A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching

Stefan K. Gehrig[1], Felix Eberli[2], and Thomas Meyer[2]

[1] Daimler AG Group Research, 71059 Sindelfingen, Germany
[2] Supercomputing Systems AG Technoparkstr. 11, Zuerich, 8005 Zuerich, Switzerland

Abstract. Many real-time stereo vision systems are available on low-power platforms. They all either use a local correlation-like stereo engine or perform dynamic programming variants on a scan-line. However, when looking at high-performance global stereo methods as listed in the upper third of the Middlebury database, the low-power real-time implementations for these methods are still missing. We propose a real-time implementation of the semi-global matching algorithm with algorithmic extensions for automotive applications on a reconfigurable hardware platform resulting in a low power consumption of under 3W. The algorithm runs at 25Hz processing image pairs of size 750x480 pixels and computing stereo on a 680x400 image part with up to a maximum of 128 disparities.

1 Introduction

3D perception is a crucial task both for automotive and for robotics applications. Besides time-of-flight systems such as RADAR or LIDAR, stereo cameras are a very popular and inexpensive choice to perform this task.

Stereo vision has been an active area of research for decades. A few years ago, a benchmark to compare stereo algorithms with respect to (w.r.t.) accuracy was established [1]. This benchmark database ranks the current published stereo algorithms. Among the top-performing algorithms in this database, we found semi-global matching (SGM) [2] to be the most efficient. We will focus on this algorithm for the remainder of this paper.

Roughly speaking, SGM performs an energy minimization in a dynamic-programming fashion on multiple 1D paths crossing each pixel and thus approximating the 2D image. The energy consists of three parts: a data term for photo-consistency, a small smoothness energy term for slanted surfaces that change the disparity slightly (parameter P_1), and a smoothness energy term for depth discontinuities (parameter P_2). Based on this algorithm, we introduce the first real-time global stereo implementation besides dynamic programming that runs at 25 Hz with less than 3W power consumption.

This paper is organized in the following way: Prior work of the field is presented in Section 2. Section 3 explains our system design. The next section describes implementation details to obtain real-time performance. Results of the hardware setup can be found in the Section 5. Conclusions and future work comprise the final section.

2 Related Work

Today, several real-time stereo vision systems are available on low-power platforms. One of the first stereo hardware engines is described in [3]. Recently, the census-based stereo system by the company Tyzx became popular [4]. Phase-based correlation has also been implemented on a field-programmable gate-array (FPGA) [5]. In this work, also a nice overview of existing reconfigurable stereo engines is provided. Dynamic Programming as a global method on a single scan line has also been implemented on an FPGA [6] .

In the automotive field, the car manufacturer Toyota delivers a stereo object detection system for the Lexus LS460 [7] produced by Denso. Autoliv cooperates with Sarnoff in the field and offers an ASIC solution for correlation stereo called Acadia [8]. The company MobilEye offers a stereo engine in its latest automotive ASIC EyeQ2 [9]. All these available stereo engines operate with local correlation-variants.

For some of the top-performing stereo algorithms near real-time implementations on the graphics card (GPU) exist. Semi-global matching runs at 13Hz for QVGA size images [10]. Belief propagation was shown to be real-time capable on the GPU for small images [11]. The power consumption of these GPUs is well above 100W.

Despite the enormous amount of work on real-time stereo vision, there is still a need for a high-performance, low-power implementation of a global stereo method. We close this gap with our contribution.

3 System Design

3.1 Design Considerations

SGM has to traverse many paths along the image for all pixels and disparities. The paths all start at the image margin so running the algorithm in one scan is not possible. The main limitation is the number of memory accesses for the accumulated costs for every pixel, disparity, and path. In order to fit modern FPGA resources we decided to run the algorithm on 8 paths, computing 4 paths simultaneously (see Figure 4) keeping the memory bandwidth on a manageable level. The intermediate results are stored into an external memory during the first scan, and then read back during the second scan for accumulation. This memory bandwidth reduction and the choice of the FPGA as computing device yield the desired low-power engine.

The core of the SGM algorithm is the cost calculation along the 8 paths. Here, parallelization is limited since it is necessary to have the result of the previous pixel in the path available. However, the costs for all disparities at the current pixel only depend on results from the previous pixel along the path (see Figure 5). We instantiate this part 16 times in parallel on the FPGA.

We consider two 750x480px input greyscale images with 12 bits per grey-value as input. Due to mounting tolerances and missing stereo overlap, the actual region of computation is limited to 680x400px. In automotive and robotics

applications with these image sizes, a disparity range of 0 through 127 suffices. For larger disparities, similarity is often hard to establish due to the significant different view angles. Ideally, SGM would be computed everywhere at full resolution. We decided to reduce the image size by a factor of 2 in width and height for the full image and to run a second SGM at full resolution for a smaller region of interest (ROI) where the full resolution is needed. This subsampling step is mandatory to keep the necessary data for path accumulation in internal memory on automotive FPGAs. The ROI can be changed on a frame-by-frame basis.

For the outdoor scenario with uncontrollable lighting conditions, a different metric to the ones proposed in [2] has to be used. We have to cope with different brightness in the left and right image and with vignetting effects due to inexpensive lenses. The proposed similarity criterion mutual information is sensitive to vignetting artifacts [12]. We slightly deviated from a pixel-wise matching cost using a small 3x3 correlation window to minimize the effect of foreground fattening. The mean-free sum-of-absolute differences (ZSAD) turns out to be very efficient and robust for the task at hand.

The experimental system explained below is a PC-based system used to validate the design and to test different SGM variants. The target system is derived from the experimental system, has limited parametrization options and runs in an embedded system. The SGM building blocks were implemented in VHDL on the FPGA from scratch.

3.2 Experimental System

The experimental system is built on a general purpose personal computer (PC) with x86 CPU. The stereo image data is received over a frame grabber from the cameras. The necessary pre-processing step to properly align the cameras, rectification, is done in software and only the SGM algorithm is executed on the PCIe-FPGA card (see Figure 1).

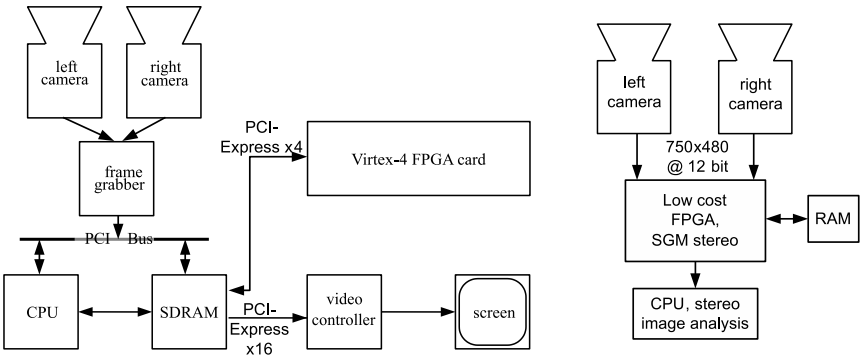

Fig. 1. Hardware blocks of the experimental system (left) and the target system (right).

Figure 2 shows the overview of the blocks running on the FPGA. The optional Gauss filter alleviates negative effects of a slight stereo decalibration [13]. The critical blocks w.r.t. latency and resources are the correlation block, the SGM path calculator and adder with the associated external RAM access controller. The pre-processing (Gauss, scaler) and post-processing (median, sub-pixel interpolation, LR-RL-combination to detect occlusions) blocks are standard operations and are only performed once per pixel. Thus, they are not time-critical blocks.

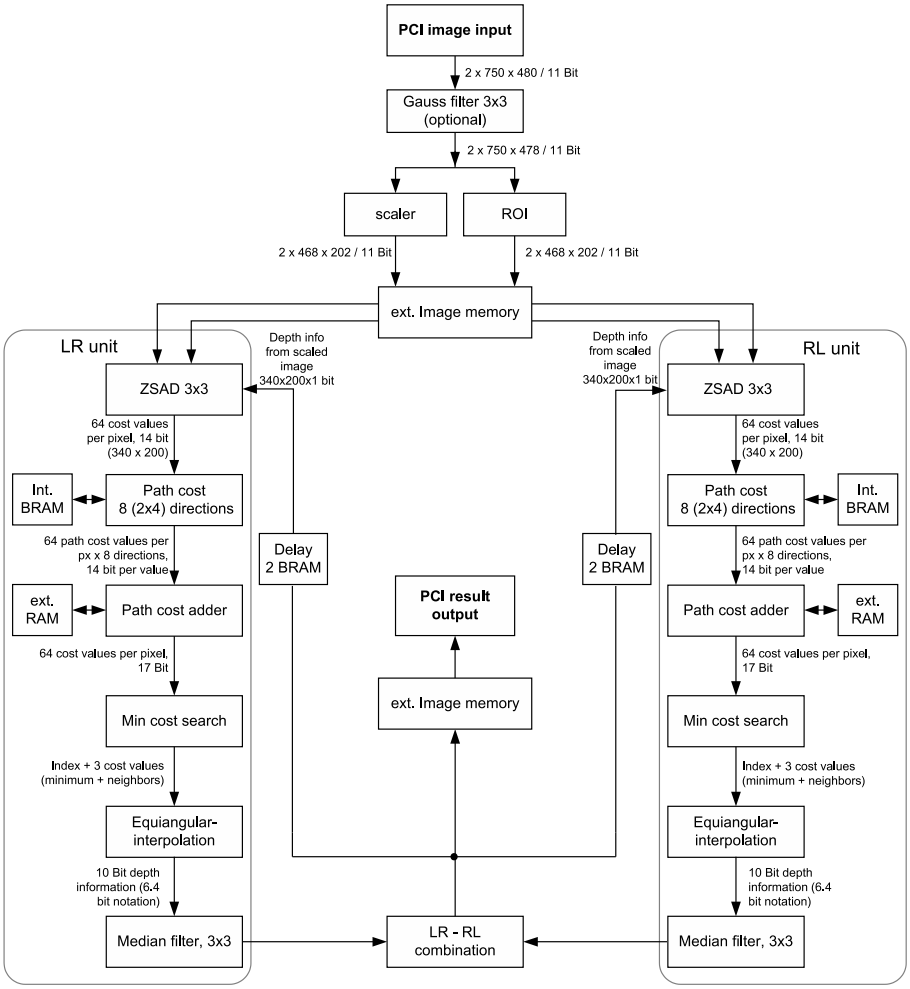

Fig. 2. System Overview. The critical blocks w.r.t. latency and resources are ZSAD correlation, path calculation, and memory communication.

3.3 Target System

In the target system, the SGM algorithm works in an embedded system with the stereo cameras directly connected to the low-cost FPGA. The FPGA executes the complete pre-processing of the input images, SGM stereo calculation and post-processing. The input images and the stereo result are then forwarded to a CPU for further analysis of the raw result images (see Figure 1). Differences to the experimental system are:

- The rectification is done by the FPGA, not in software
- The experimental system has two SGM engines in the high-end FPGA for a full right-left check, i.e. computing SGM a second time with the right image as the reference image. The target system only contains one of these, therefore the result uses a simplified right-left-check to detect occlusions.

4 Implementation

4.1 FPGA Platform for the Experimental System

The experimental system uses a Xilinx Virtex-4 FPGA on a 64-Bit PMC-Module from Alpha Data (ADM-XRC-4FX module with a Virtex-4 FX140 FPGA). The card contains a PCI/PCI-X to localbus bridge including four DMA-units. Apart from the localbus connection, the FPGA has four independent 32-bit DDR2 blocks with 256 MB each available. Virtex-4 FPGAs have an almost identical logic structure compared to the low-end Spartan-3 FPGAs which are used on the target system (Spartan-3a 3400DSP).

The image data is transferred with DMA from and to the PC main memory. The FPGA uses three external 32-bit DDR2 memories, one for image and result storage and one for each of the two stereo processing engines as temporary storage.

The image data is transferred from the PC and filtered to form the internal image format on which the algorithm works. These are two stereo image pairs, a scaled and a ROI image pair which are stored in external memory. In the first step, the core algorithm starts to read a image pair top down, line by line from left to right. In this order, the ZSAD correlation is calculated and forwarded to the SGM core.

4.2 Main SGM Blocks

The ZSAD core is fully pipelined and produces one cost value per clock cycle. To match the throughput of the SGM core, four such ZSAD units work in parallel. The line buffers for the input images, specifically the line buffers for the search image had to be designed to deliver the required bandwidth for four correlation units. As a first step in ZSAD, the means of both the reference and the search 3x3 pixel areas are calculated (see Figure 3). The 3x3 pixels of the two areas are subtracted from each other, the means are subtracted and then the absolute

values are summed to form a cost value. To reduce the number of bits for the cost value, the result is divided by 8, i.e. 3 bits are truncated. This is done to match the 14 bit cost width of the implemented SGM core.

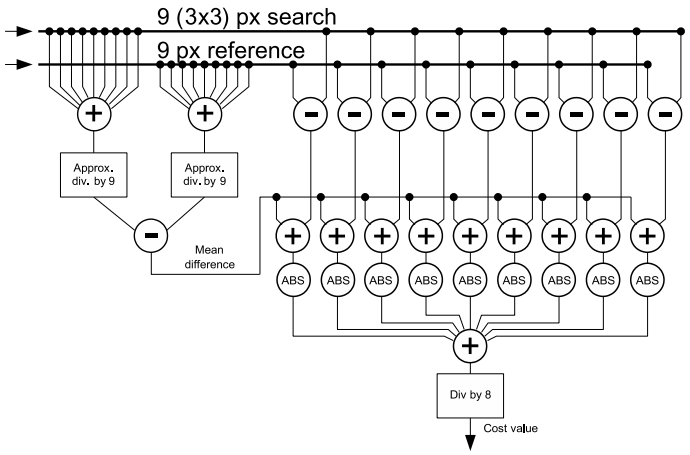

Fig. 3. ZSAD calculation core. The ZSAD cost is calculated for every pixel and disparity.

The SGM core calculates four SGM directions: one horizontal to the left and three vertical down, including two diagonal down paths (see Figure 4).

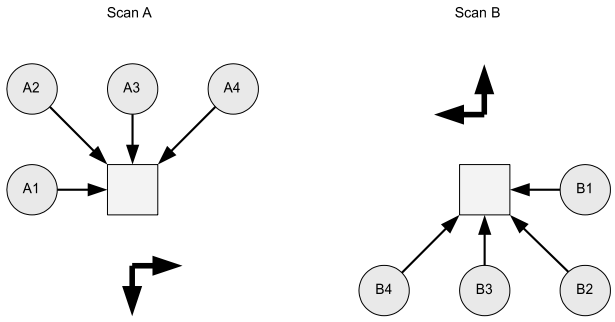

Fig. 4. Processing the 8 path accumulations in two scans.

The basic SGM operation is shown in Figure 5, where the smoothness constraint via P_1 and P_2 is incorporated ($L(p-r,d)$ is the accumulated cost from the previously visited pixel p at disparity d, $C(p,d)$ is the ZSAD value). The history of the SGM path costs is kept inside the FPGA in fast local memory. Only the accumulated costs of these first four paths are stored in external memory

which consists of an array of 64 values with 16 bits each for every pixel in the image (340x200). This external 16 bit restriction is also responsible for the 14 bit restriction for each single SGM-path and the 17 bit restriction for the final accumulated costs over eight paths

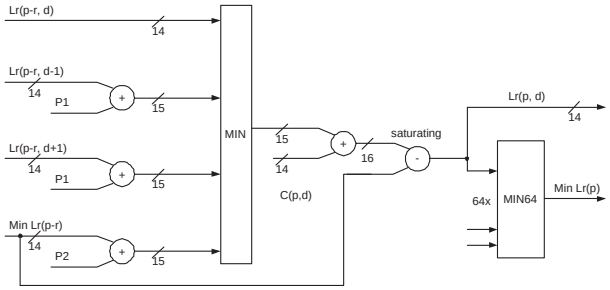

Fig. 5. The core processing unit calculating the lowest cost path. This path calculation is performed for every pixel, disparity, and path.

The second step does the same operation backwards, reading the image pair from bottom to top, again calculating the same correlation costs, but in reverse order. This allows us to calculate the missing four SGM paths in the inverse directions. During this step, the accumulated costs from the first step are read back from external memory and accumulated with the currently calculated paths to form the final costs over eight paths.

For each pixel, the minimum cost of 64 values is then calculated which yields a 6 bit index value. To further enhance the depth precision, an additional interpolation is done which expands the result to a total of 10 bits. We use the equiangular interpolation as suggested by [14] since it yields better sub-pixel precision for cost metrics based on absolute differences such as ZSAD compared to the standard parabola fit. After the interpolation, a 3x3 median filter is performed similar to [2] before the data is verified in the right-left check.

During the first and second scan, the algorithm works on the down-scaled image and produces an overview result of the whole input. Since the image is scaled by a factor of two, the 64 disparities calculated correspond to 128 disparities in the original image. During the third and fourth scan, the algorithm works on an unscaled ROI the same way as for the scaled overview to further enhance the precision of a selected part of the image. False matches in the ROI caused by the limited disparity range are avoided in the following way: We set the correlation costs for disparity 63 to 0 whenever 63 (31.5 for the scaled image) or a larger disparity has been established in the overview image. This drives the ROI SGM result towards the overview SGM result for large disparities. The combination of both results is straightforward.

In our experimental system, two SGM engines are instantiated, one with the left image as reference, one with the right image (full RL-Check). The disparity

is invalidated if the resulting disparities differ by more than 1. On the target system, we use a fast Right-To-Left consistency check traversing the accumulated costs at a 45° angle, looking again for the minimum cost and invalidating the disparity if the values differ (fast RL-Check) [2]. Thus, we only need one SGM engine.

5 Results

Computational Speed: Our stereo engine runs with clock speeds of up to 200MHz on the Virtex4 FPGA. For the low-cost FPGA clock speed of 133 MHz, we obtain a maximum frame rate of 27Hz. On the Virtex 4, a full RL-Check, i.e. computing SGM twice per disparity map, is used and two disparity images of size 340x200 per image pair are calculated. The same speed is achieved on the low-cost FPGA using the fast RL-Check. The high end FPGA has twice the performance at the same clock speed because it contains the SGM engine twice to calculate the left- and right-image referenced results at the same time.

Resources: The FPGA resources for the Virtex-4 experimental system, including DDR2 controllers, are about 60000 LUT4 logic elements for two SGM engines and 150 Block-RAMs (300kB) of internal memory. For the target system, the resources are halved using the fast RL-Check, i.e. only one SGM engine.

Power Consumption: Based on our implementation on the low-cost FPGA, the FPGA has a power consumption of less than 2W. Up to one extra Watt can be generated due to bit toggles when transferring data to the external memory.

Algorithmic Performance: We computed results on the Middlebury test data with our ZSAD similarity criterion. We obtained 5.86% wrong pixels (disparity deviates by more than one pixel to ground truth based on all pixels) for Tsukuba, 3.85% for Venus, 13.28% for Teddy and 9.54% for the Cones image pair based on all pixels with ground truth. Invalid pixels were interpolated via propagation of the last valid disparity value. These results are slightly worse than the SGM results with mutual information [2] which is not surprising since the similarity criterion has been selected for robust performance under difficult imaging conditions which are not present in the Middlebury database.

Real-World Results: We show a few results from our experimental system. In the van scene (Figure 6) with rather favorable lighting conditions and excellent lenses, our ZSAD SGM variant and the literature SGM (Birchfield-Tomasi(BT) and Hierarchical Mutual Information(HMI) [2]) versions perform comparably well with more invalid pixels for the literature variants. For the construction site scene (Figure 7), the situation is very different. Here, the lenses have a strong vignetting artifact that cannot be compensated with global statistics such as mutual information. Hence the HMI-SGM performs poorly. The BT metric fails, especially on the road, because the image pair has different brightness in addition. The ZSAD-SGM yields a dense disparity map with correct occlusion detection and no noticeable outliers. A more detailed and stringent evaluation of the algorithm and a comparison to other real-time stereo algorithms can be found in [15].

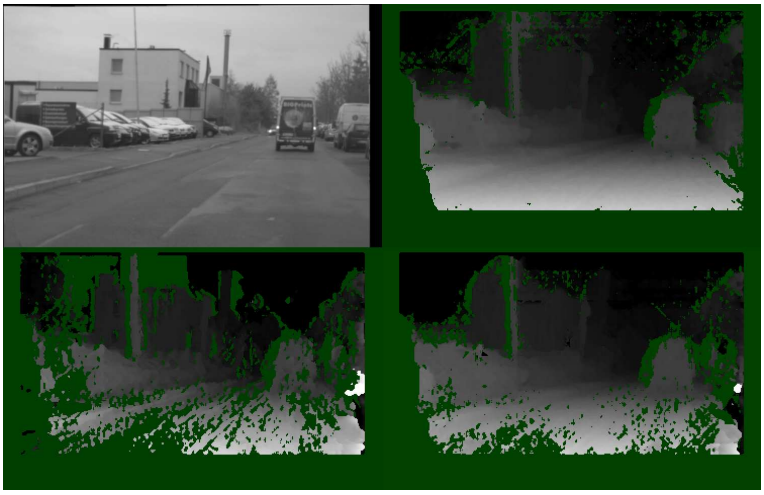

Fig. 6. Resulting Disparity Maps for the Van Scene (top left). SGM with ZSAD (top right) works well, SGM with BT (bottom left) has more invalid pixels while SGM with HMI works comparably to ZSAD (bottom right). Small disparities are mapped to black, invalid pixels are marked dark green.

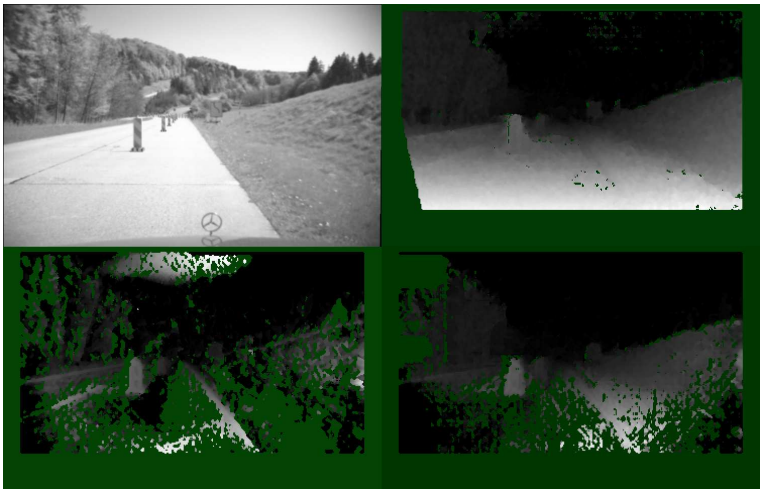

Fig. 7. Resulting Disparity Maps for the Construction Site Scene (top left). SGM with ZSAD (top right) works well while SGM with BT (bottom left) and SGM with HMI (bottom right) fail. The image pair has a strong vignetting effect and different brightness. Small disparities are mapped to black, invalid pixels are marked dark green.

6 Conclusions and Future Work

We have introduced a real-time low-power global stereo engine. The results using SGM are very encouraging. The key design choices to obtain real-time performance are subsampling and result reuse for full resolution computation, parallelization of the most time-consuming block (path calculator), and minimizing the external memory bandwidth by combining 4 paths in one scan. We expect this engine to be the basis for many future camera-based driver assistance and robotic systems. Ongoing work is the further reduction of the FPGA resources and the incorporation of issues such as calibration sensitivity [13] and additional robustness in difficult weather conditions such as rain and backlight.

References

1. Scharstein, D., Szeliski, R.: Middlebury stereo vision and evaluation page http://vision.middlebury.edu/stereo.
2. Hirschmueller, H.: Accurate and efficient stereo processing by semi-global matching and mutual information. In: Proceedings of Int. Conference on Computer Vision and Pattern Recognition 05, San Diego, CA. Volume 2. (June 2005) 807–814
3. Konolige, K.: Small vision systems. In: Proceedings of the International Symposium on Robotics Research, Hayama, Japan. (1997)
4. Woodfill, J.I., et al.: The tyzx deepsea g2 vision system, a taskable, embedded stereo camera. In: Embedded Computer Vision Workshop. (2006) 126–132
5. Masrani, D.K., MacLean, W.J.: A real-time large disparity range stereo-system using fpgas. In: Asian Conference on Computer Vision (ACCV). (2006) 42–51
6. Sabihuddin, S., MacLean, W.J.: Maximum-likelihood stereo correspondence using field programmable gate arrays. In: Int. Conference on Computer Vision Systems (ICVS), Bielefeld, Germany. (March 2007)
7. Tech-News: Toyota' lexus ls 460 employs stereo camera (viewed 2009/04/15) http://techon.nikkeibp.co.jp/english/NEWS_EN/20060301/113832/.
8. Sarnoff-Inc.: The acadia video processors - acadia pci (viewed 2009/07/15) http://www.sarnoff.com/products/acadia-video-processors/acadia-pci.
9. MobilEye: Eye q2 system - vision system on a chip (viewed 2009/07/15) http://www.mobileye.com/manufacturer-products/brochures.
10. Ernst, I., Hirschmueller, H.: Mutual information based semi-global stereo matching on the gpu. In: International Symposium of Visual Computing (ISVC), Las Vegas, NV, USA. (2008)
11. Yang, Q., et al.: Real-time global stereo matching using hierarchical belief propagation. In: British Machine Vision Conference (BMVC). (September 2006) 989–998
12. Hirschmueller, H., Scharstein, D.: Evaluation of cost functions for stereo matching. In: Proceedings of Int. Conference on Computer Vision and Pattern Recognition 07, Minneapolis, Minnesota. (June 2007)
13. Hirschmueller, H., Gehrig, S.: Stereo matching in the presence of sub-pixel calibration errors. In: Proceedings of Int. Conference on Computer Vision and Pattern Recognition 09, Miami, FL. (June 2009)
14. Shimizu, M., Okutomi, M.: An analysis of subpixel estimation error on area-based image matching. In: DSP 2002. (2002) 1239–1242
15. Steingrube, P., Gehrig, S.: Performance evaluation of stereo algorithms for automotive applications. In: ICVS 2009. (October 2009)