Hardware implementation of an SAD based stereo vision algorithm*

Kristian Ambrosch
Austrian Research Centers GmbH - ARC
A-1220 Vienna, Austria
kristian.ambrosch@arcs.ac.at

Martin Humenberger
Austrian Research Centers GmbH - ARC
A-1220 Vienna, Austria
martin.humenberger@arcs.ac.at

Wilfried Kubinger
Austrian Research Centers GmbH - ARC
A-1220 Vienna, Austria
wilfried.kubinger@arcs.ac.at

Andreas Steininger
Vienna University of Technology
A-1040 Vienna, Austria
steininger@ecs.tuwien.ac.at

Abstract

This paper presents the hardware implementation of a stereo vision core algorithm, that runs in real-time and is targeted at automotive applications. The algorithm is based on the Sum of Absolute Differences (SAD) and computes the disparity map using 320×240 *input images with a maximum disparity of 100 pixels. The hardware operates at a frequency of 65 MHz and achieves a frame rate of 425 fps by calculating the data highly parallel and pipelined. Thus an implemented and basically optimized software solution, running on an Intel Pentium 4 with 3 GHz clock frequency is 166 times outperformed.*

1. Introduction

In the field of automotive applications there is a growing need for sensors that can detect obstacles in near and far distances. For adaptive cruise control (ACC) and collision warning systems there are already embedded radar sensors in use [8]. Clearly, embedded stereo vision sensors could also be used for this kind of obstacle detection, producing much more detailed information than radar sensors. Until now the calculation of three dimensional depth maps is computationally too expensive, thus the hardware costs would be even higher than those of radar sensors currently used.

Besides the cost factor, automotive hardware platforms have to meet additional requirements. The size has to be small, because the space in the vehicle that is suitable for electronic components is very limited. To avoid a power breakdown while the engine is not running, the power consumption has to be very low. Another very challenging requirement is the temperature range of automotive equipment, which is from -40°C to +85°C, even at the most protected places within the car. This temperature range has to be reached without the need for active cooling, because moving parts intensely reduce the reliability of the system and Peltier Elements violate the requirement for a low power consumption.

Of course the use of a stereo vision system that meets all these requirements would not be limited to the automotive domain. Other applications, e. g. in the robotics domain, have quite similar requirements.

The calculation of three dimensional depth maps on signal processors that meet these requirements is very time consuming. Fortunately, many stereo vision algorithms do not enforce a purely sequential implementation and are therefore apt to parallelized solutions, leading to Field Programmable Gate Arrays (FPGAs) as a highly attractive realization platform.

We simulated and synthesized a stereo vision core algorithm implemented in VHDL for the Altera EP2S60, an FPGA that is suitable for this kind of application. The algorithm is based on the sum of absolute differences (SAD) algorithm [1]. It is small enough to enable the pre- and post-processing of the images on the same FPGA, but with a maximum disparity of 100 pixels it performs good enough for the detection of close fast moving objects.

2. Related Work

Various examples of stereo vision algorithms implemented on FPGAs exist in the literature.

Implementations that are using more than one FPGA [7, 10, 14] can be excluded for our purpose, because the hardware costs are too high and the board size does not fit as

*This work is partially funded by European Union project ROBOTS@HOME under grant #FP6-2006-IST-6-045350.

well. The same applies for works that are using PCI Cards in personal computers [3, 11, 13].

Woodfill *et al.* have proposed a stereo vision sensor, called DeepSea G2 vision system [15], that is based on an Application Specific Integrated Circuit (ASIC). The sensor contains the ASIC DeepSea processor, as well as a PowerPC chip, an Analog Devices Blackfin Digital Signal Processor (DSP), and an FPGA that handles the communication between the devices. Thus the advantages of the fast ASIC are out weighted by the high costs of the additional components and therefore this systems is not suitable for our purpose.

Yi *et al.* [16] proposed a stereo vision system based on a Xilinx Virtex II, that is using the SAD algorithm. The system can process images with a size of 270×270 at a frame rate of 30 fps. But the maximum disparity of 34 pixels is not considered as sufficiently enough for tracking close objects at high speed, which is crucial for collision warning systems.

The systems proposed in [12, 9], using a single FPGA, are also using a too small disparity for our purpose.

Han and Hwang have proposed a system [4] that can process images with a resolution of 640×480 at a frame rate of 60 fps and a maximum disparity of 128 pixels for the use in an household mobile robot. With 128 pixels, the maximum disparity is large enough to detect close objects. Because the full chip surface of the Xilinx XC2V3000 is consumed, there is no space left for the pre- and post-processing of the image data or the detection of occluded regions using a left/right consistency check.

3. Stereo Vision Algorithm

The task of a stereo vision algorithm is to analyze the images taken by a pair of cameras and to extract the displacement of the objects in both images. This displacement is counted in pixels and called disparity. All these disparities form the disparity map, which is the output of a stereo vision algorithm and enables the calculation of distances to objects using triangulation.

Stereo vision algorithms can be rough divided into feature based and area based algorithms. Feature based algorithms use characteristics in the images like edges or corners to solve the correspondence problem in the two images. The displacement between those features is used to build the disparity map and its density is directly related to the number of found features. Area based algorithms match blocks of pixels to find correspondences in the images. In the ideal case, each pixel can be found in the corresponding image as long as the search for the correct match keeps it within the image borders. The quality of the disparity map depends highly on the textures in the images. Common methods for the matching in area based algorithms are the sum of squared differences (SSD) [1] and absolute differences (SAD). There exist also various other algorithms for area based matching, but most of them are computational too expensive.

In our work we use area based matching, because the performed matching steps are independent of the calculations history and thus can be calculated in parallel for all analyzed disparities. The first part of the matching procedure is the calculation of the disparity space image [2], that contains the matching costs (absolute differences) for each disparity. The used disparity range is 0 to 100. With this range and the setup of our stereo camera head a detection of close objects is possible. The size of the input images is 320×240 and 220×240 for the output disparity map.

Close objects can also be detected using a smaller disparity range, where the minimum value is not zero, but at a dynamically chosen minimum value. This minimum disparity is detected for each object in the image. If there is a new object in the image, the algorithm needs to recover from the mismatch. Miyajima *et al.* [10] proposed a relatively large system using four Altera Stratix S80 FPGAs, that still has a worst-case time to recovery of 233 ms. Even with the high resource usage, the recovery time is unacceptable for automotive applications, because in this time span a car driving 130 km/h already moved a distance of 8.4 m. Taking into account that in dense traffic - especially at big crossings - the number of new objects in one second can be pretty high, the frame rate of the stereo vision system may drop to 4.3 fps, which is definitely not enough for our purpose even if we had these enormous hardware resources available. This is the reason why we rather use a large disparity than a small but dynamic disparity.

As shown in [6] the SSD algorithm performs only a little better than the SAD algorithm and this does not justify the high hardware resources, that are required for the implementation of the square operation. Thus we use the SAD algorithm in our implementation. The chosen block size for the SAD is 3×3, which keeps the required chip surface even lower. After the SAD is calculated for all stages of the disparity space image, the block with the lowest sum is chosen as the pixel value for the disparity map.

In our algorithm we match from the left image to the right one. Figure 3 shows an example result of the algorithm using the images in figure 1 and 2 as the input images. The shown input images come from calibrated cameras and are rectified to fulfill the epipolar geometry [17].

There is no detection of occluded areas included, but one reason why we are keeping the algorithm small is to have enough time for the calculation of two disparity maps in one frame. This offers the possibility of a left/right consistency check to detect the occluded areas.

The chosen algorithm is only a core stereo vision algorithm. As can be seen in the given example, the resulting disparity map shows only sparse three dimensional information of the scene and it is up to the additional pre- and post-processing stage to compute this information and create a detailed depth map.

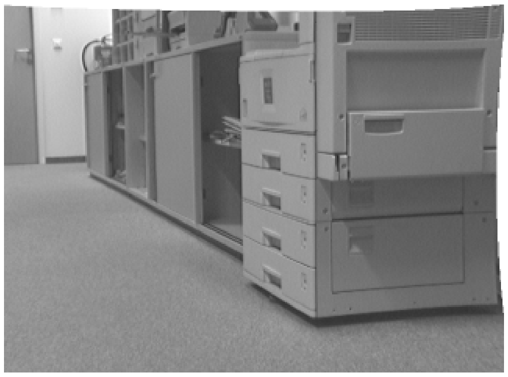
Figure 1. Left Stereo Image

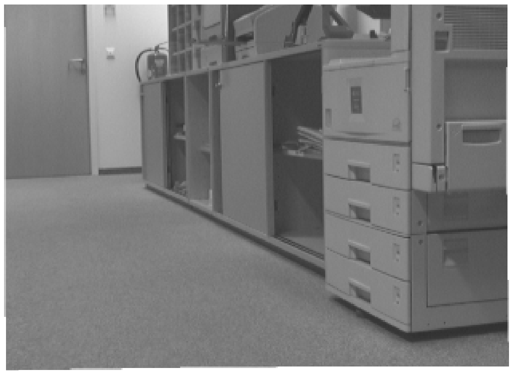
Figure 2. Right Stereo Image

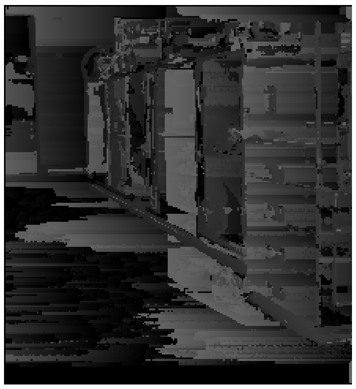
Figure 3. Resulting Disparity Map

4. Hardware Implementation

4.1. Architecture

The algorithm was implemented in VHDL and synthesized for an Altera EP2S60 using Quartus II. Since only the core algorithm was implemented, we used a 32 bit input port and an 8 bit output port, as well as two 1 bit interface lines to indicate an input or output transfer.

The calculation of the disparity map is performed line per line. Each calculation starts with the input of the two image lines, which are stored in the internal memory. The size of the input images is 320×240 in 8 bit gray scale, thus 160 $(2 \cdot 320 \cdot 8/32)$ clock cycles per line are needed for the data transfer. The data for the disparity map is written to the output port as soon as the corresponding pixel value is calculated, requiring no further time for the data transfer. We assume that the data transfer to or from an external memory or the pre-/post-processing stages are performed by another hardware block, to keep the algorithm more flexible.

The algorithm is divided in two major parts. The first part is the calculation of the disparity space image and the second part is the calculation of the SAD with the selection of the lowest block sum in the disparity space image and the resulting disparity.

The interface between these parts is the internal memory that contains the calculated disparity space image. For each stage of the disparity space image, two 4 kbit dual ported memory blocks with 32 bit data ports are used to store the last 3 lines. To reduce the time for the memory access, the pixels with the same positions in the horizontal direction are stored together in one data word. The data word itself is organized as a cyclic memory, where the position of the last written byte indicates the current line. Thus it is possible to access the pixel values of a specific position for the last 3 lines and all disparities in one clock cycle, resulting in a data transfer rate of 2400 bit $(3 \cdot 100 \cdot 8bit)$ per clock cycle.

Figure 4 gives an overview of the implemented hardware architecture.

4.2. Implementation

For the computation of the disparity space image, the absolute difference is calculated in parallel for all 100 disparities of one pixel. By buffering and shifting of the image data, an extra transfer time to read the image data is only necessary at the beginning of the computation. All further read transfers are performed in parallel with the calculation. This also includes the input of the already calculated disparity space image, that is stored in the internal memory and has to be combined with the new data. The storage of the calculated data takes an additional clock cycle and is pipelined with the calculation of the absolute differences. This part of the algorithm takes 247 clock cycles in total: 221 clock cycles for the calculation and 26 for the initial data transfer.

The calculation of the disparity map is very time consuming. Thus it is split up into 6 pipeline stages. Before the computation of the disparity map can start, the buffers have to be filled with the initial data. This is done by performing

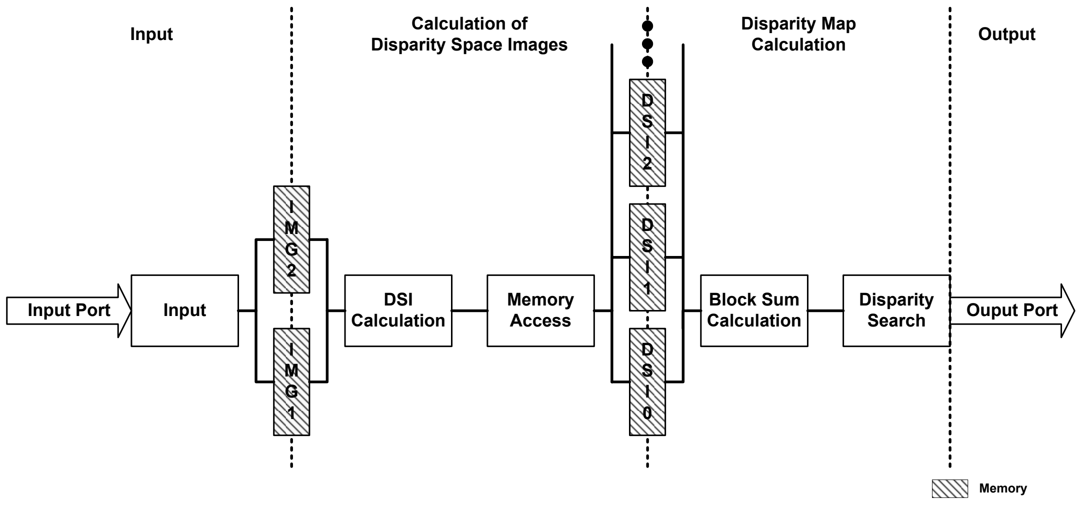

Figure 4. Architecture of the stereo vision hardware algorithm

the first two pipeline stages three times in a row.

The first stage of the pipeline loads the values of all disparities of the last three lines for the next pixel from the memory.

The next stage calculates the sum of the three lines, i. e. the vertical sum of the blocks, and stores it in a buffer. This buffer contains the vertical block sums for the last three pixels and is shifted before each new value is stored

The splitting of the blocks into three vertical block sums not only enhances the possibility to pipeline the computation, but also reduces the consumed hardware resources, because two of the three block sums can be reused for the computation of the next block sum. This way only three bytes have to be read from the internal memory for each calculated block. The used organization of the internal memory ensures, that these three bytes are within a single data word and can be read in one clock cycle, i. e. in the first pipeline stage. The amount of implemented registers is also reduced, because only two 10 bit values for the vertical sums have to be buffered, instead of six 8 bit values for the pixel values.

The summation of these three vertical block sums, to get the final block sum is the third stage of the pipeline.

The left part in figure 5 shows the block diagram of these three pipeline stages.

Now the block sums for all disparities are calculated and the block with the smallest value has to be found. This is performed by the next three pipeline stages. First groups of five blocks are compared. The value and the position of the smallest block in each group is handed over to the next stage. Then these values are grouped again by fives. Again the value and position of the smallest blocks are handed over to the next stage, adding the position of the corresponding groups to the positions of the blocks found so far.

In the final stage, the resulting four blocks are compared. The position of the smallest block is the value for the pixel in the disparity map, which is immediately written to the output port. The right part in figure 5 shows the block diagram of the search for the resulting disparity.

We evaluated different group sizes for this search and the best performance was given, when groups of fives with four final values were chosen. Further pipelining would only achieve improvements, if the calculation of the disparity space image was further pipelined as well, because the propagation delay of this calculation defines the maximum clock frequency.

In our solution, the calculation of the disparity map takes 4 clock cycles for the initialization and 225 clock cycles for the calculation itself, which amounts to 229 clock cycles in total.

The processing of the images is controlled by a further hardware block, that clears all buffers after the computation of a whole stereo image pair has finished and controls the interface line that indicates the start of an input of new image lines. This simple state machine takes 2 additional clock cycles for each calculated line.

5. Results and Comparison

5.1. Results

The total computation of a stereo image pair takes 153120 clock cycles. When synthesized with Altera Quartus II, the hardware can operate with a maximum frequency of 65 MHz. Thus the total computation takes $2.352ms$, the achieved frame rate is 425 fps.

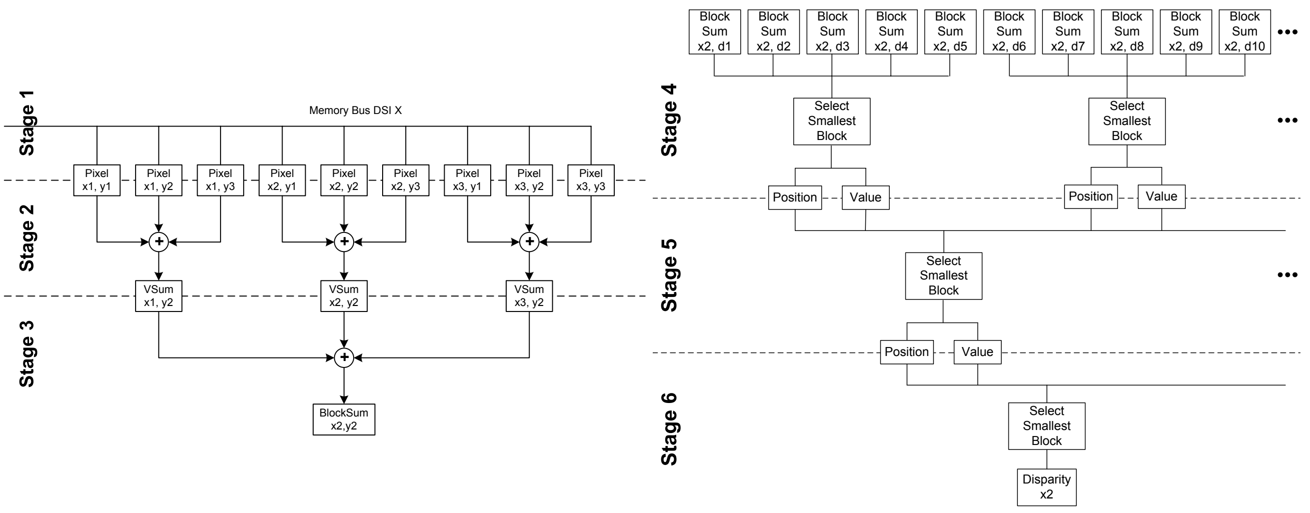

Figure 5. Left: Pipeline stages for the calculation of the block sums Right: Pipeline stages for search for the correct disparity

Author	Frame Rate	Image Size	Max. Disp.	Algorithm	Block Size	Platform
Proposed impl.	425 fps	320×240	100	SAD	3×3	1 FPGA
Software impl.	1.24 fps	320×240	100	SAD	3×3	PC
Niitsuma *et al.*	30 fps	640×480	27	SAD	7×7	1 FPGA
Lee *et al.*	122 fps	320×240	64	SAD	16×16	1 FPGA
Kim *et al.*	30 fps	1300×640	NA	Trellis based	NA	2 FPGAs
PARTS	42 fps	320×240	24	LW Phase Corr.	NA	16 FPGAs
Masrani *et al.*	30 fps	640×480	dyn	LW Phase Corr.	NA	4 FPGAs
Niitsuma *et al.*	840 fps	320×240	121	SAD	7×7	1 FPGA + PC
Miyajima *et al.*	18.9 fps	640×480	200	SAD	7×7	2 FPGAs + PC
DeepSea	200 fps	512×480	52	Census Transf.	NA	ASIC

Table 1. Comparison of stereo vision implementations

The algorithm consumes 15616 arithmetic look up tables, which is just about 32% of the available FPGA resources and equivalent to 19520 logic elements. There is a high amount of hardware resources left for the pre- and post-processing of the images, although we are not using a high end FPGA.

The memory usage is 884736 bits or 35% of the chip's internal memory. Since we use about 80% of the 4 kbit block RAM in the chip and the rest of the memory is split into 512 kbit blocks, other processing stages can hardly employ parallel memory access.

Table 2 summarizes the performance data of the implemented stereo vision algorithm.

The high frame rate gives us the opportunity to calculate a left/right consistency check for the detection of occluded areas. This can be done by computing the same stereo pair once again, using the left image as the main image and comparing the results. Even if our algorithm always assumes that the right image is the main image and therefore searches for the disparity in the other image in the right direction, this can be overcome by horizontal flipping of the images at the input stage.

Input Image Size	320×240
Search Window	100 Disparities
Block Size	3×3
Frame Rate	425 fps
Logic Elements	19520
Memory Bits	884736

Table 2. Performance of the proposed algorithm

5.2. Comparison

For a better evaluation of the performance benefits of our hardware implementation in an FPGA instead of processor based systems, we implemented the same algorithm in software as well. This software implementation is an optimized implementation using Intel's Open Source Computer Vision

Library [5]. The test platform was an Intel Pentium 4 with 3 GHz clock frequency and 1 GB memory. The processing time for one frame was 391 ms. This is 166 times slower than the proposed hardware implementation and it seems obvious that, even with the best optimizations, the processor based system cannot outperform the FPGA based solution.

We also compared our system to other FPGA implementations as shown in table 1. Since the used stereo vision algorithms are very different and so the quality of the resulting disparity maps differs as well, direct comparisons of logic elements against frame rate would be misleading.

6. Conclusion

A cost efficient hardware implementation of a real-time stereo vision algorithm using an FPGA for the calculation of disparity maps is proposed. Our small sized algorithm leaves enough resources for the implementation of pre- and post-processing stages. It performs well enough for the detection of fast moving objects by using a large disparity range. The high frame rate gives the opportunity for further expansions of the algorithm.

The small hardware costs of the chosen SAD algorithm allowed us to completely outperform an already optimized software implementation. This shows, that FPGAs or ASICs are an excellent choice for the realization of low cost real-time stereo vision systems for automotive or robotics applications.

7. Future Works

The results for our stereo vision hardware algorithm are very promising. We plan to extend our algorithm with a left/right consistency check for the detection of occluded regions, as already mentioned and evaluate the use of different blocks sizes.

Furthermore, we will use different port sizes for our memory access, which enables us to write the calculated disparity space image into the internal memory without affecting the data already stored in the 32 bit data word. Thus no more read operations from this memory are necessary, which enables us to make full use of the dual ported memory and pipeline the computation of the disparity space image with the calculation of the disparity map. This will further reduce the processing time for one frame significantly.

We also intend to combine the algorithm with a pre- and post-processing stage and build a prototype of a stereo vision system, that is small enough for the operation in automotive and robotics applications.

References

[1] J. Banks, M. Bennamoun, and P. Corke. Non-parametric techniques for fast and robust stereo matching. In *Proceedings of IEEE Conference on Speech and Image Technologies for Computing and Telecommunications*, 1997.

[2] A. Bobick and S. Intille. Large occlusion stereo. *International Journal of Computer Vision*, 33(3):181–200, 1999.

[3] P. Corke and P. Dunn. Real-Time Stereopsis Using FPGAs. In *Proceedings of the IEEE Conference on Speech and Image Technologies for Computing and Telecommunications*, 1997.

[4] D. Han and D.-H. Hwang. A novel stereo matching method for wide disparity range detection. *Lecture Notes in Computer Science*, 3656:643–650, 2005.

[5] Intel Corporation, Santa Clara, CA. *Intel Open Source Computer Vision Library, www.intel.com/technology/computing/opencv/*, 2007.

[6] T. Kanade. Development of a video-rate stereo machine. In *Proceedings of 1994 ARPA Image Understanding Workshop*, 1994.

[7] S. Kim, S. Choi, S. Won, and H. Jeong. The coil recognition system for an unmanned crane using stereo vision. In *Proceedings of the 30th Conference of IEEE Industrial Electronics Society*, 2004.

[8] S. Le Beux, P. Marquet, O. Labbani, and J.-L. Dekeyser. FPGA implementation of Embedded Cruise Control and Anti-Collision Radar. In *Proceedings of the 9th EUROMICRO Conference on Digital Systems Design*, 2006.

[9] S. Lee, J. Yi, and J. Kim. Real-time stereo vision on a reconfigurable system. *Lecture Notes in Computer Science*, 3553:299–307, 2005.

[10] D. K. Masrani and W. J. MacLean. A Real-Time Large Disparity Range Stereo-System Using FPGAs. In *Proceedings of the IEEE International Conference on Computer Vision Systems*, 2006.

[11] Y. Miyajima and T. Maruyama. A Real-Time Stereo Vision System with FPGA. In *Proceedings of the 30th Conference of IEEE Industrial Electronics Society*, 2003.

[12] H. Niitsuma and T. Maruyama. Real-time detection of moving objects. *Lecture Notes in Computer Science*, 3203:1155–1157, 2004.

[13] H. Niitsuma and T. Maruyama. High speed computation of the optical flow. *Lecture Notes in Computer Science*, 3617:287–295, 2005.

[14] J. Woodfill and B. Von Herzen. Real-time stereo vision on the PARTS reconfigurable computer. In *Proceedings of the 5th IEEE Symposium on FPGAs for Custom Computing Machines*, 1997.

[15] J. I. Woodfill, G. Gordon, D. Jurasek, T. Brown, and R. Buck. The Tyzx DeepSea G2 Vision System, A Taskable, Embedded Stereo Camera. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recoginition Workshop*, 2006.

[16] J. Yi, J. Kim, L. Li, J. Morris, G. Lee, and P. Leclercq. Real-time three dimensional vision. *Lecture Notes in Computer Science*, 3189:309–320, 2004.

[17] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27(2):161–195, 1998.