Real-Time Semi-Global Matching on the CPU

Stefan K. Gehrig
Daimler AG
Group Research & Advanced Engineering
HPC 059-G024
71059 Sindelfingen
Germany
stefan.gehrig@daimler.com

Clemens Rabe
University of Ulm
Albert-Einstein-Allee 41
89081 Ulm
Germany
clemens.rabe@gmail.com

Abstract

Among the top-performing stereo algorithms on the Middlebury Stereo Database, Semi-Global Matching (SGM) is commonly regarded as the most efficient algorithm. Consequently, real-time implementations of the algorithm for graphics hardware (GPU) and reconfigurable hardware (FPGA) exist. However, the computation time on general purpose PCs is still more than a second.

In this paper, a real-time SGM implementation on a general purpose PC is introduced. Parallelization and image subsampling is used while ensuring the full disparity resolution for small disparities.

This approach is especially beneficial for robotic and driver assistance systems. The system is able to compute 640x320 image pairs at more than 14Hz, leaving also computational resources for subsequent processing such as free space computation or object detection. The algorithmic approach is portable to multi-core embedded processors.

1. Introduction

3D perception is a crucial task both for automotive and 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 accuracy was established [15]. This benchmark database ranks the currently published stereo algorithms. Among the top-performing algorithms in this database, we found semi-global matching (SGM) [8] 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 a real-time stereo implementation that runs at framerates beyond 10Hz on a general purpose PC.

Both GPU and FPGA implementations of SGM exists. However, GPUs that deliver real-time performance consume more than 100W power: FPGA solutions offer low-power performance, however, they are not straightforward to integrate in laptop-like systems. In addition, even small algorithmic adaptions and extensions require a new time-consuming bitstream generation. With framerates beyond 10Hz robotic applications and urban driver assistance functions can be implemented on the same platform using this algorithm. Our system analysis indicates that SGM can also run on embedded processors such as Cortex A9 quad core with clock rates beyond 800MHz [1] or multi-core digital signal processors (see *e.g.* [6]).

Our main applications are robotic and driver assistance applications running on mobile multi-core processor technology. SGM has already proven its robustness and suitability for driver assistance applications [18]. On such a multi-core platform the SGM parameters and evaluation ranges may be changed on a frame-by-frame basis. The optimizations presented in this paper allow not only the real-time computation of the stereo data, but also its analysis by high-level applications such as object detection on the same platform. Fast algorithms performing free space computation or object detection are available (see *e.g.* [5]) for that purpose.

This paper is organized in the following way: Related work to our contribution is presented in Section 2. Section 3 explains our system design followed by a section on implementation details to obtain real-time performance. Results of the implementation including timings and algorithmic results can be found in the Section 5. Conclusions and future work comprise the final section.

2. Related Work

Today, many real-time stereo vision systems are available. Most of the stereo engines operate with local correlation-variants, *e.g*. the census-based stereo system by the company Tyzx [19].

As for global stereo methods, only dynamic programming stereo that operates on one scanline only is available [4]. Dynamic Programming on a single scan line has also been implemented on an FPGA [14] .

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 [3]. An earlier implementation on the GPU reported slightly slower framerates at comparable image sizes [13]. Belief propagation was shown to be real-time capable on the GPU for small images [20] and disparity ranges. The power consumption of these GPUs is well above 100W.

Semi-global matching has been successfully ported on an FPGA [7]. This implementation yields 25Hz on 680x400px image pairs processing 128 disparities. To keep the memory bandwidth to an acceptable level, two SGM engines operate on 340x200px image pairs with a range of 64 disparities per frame. The results were combined to a 680x400 disparity map.

Despite the enormous amount of work on real-time stereo vision, there is still a need for a high-performance implementation of a global stereo method on the CPU to keep stereo and downstream algorithms on one platform. We close this gap with our contribution.

3. Algorithm Design

3.1. Design Considerations

For mobile robotics and driver assistance, VGA size images are typically evaluated. With the desire to keep the views of corresponding points sufficiently similar, a disparity range of 128 is a common value. For stereo camera systems, the depth uncertainty rises quadratically with the measured distance. However, most robotic and driver assistance systems have accuracy demands that remain mostly constant over the operating range. This fact can be exploited by using lower resolution imagery for nearby objects and the full resolution for far-away objects.

The main parts of the SGM algorithm are the following: After computing the similarity criterion, Census on a 5x5 window in our case, the path accumulation over 8 paths is conducted. In this disparity cube, the minimum costs are obtained, a sub-pixel interpolation is performed, the disparity map is median-filtered and occluded regions are invalidated with right-to-left consistency check (RL-check).

Summarizing, the algorithmic steps for the SGM computation are:

1. Census computation
2. Path accumulation
3. Minimum cost determination
4. Sub-pixel interpolation
5. Median filtering
6. Right-to-left consistency check

Ideally, a full resolution SGM computation over the full disparity range should be performed. However, the often demanded framerates beyond 10Hz require image subsampling to reduce the computational load. The resulting loss in range accuracy due to fewer disparity steps is minimized by organizing the subsampling in the following way:

Figure 1 shows the depth uncertainty for our stereo camera system. The dotted line shows the quadratic increase in uncertainty over the measured distance. For most driver assistance systems, it is desirable to keep a depth uncertainty of less than 1m over the operating range (see *e.g*. [12]). This offers options to subsample the image multiple times - the resulting uncertainties are shown with the solid line. Up to disparity 16 the full resolution has to be used. Between 16 and 32 disparity levels, half the resolution is sufficient. For the ultra-near range, quarter resolution is sufficient. The underlying parameters for the camera system are a focal length of 830px and a baseline of 30cm resulting in a stereo camera constant of 250$\mathbf{px} \cdot \mathbf{m}$. With $z = \frac{f \cdot b}{d}$, and $\Delta z = \frac{z^2}{f \cdot b} \cdot \Delta d$ we can limit the uncertainty to less than 1m despite the smaller image resolution. In this graph we assume a conservative disparity uncertainty of 0.5px standard deviation which is easily achieved, provided sub-pixel interpolation is performed.

An example of the subsampled images and disparity ranges we use is shown in Figure 2. The same principle can be applied to stereo camera systems with other baselines and focal lengths. The disparity ranges might have to be adapted.

Note that this approach is not equivalent to the classic coarse-to-fine approach where determined disparity values are only refined at larger image resolution [17]. We compute the small disparities from scratch only blanking out the parts that are not within this disparity range. For the remainder of the paper we refer to this algorithm variant as Real-time SGM (RTSGM).

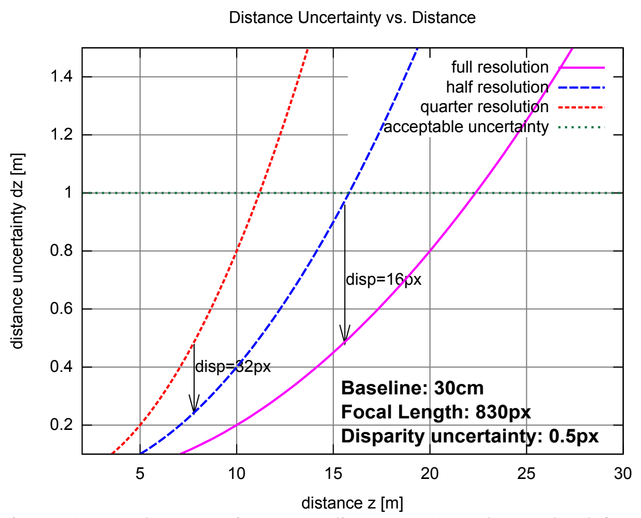

Figure 1. Depth uncertainty vs. distance. At 7.8m and 15.6m changes in resolution occur. Up to 22m distance, the depth uncertainty stays below 1m.

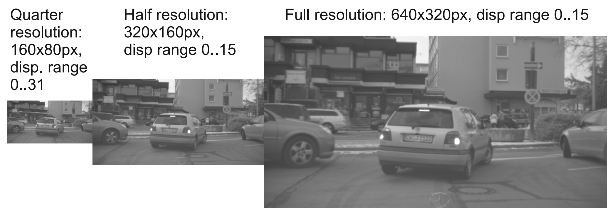
Quarter resolution: 160x80px, disp. range 0..31
Half resolution: 320x160px, disp range 0..15
Full resolution: 640x320px, disp range 0..15

Figure 2. Example image and corresponding subsampled images of half and quarter resolution.

3.2. Algorithm Analysis

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. We run the algorithm on 8 paths, similar to [3] and [7].

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. This part can be run in parallel.

We consider two 640x480px input greyscale images with 12 bits per greyvalue as input. Due to the lack of relevant data, the actual region of computation is limited to 640x320px, the rest of the image shows the sky or the own vehicle in our setup. In automotive and robotics applications with these image sizes, a disparity range of 0 through 127 suffices. Ideally, SGM would be computed everywhere at full resolution. We decided to reduce the image size twice by a factor of two in width and height. For the quarter resolution image pair, SGM is calculated with up to 31 disparities, which corresponds to 127 disparities at full resolution. The second SGM runs on half resolution image pairs with up to 15 disparities, corresponding to 31 disparities. The full resolution SGM is computed for 16 disparities on the full image. This subsampling step is mandatory to keep the data bandwidth to a manageable level and has little effect on depth accuracy.

For the outdoor scenario with uncontrollable lighting conditions, the census metric has been found to be very effective [11]. We have to cope with different brightness in the left and right image and with vignetting effects due to inexpensive lenses. In contrast to these requirements, the similarity criterion mutual information proposed in [8] is sensitive to vignetting artifacts [10]. As a consequence, we deviate from a pixel-wise matching cost using a small 5x5 census window to keep the effect of foreground fattening small. For the subsampled images, we stick to the same metric.

4. Implementation

4.1. Similarity Criterion

We use a 5x5 Census mask to determine correspondences. The cost computation step has to be performed $w \cdot h \cdot n_{disp}$ times for every cycle and is hence time-critical. Here n_{disp} is the number of disparities, w the image width and h the image height. For the similarity criterion, an OpenMP parallelization was implemented. Every core runs on one line of the image independently. As a first step, the census-transformed images are computed using the same line-wise parallelization with OpenMP. A 32bit integer image suffices to store the result ($25 bits$). These images are then used to compute the Hamming distance, the absolute distance of the census-transformed image pixels, quickly using a combination of XOR-operations and bit-shifting operations as explained in [21].

4.2. Image Subsampling

We run SGM on subsampled images to reduce memory bandwidth and computational burden. The original image region-of-interest of 640x320px is reduced to 320x160px by simply averaging the 4 greyvalues from the original image and storing the result at the corresponding position in the subsampled image. For the quarter resolution image, the image subsampling step is repeated. This procedure could lead to aliasing artifacts without proper low-pass filtering but for the disparity computation using Census we have not observed such problems.

4.3. Thin Objects and Lateral Resolution

The chosen image subsampling strategy meets the requirement of limited distance uncertainty. However, with image subsampling the lateral resolution of the disparity image deteriorates since, for nearby objects, one disparity result from the subsampled image either sets the value for 2 (half resolution) or 4 pixels in one row of the full resolution image. This clearly reduces the lateral resolution. In addition, this can become critical if thin objects at close distances appear in the scene. With our camera setup 1px corresponds to 0.8cm width at 7.8m distance. At quarter resolution, we need at least 4px to see an object in the subsampled image, better 8px to prevent smoothing over the object. This yields a minimum object width of 6.4cm which is fortunately the case for most objects encountered in traffic scenes. Thinner objects could be detected when applying the Magnitude-Extended Laplacian Pyramid [17]. A result on a thin object is shown in Section 5.3.

4.4. Parallelization

For the path accumulation step , a parallelization is crucial to increase computational speed. In essence, path accumulation is performed by Equation 12 from [8],

$$L'_r(\mathbf{p},d) = C(\mathbf{p},d) + \min(L'_r(\mathbf{p}-r,d), L'_r(\mathbf{p}-r,d-1) \\ +P_1, L'_r(\mathbf{p}-r,d+1) + P1, \min_i L'_r(\mathbf{p}-r,i) + P_2). \quad (1)$$

This operation is performed $8 \cdot w \cdot h \cdot n_{disp}$ times for every cycle. We use SIMD (single instruction, multiple data) instructions, namely SSE2, SSE3, and SSSE3 instructions to compute 16 disparities in one step. With the chosen similarity criterion, a 5x5 Census, it is sufficient to store only one byte for the cost values and one byte for the accumulated cost value. Hence, 16 values fit in one 128-bit register. This yields a speed-up factor of about six compared to the plain C variant. One thing to consider is the use of a saturating packed add (PADDSB) command to prevent overflows from having an effect when accumulating the path. The penalties P_1 and P_2 have to roughly be below $4 \cdot P_1 + 4 \cdot P_2 < 255$ to ensure unsaturated accumulated costs at least for the values around the accumulated cost minimum. The optimum penalties for our Census similarity criterion stay below that limit.

Another parallelization we implemented is computing the independent rays on one path direction simultaneously via OpenMP parallelization. This way, a four-fold speed increase of that part is achieved. One additional option to reduce load-store operations by a factor of 4 is to compute 4 paths in one scan similar to [7]. Then, above parallelization is limited to 2 cores, leaving more resources to stereo post-processing. To minimize load-store operations, the SGM core would calculate four SGM directions simultaneously: one horizontal to the left and three vertical down, including two diagonal down paths (see Figure 3).

We decided to compute the 8 paths independently leaving us with more flexibility to turn single paths on or off.

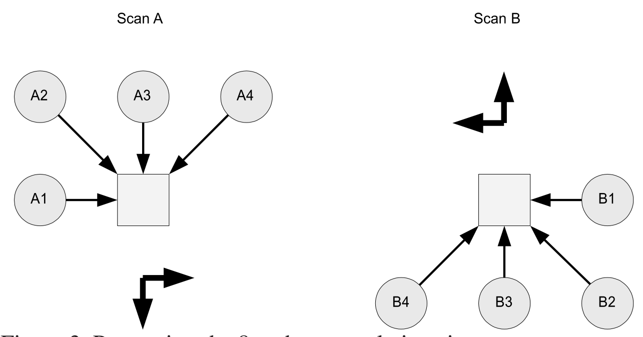

Figure 3. Processing the 8 path accumulations in two scans.

After traversing all 8 paths, the minimum cost of all disparity values has to be found for every pixel. To further enhance the depth precision, an additional interpolation is conducted which yields a float disparity value. We use the equiangular interpolation as suggested by [16] since it yields better sub-pixel precision for cost metrics based on absolute differences such as the Hamming distance compared to the standard parabola fit. After the interpolation, a 3x3 median filter is performed similar to [8] before the data is verified in the right-left check.

4.5. Disparity Map Combination

When propagating a disparity map from a subsampled image into the higher resolution, one has to avoid false matches caused by the limited disparity range. This is done in the following way: We set the matching cost for the maximum disparity of the higher resolution cost cube to 0 whenever this disparity value or a larger disparity has been established in the subsampled image ("cost prior" in Figure 4). This gives a strong prior in the higher resolution SGM result towards the subsampled SGM result for large disparities. The combination of both results ("fill near disparities" in Figure 4) is straightforward, computing the final disparity $d(x, y)$ via

$$d(x,y) = \begin{cases} d_{hr}(x,y),\ if\ d_{hr}(x,y) < d_{max} - 2\ \&\ d_{hr}\ valid \\ 2 \cdot d_{ss}(x/2, y/2)\ \ otherwise. \end{cases} \quad (2)$$

d_{hr} is the disparity of the higher resolution image, d_{ss} the disparity value of the subsampled image. The higher-resolution disparity image is used, only if the disparity value is at least 2 disparity steps less than the maximum disparity value. The costs for the maximum disparity has been changed to 0 for near pixels to favor a reconstruction with a large disparity value. To be sure to model the correct penalties for disparity jumps we do not use the disparity value $d(x, y) = d_{max} - 1$. Here, it is not known whether the real disparity is to be penalized with P_1 or P_2.

To find invalid matches, *e.g.* in occluded regions, 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 minima differ [8].

The full workflow to compute RTSGM is shown in Figure 4. The block "SGM" comprises all steps listed in Section 3.1 ranging from to Census Computation to Right-to-left consistency check.

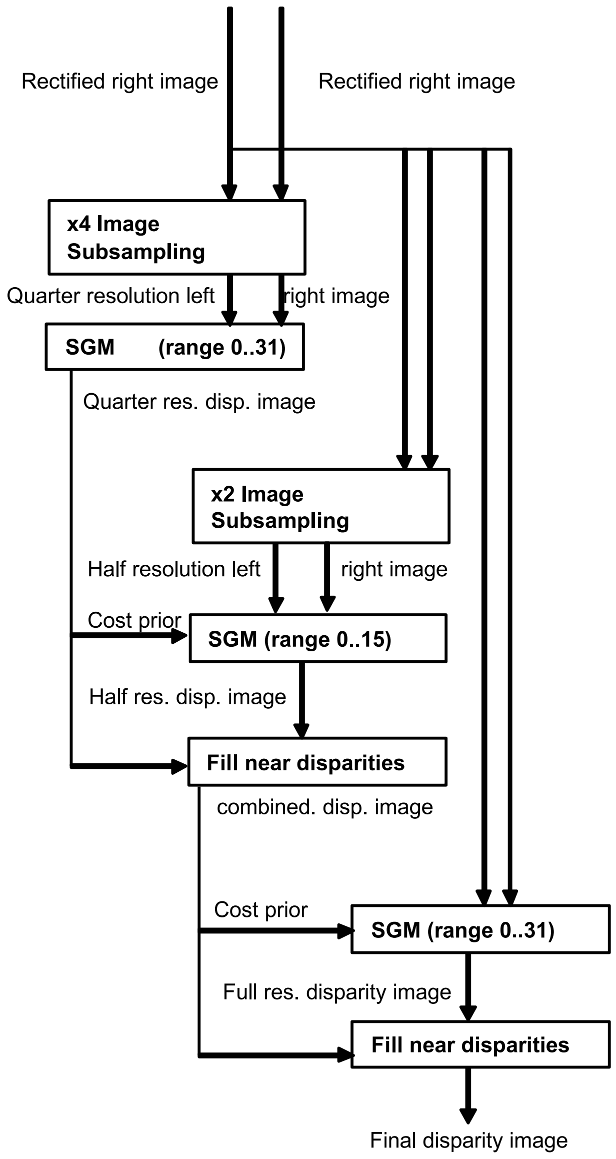

Figure 4. Algorithm flow chart for RTSGM.

5. Results

5.1. Algorithmic Performance

We computed results on the Middlebury test data with our Census similarity criterion using a 5x5px mask. We compare running SGM at full resolution using Census (Census SGM) to our algorithm variant proposed above (RTSGM) and to the mutual information literature SGM variant from [8]. The results are presented in Table 1. All disparities deviating more than 1px from ground truth are counted as errors. The error percentage is stated for all pixels. For Census SGM and RTSGM, the disparity interpolation algorithm from [9] was used and the remaining invalid pixels at the image edge are filled by propagating the last valid disparity to the image edge. We use $P_1 = 7$ and $P_2 = 20$ as smoothness penalties. The results of mutual information SGM [8] are comparable to the Census SGM results, Census SGM even being slightly better when using the bad pixel average as comparison measure (overall 6.74% vs. 6.86% error). This is somewhat surprising as the Census similarity criterion has been selected to cope with difficult imaging conditions which are not present in the Middlebury database. RTSGM yields the same result for Tsukuba, since the maximum disparity is 15. For all other image pairs, RTSGM obtains more wrong disparities due to the blocky results for large disparities resulting from the computation on subsampled images. The difference between Census SGM and RTSGM is significantly smaller when allowing 2px disparity deviation.

Method	Tsukuba	Venus	Teddy	Cones
MI SGM	3.96	1.57	12.20	9.75
Census SGM	3.38	2.00	12.81	8.79
RTSGM	3.38	3.29	16.54	13.04

Table 1. Error percentages for stereo ground truth data sets. The census similarity criterion performs similar to mutual information. RTSGM has a weaker performance. Error threshold is 1px disparity deviation and all pixels are evaluated.

5.2. Computational Speed

Our timings have been measured on an Intel Core i7-975ex (four cores) with 3.3GHz clock rate. We compare our timings to the results of a GPU implementation [3] and an FPGA implementation [7]. Also their disparity ranges and image sizes have been timed using our CPU SGM variant. The utilized image sizes and disparity ranges are listed in Table 2. Our CPU implementation outperforms the GPU variant in terms of speed. Note that the utilized GPU in [3], a NVidia Geforce 8800 Ultra, offers less performance than current GPU models. Comparing to the FPGA implementation, our system runs at slightly less than half the speed. For RTSGM, we obtain a cycle time of 69ms, corresponding to 14.5Hz when no further stereo postprocessing is performed. The necessary pre-processing step rectification takes less than 3ms when using SSE optimization. We also compared the number of crunched disparities (image size times number of disparities) per second. Our implementation performs up to 117 million disparity computations per second, only topped by the FPGA implementation. This number is achieved running the full resolution Census SGM. Despite the great disparity crunching perfor-

Method Unit	image size [px]	#disp [px]	cycle [ms]	disp/s [$10^6/s$]
GPU	320x240	64	76	64
CPU	320x240	64	51	96
FPGA	2· 340x200	2·64	40	218
CPU	2· 340x200	2·64	94	92
CPU	160x80	32	8	50
	320x160	16	16	50
	640x320	16	45	73
CPU	268800	128	69	65
Census	640*320	128	224	117

Table 2. Timings of our CPU SGM algorithm compared to a GPU and an FPGA implementation. Despite the limited parallelism on the CPU the results are comparable. The column "disp/s" states the number of crunched disparities per second. The last row shows the timings for full resolution Census SGM as reference.

Algorithmic step Computation time	$\frac{1}{4}$ Res. [ms]	$\frac{1}{2}$ Res. [ms]	$\frac{1}{1}$ Res. [ms]	All [ms]
Census computation	2	5	12	19
Path accumulation	2	7	22	31
Sub-pixel interpol.	1	1	3	5
RL-check	2	2	7	11

Table 3. Detailed timings of the implemented RTSGM algorithm. Most of the time is used for cost computation and path accumulation.

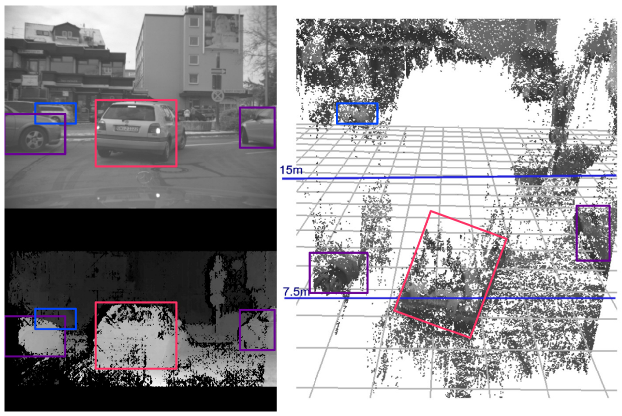

Figure 5. car scene and disparity map result using mutual information as similarity criterion [8].

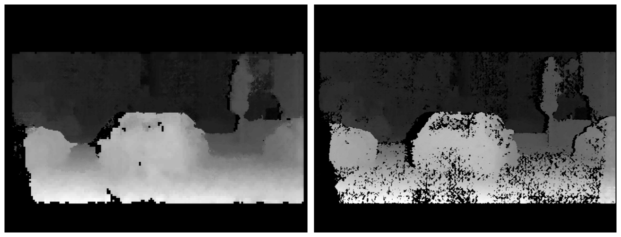

Figure 6. Disparity map of above scene for RTSGM (left), Census SGM with full resolution (right). The foreground car appears more blocky on the left.

mance, the resulting cycle time of 224ms is not sufficient for robotics or driver assistance tasks.

Table 3 lists detailed timings for the different processing steps. The parallelized blocks take most of the computation time with the fast RL-check taking up most of the unparallelized time. The remaining parts median filtering, image subsampling, and cost value injection from subsampled images take about 3ms in total. The timing for cost accumulation also includes finding the minimum disparity. Most of the time is used for the full resolution SGM computation with the number of crunched disparities per second being roughly constant for all resolution stages.

5.3. Real-World Results

We show a few results for RTSGM using driver assistance imagery. The first scene, an urban scene, shown in Figure 5, contains multiple vehicles at various distances and a traffic sign post at 16m distance. For reference, SGM using mutual information as similarity criterion is shown. The vehicles and the traffic sign post are correctly measured. However, only few measurements on the street are obtained although we used 10 bit greyvalues as input for matching.

Figure 6 left shows the disparity map using RTSGM. Here we used $P_1 = 10$ and $P_2 = 50$ to cope with the low-texture of the scene. A dense disparity map is obtained, slight blocking effects occur for the closest car in the middle of the scene (red box in Figure 5). Figure 7 depicts the 3D view of the RTSGM disparity map. One can see the change in resolution at about 7.5m and 15m distance. The clipped cars on the left and right side have already comparable 3D views to Census SGM running at full resolution (Figure 6 left and Figure 6 right, cyan box in Figure 5). The bright station wagon at the left at 25m distance (blue box in Figure 5) has almost exactly the same triangulated points as for the full resolution Census SGM variant. This is due to the fact that up to a disparity value of 16px, the full resolution SGM is computed in case of RTSGM. The traffic sign post at about 16m is located at the border going from half resolution to full resolution SGM. The transition from half resolution SGM to quarter resolution SGM can be viewed at the car in the center of the scene. The rear is at 7m distance computed at quarter resolution, from the rear door onwards, the points on the car are computed with half resolution SGM.

Our second traffic scene, a rural scene (Figure 8 top left) with a small street sign on the right and reflection posts shows the effect of image subsampling on thin structures. Figure 8 bottom left and the right figure show the disparity map and 3D result using SGM with mutual information. Again, the disparity map on the street is quite sparse. In ad-

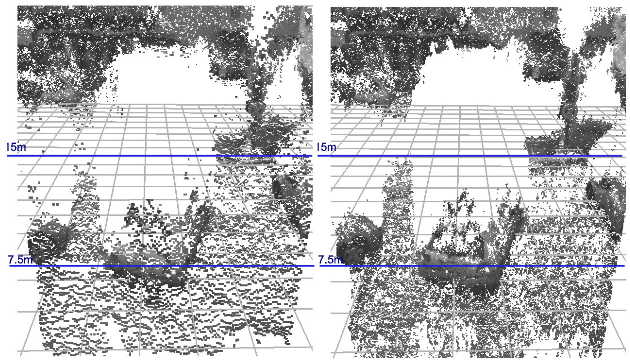

Figure 7. 3D view of above scene for RTSGM (left), Census SGM with full resolution (right). The foreground car appears more blocky but no significant depth errors occur.

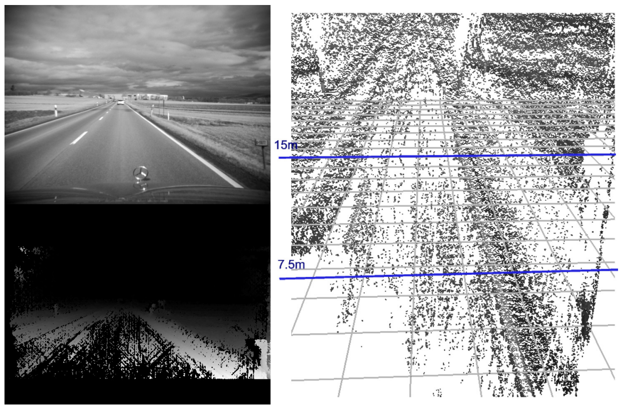

Figure 8. pole scene and disparity map result using mutual information as similarity criterion [8].

dition, the pole on the bottom right appears fattened and the distance determination noisy. This is due to the vignetting of the lens which is most prominent at the image edges. Also, some false correspondences in the sky occur at the left edge of the disparity image. This shows once more that mutual information is sensitive to vignetting artifacts [11]. In Figure 9 on the left side, the disparity map for RTSGM is shown. The street sign is clearly visible but appears blocky and the silhouette of the sign turns out enlarged (foreground fattening). Census SGM reconstructs the street sign very well. This can be seen better in the 3d view in Figure 10. Note that the disparity map is less dense for Census SGM. For RTSGM, the low-textured parts can be better measured due to considering a larger neighborhood in the matching step. The 5x5 Census mask operating on half and quarter resolution corresponds to 10x10 and 20x20 masks, respectively. On the street the transitions from quarter resolution computation to half resolution at about 7.5m and the transition from half resolution to full resolution are visible in the 3D views. For the ultra-near range, 16 pixels of the original image are mapped to one disparity value. this corresponds to an angular resolution of about 0.25°, which is sufficient for most robotics and driver assistance tasks in the near range.

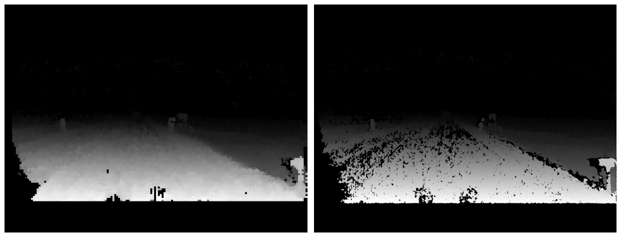
Figure 9. Disparity map of above scene for RTSGM (left), Census SGM with full resolution (right). The pole appears more blocky on the left and exhibits a slight halo effect.

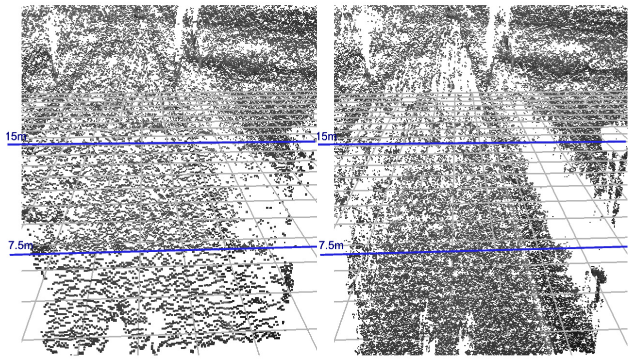

Figure 10. Disparity map of above scene for RTSGM (left), Census SGM with full resolution (right). The pole appears more blocky on the left and exhibits a slight halo effect, but is visible on the full pole length.

6. Conclusions and Future Work

We have introduced a real-time global stereo engine on a general purpose computer. The results using SGM are very encouraging. With this implementation, it is possible for the first time to compute high-quality disparity maps in real-time without specialized hardware . The key design choices to obtain real-time performance are image subsampling and result reuse for full resolution computation as well as parallelization of the most time-consuming blocks. We expect this stereo engine to be a very good basis for many future camera-based driver assistance and robotic systems.

For future work, we consider porting RTSGM to high-end embedded processors with up to 4 cores at beyond 1GHz clock rates which offer sufficient computing power [1]. This way, an embedded system with SGM real-time performance and very little power consumption seems feasible. The potential of using OpenMP on such a platform has been shown by [2].

References

[1] ARM. Cortex-a9 mpcore technical reference manual. Technical report, ARM Holding, viewed 2010/03/08. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0407e/index.html. 1, 7

[2] H. Blume and et al. Openmp-based parallelization on an mpcore multiprocessor platform. *Int. Journal of Systems Architecture*, 54:1019–1029, 2008. 7

[3] I. Ernst and H. Hirschmueller. Mutual information based semi-global stereo matching on the gpu. In *International Symposium of Visual Computing (ISVC), Las Vegas*, 2008. 2, 3, 5

[4] S. Forstmann and et al. Real-time stereo by using dynamic programming. In *Computer Vision and Pattern Recognition Workshop on Real-Time 3D Sensors*, pages 29 – 36, 2004. 2

[5] U. Franke. Real-time stereo vision for urban traffic scene understanding. In *Proceedings of the Intelligent Vehicles 00 Symposium*, 2000. 1

[6] Freescale. Msc8144 quad-core dsp. Technical report, Freescale Semiconductor, viewed 2010/03/08. http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MSC8144. 1

[7] S. Gehrig, F. Eberli, and T. Meyer. A real-time low-power stereo vision engine using semi-global matching. In *ICVS 2009*, pages 134–143, October 2009. 2, 3, 4, 5

[8] H. Hirschmueller. 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, pages 807–814, June 2005. 1, 3, 4, 5, 6, 7

[9] H. Hirschmueller. Stereo vision in structured environments by consistent semi-global matching and mutual information. In *Proceedings of Int. Conference on Computer Vision and Pattern Recognition 06, New York, NY*, pages 2386–2393, June 2006. 5

[10] H. Hirschmueller and D. Scharstein. Evaluation of cost functions for stereo matching. In *Proceedings of Int. Conference on Computer Vision and Pattern Recognition 07, Minneapolis, Minnesota*, June 2007. 3

[11] H. Hirschmueller and D. Scharstein. Evaluation of stereo matching costs on images with radiometric distortions. *IEEE Transact. Pattern Analysis & Machine Intelligence*, 31(9):1582–1599, 2009. 3, 7

[12] K. Huh, J. Park, J. Hwang, and D. Hong. A stereo vision-based obstacle detection system in vehicles. *Optics and Lasers in Engineering*, 46(2), February 2008. 2

[13] I. D. Rosenberg, P. L. Davidson, C. M. R. Muller, and J. Y. Han. Real-time stereo vision using semi-global matching on programmable graphics hardware. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*. ACM Press, 2006. 2

[14] S. Sabihuddin and W. J. MacLean. Maximum-likelihood stereo correspondence using field programmable gate arrays. In *Int. Conference on Computer Vision Systems (ICVS), Bielefeld, Germany*, March 2007. 2

[15] D. Scharstein and R. Szeliski. Middlebury online stereo evaluation. http://vision.middlebury.edu/stereo. 1

[16] M. Shimizu and M. Okutomi. An analysis of subpixel estimation error on area-based image matching. In *DSP 2002*, pages 1239–1242, 2002. 4

[17] M. Sizintsev. Hierarchical stereo with thin structures and transparency. In *Canadian Robot Vision*, pages 97 – 104, 2008. 2, 4

[18] P. Steingrube, S. Gehrig, and U. Franke. Performance evaluation of stereo algorithms for automotive applications. In *ICVS 2009*, pages 285–294, October 2009. 1

[19] J. I. Woodfill and et al. The tyzx deepsea g2 vision system, a taskable, embedded stereo camera. In *Embedded Computer Vision Workshop*, pages 126–132, 2006. 2

[20] Q. Yang and et al. Real-time global stereo matching using hierarchical belief propagation. In *British Machine Vision Conference (BMVC)*, pages 989–998, September 2006. 2

[21] E. Zinner, M. Humenberger, K. Ambrosch, and W. Kubinger. An optimized software-based implementation of a census-based stereo matching algorithm. In *International Symposium of Visual Computing (ISVC), Las Vegas*, 2008. 3