A High-Performance Dense Block Matching Solution for Automotive 6D-Vision

Henning Sahlbach, Sean Whitty, Rolf Ernst
Institute of Computer and Network Engineering
Technische Universität Braunschweig, Germany
sahlbach|whitty|ernst@ida.ing.tu-bs.de

***Abstract*—Camera-based driver assistance systems have attracted the attention of all major automotive manufacturers in the past several years and are increasingly utilized to differentiate a vendor's vehicles from its competitors. The calculation of depth information and Motion Estimation can be considered as two fundamental image processing applications in these systems, which have already been evaluated in diverse research scenarios. However, in order to push these computation-intensive features towards series integration, future in-vehicle implementations must adhere to the automotive industry's strict power consumption and cost constraints.**

As an answer to this challenge, this paper presents a high-performance FPGA-based dense block matching solution, which enables the calculation of both object motion and the extraction of depth information on shared hardware resources. This novel single-design approach significantly reduces the amount of logic resources required, resulting in valuable cost and power savings. The acquired sensor information can be fusioned into 3D positions with an associated 3D motion vector, which enables a robust perception of the vehicle's environment. The modular implementation offers enhanced configuration features at design and execution time and achieves up to 418 GOPS at a moderate energy consumption of 10 Watts, providing a flexible solution for a future series integration.

I. Introduction

Current urban traffic scenarios pose challenging tasks for the drivers of today's vehicles, which will worsen in the future due to increasing traffic density. In order to support the drivers, automotive research has focused on the development of different sensor types with associated Advanced Driver Assistance Systems (ADAS), which are already capable of driving autonomously in predefined scenarios. As the executed ADAS applications often come with massive computation requirements, prototypic systems usually combine their sensors with several high-performance PCs. However, as these research systems cannot meet the severe cost and power constraints of production vehicles, specialized hardware-accelerated implementations are inevitable.

As a consequence, various Graphics Processing Unit (GPU) solutions, Field Programmable Gate Array (FPGA) implementations and other automotive accelerators [1] have been presented in the last few years. Because of their flexible interconnect, the ability of in-field updates, and high processing performance at modest power consumption, FPGAs provide several valuable advantages for an automotive use case when compared to other accelerators. However, only a select few FPGA architectures provide support for run-time reuse for multiple applications [2]. Instead, they focus on static execution of one fixed application, resulting in considerable resource, cost and power overheads due to the requirement of multiple devices.

978-3-9810801-8-6/DATE12/

In the considered application class, Stereo Vision (SV) and Motion Estimation (ME) are important representatives with more than one algorithmic implementation, including dense block matching in both cases. Exploiting this common variant, this paper presents an efficient implementation of both applications on shared hardware resources. The implementation is based on an existing stream processing architecture, which enables a modular design approach and run-time reuse of hardware components [3], and achieves high-performance processing of 418 GOPS at modest power consumption of 10 Watts. From the ADAS perspective, the obtained processing results can be utilized for the construction of 6D image coordinates, consisting of a 3D pixel with associated 3D motion information. Similar to other vision approaches [4], these coordinates enable a robust perception and reconstruction of the vehicle's environment, which is crucial in the targeted application domain.

The rest of the paper is organized as follows. Section II gives an overview of existing approaches for ME and SV, followed by the presentation of the selected variant and its implementation in Section III. A thorough application evaluation is presented in Section IV and Section V concludes the paper.

II. Algorithm Evaluation

In order to guarantee the suitability of the selected algorithmic approach, a comparsion with other automotive solutions was conducted before implementation. For the reconstruction of the 3D coordinate of a certain world point out of 2D image information, a matching in at least two camera image planes is necessary. In the past, different matching approaches have been presented. Usually, a cost table for different disparities is constructed, which is typically the most computation intensive task. For the optimization of the obtained disparity fields, different approaches such as a simple winner-takes-it-all algorithm or global optimization techniques, graph-cuts or semi-global matching are implemented, which are compared in an existing evaluation database [5]. For the calculation of disparity costs, two major techniques are the census transformation [6] and derivatives of the sum-of-absolute differences metric [7], which are also used in our solution.

For ME, several methods have been developed in the past, which typically are based on differential or block matching techniques. In the field of differential algorithms, most solutions use optical flow or gradient calculations for the determination of motion [8], [9]. Several hardware implementations

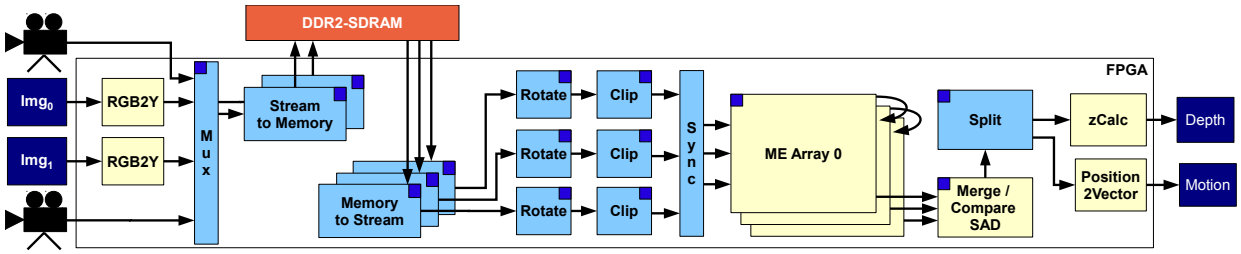


Figure 1. Dense Matching Architecture

are based on these two differential approaches [10]; however, these implementations typically relax the optimization criterion or do not perform sufficient iterations to refine the flow vector field to an acceptable quality level as found in software solutions. Block matching techniques are typically used in the field of video encoding and frame interpolation. Thanks to the regular computational structure, they are well-suited for hardware implementations [11]. In contrast to video encoding, ADAS require the most accurate match for each pixel and even sub-pixel accuracy to enable a robust and exact classification of moving objects. To our knowledge, this paper's dense block matching solution is currently the only full search FPGA implementation achieving the required processing performance for this ambitious automotive use case.

III. Dense Block Matching

For our implementation, the combination of a full block search variant and Sum of Absolute Differences (SAD) as the matching kernel has been selected in order to achieve high-quality processing results. This variant comes with massive computation requirements, as up to 10^5 algorithmic operations must be executed for each pixel, depending on block size and search area. The basic formula for the calculation of a single $SAD(x, y, s_x, s_y)$ for a given block B with origin x, y and search offsets s_x, s_y is presented in Equation 1,

$$SAD(x, y, s_x, s_y) = \sum_{i=x}^{x+B} \sum_{j=y}^{y+B} |b_0(i, j) - b_1(i + s_x, j + s_y)| \quad (1)$$

with $b_0(i, j)$ as the reference block pixels and $b_1(i+s_x, j+s_y)$ as the search block pixels provided by the previous or next frame. The minimum SAD is then utilized to determine the best match $M(x, y)$ in a defined search area (Equation 2).

$$M(x, y) = min\{SAD(x, y, s_x, s_y) | \forall s_x \in S_x, \forall s_y \in S_y\} \quad (2)$$

with $S_x, S_y \in [-N..N]; N \in \mathbb{N}$. 2D motion vectors dx, dy can then be derived from the SAD's position in the search area.

For the calculation of depth information, a second stereoscopic camera is added. Using dense block matching for the identification of corresponding pixels in the two rectified stereo frames enables the calculation of the pixel's distance $z(x, y)$ via the intercept theorem (Equation 3),

$$z(x, y) = \frac{b * f}{d(x, y)} \quad (3)$$

with b as the distance of the cameras, f as the camera's focal length and $d(x, y)$ as the pixel's disparity, which is derived from the horizontal component of the motion vector.

Using a pixel's 2D motion vector dx_n, dy_n and depth information from two neighboring frames n and $n-1$, the missing motion vector dz_n can be easily calculated in a post processing stage as

$$dz_n = z(x, y)_n - z(x - dx_n, y - dy_n)_{n-1}. \quad (4)$$

A. FPGA-based Architecture

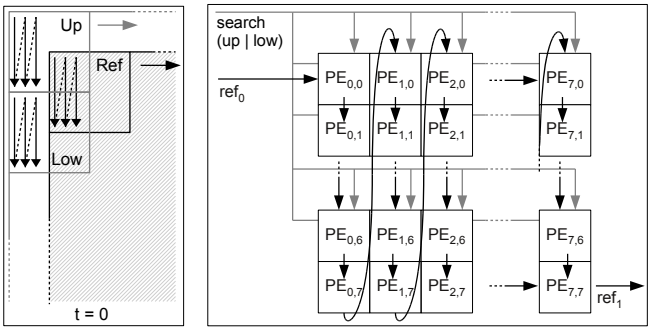
The general architecture of the implementation can be seen in Figure 1. Grayscale image data is either directly received from automotive cameras or can be transferred from an external PC harddisk via PCI-Express (PCIe), combined with on-chip colorspace conversion modules. As ME requires two succeeding frames, the images must be stored in an external framebuffer with two write and three read ports, which implement the required block-based memory accesses. This is achieved by run-time programmable address generators, which enable the execution of multiple address patterns for the two different applications. All programmable elements, which are marked by small boxes in the element's corners in Figure 1, are customizable by element-specific instructions, which can be efficiently modified at application run-time via a dedicated parameter interface. This technique, which is explained in detail in [3], enables efficient run-time modification and reuse of hardware elements in multiple designs and applications.

Next, image data is rotated and image clipping and padding is performed in order to fill the image borders of all three memory streams. Block matching is then performed in a cascade of matching arrays, which are explained in more detail in Section III-B, and whose results are merged in a dedicated compare/merge unit. As each application has a unique post-processing stage, result data is split into two different branches, which also return the final processing results to an external PC. Except for the post-processing stages, both applications utilize the same hardware resources, resulting in an efficient implementation with a high degree of hardware reuse.

B. Matching Array Implementation

The proposed block matching implementation is based on an existing sparse matching approach [12], which is enhanced with several valuable extensions: the Processing Elements (PE), which are presented in Section III-C, have been redesigned and are now able to calculate dense block matchings, the search area size can be easily customized and extended in horizontal and vertical directions, and the run-time programmable approach enables a utilization in multiple application scenarios. Furthermore, the implementation scales efficiently in space and time, exploiting recent FPGAs' fast clock frequencies via time-multiplexed reuse of matching arrays and large FPGA chip areas via the cascaded assembly of arrays, each covering a certain part of the search area.

Figure 2b shows an example configuration with one array and a block size of $B=8$. The array is organized in a systolic



(a) Array Data Flow (b) Dense Matching Array with block size B=8

Figure 2. Matching Array Architecture

fashion and contains B^2 elements, which are chained in a deep pipeline. Reference pixels are forwarded between the elements with a one cycle delay, while search pixels are transmitted to each PE synchronously. Inside the array, each single PE represents a specific matching position, which results in B^2 different block matchings per array. Depending on the PE's vertical position in the array, the upper or lower part of the search window are required to calculate the correct block matching, which also results in the need for a third image stream (Figure 2a). The previously performed rotation of pixel blocks from a row- to a column-wise representation enables the exploitation of data localities in the pipeline as matching columns are reused for the calculation of multiple matchings inside the array. This results in a gapless pixel stream within one complete frame, as overflow effects occuring at the image borders can be disregarded.

C. Processing Element Architecture

Besides an efficient internal organization, the architecture of the PEs is a key aspect for a high-performance implementation, which is due to the large amount of PEs per array and the requirement for a fast and area efficient implementation in the automotive context. The VLSI architecture of the PEs is shown in Figure 3. The PEs consist of one subtractor for the calculation of the absolute difference and up to B accumulators. These are organized in a shift register fashion, each storing an SAD for a different position. At reset, each accumulator forwards its intermediate result to the next stage and the element outputs the result of the last accumulator into its output multiplexer. In the output stage, all results of one vertical matching column are multiplexed and forwarded to a second multiplexing level that merges associated SADs for the following selection of the best matches.

D. Array Customizations

An automotive block matching solution demands a high degree of flexibility in order to cover different environmental situations. The required search area can vary for each camera device, its position, the traffic scenario and the vehicle's velocity. Furthermore, the matching quality is influenced by various parameters such as the pixel bit width, which also are related to the performance of the implementation and the amount of logic resources consumed.

In order to obtain a flexible yet efficient hardware implementation, all parameters with a strong influence on the required chip area such as the image block size or the

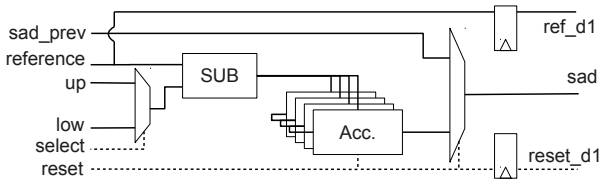


Figure 3. Processing Element Architecture

amount of parallel matching arrays, can be customized at synthesis time via VHDL generics. Secondary parameters such as the SAD accuracy or the bit width of the grayscale pixels also influence the array size and matching quality, and are customizable as well. In addition, the implementation provides several run-time configurable parameters such as the degree of time-multiplexed reuse, which can be controlled via the flexible address generators. Using these parameters, multiple configurations become feasible, a selection of which is evaluated in the next section.

IV. Experimental Results

In order to quantify the general processing performance and latency of the implementation and to evaluate the benefits of the shared resource approach, several experiments have been conducted. All presented variants have been implemented on a Xilinx Virtex 5 FPGA (XC5VSX240T) running at 125 MHz. For all experiments, a sequence of RGB images with an automotive image size of 512x384 pixels and 10 bits per color component has been used as input data. During the tests, the execution time is measured after each 250 images and an average value for the frames per second is calculated based on all received results.

All implemented variants are based on a block size of 8x8 pixels. The performance and latency of the shared resource approach is compared with separated versions of both of the selected applications, and a sparse matching version, which represents a typical video processing use case. Finally, the implementation's power and cost efficiency is evaluated with regard to automotive constraints.

A. Performance Evaluation

The performance results of the different implementations are summarized in Table I. All benchmarked variants implement a cascade of three matching arrays, which are also reused in a time-multiplexed fashion for three times if applicable. Sparse ME (line 1), which calculates only one matching position per block, achieves the best results in terms of frames per second, which is due to the reduced computation requirements when compared with a dense ME implementation calculating B^2 positions.

The dedicated implementations of Dense ME (line 2) and SV (line 3) are able to fulfill the frame rate requirement of 24 FPS and show high processing performance of several hundred GOPS, which specifies the amount of executed integer operations within one second. Compared to the standard SV application using two cameras, a modified Motion Stereo version (line 4) based on two succeeding frames from one monocular camera, provides minor performance advantages. This is explained by a reduced amount of memory transfers, as only one instead of two parallel image streams need to be buffered. This comparison also shows the general influence

	Application	FPS	GOPS	Latency in μs
1	Motion Estimation (Sparse)	211.00	72.50	11.12
2	Motion Estimation (Dense)	26.86	417.69	77.14
3	Stereo Vision	76.65	397.27	5.9
4	Motion Stereo	80.53	417.51	4.3
5	Motion + Stereo	20.15	418.00	120.4

Table I
PERFORMANCE RESULTS: BLOCK SIZE 8X8, SEARCH AREA SIZE 32X32

of memory performance in the targeted application domain, which often is the limiting factor.

The joint implementation of both parts (line 5) achieves the best performance in terms of GOPS, but slighty fails to achieve the original frame rate requirement. However, this can be attributed to the comparatively slow application clock frequency of 125 MHz, which does not represent the performance limit of the matching array. Therefore, it is expected that the existing small gap can be easily closed via a slightly faster array clock frequency in future application revisions.

The last column of Table I presents the latency results of the implementations, which is the measured time between the start of a memory read transfer and the availability of the corresponding results at the FPGA output. For an automotive setup, all latencies are not considered critical, as the application latency is dominated by the image capturing process, which typically requires up to 40 ms. When compared to all other implementations, the SV latencies differ significantly. This is related to the absence of run-time reuse, resulting from the fixed mounting of the cameras on the same vertical axis.

B. Power and Cost Considerations

As mentioned before, power consumption and hardware costs are important aspects for automotive computer architectures, as systems for series vehicles must adhere to strict constraints in both areas. In order to evaluate these aspects, Table II presents the chip area requirements, the FPGA power dissipation, and the computation density for all considered applications. The power consumption figures have been calculated by Xilinx XPower, which provides an analysis of FPGA power consumption for Xilinx devices.

With the exception of Sparse ME, which exhibits reduced application complexity, all implementations require approximately the same amount of resources. This is explained by the same physical size of the matching array, which consumes the largest part of the chip area. Regarding the power consumption, all implementations show similar values, which is related to the selected Virtex 5 FPGA device that does not provide power saving techniques such as clock or power gating of unused resources. It is expected that the power consumption figures for recent low-power FPGA generations providing these features would be significantly smaller and correlated to the consumed chip area.

The last column presents the computation density of the implementations expressed in GOPS/W. Compared to the dedicated implementations, the joint solution of SV and ME reaches similar performance results, which clearly demonstrates the efficiency of the shared resource approach. Furthermore, a direct comparsion with two separated implementations, each running on a dedicated chip (line 6), reveals the advantages of the presented solution (line 5) for the automotive domain: one single chip, which still achieves the required processing performance via a high computation density, consumes significantly less electrical power, requiring only one single hardware device for the implementation of two distinct applications.

	Application	Chip Area	Power Cons.	GOPS/W
1	Motion Estimation (Sparse)	27 %	10.19 W	7.11
2	Motion Estimation (Dense)	43 %	10.48 W	39.86
3	Stereo Vision	44 %	10.46 W	37.98
4	Motion Stereo	43 %	10.47 W	39.88
5	Motion + Stereo	45 %	10.56 W	39.58
6	Motion + Stereo add.	43 % + 44 %	20.95 W	38.90

Table II
POWER CONSUMPTION EVALUATION

V. CONCLUSION

In this paper, we have presented a high-performance dense block matching solution for automotive image processing, which permits the extraction of depth as well as motion information. These results can be fusioned into 6D image coordinates, which enable robust object detection and environment perception. Major parts of both considered applications are executed on shared hardware resources, which is achieved by flexible computation and data transport elements that enable run-time reuse of logic resources. This results in high computation density and an efficient implementation in terms of chip area and power consumption, which are both crucial factors for the future integration of such systems in cost- and energy-constrained series vehicles. Although several aspects such as safety questions or matching quality improvements remain unexplored, the general applicability of the presented novel single-chip approach has been successfully demonstrated, showing a viable path for an application integration in future series vehicles.

REFERENCES

[1] S. Kyo and S. Okazaki, "In-vehicle vision processors for driver assistance systems," in *Proc. of ASP-DAC '08*, 2008.

[2] C. Claus, W. Stechele, and A. Herkersdorf, "Autovision – A Run-time Reconfigurable MPSoC Architecture for Future Driver Assistance Systems," *Information Technology*, vol. 49, pp. 181–187, 2007.

[3] A. do Carmo Lucas, H. Sahlbach, S. Whitty, S. Heithecker, and R. Ernst, "Application development with the FlexWAFE real-time stream processing architecture for FPGAs," *ACM TECS, Special Issue on Configuring Algorithms, Processes and Architecture (CAPA)*, vol. 9, p. 23, 2009.

[4] U. Franke, C. Rabe, H. Badino, and S. Gehrig, "6d-vision: Fusion of stereo and motion for robust environment perception," in *Pattern Recognition*. Springer Berlin / Heidelberg, 2005.

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

[6] K. Ambrosch and W. Kubinger, "Accurate hardware-based stereo vision," *Computer Vision and Image Understanding*, vol. 114, no. 11, pp. 1303–1316, Nov. 2010.

[7] S. Gehrig, F. Eberli, and T. Meyer, "A real-time low-power stereo vision engine using semi-global matching," in *Computer Vision Systems*. Springer Berlin / Heidelberg, 2009.

[8] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," *Artifical Intelligence*, vol. 17, pp. 185–203, 1981.

[9] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of Imaging Understanding Workshop*, 1981.

[10] P. Cobos and F. Monasterio, "Fpga implementation of the horn & shunk optical flow algorithm for motion detection in real time images," in *Proc. of DCIS' 98*, 1998.

[11] M. Kim, I. Hwang, and S. Chae, "A fast vlsi architecture for full-search variable block size motion estimation in mpeg-4 avc/h. 264," in *Proc. of ASP-DAC '05*, 2005.

[12] C. Sanz, M. J. Garrido, and J. M. Meneses, "VLSI Architecture for Motion Estimation using the Block-Matching Algorithm," in *EDTC*, 1996, p. 310.