Stereo Vision on an ARM/DSP Multicore Platform based on Code Generation using the MATLAB Embedded Coder

Dennis Schuldt[1], Jörg Thiem[1] and Semir Mustedanagic[2]
[1]*Department of Information and Electrical Engineering, University of Applied Sciences and Arts Dortmund, Sonnenstraße 96, 44139 Dortmund, Germany*
[2]*Smart Mechatronics GmbH, Kronenburgallee 2, 44141 Dortmund, Germany*

Keywords: MATLAB, MATLAB Embedded Coder, ARM Cortex-A8, DSP C674x, VRmagic D3 Platform, Texas Instruments.

Abstract: Vision based techniques are by now well established means for contactless measuring applications, e.g. in industrial automation, optical inspection, medical imaging or robot navigation. However, the development and implementation of these applications on small-sized embedded systems is still challenging, because image processing algorithms require quite a lot CPU performance. Therefore, code optimization has to be considered in the development process. In this work we present our results on implementation and benchmarking of stereo disparity algorithms on an ARM/DSP embedded multicore platform. The algorithms are developed with the computer algebra framework MATLAB from Mathworks®, which allows to generate generic and processor-specific C/C++ code automatically. The analysis of the code generation process and the benchmark of the target performance are the main focus of this contribution.

1 INTRODUCTION

Image processing algorithms, especially stereo reconstruction algorithms on small-sized embedded platforms, e.g. mobile robots, make high demands on the overall processor performance. For this reason, many research groups all over the world are involved with this field of investigation. Thus, there already exist several approaches to get faster disparity implementations, i.e. faster algorithms or of course more processor performance.

To mention examples concerning the first approaches, region-based methods are proposed which are faster compared to dense stereo reconstruction but only result in rough disparity maps. In (Mustafah et al., 2012) the disparity calculation is based upon object-background estimation. Therefore, only the measurement of the distance and size of objects using a stereo vision system is possible. The algorithms are simple and fast but the achieved accuracy is poor, e.g. the distance error is about ±10 cm @ 3m. Lixin et al. (2009) present their results based on color-segmentation as comparable to existing methods as far as the disparity error is concerned. Unfortunately, the calculation time of this implementation is not supplied in detail and therefore this cannot be compared.

The second approach is the usage of more powerful processors, like DSPs (Digital Signal Processor) or processor optimized algorithms, which of course should result in faster implementations. Nevertheless, one has to keep in mind, that with changing the hardware, a redesign of the implemented code is required or recommended.

In (Demirovic et al., 2014) the authors show their results obtained with ARM NEON instructions for mobile devices using the example of image registration. Although we can expect significant speed improvements, mobile devices don't seem to compete yet with powerful desktop CPUs. Nevertheless, mobile processors do have a better performance per watt.

Welch et al (2012) present their investigations about the performance speed-up of SIMD architectures (single instruction multiple data), like Intel's SSE or ARM's NEON, which are advantageous when the calculation of one pixel does not affect another one and the algorithms are parallelizable. Their study is focused on the NEON SIMD instructions applied on two image processing algorithms. The bilinear interpolation algorithm

Schuldt, D., Thiem, J. and Mustedanagic, S.
Stereo Vision on an ARM/DSP Multicore Platform based on Code Generation using the MATLAB Embedded Coder.
DOI: 10.5220/0005699405580562
In *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2016) - Volume 2*, pages 558-562
ISBN: 978-989-758-198-4

results in a speedup of about 2 when using SIMD instructions. The second test, a distortion algorithm, achieves a speedup factor of 3.

Goldberg and Matthies (2011) presents their 7x7 SAD implementation for disparity estimation on an ARM/DSP platform for small robots. The stereo algorithm is implemented on the C64x+ DSP core@520MHz of the multimedia processor OMAP3530. This contribution also reveals a good comparison of several disparity implementations on DSP platforms, like (Ambrosch et al., 2010), (Humenberger et al., 2009) and (Khaleghi et al., 2008). With these results, we can get a well-founded idea of the expected embedded processor power. However, the implementations may have to be ported and optimized for a different kind of processor, if this is required.

A way out could be the model-based development of algorithms and the use of automatic generic or processor-specific code generation, which is already well-established for example in the development process of embedded control systems. Regardless of how efficient the generated image processing code may be, this approach would help to test the functionality of the algorithm in a very early stage on an embedded prototyping system. The upgrade to a more powerful processor or to a different multicore-architecture is easier than with manually implemented code. Of course, this advantage also applies if there is a hardware redesign during the development process for other reasons.

2 SYSTEM SETUP

2.1 Camera Hardware Architecture

In Fig. 1 you can see the D3 Intelligent Camera Platform from VRmagic with two camera modules mounted on the stereo-rig. The evaluation board provides input and output ports like Ethernet, HDMI, GPIO or audio (VRmagic GmbH, 2015).

The two camera modules (VRm-S 12/C) have a 1/3" color CMOS sensor with a resolution of 754x480 pixels and a pixel size of 6µm x 6µm. With a maximal frame rate of 69 Hz the cameras are able to take monochrome and color images (VRmagic GmbH, 2015).

The architecture of the D3 camera is shown in Fig. 2. The DaVinci dual core platform consists of an ARM-Cortex A8 general purpose processor and a DSP C674x digital signal processor. Both have a floating point unit (FPU). Communication between these two cores is done over a 64-Bit interface, which

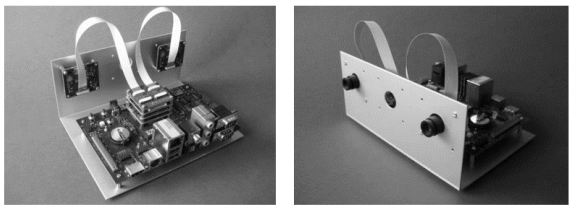

Figure 1: VRmagic D3 intelligent ARM/DSP camera (EVM).

is controlled by the Texas Instruments Codec Engine. There is also a separate interface for interrupt requests. The ARM core is clocked with 1 GHz and the DSP with 700 MHz respectively. Both cores can access a shared memory of the overall 2 GB DDR3-800 RAM which is clocked with 400 MHz. Other areas of the partitioned RAM are used e.g. for graphics output. The ARM-Cortex A8 runs a customized version of Ubuntu 12.04 long term support (LTS) with the Linux kernel 2.6.37 and without a graphical user interface. The communication between the ARM and DSP core is implemented with the help of the TI Codec Engine. The algorithm on the DSP is using the IUNIVERSAL interface.

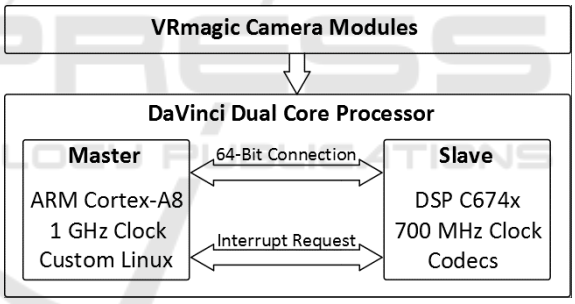

Figure 2: TI DaVinci DM8148 multicore architecture (VRmagic GmbH, 2015).

2.2 Embedded Toolchain

Fig. 3 shows an overview of the VRmagic development system. As you can see the implementation of the desired algorithms is done on an x86 host computer and the binary files are transferred to the target (VRmagic GmbH, 2015).

The target can either be an x86 or an ARM based computer, of course the appropriate runtime libraries are needed for both systems. The x86 host computer runs with a custom version of Ubuntu 12.04 (LTS) which is offered by VRmagic. To generate binary files for the different targets the toolchain provides a cross-compiler suite, in our case the GNU Compiler Collection (GCC), and all required development libraries for the different target platforms. The custom

Ubuntu operating system can be installed on a PC/Laptop or a virtual machine.

2.3 Automatic Generation of C/C++ Code

As seen above, the compiler toolchain is homogenous. The host machine as well as the embedded target is based on an Ubuntu distribution with an appropriate cross-compiler for the processor. In addition, we are able to use the same generated code from the algorithm function, if we use the MATLAB Coder for generic code generation. If the MATLAB Embedded Coder is used, of course, the code is optimized for the target processor. Nevertheless, the homogenous toolchain is still profitable.

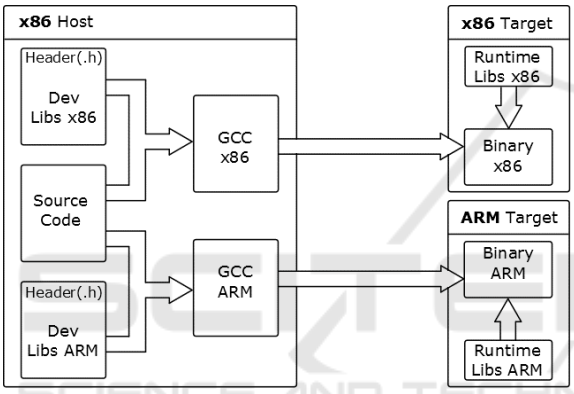

Figure 3: Toolchain of the used VRmagic camera (VRmagic GmbH, 2015).

The process of automatic code generation sounds easy. However, there are some preliminaries which have to be done before automatic code generation. The MATLAB language determines the type of variables at runtime, which is called, dynamic typing but the variable types for a static typing language like C/C++ has to be set at compile time and therefore this has to be done first. It should also be checked if the used MATLAB functions are supported by the MATLAB code generation framework. For example we are not using the MATLAB toolbox function for calculating the disparity, because it does not support ARM based processors and therefore this approach would not be implementable on our D3 camera system. The disparity function of MATLAB needs a library which is only available for x86 target systems. For this reason, we are using an own SAD implementation for our investigations.

3 RESULTS

The regarded algorithm performs the computationally expensive stereo matching calculation based on rectified images. The Windows 7 host platform is running on an Intel Core-i5 3570k@3.4 GHz. We use the sum of absolute differences (SAD) for correlation as in (Goldberg & Matthies, 2011). The greyscale stereo image pair Tsukuba from the Middlebury Dataset (Middlebury, 2015) has the size 384x288 at 8 Bit depth. The calculated disparity image has integer precision. In Fig. 4 the stereo image pair, the true disparities and the calculated disparity map is shown using a SAD window size of 7x7 and a disparity range of 16.

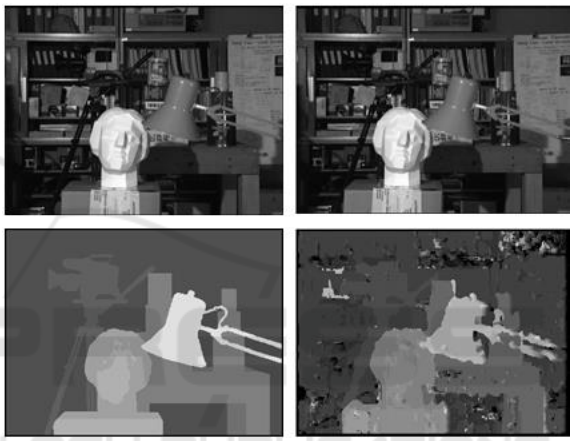

Figure 4: Stereo image pair (above), ground truth disparity map (bottom left) and calculated disparities (bottom right).

To evaluate the performance of our code on the embedded platform, we measure the elapsed time of the implemented code with the accurate time.h Linux Library and the system function gettimeofday() (Ubuntu Manpage, 2015). Because of our homogenous toolchain, i.e. the host machine and the embedded target are based on an Ubuntu distribution, there are no variants in code concerning the measurement method. The elapsed time within MATLAB is calculated with tic/toc, which gives us adequate precision in milliseconds.

To be able to compare our results with existing approaches, especially (Ambrosch et al., 2010), (Goldberg & Matthies, 2011), (Humenberger et al., 2009) and (Khaleghi et al., 2008), we calculate the performance parameter Mde (million disparity estimations per second) out of our measurements. Finally, we refer these results to the rate of the processor cycle frequency and get Mde/GHz.

As one can see in TABLE 1 the implementation of the C-Code, generated by the MATLAB

Embedded Coder achieves a performance value of 3.7 Mde/GHz and therefore cannot compete with more powerful desktop CPUs. This finding corresponds to other known publications (Demirovic et al., 2014), (Welch et al., 2012). Here, the generic (not platform specific) C-Code of the algorithm produced by the MATLAB Coder results in 5.8 Mde/GHz as MATLAB mex library and 6.5 Mde/GHz in the Ubuntu Host environment. However, our embedded performance is comparable to and better than the MATLAB engine itself on a Core-i5 desktop CPU which achieves 2.8 Mde/GHz.

In the section above the performance values belong to MATLAB Code which has been optimized for the code generation process. In TABLE *2* the performance improvements are shown in detail. The variant D is used in the preceding section. All results are calculated by averaging over up to 100 measurements.

Table 1: Comparison of the MATLAB execution time and the embedded implementation of the final optimized algorithm.

Platform	time/ms	Mde/GHz
Core-i5 3570k@3.4Ghz, MATLAB	184	2.8
Core-i5 3570k@3.4Ghz, MATLAB mex	89	5.8
Core-i5 3570k@3.4Ghz, Ubuntu Host	80	6.5
ARM Cortex-A8@1GHz	476	3.7
VLIW DSP C674x@700Mhz	428	5.9

One outcome of the investigations is that the processor specific optimization of the MATLAB Embedded Coder which results in using the NEON SIMD instruction set seems to be not as noticeable as expected. In the case of the final optimization D, we can state a speed-up of only 12%.

In code variant A the SAD matching algorithm is based on matrix-templates that are cut out of the left and right image. Next, for each column in the reference image and each disparity value the absolute difference of the matrix-templates are calculated. Obviously, addressing all window-elements of the sub-matrices is rather time consuming, i.e. the calculation of the depth map requires approx. 3.6 seconds (0.5 Mde/GHz). The performance can be significantly increased by calculating the SAD metric successively column by column. With this modification variant B shows a speed-up of about 3 (1.2 seconds or 1.4 Mde/GHz). Finally, the accumulation for the SAD metric in vertical direction can also be optimized, that results in a calculation time of about 500 milliseconds or 3.6 Mde/GHz (variant C,D).

Table 2: Performance optimization of the generated C-Code for the embedded implementation.

Algorithm	time/ms		Mde/GHz	
	w/o NEON	w NEON	w/o NEON	w NEON
A: SAD with sub-matrices	3625	n/a	0.5	n/a
B: SAD column-wise	1274	1258	1.4	1.4
C: Sum up by conv2	601	497	2.9	3.6
D: Sum up manually	543	476	3.3	3.7

3.1 Comparison

In TABLE 3 our gathered performance parameter Mde/GHz of our ARM implementation is compared to given contributions that are based on different embedded platforms, mainly on a DSP or ARM/DSP processor. As expected, the implementation on the ARM core of our system shows worse performance characteristics compared to optimized DSP implementations (Goldberg & Matthies, 2011), (Ambrosch et al., 2010).

However, we can also compare our results to another ARM Cortex-A8 implementation (Agadakos, 2015) that has already been optimized as stated by the author. Our approach shows the final implementation with the MATLAB Embedded Coder and our optimizations for the SAD calculation which therefore outperforms the given contribution.

Using the generated generic C-Code on the DSP core yields to a performance increase of about 10% compared to our ARM implementation. Although if we compare that to other DSP implementations (Goldberg & Matthies, 2011) and (Ambrosch et al., 2010), we can see that the performance is still multiple times slower than what can be achieved with highly optimized code. Nevertheless, the scope of our investigations was the analysis of automatically generated code, rather than processor optimized implementations.

Table 3: Comparison of the embedded stereo implementations.

Author	Method	Mde/GHz
Khaleghi	3x3 Census	19.2
Humenberger	8x8 Census	78.38
Ambrosch	Sparse Census	120.23
Goldberg	7x7 SAD	280.77
Agadakos	7x7 SAD	0.58
This work (ARM, variant D)	7x7 SAD	3.7
This work (DSP, variant D)	7x7 SAD	5.9

4 CONCLUSION

In this work we have illustrated the implementation of an SAD based stereo algorithm on an embedded ARM/DSP vision platform. The algorithm is developed as MATLAB function code that feeds the MATLAB Embedded Coder to generate generic or platform-specific C-Code code, in our case using the NEON instruction set of the ARM Cortex processor. We have evaluated the performance of the implemented code by comparing the speed performance in million disparity estimations per second in respect to the processor cycle frequency.

The main outcome of our work is the awareness that the performance of the embedded implementation highly depends on the developed MATLAB function code. The MATLAB Embedded Coder seems to generate more efficient code if the underlying algorithm is based on vector calculations rather than addressing sub-matrices. This could be a drawback, because the resulting MATLAB code could be less compact and readable.

With the described optimizations our generated C-Code achieves a performance of 3.7 Mde/GHz and therefore outperforms an existing ARM implementation of this algorithm. Unfortunately, no further ARM-only implementations of the considered algorithm are known.

Additionally, our algorithm on the DSP shows worse performance characteristics compared to existing DSP implementations. We used generic C-Code generated with the MATLAB coder which has a major drawback because we don't specifically take advantage of the DSP architecture (apart from TI compiler optimizations). Therefore future work should investigate the possibility of optimized code generation for the DSP with the MATLAB Embedded Coder and the TI C6000 Hardware Support Package. Furthermore improvements should be made to the SAD algorithm with special emphasis on the DSP code generation.

REFERENCES

Agadakos, Y., 2015. *A Heterogenous System Approach To The Real Time Stereo Problem*. [Online] Available at: http://artemis.library.tuc.gr/MT2013-0114/MT2013-0114.pdf [Accessed 03 March 2015].

Ambrosch, K., Zinner, C. & Leopold, H., 2010. A miniature embedded stereo vision system for automotive applications. *Electrical and Electronics Engineers in Israel (IEEEI)*, pp.000786-89.

Demirovic, D., Serifovic Trbalic, A., Prljaca, N. & Cattin, P. C., 2014. Evaluation of image processing algorithms an ARM powered mobile devices. *Information and Communications Technolog, Electronics and Microelectronics (MIPRO)*, pp.417-20.

Goldberg, S. B. & Matthies, L., 2011. Stereo and IMU assisted visual odometry on an OMAP3530 for small robots. *Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp.169-76.

Humenberger, M., Zinner, C. & Kubinger, W., 2009. Performance evaluation of a cencus-based stereo matching algorithm on embedded and multi-core hardware. *Image and Signal Processing and Analysis*, pp.388-93.

Khaleghi, B., Ahuja, S. & Wu, Q., 2008. An improved real-time miniaturized embedded stereo vision system (MESVS-II). *Computer Vision and Pattern Recognition Workshops*, pp.1-8.

Lixin, Z., Dong, Y. & Zhengguang, X., 2009. A Segment Based Dense Stereo Matching Algorithm. *Information Science and Engineering*, pp.574-78.

Manpage, U., 2015. *Ubuntu Homepage*. [Online] Available at: http://manpages.ubuntu.com/manpages/lucid/man2/gettimeofday.2.html [Accessed 03 March 2015].

Middlebury Datasets, 2015. *Middlebury*. [Online] Available at: http://www.middlebury.edu/stereo [Accessed 03 March 2015].

Mustafah, Y. M., Noor, R., Hasbi, H. & Azma, A. W., 2012. Stereo vision images processing for real-time object distance and size measurements. *Computer and Communication Engineering*, pp.659-63.

Scharstein, D., Szeliski, R. & Zabih, R., 2001. A taxonomy and evaluation of dense two-frame stereo correspondance algorithms. *Stereo and Multibaseline Vision*, pp.131-40.

VRmagic GmbH, 2015. *VRmagic Hompage*. [Online] Available at: http://www.vrmagic.com/downloads/imaging/protected/D3_intelligent_camera_user_guide_150128_EN.pdf [Accessed 03 March 2015].

Welch, E., Patru, D., Saber, E. & Bengston, K., 2012. A study of the use of SIMD instructions for two image processing algorithms. *Image Processing Workshop (WNYIPW)*.