Dense realtime stereo matching using a memory efficient Semi-Global-Matching variant based on FPGAs

Maximilian Buder[a]

[a]German Aerospace Center, Robotics and Mechatronics Center, Rutherfordstr. 2, 12489 Berlin, Germany

ABSTRACT

This paper presents a stereo image matching system that takes advantage of a global image matching method. The system is designed to provide depth information for mobile robotic applications. Typical tasks of the proposed system are to assist in obstacle avoidance, SLAM and path planning. Mobile robots pose strong requirements about size, energy consumption, reliability and output quality of the image matching subsystem. Current available systems either rely on active sensors or on local stereo image matching algorithms. The first are only suitable in controlled environments while the second suffer from low quality depth-maps. Top ranking quality results are only achieved by an iterative approach using global image matching and color segmentation techniques which are computationally demanding and therefore difficult to be executed in realtime. Attempts were made to still reach realtime performance with global methods by simplifying the routines. The depth maps are at the end almost comparable to local methods. An equally named semi-global algorithm was proposed earlier that shows both very good image matching results and relatively simple operations. A memory efficient variant of the Semi-Global-Matching algorithm is reviewed and adopted for an implementation based on reconfigurable hardware. The implementation is suitable for realtime execution in the field of robotics. It will be shown that the modified version of the efficient Semi-Global-Matching method is delivering equivalent result compared to the original algorithm based on the Middlebury dataset.

The system has proven to be capable of processing VGA sized images with a disparity resolution of 64 pixel at 33 frames per second based on low cost to mid-range hardware. In case the focus is shifted to a higher image resolution, 1024×1024-sized stereo frames may be processed with the same hardware at 10 fps. The disparity resolution settings stay unchanged. A mobile system that covers preprocessing, matching and interfacing operations is also presented.

Keywords: stereo matching, realtime, FPGA, Semi-Global-Matching, mobile robotics

1. INTRODUCTION

In this paper a stereo image matching system is presented that targets mobile robotic applications. A major requirement for those systems includes the capability to operate with very limited energy and computational resources. Also it is commonly demanded to deliver dense and high quality VGA sized depth maps at 25 fps within an uncontrolled environment. Demanding a high-quality disparity maps, a high geometrical resolutions and a high frame rate on a restricted hardware platform is strongly contradicting. This paper proposes a FPGA based stereo matching system that uses a memory efficient variant of Semi-Global-Matching [1] * algorithm that fulfils above mentioned requirements.

Most of the realtime stereo matching solutions available today still rely on local matching techniques and a specialized hardware platform in order to meet the requirements within the robotic sector. Compared to state of the art global methods, their matching results are inferior and not very robust in real world scenes. The inherent static local windowing causes the disparity map to be blurred at depth discontinuities. Decreasing the window results in a higher signal-to-noise ratio and increases the ambiguity. Alternative local methods have been proposed to improve the matching results. Fixed multi-Window approaches select a window shape from a predefined set [2]. Because of the non-general character of the window-model, it is not possible to handle arbitrary depth discontinuities properly. Color segmentation during pre-processing may be used to find better suited window shapes which is at the end not feasible for realtime applications. A local color

Further author information: (Send correspondence to Maximilian Buder)
DLR: E-mail: maximilian.buder@dlr.de, Telephone: ++ 49 (0)30 67055 632
*to be published

segmentation technique is used by Yoon et at. [3] which assigns each window position a weight that is based on color and spatial information derived from the neighborhood. The shape of the support region changes arbitrarily at each pixel. This class of algorithm delivers better results than the basic windowing techniques but leads to increased complexity which is in contrast to global matching algorithms not justified, if the quality of the depth maps are compared.

Currently the class of global stereo matching algorithm deliver the best matching results [4]. Usually they are complex and require a high amount of memory to store intermediate results. Commonly they depend on an iterative color segmentation and multiple stereo matching subroutines which are used to refine the disparity at each step. To execute these algorithm on a standard dataset [4] it still takes several seconds on high performance CPUs. In recent years, GPUs became a popular computing platform which were used to increase the runtime of the top ranking stereo matching algorithm. These variants deliver either a high quality depth map for only small image dimensions and disparity ranges [5] or high geometric resolutions [6]. The second design target usually leads to heavily degraded depth maps. Additionally GPUs are not usefully integrated in mobile robotic systems.

The Semi-Global-Matching method introduced by H. Hirschmüller [7] follows a hybrid approach. Like the majority of global methods, SGM defines a global energy function E as in equation (1). The correspondence search problem is reformulated as an optimization problem. The Semi-Global-Matching method [7] relaxes the optimization problem from a NP-hard class to a linear approximation. Dynamic Programming follows the same approach. In contrast to Dynamic Programming, not only the scanline direction is being used. Instead, it is performed symmetrically from eight directions towards all pixels in the image (see figure 1). SGM does not suffer from streaking artefacts like Dynamic Programming and does not require iterations like Belief Propagation. The rather simple and regular integer operations of the SGM algorithm make it suitable for implementations on the GPU [8, 9, 10] and FPGA [11, 12] which permits realtime performance.

In practice, SGM has proved to be robust and insensitive to the choice of parameters in contrast to other methods like the one proposed by Xing et al. [13]. This makes SGM suitable for real world applications [14]. The major disadvantage of SGM is its memory consumption that depends on the number of pixels and the disparity range. In this paper, a memory efficient variant of the Semi-Global-Matching algorithm (SGM) is being used which is named eSGM. The algorithm has a memory requirement that is linear to the number of pixels, but independent from the disparity range.

2. REVIEW OF THE ESGM-METHOD

The objective of the eSGM method is to approximately minimize a global energy function

$$\begin{aligned} E(D) = \sum_{\mathbf{p}} (C(\mathbf{p}, D(\mathbf{p})) \quad &+ \quad \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_1 \, \mathrm{T}[|D(\mathbf{p}) - D(\mathbf{q})| = 1] \\ &+ \quad \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_2 \, \mathrm{T}[|D(\mathbf{p}) - D(\mathbf{q})| > 1]). \end{aligned} \tag{1}$$

Function 1 consists of three terms. The first term sums all costs C over all pixels $\mathbf{p}$ and disparities $D(\mathbf{p})$. The function $C(\mathbf{p}, D(\mathbf{p}))$ may be any cost function suitable. In the following, Census [15] is being used as matching cost, as it is well suited for hardware realization. It has been shown by Hirschmüller et al. [16] that Census delivers comparatively good results for scenes with difficult lightning.

Term two and three introduce a local smoothness constraint. Due to noise or untextured regions low disparity values will usually have smaller costs than the correct ones. Term two penalizes disparity changes of one pixel with a small $P1$ in order to allow slanted curves. $P2$ on the other hand is being set to a high value to allow disparity discontinuities. Depth discontinuities usually coincide with intensity changes which may be detected by calculating the intensity gradient. Therefore $P2$ is defined as $\frac{P2_{const}}{|p_M - p_B|}$.

The base algorithm aggregates the costs for the disparity from at least eight directions in a data volume S, from which the final disparity is selected.

$$S(\mathbf{p}, d) = \sum_{\mathbf{r}} L_{\mathbf{r}}(\mathbf{p}, d). \tag{2}$$

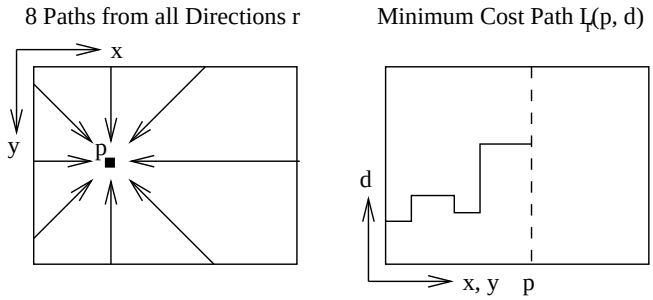

Figure 1. Aggregation of costs in disparity space. [1]

Therefore the disparity for each pixel corresponds to the minimum cost which is set after all directions have been calculated.

$$D_{SGM}(\mathbf{p}) = \underset{d}{\operatorname{argmin}}\, S(\mathbf{p}, d). \tag{3}$$

$D_{SGM}(\mathbf{p})$ equals the index where the sum of the cost from all 8 directions reaches its minimum. The aggregation step is a mathematical formulation to the idea that at least one direction contributes sufficient information to dissolve any ambiguity and depth discontinuity. This is the reason why SGM works well in contrast to the Dynamic Programming approach, where only one direction is being used.

The new memory efficient SGM method recognizes the original intention of the SGM algorithm which states that the correct disparity value is either represented by the cost $L_{\mathbf{r}}(\mathbf{p}, d)$ from at least one direction $\mathbf{r}$ or found by chance. Figure 2 shows the costs of the eight paths from different directions at pixel $\mathbf{p}$ for all disparities. In the majority of cases the minimum of S equals the minimum of all directions $L_0 \dots L_7$. At depth discontinuities some directions will have a different local minimum, but the global minimum of the sum still holds the true disparity. It is also possible that none of the directions is approximating the correct disparity. This will likely happen in textureless regions and either a wrong disparity is selected or the correct one by random as shown in figure 2.

The novel eSGM method requires to search for the minimum of each direction separately. The index i is saved for each pixel and direction in $\mathcal{I}(\mathbf{p}, \mathbf{r})$.

$$\mathcal{I}(\mathbf{p}, \mathbf{r}) = \underset{i}{\operatorname{argmin}}\, L_{\mathbf{r}}(\mathbf{p}, i). \tag{4}$$

The disparity value is then selected from

$$D_{eSGM}(\mathbf{p}) = \underset{t \in \mathcal{I}(p,r)}{\operatorname{argmin}}\, S(\mathbf{p}, t). \tag{5}$$

Therefore only eight indexes and values need to be saved. All other values are discarded which relaxes the memory constraint from $\mathcal{O}(WHD)$ to $\mathcal{O}(WH)$. In a mathematical sense this approach is not justified. It may be argued that the minimum of a sum does not have to be equal the minimum of a subset which is of course true. It is also questionable wether or not the disparities are pruned to early. Two cases are lined out to show that from a conceptual view, SGM and eSGM still follow the same intend.

1. Whenever SGM returns a disparity value that is also a local minimum, then this value will also be calculated by the eSGM method.
2. Whenever SGM returns a disparity value that is not a local minimum, then this value may diverge from the disparity found by the eSGM method.

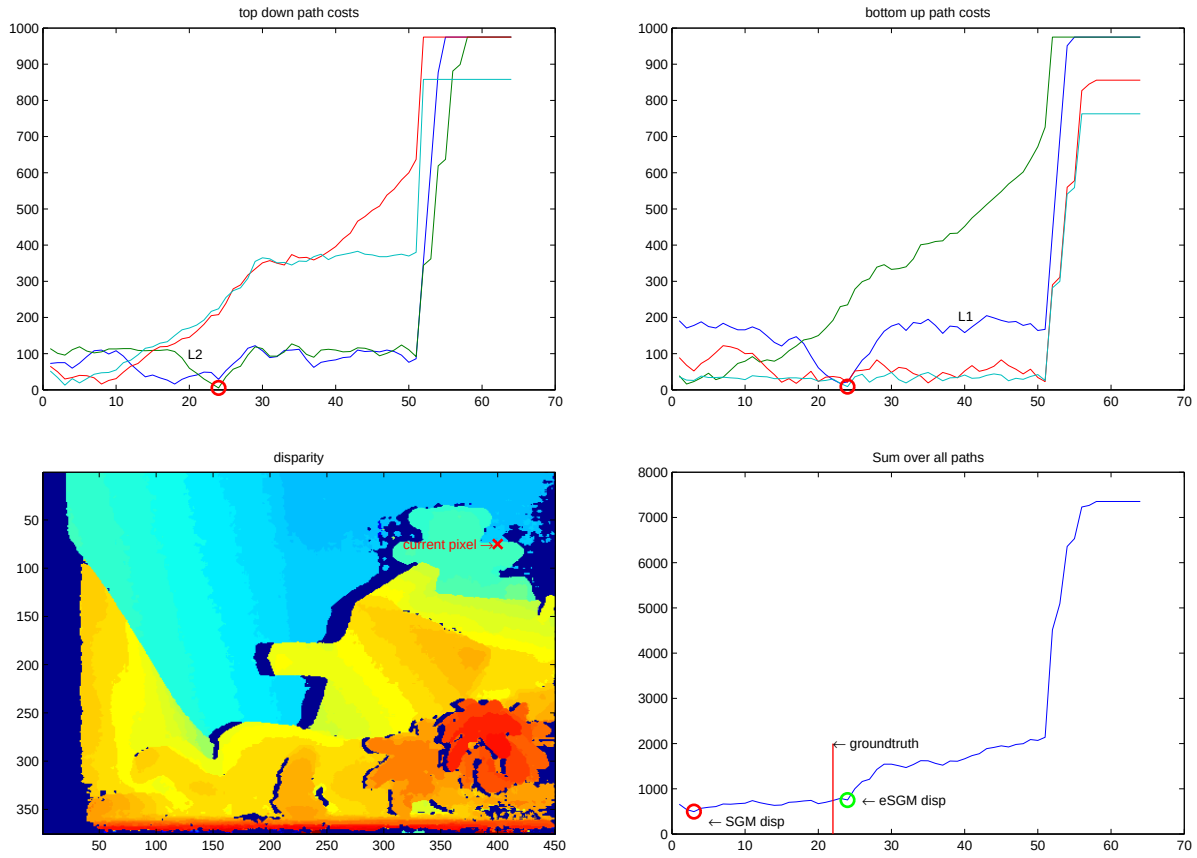

Figure 2. Costs for all disparities at a certain pixel. The first two images show the individual costs from the paths of eight different directions that are summed to S (lower, right image). The minimum of S may differ from the minima of the pathwise costs.

The first case holds true because $D_{SGM}(\mathbf{p})$ is by definition in equation (3) a global minimum. Let $k = D_{SGM}(p)$ be the index that fulfils equation (3). Therefore it follows that

$$\exists r' \in \mathcal{R}, \quad \text{with } \underset{k}{\operatorname{argmin}}\, L_r(\mathbf{p}, k). \tag{6}$$

In case the base SGM algorithm delivers a disparity value which is not derived from a local minimum, then the disparity is either wrong (not identical to the groundtruth) or correct by random. The value can not be usefully interpreted by means of the SGM algorithm since. In contrast, the eSGM method will not allow such a derivation. It will always select a minimum that is justified by at least one direction and has therefore an even higher possibility to detect the correct disparity value compared to SGM. Figure 2 gives an example how eSGM finds a disparity value that is closer to the groundtruth than SGM does, since only L1 and L2 represent a reasonable confidence.

The realization of the eSGM method has to be done in three passes.

1. The first pass is identical to the SGM method. As outlined in equation (4), only the index and the sum at each index have to be stored. For sub-pixel interpolation, the adjacent values $S(i_{\pm 1})$ have to be saved. Figure 3 shows the values that have to be stored for one random pixel.

2. The second pass computes the remaining four directions from the bottom up. The already found values from the first run are completed and an intermediate minimum may be calculated. This will free the memory locations from the first pass. The freed memory may be used to store the new indexes and values from the bottom up direction.

3. The aggregation is completed by a third pass that follow the same top down directions from the first pass. This run is needed to update the costs at the minima that where identified in the second pass. The final minimum of each pixel can be selected among these four minima and the intermediate result of the second pass.

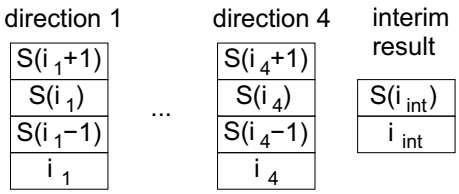

Figure 3. Definition of 18 data elements that need to be stored for each pixel in the eSGM method. [1]

Thus the amount of temporary memory is just

$$M_{esgm} = w \times h \times 18 + 3 \times w \times d_{max} + d_{max}. \tag{7}$$

The computational effort of eSGM is increased by 50% in comparison to SGM due to the necessary third pass. In contrast to dynamic programming approaches, it is not possible to detect occlusions directly. This is due to the fact that the SGM method aggregates the cost from directions which are not aligned to the epipolar line. Therefore occlusion detection is done by comparing the disparity of the Left-Right- and Right-Left-matches. If the disparities do not match, the value is invalidated. A diagonal search as described in [7] is also possible but not feasible for FPGA implementations.

3. RESULTS

Programmable hardware has been used for stereo matching based on the SGM algorithm in [12, 11]. FPGAs have a low power consumption and high computational power that make them suitable for energy aware embedded systems that demand realtime image processing. A FPGA implementation of the original SGM algorithm on a high-end FPGA board matches two pairs of 320x200 images at 27 fps using ZSAD as matching cost [11]. During the FPGA implementation efforts it became clear that the required memory bandwidth is the limiting factor for increased image resolutions or higher frame rate respectively. The novel eSGM method overcomes this limitation at increased computational cost. The additional third pass uses the same logic resources as the first two rounds avoiding extra logic occupation. In contrast to SGM, the logic pipelines are now fed constantly permitting an overall higher output data rate.

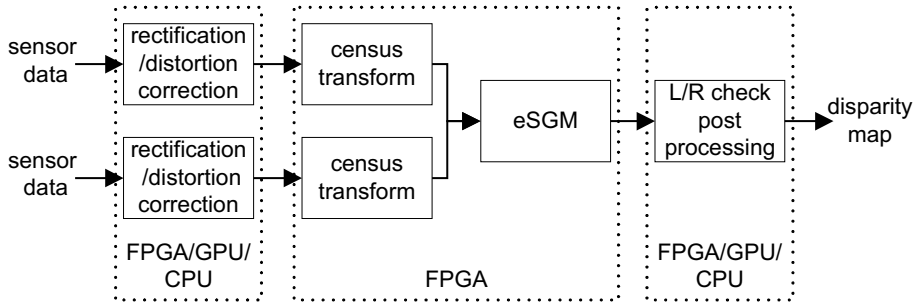

Figure 4. Overview of the functional blocks and the underlying heterogeneous hardware platform.

Rectified images are expected as input data for the stereo matching. In order to allow as much logic resources as possible to be dedicated to the matching routines, rectification is currently performed on a second FPGA. Alternatively a GPU or CPU may be used for rectification. It has to be noted that the last two are either not available or not fast enough on mobile robots.

Unlike other FPGA implementations [11] we are using Census as matching cost, because of above stated advantages. The window size of the Census transform can be set to any arbitrary size. For our realtime setup, the census size was

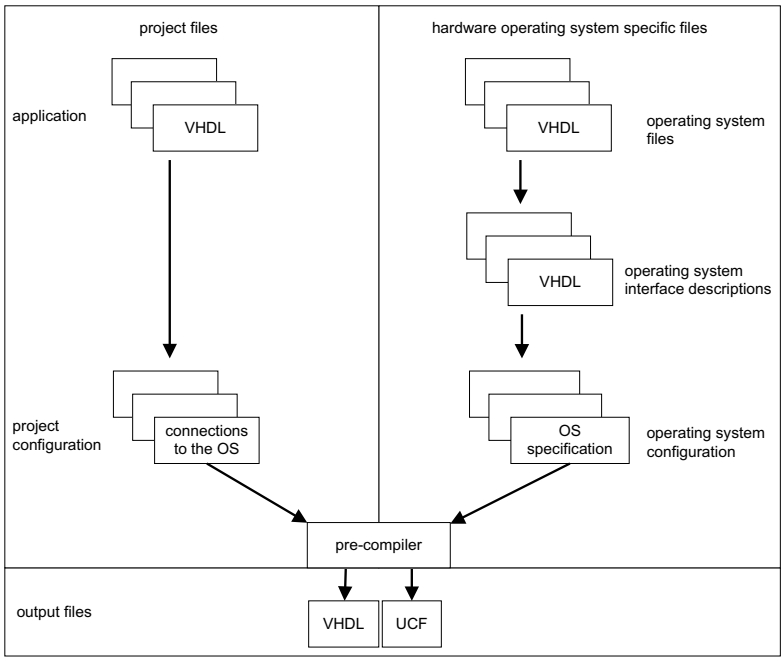

Figure 5. Workflow of the hardware operating system.

chosen to be 5×5, since we found that it gives the best trade-off between hardware impact and output result based on the benchmark.

All hardware components are designed with the hardware description language VHDL, following a hardware operating system concept (HW-OS). The objective of the HW-OS is to provide a platform independent hardware description that can be synthesized by standard tools. Equivalent to the concept of a software operating system, the HW-OS defines standard interfaces to physical hardware resources (external memory or communication interfaces) and common modules like communication protocol stacks. It also arbitrates shared resources and makes them transparent to the application. The advantage of the HW-OS results from the ability to exchange components relatively easy and to reuse proven IP-Cores in new projects. A precompiler is necessary to alleviate the insufficiency of the VHDL-language. It takes a project description and the VHDL-source code as input and generates a new VHDL file that can be used in the following synthesis process. In contrast to tools like C-to-VHDL compiler or other high-level code entry tools like LabView or MATLAB for FPGA development, it is still necessary to describe the application in VHDL. Encoding the algorithm in VHDL might be seen as a burden but is until today the only way to develop an optimal FPGA design.

3.1 Stationary System

The eSGM core is customizable to any disparity and image resolution which must be a power of two. Several FPGA hardware platforms are currently in use. The stationary setup consists of a Xilinx Virtex 5 FPGA (XC5VSX95T) with a PCIe interface to the image sensors and a GPU. The GPU is used for pre- and post-processing as shown in figure 4. Based on this hardware environment, a stereo image pair with a resolution of 1024×1024 pixel and a maximum of 64 disparity steps can be processed at nearly 10 Hz. VGA sized image pairs are processed at 33 Hz with the same disparity range. The consistency check doubles the execution time. If needed, a second independent matching core may be instantiated in a more powerful FPGA or a second FPGA-card may be installed.

3.2 Mobile System

A Spartan FPGA-kit is used to embed the stereo matching system in mobile systems. The kit consists of two Xilinx Spartan 6 LX150 devices that are used for both stereo matching and image rectification as shown in figure 6. The image data is captured directly by the FPGA and forwarded to the rectification unit. The rectification and distortion correction is done in realtime and the output sent over a localbus to the second FPGA. The slave device is capable of processing the image data at 25 Hz with image sizes up to 512×512 pixel and a depth resolution of 64 pixel. The resulting depth

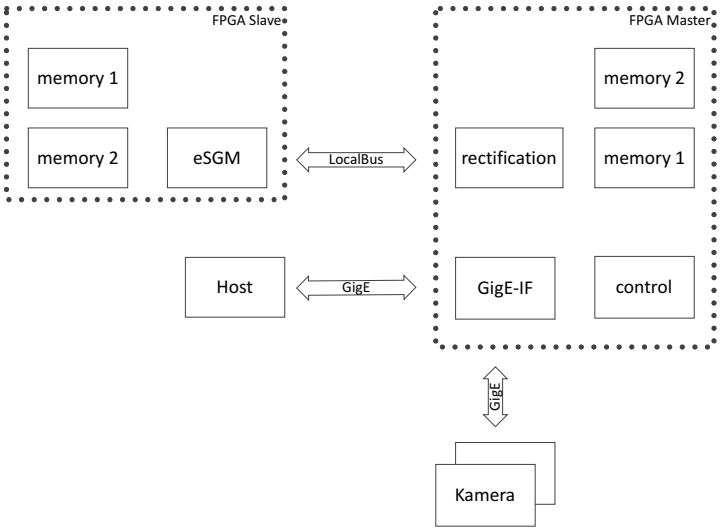

Figure 6. Functional overview of the FPGA-kit.

map is also communicated via gigabit Ethernet to the attached host system. Occlusion detection is done by a consistency check and doubles the execution time if no additional hardware resources are available. The FPGA-kit has a very small mechanical outline and a very low energy consumption that makes it ideal for UAVs.

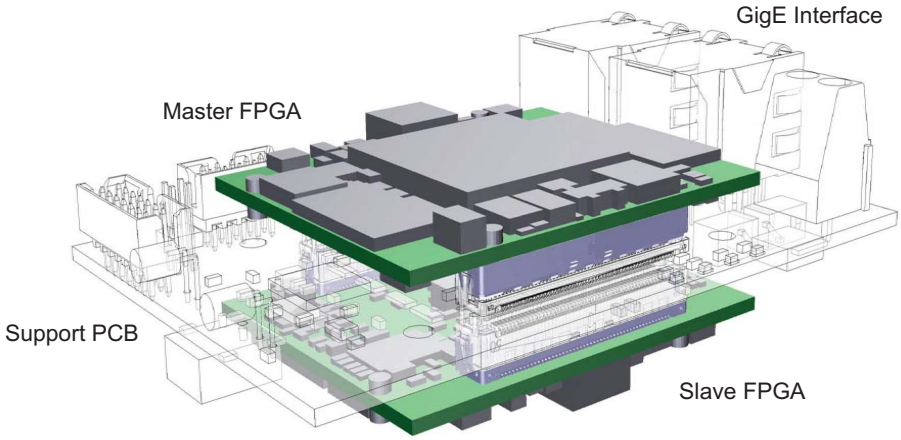

Figure 7. CAD overview of the FPGA-kit.

3.3 Evaluation on Middlebury Datasets

The implementations of SGM and eSGM where tested against each other on the standard test set [4, 17, 18]. The same parameter settings were used for SGM and eSGM and for all images. The consistency check was used for identifying occlusions. The disparity images of SGM and eSGM are almost identical.

Table 1. Errors in non-occluded areas of the Middlebury datasets using the standard threshold of one pixel.

Algorithm	Tsukuba	Venus	Teddy	Cones	Average error
AdaptingBP [19]	1.11	0.10	4.22	2.48	4.23
eSGM (Census)	3.25	0.60	5.38	2.87	7.16
SGM (Census)	3.26	0.61	5.41	2.85	7.17
SGM (HMI)[7]	3.26	1.00	6.02	3.06	7.50

All pixels that differ by more than one pixel from the ground truth are counted as error. Table 1 shows the results over all non-occluded pixels. The performance of SGM and eSGM is very similar which confirms that the eSGM method produces virtually the same output as the SGM method. The original SGM implementation is listed as SGM (HMI) in the table. The difference to the eSGM version is the matching cost. In contrast SGM (HMI) uses Mutual Information which is slightly inferior [16].

4. DISCUSSION

The reduction of the external memory bandwidth and capacity required to store the cost volume S leads to a new bottleneck. Now it is the buffer that stores $3 \times w \times d_{max} + d_{max}$ elements of the four paths shown in figure 1 which becomes the limiting factor. FPGAs that are typically equipped with ≈ 500 KByte to ≈ 2 MByte total on-chip memory are sufficient to store the pathwise costs for relatively small image sizes. An alternative for future improvements is the work of Banz et al. [12], who processed to process multiple scanlines in parallel. Within each block, the path costs are stored within three shift registers of different length. The costs are passed to the succeeding line. Only the path costs of the last scanline within each block needs to be transferred to off-chip memory. This technique reduces the amount of external memory bandwidth by a factor that depends on the number of scanlines within each block.

5. CONCLUSION

A realtime stereo matching system was presented that is capable of processing VGA sized images at 25 Hz or 33 Hz respectively. The matching algorithm calculated high quality depth maps with up to 64 pixel resolution that are useable in real world environments. The hardware operating system permits a partitioning of the subroutines to different hardware platforms depending on the available resources.

REFERENCES

[1] Hirschmüller, H., Ernst, I., and Buder, M., "Memory efficient semi-global matching," in [*International Annals of Photogrammetry and Remote Sensing*], (2012).

[2] Veksler, O., "Fast variable window for stereo correspondence using integral images," in [*Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*], **1**, I–556–I–561 vol.1 (2003).

[3] Yoon, K. J. and Kweon, I. S., "Adaptive support-weight approach for correspondence search," *Ieee Transactions on Pattern Analysis and Machine Intelligence* **28**(4), 650–656 (2006).

[4] Scharstein, D. and Szeliski, R., "Middlebury stereo website." www.middlebury.edu/stereo (2010).

[5] Wang, L., Liao, M., Gong, M., Yang, R., and Nister, D., "High-quality real-time stereo using adaptive cost aggregation and dynamic programming," in [*3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*], 798–805, IEEE Computer Society, Washington, DC, USA (2006).

[6] Yang, Q., Engels, C., and Akbarzadeh, A., "Near real-time stereo for weakly-textured scenes," (2008).

[7] Hirschmüller, H., "Stereo processing by semiglobal matching and mutual information," *IEEE Trans. Pattern Anal. Mach. Intell.* **30**(2), 328–341 (2008).

[8] Rosenberg, I. D., Davidson, P. L., Muller, C. M. R., and Han, J. Y., "Real-time stereo vision using semi-global matching on programmable graphics hardware," in [*SIGGRAPH*], (2006).

[9] Gibson, J. and Marques, O., "Stereo depth with a unified architecture gpu," in [*IEEE CVPR*], (2008).

[10] Ernst, I. and Hirschmüller, H., "Mutual information based semi-global stereo matching on the gpu," in [*ISVC*], **LNCS 5358, Part 1**, 228–239 (December 2008).

[11] Gehrig, S., Eberli, F., and Meyer, T., "A real-time low-power stereo vision engine using semi-global matching," in [*ICVS*], **LNCS 5815**, 134–143 (October 2009).

[12] Banz, C., Hesselbarth, S., Flatt, H., Blume, H., and Pirsch, P., "Real-time stereo vision system using semi-global matching disparity estimation: Architecture and fpga-implementation," in [*IEEE Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*], (July 2010).

[13] Xing, M., Xun, S., Mingcai, Z., Shaohui, J., Haitao, W., and Xiaopeng, Z., "On building an accurate stereo matching system on graphics hardware," in [*Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*], 467–474.

[14] Steingrube, P., Gehrig, S., and Franke, U., "Performance evaluation of stereo algorithms for automotive applications," in [*ICVS*], **LNCS 5815** (October 2009).

[15] Zabih, R. and Woodfill, J., "Non-parametric local transforms for computing visual correspondence," in [*Proceedings of the third European conference on Computer Vision (Vol. II)*], 151–158, Springer-Verlag New York, Inc., Stockholm, Sweden (1994).

[16] Hirschmüller, H. and Scharstein, D., "Evaluation of stereo matching costs on images with radiometric differences," *IEEE TPAMI* **31**, 1582–1599 (September 2009).

[17] Scharstein, D. and Szeliski, R., "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *IJCV* **47**, 7–42 (April-June 2002).

[18] Scharstein, D. and Szeliski, R., "High-accuracy stereo depth maps using structured light," in [*IEEE CVPR*], **1**, 195–202 (June 2003).

[19] Klaus, A., Sormann, M., and Karner, K., "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure," in [*ICPR*], (2006).