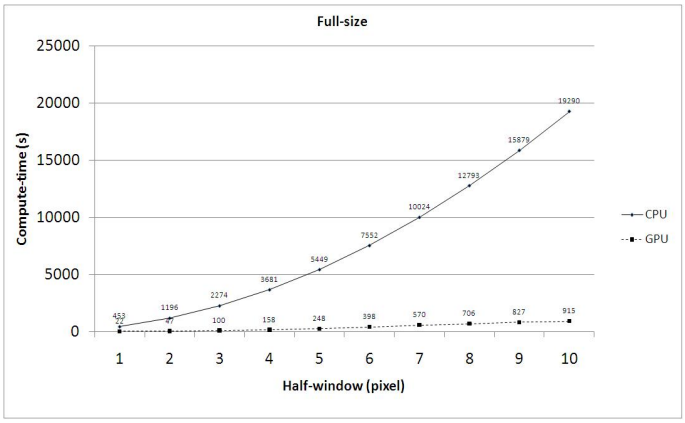
A matching is considered as valid when the absolute difference between the estimated disparity and the given benchmark $d_l^{\omega}(x_l, y)$ is lower or equal to δ, which is the disparity error tolerance. In the experiments we set this coefficient to 1.

At the end, a disparity map, as illustrated in Fig. 2(e), is estimated for each pair of image.

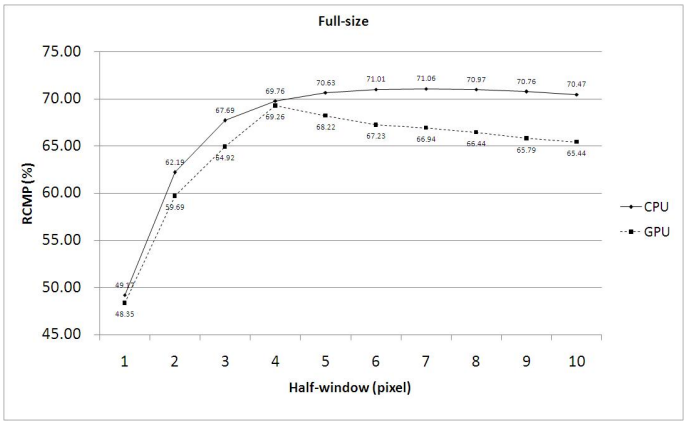
IV. Experimental Results and Discuss

The experiment are executed on all the ten chosen datasets. Here, we take the 'Aloe' image group as an example. As it is shown in Fig. 4 that with the increase of the half-window, the computation time increases significantly. When the half-window is given as ω, for every pixel in the left image and for their every possible candidate pixel in the right image, we should compute a group of $(2\omega+1) \times (2\omega+1)$ pixels to obtain the matching cost. So, when the half-window ω increases, the

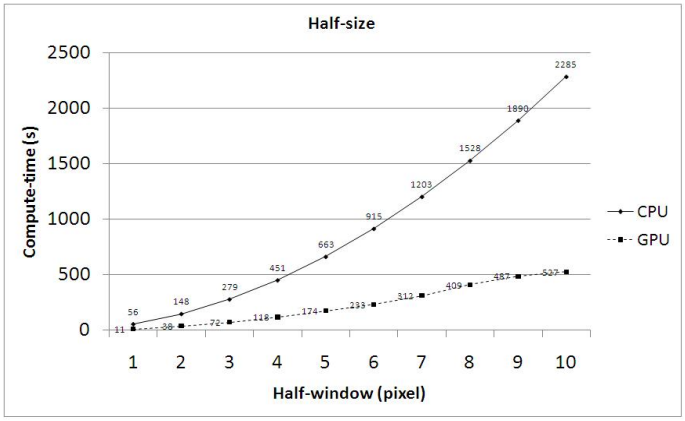
 ISBN: 978-1-61208-310-0



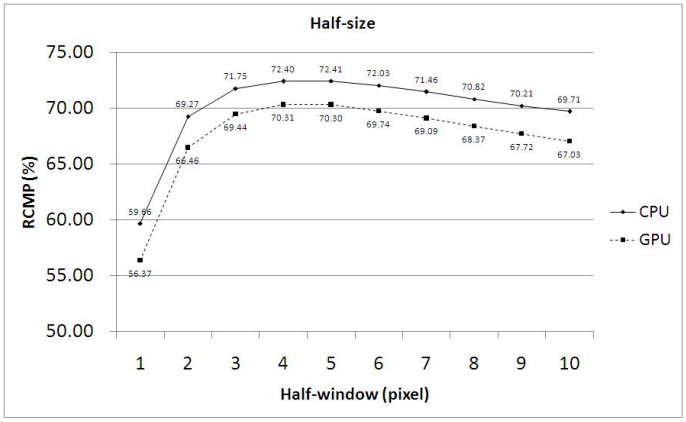
(a) Compute-time of GPU and CPU on full size datasets



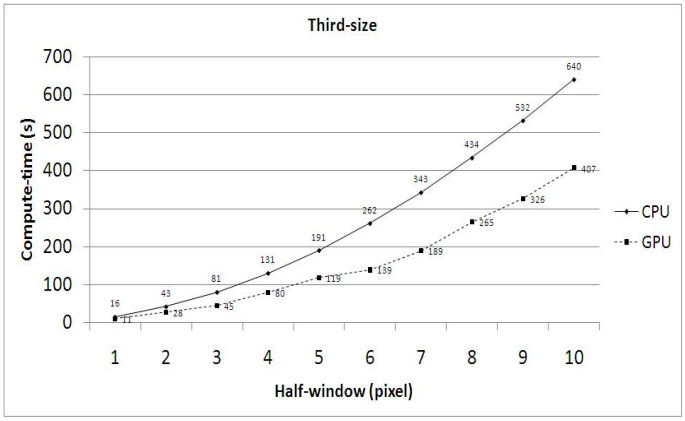
(b) RCMP of GPU and CPU on full size datasets



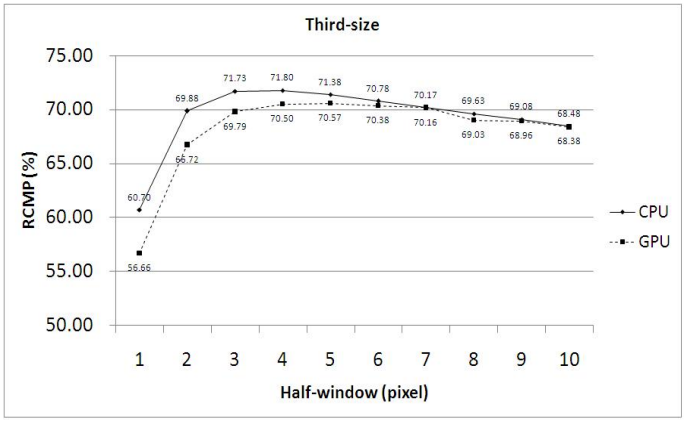
(c) Compute-time of GPU and CPU on half size datasets



(d) RCMP of GPU and CPU on half size datasets



(e) Compute-time of GPU and CPU on third size datasets



(f) RCMP of GPU and CPU on third size datasets

Fig. 4: Compute-time and rate of correctly matched pixels (RCMP) obtained with the adapted SSD computed on the full sizeAloe stereo image pair for δ set to 1.

 ISBN: 978-1-61208-310-0

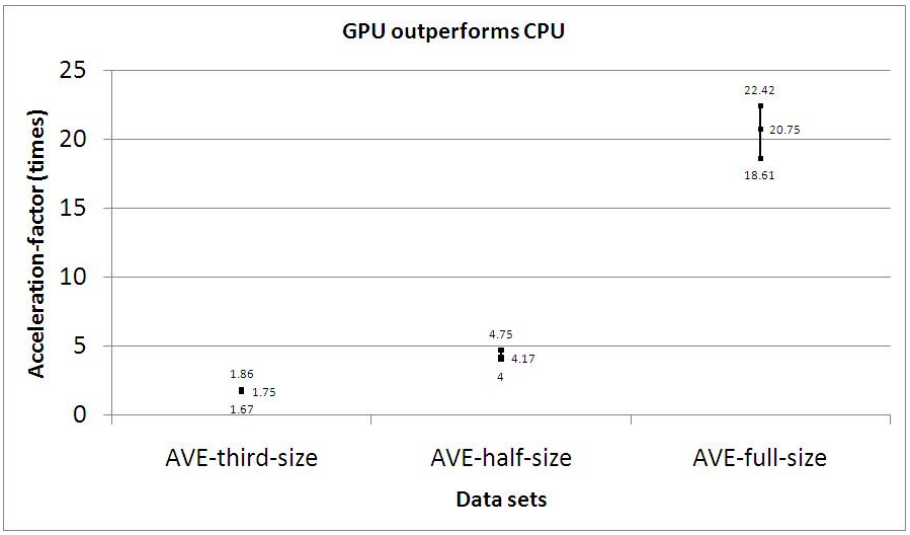


Fig. 5: GPU outperforms CPU in terms of computation time. The maximum, the minimum and the average acceleration factor obtained in the executions on 'Aloe' image pair based on half-window from 1 to 10 are marked. The acceleration increases along with the image dimensions.

TABLE I: COMPARATIVE EVALUATION ON THE FOUR MIDDLEBURY DATA SETS OF THE THREE DIFFERENT SIZES BY DIFFERENT GPU PROGRAMMING VERSION. 'COMPUTATION TIME (CT) ' AND 'PERCENTAGE OF BAD PIXELS (PBP)' .

		Partial_Demosaiced		Full_Color		Gray_level	
Image	Size	PBP	CT (s)	PBP	CT (s)	PBP	CT (s)
Rocks1	Third size	28.76	0.214	25.81	0.191	24.58	0.146
	Half size	32.24	0.679	27.87	0.715	25.82	0.913
	Full size	37.06	5.811	31.20	5.717	28.93	8.235
Aloe	Third size	27.08	0.214	24.96	0.192	23.96	0.115
	Half size	29.33	0.698	25.78	0.697	24.94	0.68
	Full size	30.26	5.955	28.98	5.875	26.04	5.595
Cones	Third size	39.18	0.234	29.77	0.229	28.76	0.276
	Half size	46.98	2.123	37.36	1.548	36.91	3.024
	Full size	52.14	20.569	44.85	11.502	45.16	19.881
Teddy	Third size	45.27	0.23	32.23	0.23	29.86	0.284
	Half size	53.69	2.12	38.38	1.55	37.77	3.246
	Full size	57.36	27.09	46.58	11.51	47.37	24.922
Avg.		39.95	5.495	32.82	3.330	31.68	5.610

computing complexity is a squared function of ω, which is a real challenge to the capacity of the processors.

Meanwhile, with the increase of the half-window, and so, the computation intensity, CPU's computation time increases more importantly than GPU's computation time. Moreover, as illustrated in Fig. 5, as the image dimensions augment, the acceleration factor obtained by using GPU has expanded from 1.75 to 20.75. This means that the GPU offers powerful computation capacity in intense computation thanks to the parallel organization of the *blocks* and the threads on GPU.

The performance of GPU is significantly influenced by the dimension of the image. The nearer to the complete load of the employed streaming multiprocessor on GPU, the higher performance we get.

The Ratio of Correctly Matched Pixel (RCMP) is the percentage of the well matched pixels in all the pixels of the image to be matched. As it is shown in Fig. 4, for 'Aloe ' image pair, the RCMP reaches the smooth peak when the half-window is between 4 and 8 (for those textureless image pairs, the half-window should be bigger to have their peak of RCMP). In fact, whatever the image type, the matching performance increases with aggregation window half-width. Small windows do not contain enough information to allow a correct matching. At the opposite, too large aggregation windows may cover image regions containing pixels with different disparities, which explains the decrease of matching

 ISBN: 978-1-61208-310-0

performance. Though on CPU and on GPU, all the single-precision floating-point computations follow the same accuracy standards. The accuracy-loss is more important on GPU owning to the accuracy problem of floating-point, that is why the results by CPU and that by GPU may have some deviations, especially when some cumulations in the computation exists. In our experiments, we use the floating point in the computation of matching cost with SSD and in the aggregation based on fixed support window, the deviation occurs at these computations, which can explain the tolerances between the RCMP by CPU and that by GPU shown in Fig. 4(b).

In addition, this method is also compared with the classic fixed windows matching methods (treating full color image and gray-level image separately) we used in former experiments, the results are shown in Table I. This method performs worse on all these four pairs both in matching quality and in computation time. That is because this method requires too many refers to the logical operations in the programming and too many branches in data treatments, which is the real weakness of GPU architecture. These branches and logical operations lead to great load on to the GPU system when loading data from the memory space and wastes GPU's CUDA cores, which are powerful in arithmetical operation, by making them do logical works.

V. Conclusion

This paper presents a GPU implementation of a partial demosaicing scheme specially designed for stereo matching of CFA image. By analyzing the CFA image directly, this method can handle the stereo matching works in case of single-CCD cameras usage. This method has three main techniques characters: the adapted matching cost for CFA image, the estimated Second color component based on Hamilton's estimate method, and the robust cost aggregation window. Experiments carried on show the performance of this method on GPU. The results show that this method can benefit from GPU's parallel architecture. The experiments also show that this method performs faster on GPU as the image size grows. Future improvements should come from a better setting of the memory arrangement in order to allow more coalescing accesses.

References

[1] S. Battiato, M. Guarnera, G. Messina, and V. Tomaselli, "Recent patents on color demosaicing," *Recent Patents on Computer Science*, pp. 1(3):194–207, November 2008.

[2] Y. Yang, O. Losson, and L. Duvieubourg, "Quality evaluation of color demosaicing according to image resolution," *Proceedings of the 3rd International Conference on Signal-Image Technology & Internet-based Systems, pages 640–646,Shanghai Jiaotong University, China*, December 2007.

[3] H. Halawana, "Partial demosaicing of CFA images for stereo matching," Ph.D. dissertation, Université de Lille 1, 2010.

[4] J. Hamilton and J. Adams, "Adaptive color plan interpolation in single sensor color electronic camera," *US patent 5,629,734, to Eastman Kodak Co., Patent and Trademark Office, Washington D.C.*, May 1997.

[5] S. A. N. Brad Hiebert-Treuer and D. Scharstei, "2006 Stereo Datasets," *http://vision.middlebury.edu/stereo/data/scenes2006/*, Nov 2006.

[6] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two- frame stereo correspondence algorithms," *International Journal of Computer Vision, 47:7–42*, 2002.

 ISBN: 978-1-61208-310-0