Proceedings of the
2007 IEEE Intelligent Vehicles Symposium
Istanbul, Turkey, June 13-15, 2007

Application Development of Camera-based Driver Assistance Systems on a Programmable Multi-Processor Architecture

Axel Techmer

Abstract— **The VIP-II, a new multi-processor architecture developed by Infineon, provides the required performance for future camera-based advanced driver assistance systems. The integration of data-processing, control and communication, an optimized set of peripherals and low power consumption will help to reduce the overall systems costs required for a future wide market penetration. The VIP-II software tool-chain supports a PC-based application development process from a first functional reference design up to further development steps including parallel programming. The paper proposes a multi-threading processing framework for image feature extraction and motion estimation. Performance measurements show an achievable performance in the range of today's desktop processors.**

I. Introduction

Advanced driver assistance systems (ADAS) are one of the emerging automotive markets, driven by a wide public and industrial interest in improved safety and comfort. Applications like lane departure warning, automatic cruise control, parking aids or night vision have already been introduced. Further and more advanced applications like pedestrian detection, emergency brakes or automatic lane change assistance are under development. ADAS sense the driving environment to provide information or vehicle control, where several sensors per vehicle are necessary to provide a complete understanding of the current traffic situation. Today products are available for commercial cars or as options in the luxury passenger vehicle segment. To address the full potential of the automotive market the reduction of the overall system costs is one important step.

Besides radar, lidar or ultrasonic cameras are one key technology for ADAS. In contrast to other sensors cameras produce a projection of the driving environment with a high spatial resolution, which can be directly displayed to the driver (e.g. night vision, rear view cameras) or interpreted for lane departure warning, blind spot detection, pedestrian detection etc. But the interpretation of images with high spatial resolution requires powerful data processing capabilities. Today, prototypes and products of camera-based ADAS are based on high performance general purpose DSPs, combinations of FPGAs and DSPs as well as application specific processors ([6], [5], [2]). These solutions point out the need of a computational performance in the range of desktop processors. At the same time the automotive embedded systems must fulfil automotive requirements like automotive qualification and reduced power consumption. For all approaches a reduced power consumption results from parallel processing capabilities (VLIW, SIMD, multi-core) and decreased processor frequencies.

Infineon researches new multi processor architectures for communication and video applications. The current prototype, which is further called VIP-II (VIP-II stands for the 2nd generation of Vision Instruction Processor), offers the required computational performance, supports SIMD as well as MIMD operations, and fulfils low power requirements as it is also designed for handheld devices. Pure SIMD architectures, proposed in [2] and [6], can only be used for low level processing. In contrast to this the VIP-II supports the complete processing chain. Due to its low power consumption and optimized interfaces for camera applications, a cost advantage to general purpose processors is expected. On the other side parallel programming at the data and task level is required to benefit from the VIP-II's SIMD and MIMD support. The paper will discuss how camera-based application development is supported by the development environment of the VIP-II. An application example - including a motion estimation approach based on a tracking of feature points - is presented.

Section II and III describes the main features of new multi processor architecture and its software environment. Section IV discusses the process of application development. The paper ends with summary and outlook in section V.

II. Processor Architecture

Application software for camera-based ADAS shows different kinds of parallel processing. Figure 1 illustrates the principle processing chain. First steps of the image processing chain are data intensive. The same instructions are performed at every pixel to identify significant image parts or feature points. This can be expressed by SIMD-instructions (single instruction multiple data). Further processing steps cover object detection, tracking and classification up to application specific situation analysis. In contrast to the first steps the processing is now based on a reduced but more abstract dataset. The required algorithms

Axel Techmer is with the Corporate Research of the Infineon Technologies AG, D-81730 Munich, Germany; phone: ++49 89 234 40055; e-mail: axel.techmer@infineon.com

1-4244-1068-1/07/$25.00 ©2007 IEEE.

are more complex and control-based. Parallelism can be found on object level (e.g. tracking of several vehicles parallel to lane detection and/or traffic sign recognition).

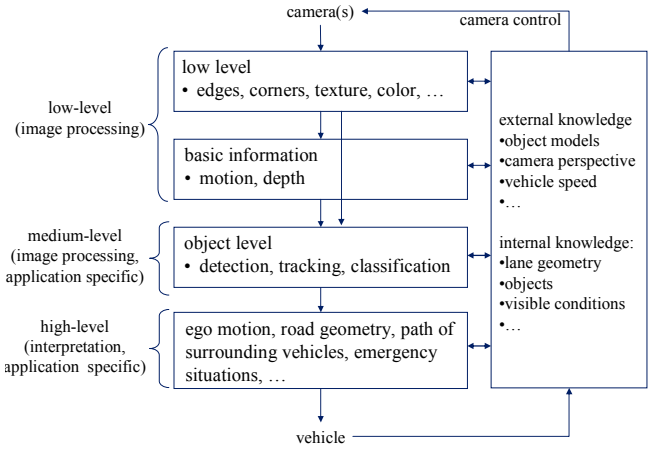

Figure 1: General processing chain for automatic video surveillance systems.

The VIP-II integrates all processing modules required for camera-based ADAS (see figure 2.). Cameras can be connected via the video interface. The video interface provides parallel and serial (LVDS) data links for input and output. The interface supports a transfer to and from the on-chip memory at a data rate of up to 22MSamples/s without any control of the SIMD cores. A sample corresponds to a 8 or 16 bit data values. Higher data rates are possible, but requires control of the SIMD cores. The VIP-II prototype offers 512kByte on-chip memory. Additional external memory is accessible via a 16bit wide external bus unit. To support the complete processing chain the VIP-II provides a multi processor approach to support SIMD as well as MIMD operations. The principle architecture of the VIP-II is illustrated in figure 3.

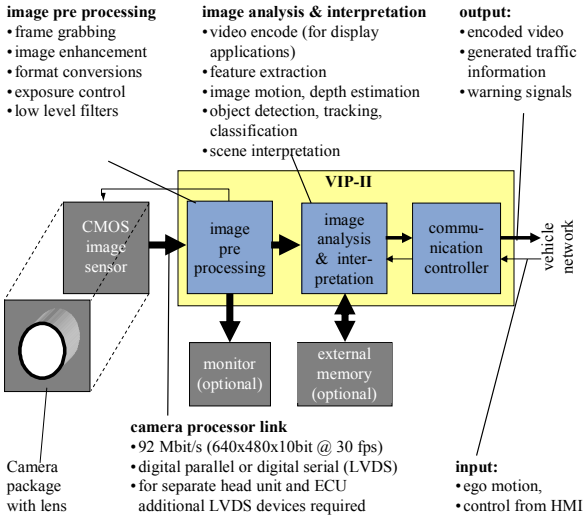

Figure 2: Components of a camera system for ADAS. The VIP-II is designed to integrate all processing modules form image capturing to vehicle communication.

The multi-processor approach based on a cluster of multi-tasked SIMD cores is suited for the data-intensive low-level operations as well as for medium- and high-level operations. This scaleable approach supports different performance requirements of different applications as well as a further evolution of applications.

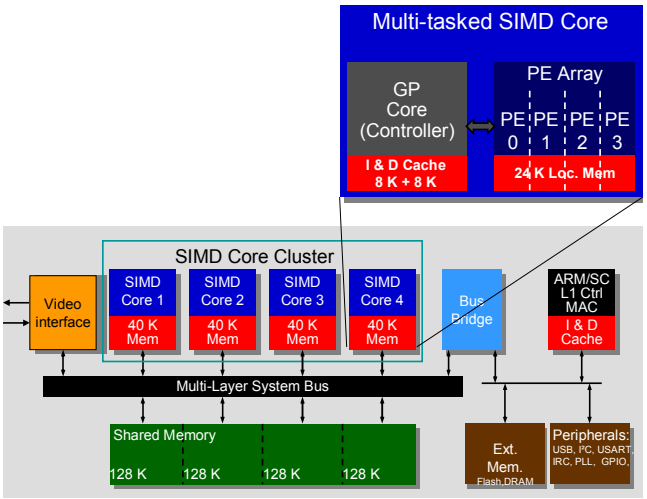

Figure 3: Principal architecture of the VIP-II

An additional general purpose processor core, the ARM9, is linked via a bus bridge to the SIMD-cluster. The ARM9 was integrated with a rich subset of its standard peripherals. For ADAS applications the ARM9 can be used for communication tasks and the main control-flow.

A multi-layer bus system provides simultaneous memory accesses of the SIMD cores. The first prototype consists of a cluster of four SIMD cores.

The SIMD core is based upon previous work [3]. Each SIMD core contains four processing elements and operates with a clock frequency of 300 MHz. It supports special instructions like saturating operations and finite-field arithmetic, and long-instruction word (LIW) features for performing arithmetic operations and memory accesses in parallel. The execution pipeline is four stages long, which is used to relax the timing requirements for the memories and to reduce the memories' supply voltage and thereby the power consumption. However, a long pipeline reduces the performance in case of data dependencies between instructions from the same task. That is why here each of the pipeline stages contains an instruction from one of four separate tasks. Therefore, each SIMD core can be seen as a multi-tasked SIMD core. To execute four independent tasks on one core without any penalty due to context switching, the instruction caches, the local memories and the registers are replicated to store four different contexts at the same time. Every clock cycle a new context is selected in a round-robin schedule. A general purpose core inside every SIMD core controls the processing of the four processing elements. Corresponding to the multi-tasked SIMD core a multi-tasked general purpose processor is required, which can also be used for multi-tasked sequential processing.

Form the programmer's perspective the VIP-II can be seen as a cluster of 16 independent task-cores (four multi-tasked SIMD cores, where each core can handle four tasks in parallel). As the system runs with 300MHz the remaining performance of a single task-core is 75MHz. Each task-core consists of a general purpose core and an array for four processing elements. The current prototype provides a computational performance of 14.400 MIPS (including the PE-array), 3000 MIPS (without the PE-array) and 9.600 MMACs (16b input and 32b output). It is designed in 90nm CMOS technology and available in a BGA package (12mm x 12mm) with 310 pins. The power consumption is expected to be lower than 500mW (the first VIP-II prototype is currently under test.).

III. Software Environment

The general purpose controller of the SIMD cores and the additional ARM9 support "C" programming. The complete "C" based tool chain, a library for image processing and an operating system are available. Figure 4 illustrates the compilation process. Because "C" does not support data parallel programming a SIMD-compiler based DPCE language was developed (data parallel C extension). The SIMD-compiler supports the declaration of multi-dimensional objects and their operations in a "C" oriented way. Assembling on GP cores, the PE array and the ARM9 is also supported.

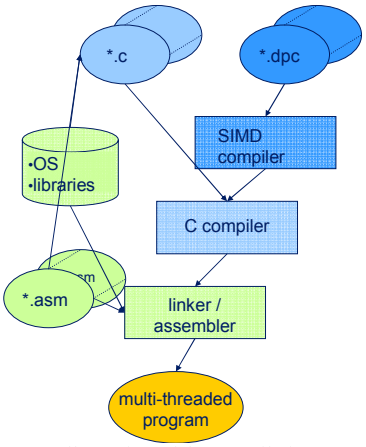

Figure 4: Illustration of the compilation process

Application development requires a real-time operating system (RTOS) running on the general purpose core and on each SIMD core. The RTOS contains all the necessary functions for thread creation and synchronization, interrupt handling, access to peripherals, and input/output. The operating system is available as a "C" library. Each core of the VIP-II is able to execute the main thread and to create and control further threads. Currently no dynamic scheduling is supported. Therefore, the thread schedule must be defined before execution time. Scheduling can be developed and simulated on a standard PC running with an operating system with a multi-threading functionality. If the required functions of the VIP-II's operating system are renamed to the functions of the PC's operating system, the same "C"-sources of the multi-threaded application can be used in both environments.

The API provides access to a library of optimized image processing functions, peripherals, and operating system functions. The image processing library follows Intel's "Integrated Performance Primitives" library (IPP), by keeping a close relation to the IPP API. The image processing library provides a broad set of often used low-level operators for image processing, which can easily be integrated into "C"-programs. The image processing library of our vision processors can be easily mapped on Intel's IPP. This allows a comfortable and efficient application development in a PC-environment.

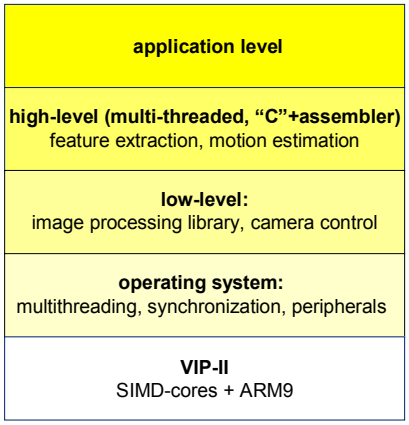

Figure 5: VIP-II's software architecture

On higher application levels reference examples explain the way of multi-threaded programming and provide a framework for general application tasks like frame capturing, external memory access, thread synchronization and data output.

IV. Application Development

To illustrate the application development process on the VIP-II's multi-processor architecture a reference application example is discussed. The application covers the low-level processing step of the image processing chain and performs an extraction of different image features (edges, corners, lane markers) and a motion estimation on corners (see figure 6). Further high level application parts could be integrated into the proposed processing framework based on the extracted feature set.

Images are processed on the fly (without storing a complete image), only the extracted features with their assigned information like motion trajectories are stored. Therefore, the implementation offers low memory consumption and requires only the on-chip memory. This could be a benefit for a future embedded solution

To develop this reference application the following

approach was used:

1. A first implementation was programmed in "C" on a standard PC.
2. Multi-threading and data parallel operations were introduced in the PC-based implementation:
   a. The multi-threading support functions of VIP-II's operating system were mapped on Window's multi-threading library. Only these functions were used to generate a multi-threading program on the PC.
   b. The functions of VIP-II's image processing library were mapped on Intel's IPP. Only these functions were used to introduce data parallel processing on the PC. As a side effect the performance of the PC-based implementation was increased.
3. The PC-based implementation generated in step 2 was compiled with the VIP-II tool chain and executed on the VIP-II's virtual prototype. With the limitation of a file based input and output, no additional modification on the source code is required. The VIP-II's virtual prototype produces cycle-accurate performance values.
4. Depending on the performance results of step 3 optimization steps of the multi-threading program became necessary. This was first implemented and verified on the PC and afterwards on the virtual prototype.
5. The application is than loaded and performed on the real hardware.

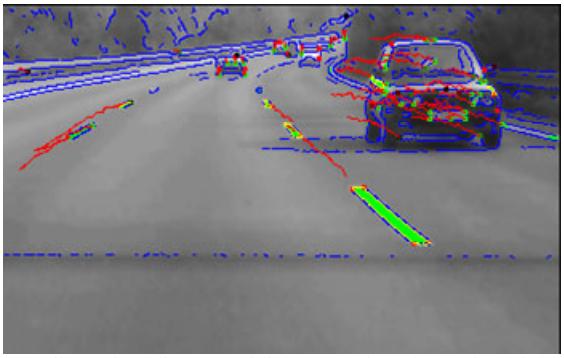

Figure 6: Illustration of extracted features and motion trajectories: edges (blue); corners and lane markers (green); trajectories (red)

The reference application extracts three different features:

- **Edges** correspond to a significant change of the local image contrast. Typically, they are detected as local maxima of the image gradient (see figure 7). The result of this edge detection approach is the information at every pixel, if it is an edge or not. Additionally, the magnitude and the direction of the image gradient for each edge point are provided. The gradient magnitude can be used as a confidence value. Figure 6 displays the result of edge detection. [4] provides further details of the implemented approach.
- **Corners** correspond to pixels where the change of the contrast describes a geometrical corner. The implemented corner detection uses an approach of Harris and Stephens [1]. The implemented approaches of edge and corner detection uses the same preprocessing (image smoothing and the computation of the image gradient; see figure 7). Similar to edges a value is computed, which can be used as a confidence value for corners. Figure 6 shows the result of the corner detection approach.
- **Lane Markers:** Particularly for camera based driver assistance systems the detection of lane markers and the road geometry is a basic processing task. Typically, lane markers are identified as dark-bright-dark contrast changes. The implemented approach detects these contrast changes only in row direction and only at pixels with significant image intensity. The implementation uses the output of the image gradient computation, which is also required for edges and corner detection. Figure 6 shows the result of the implemented approach.

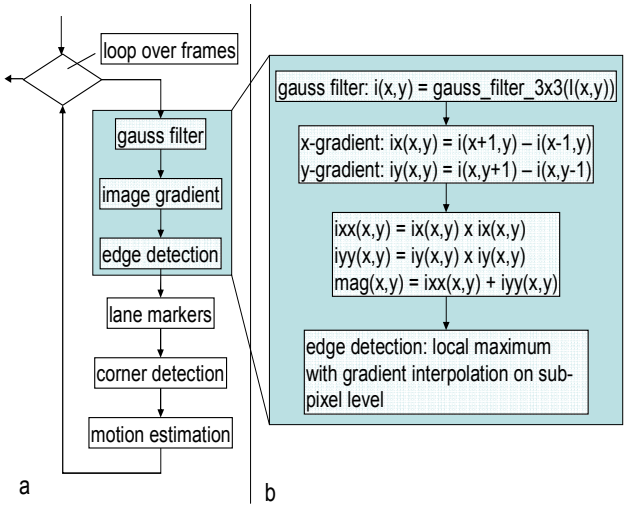

Figure 7: Illustration of the main processing steps of the reference application (a) and the required operations for edge detection (b)

In addition to the extracted features, corners are tracked over time to provide motion information to higher application software levels. As the implemented corner detection approach produces stable corners over time, a simple and fast tracking algorithm is used. For every frame the detected corners are saved as a list of points including a description of the image neighborhood around this corner pixel. To find a corresponding corner in the previous frame, the list for previous corners is scanned. The corner, which posses the highest similarity of the local image descriptor and which fulfills a distance constraint, is defined as a predecessor of a new corner. The motion trajectory is updated and linked to this new feature point. The result of motion estimation is also displayed in figure 6.

Figure 7 displays the main processing step of the presented feature extraction approach. Data parallel operations are easily identified in the image based processing steps of the Gaussian filtering, the image gradient computation and the following edge detection step. Corresponding library functions are provided in Intel's IPP library as well as in the VIP-II's image processing library.

The final local maximum detection to define single pixels as edges is implemented in standard "C" on the PC as well as on the VIP-II.

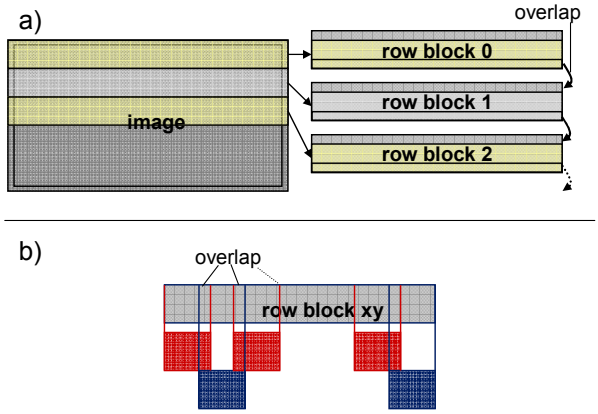

Figure 8: The camera image is divided and processed in single row-blocks (a); every row-block is divided in several sub-blocks, which are processed in parallel (b).

In contrast to use of data parallel processing power by including optimized library functions it is much more challenging to use the multi-thread capabilities. To reduce the required memory of the final application the camera image is divided and processed in single row-blocks (see figure 8). At least the memory for two row-blocks is hold in the on-chip memory, where one row-block is required for reading pixels from the camera interface and the other row-block is required for processing. This allows reading and processing in parallel. Because feature extraction requires local neighborhood operations an overlap of the row-blocks must be considered (see figure 8).

For further multi-threaded processing each row block is split into several sub-blocks. Each sub-block is processed on a single task-core of the VIP-II. Therefore the maximum number of sub-blocks per row-block should be 16 according to the number of available task-cores on the VIP-II. Due to the real-time constraints of a specific application lower numbers of sub-blocks are reasonable. According to the problem of local neighborhood operations as mentioned above an overlap of the sub-blocks must also be considered (see figure 8).

Figure 9 shows a general block diagram of the multi-threading program. The main thread, which is executed on one of the SIMD-task-cores, runs some initialization steps including the creation of the processor threads and the corresponding synchronization objects. The main thread reads the incoming pixels from the camera interface and distributes them into a memory structure, which is prepared for the following parallel processing. After all pixels of one row-block are captured, the sub-blocks are ready for processing and the corresponding synchronization object is updated accordingly. Triggered by this synchronization signal the worker-threads start the processing of their sub-blocks. Parallel to the worker-threads the main-thread continues to read and distribute the incoming pixels to a second memory field. This second memory field is necessary to dissolve accesses from worker-threads and the main-thread. After all row-blocks are processed the main-thread waits until the worker threads have stored the extracted features to a common data structure. Then the schedule is repeated for the next frame.

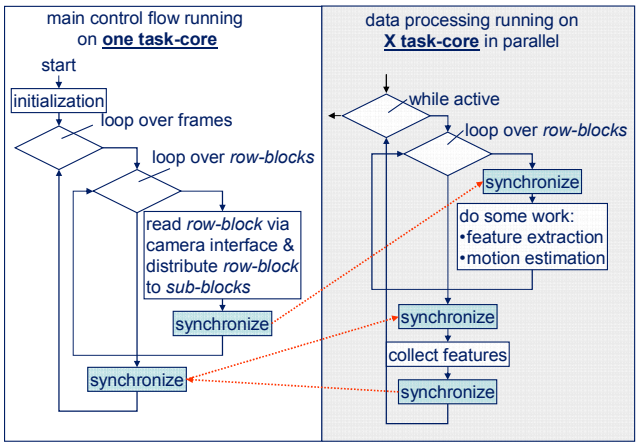

Figure 9: Scheduling and synchronization of tasks for parallel feature extraction in sub-blocks.

Figure 10 displays performance results of the proposed multi-threading feature extraction approach. The figures are based on VGA resolution and the processed data is 8 or 16bit wide. For the VIP-II implementation, the image was divided in 10 row-blocks and 16 sub-blocks. The performance measurement includes all synchronization overhead. On the Pentium the image was processed as one single block to minimized effects of thread synchronization. Furthermore Intel's IPP functions were used to exploit the full computational power of the Intel architecture. The figures demonstrate that the processing power of the VIP-II exceeds the Pentium IV by a factor in the range of 2 to 6. It should be noted that the VIP-II operates at 300MHz. The four SIMD-cores provide computational performance of 1.2GHz, which is less then the 1.8GHz of the Pentium.

Computational performance on VGA resoluton (640x480)		
	VIP-II [ms]	PIV, 1.8GHz [ms]
basis image operations		
copy (16bit)	0,35	1,92
add (a+b=c, 16bit)	0,57	3,02
sub (a-b=c, 16bit)	0,57	3,04
mul (a*b=c, 16bit)	0,58	3,02
edge detection		
3x3 image filter (8 bit)	0,80	2,00
image gradient (x and y, 16bit)	1,20	4,23
gradient magnitude (16bit)	1,56	6,94

Figure 10: Performance figures of VIP-II versus Pentium IV. The performance values of the PC are based on Intel's IPP implementation running on a 1.8GHz Pentium IV processor.

V. SUMMARY AND OUTLOOK

First camera-based ADAS have been introduced into the automotive market in commercial cars as well as an option in the luxury passenger vehicle segment. To expand the benefit of these new safety features as a standard option over all vehicle segments, the system costs must be reduced. Here, new processor architectures, which offer the required computational performance and decrease the systems costs, will be a significant factor. The VIP-II, a new multi-processor architecture developed by Infineon, provides the required performance for advanced ADAS. The integration of data-processing, control and communication, the optimized set of peripherals and the low power consumption will reduce the overall systems costs. Because the processor is scalable in hardware as well in software, a wide range of varying applications could be supported by a future processor family strategy. The application developer will profit form the software scalability in terms of a reduced time to market.

On the other side the multi-processor approach requires multi-threaded programming as well as the introduction of data-parallel operations. The software tool-chain of the VIP-II supports a PC-based application development process. The PC can be used for the first functional reference design as well as for all further development steps (introduction of data-parallel and task parallel processing). Another useful aspect of development on PC is benchmarking of algorithms. In particular the available operation system and image processing library supports an application development on "C" level. The available virtual prototype of the VIP-II can be used to verify the application directly in a PC environment.

To explain the application development process the paper presents a multi-thread feature extraction framework. The achieved performance shows that VIP-II allows to integrated further processing steps without real-time conflicts. Furthermore the framework fits into the on-chip memory and allows smart system integration.

Future work will cover hardware and software aspects. On the hardware side the peripheral set will be further improved and optimized according to the requirements of camera-based ADAS. Application analysis to identify parallelism and a parallel programming support will improve the current software tool chain.

VI. ACKNOWLEDGEMENT

The work on the application demonstrations for the VIP-II is funded by the BMBF research projects KASS and AUTOSAFE.

VII. SREFERENCES

[1] Ch. Harris, M. Stephens, A Combined Corner and Edge Detector, 4th Alvey Vision Conference, pp 147-151, 1988

[2] W. Raab, N. Brüls, U. Hachmann, J. Harnisch, U. Ramacher, C. Sauer, A. Techmer, *A 100-GOPS Programmable Processor for Vehicle Vision Systems*, IEEE Design & Test of Computers, vol. 20, no. 1, Jan. 2003.

[3] W. Raab, H.-M. Blüthgen, U. Ramacher, *A Low-Power Memory Hierarchy for a Fully Programmable Baseband Processor*, 3rd Workshop on Memory Perfromance Issues WMPI-2004, June 20, 2004, Munich, Germany.

[4] A. Techmer, Hans-Martin Bluethgen, Cyprian Grassmann, Ulrich Hachmann, Wolfgang Raab, Ulrich Ramacher, *Embedded Vision Platform for Video Surveillance Systems,* 17th Symposium on Electronic Imaging, San Jose, January 16-20, 2005

[5] NEC Electronics, The IMAPCAR® parallel processor for image recognition and its contribution to the realization of pre-crash safety solutions for automobiles, NEC Electronics Web Magazine, Volume 63 (Dec 27, 2006), http://www.necel.com/en/channel/vol_0063/vol_0063.pdf

[6] Elchanan Rushinek, MobilEye, EyeQ™ Data Sheet, datasheet, Revision: 3.14, 2006 Rev.: 3.14 , http://www.mobileye.com/EyeQ_DataSheet.pdf