A Framework for a Rapid Prototyping Environment for Stereo Vision Systems

Bastian Nordmeyer, Olaf Lüke, Frank Schmidtmeier,
Mohamed Shafik, and Bärbel Mertsching

University of Paderborn, GET Lab,
Pohlweg 47-49, 33098 Paderborn, Germany
{nordmeyer, lueke, schmidtmeier, shafik, mertsching}@
get.uni-paderborn.de
http://getwww.uni-paderborn.de/

Abstract. In this paper a cost-efficient framework of an FPGA based rapid prototyping environment for the development of stereo vision algorithms is presented. The hardware platform consists of the Xilinx SP601 development board and an adjustable camera mount with two cameras and adaptors for the connection to the board. On the FPGA the framework includes interfaces to the cameras, the DDR memory and the Ethernet. As a special feature it provides interfaces for the C-To-FPGA tool *Impulse C* which allows a software-oriented implementation of the image processing or stereo vision algorithms. As an example application the relatively complex *scan-line segment based stereo vision* algorithm with preprocessing image filters is implemented.

1 Introduction

The development and application of stereo vision algorithms is an active research area for more than 30 years. The resulting depth information are used in interesting applications such as autonomous mobile robots, augmented-reality and automatic vehicle guidance [**?**].

Our focus lies on the development of stereo vision algorithms for medium-sized mobile robots. Different hardware for stereo vision application exist on the market, mostly are cameras mounted in a stereoscopic manner without the possibility of image processing. Few complete stereo vision systems with included image processing exist on the market, for example the *MobileRanger C3D*[1]. These solutions have a price comparable to the price of a laser scanner and mostly do not allow the development of other algorithms on the system. Compared to these solutions our system is designed as a low-cost stereo vision system which allows an easy development of stereo vision algorithms by developers without much hardware knowledge.

We apply FPGAs for the image processing, which carries out the calculation of the disparity map containing the depth information. This unloads the host PC and leaves more computational power for other tasks such as navigation. The FPGA supports the often inherent parallelism of image processing algorithms which could lead to

[1] mobilerobots.com

higher performance of the application. Due to their reconfigurability FPGAs are flexible enough for the testing of different algorithms. The cost-efficiency is achieved by using Spartan-6 FPGAs, since these are low-price high performance FPGAs and with the Xilinx SP601[2] a suitable development board exists.

Usually, image processing algorithms are firstly implemented in a high-level languages such as C/C++. This level of abstraction provides the necessary flexibility for developing innovative algorithms, which reduces the development time and supports the portability of the algorithm. Unfortunately, one has to convert the implementation to transfer it on the FPGA. This could be done manually by reimplementing the algorithm in VHDL or automatically by so called C-To-FPGA tools such as *Impulse C* from Impulse Accelerated Technologies[3]. For the first method, one needs an experienced hardware developer who could get a very high performance out of the implementation. But still this method is time consuming and not very flexible. The latter can be done by software developers without deeper knowledge of hardware and could lead to a reasonable compromise between performance and implementation time [**?**]. Additionally, this approach offers a greater flexibility. Thus, we decided to use Impulse C in the framework for the implementation of the stereo vision algorithms.

Another important aspect of image processing are cameras and the link to the processing unit. The rapid prototyping framework should support the usage of different cameras for test purposes. Typical cameras with GigE or CameraLink interfaces do not fulfill our cost requirements. We decided to use image sensor development boards offered by image sensor manufactures. A wide range of these low-priced boards are offered, such that a suitable camera can be obtained for each test purpose. To interface these sensors with the SP601 development board own hardware components are necessary.

Our system is divided into three components: the *hardware system*, the *vision framework* and the *vision core*. These components are explained in more detail in the following sections after the related work. A brief conclusion will close this contribution.

2 Related Work

C-To-FPGA tools are available for many years. Among these tools Impulse C is well established in the market and compared to *CatapultC* from Mentor Graphics[4] it is a low-cost product. Nevertheless, it produces good results in terms of short development time, user-friendliness (easy-to-use criterion) and sufficient performance [**?**]. A comparison of different C-to-FPGA tools can be found in [**?**]. Impulse C is especially used for developing video and image processing applications, e. g. Hella Aglaia, a company that develops intelligent visual sensor systems, utilizes Impulse C and Xilinx FPGAs as a prototyping platform for ASICs [**?**]. Other publication in the field of image processing make use of the hardware/software co-design capabilities of Impulse C [**?**] or optimize relative small applications like a bilateral filter to a very high performance [**?**]. As an example application we chose the *scan-line segment based stereo vision* algorithm [**?**].

[2] `http://www.xilinx.com/products/devkits/EK-S6-SP601-G.htm`
[3] `http://www.impulseaccelerated.com`
[4] `http://www.mentor.com`

This is a more complex stereo vision algorithm than the often used standard algorithms like *census transform* or *sum of absolute differences* (SAD).

3 Hardware System

We propose to build a cost-efficient system, therefore low-priced components are necessary. In figure **??** the hardware system of our work is depicted. It consists of a mounting, SP601 development board and self-made hardware. We chose the SP601 board for our application since it is a new low-priced board with an powerful FPGA, which is equipped with an gigabit Ethernet transceiver, DDR2 memory and a connector with a large number of I/Os.

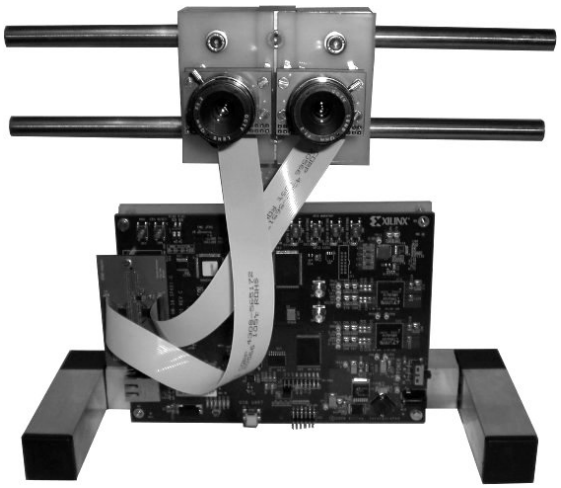


Fig. 1: Hardware system (top down): *Camera mount* with two cameras and *camera adapter* boards connected via FFC cables with the *SP601 adapter* on the FMC-LPC connector of the SP601 development board.

For image capturing two different types of cameras are utilized: An OV7720 image sensor based camera from Omnivision Technologies[5] and a MT9V032 image sensor based camera by Aptina Imaging[6]. These cameras are connected by *camera adapter* boards and FFC cables with the SP601 development board.

Since a direct connection is not possible, we extended the development board by a self-made adapter attached on the board (*SP601 adapter*). This adapter contains the power supply for the camera, performs the necessary voltage translations and assures an interference-free image data transmission. A robust camera mount is essential for a parallel orientation with adjustable distance between the cameras.

[5] `http://www.ovt.com`
[6] `http://www.aptina.com`

4 Vision Framework

The vision framework allows vision algorithm developers to implement their algorithms with Impulse C without the need to be concerned with difficult low-level HDL implementations. It handles image capturing and buffering as well as data output for the implemented algorithm.

Images are simultaneously captured and buffered by the framework. The developed algorithms are encapsulated in a component called *vision core* which can access each pixel of these images individually. Data generated by the vision core is transmitted over gigabit Ethernet to a host PC. To achieve this, the developer has to frame the data and send it over a stream to the component that handles the Ethernet communication.

The focus was set on developing a generic interface for each component, such that it is flexibly usable in other applications or that it can easily be replaced by a component with other behaviors. For example the camera data input can be substituted by a component which reads out another type of camera without the need to change any of the other components. Vision algorithms developed on our platform can be used on other platforms by adapting the framework without the need to reimplement the vision algorithms.

The whole RTL system except for some used IP cores is written in VHDL. Figure **??** illustrates the basic architecture and the data flow in the stereo vision system.

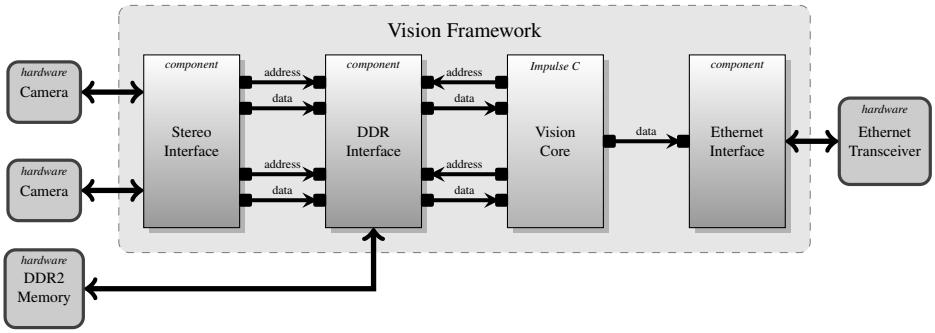


Fig. 2: Illustration of the stereo vision system architecture consisting of a stereo interface for stereo image capturing, DDR interface for data storage, vision algorithms encapsulated in the vision core and Ethernet interface for data output.

Two functions are provided by the *stereo interface* component. First, it configures the connected cameras to behave as user defined. E.g. it can set the resolution or the output format of the cameras. Second, if both cameras are configured, it reads out the camera images and guaranties stereo image consistency, i. e. it assures that only images which are captured at the same time, are transmitted. The captured images are stored in two different buffers in the DDR memory by setting the corresponding address and data.

The *DDR interface* provides an interface to the external DDR memory on the SP601 and adds some double buffering features, i. e. a simultaneous read and write on two individual memory areas, can be performed. After the reader and the writer have finished their operations these buffers are swapped. Two double buffers equipped with a write and a read port for each are provided. With both write ports image data is written into memory.

An independent readout of the stored image data is performed by the vision core. Each DDR double buffer features an *address stream* and a *data stream*. The component can request data by transmitting an address over the address stream and receives as response the corresponding data over the data stream.

Data that is generated by the vision core is transmitted over an Impulse C stream to the *Ethernet interface*, which transmits the data to a connected host PC.

5 Example Vision Core

As an example application for the vision framework, we implemented a vision core in Impulse C which uses the scan-line segment based stereo [**?**] as a stereo vision algorithm with a preprocessing bilateral filter [**?**].

Impulse C is a C-based development system for programmable hardware with or without processors. The language itself is a subset of ANSI C and additional Impulse C libraries. The Impulse C programming model consists of *processes* for computations and *streams* for data exchange.

Figure **??** shows the architecture of the vision core. The vision core is encapsulated by the *DDR interface* and the *Ethernet interface* provided by the vision framework. Impulse C nodes communicate only over these two components with the rest of the system.

Each image that is taken by the cameras traverses the following path in the vision core: The *index generator* (IG) generates indices that are in the correct order to be used in the *bilateral filter* (BF), i. e. indices to the pixels in the filter kernel that are added or removed when the filter kernel window is moved.

Using the information from the index generator, the VHDL DDR interface can produce the correct pixels and send them to the *bilateral distributor* (BD) processes which subsequently segments the image data in six different color channels to be processed by six parallel bilateral filter processes (for the red, green and blue color channels of two input images). Thereafter, the filtered image data is merged again with the *bilateral serializer* (BS) processes.

Computing the actual disparity information is done with the scan-line segment based stereo process. In this basic design it is implemented in one single process, although a parallelization with up to one process per line of the input image is possible.

After the computational part of the system is done, the computed data is combined into IEEE 802.3 conform Ethernet packets in the *packet producer* (PP). The Ethernet packets are merged into one continuous stream via the *packet serializer* (PS) to be handled by the VHDL Ethernet interface component (ETH).

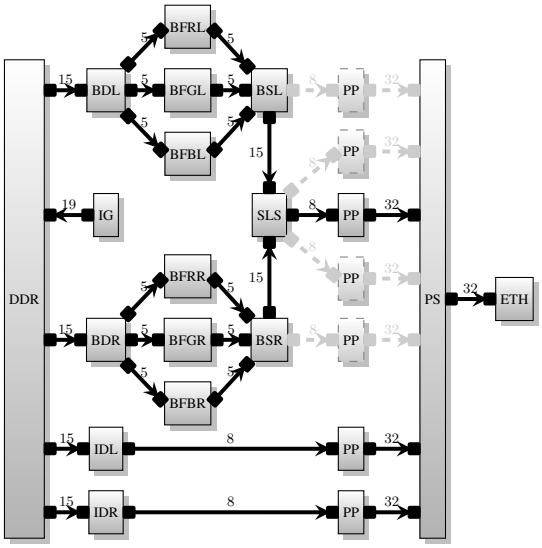


Fig. 3: Impulse C process flow chart of the image processing application. Impulse C processes consist of an index generator (IG), bilateral distributor for left and right images (BD(R|L)), bilateral filter for the left and right image as well as for the red, green, and blue color channels (BF(R|G|B)(L|R)), a bilateral serializer for left and right images (BS(L|R)), the scan-line segmentation (SLS), the packet producer (PP), and the packet serializer (PS). The dotted lined streams and processes can be used for debugging.

In parallel to the processes described above, the plain image output of the cameras is also packaged and subsequently serialized and send to the host PC via the Ethernet interface.

6 Results

In table **??** the hardware requirements are illustrated in respective to *slices*, *slice registers*, *slice LUTs* and *block RAMs* (The hardware requirements of instantiated components are added to the requirements of the main component).

The *stereo interface* is depicted twice because this component can be configured to handle OV7720 or MT9V032 cameras. In case of an OV7720 camera an *OV7 interface* is instantiated, the hardware requirements of the OV7 interface are added to those of the stereo interface. As depicted, the hardware requirements of the different configured stereo interfaces does not vary much. The only appreciable difference is the usage of two additional block RAM blocks by the MT9 interface. Each MT9 interface needs one additional block RAM, compared to an OV7 interface, to store the last image line such as that a debayering can be performed since a direct RGB output is not supported by the image sensor.

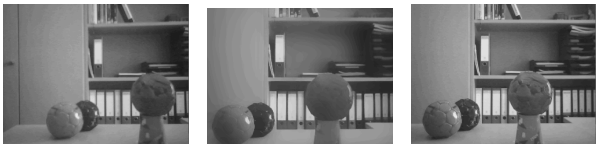
[7] $RAMB16BWER$ blocks

Component	Slices	Slice Register	Slice LUTs	BRAM blocks
Available on SP601	2278	18224	9112	32[7]
Clock manager	5	25	1	0
Stereo interface (OV7)	382	675	889	2
OV7 interface	181	295	407	0
Stereo interface (MT9)	374	675	909	4
MT9 interface	173	295	417	2
DDR interface	336	634	668	0
Ethernet interface	536	1259	1040	4

Table 1: Resource requirements of vision framework components.

The most resources are occupied by the *Ethernet interface*, whereas the largest amount is used by the Ethernet IP Core. To reduce the slice requirements only the transmit part of this core is operated such that the receive part could be removed.

As mentioned we implemented the scan-line segment based stereo vision algorithm. Figure **??** illustrates example images from the OV7 camera with corresponding filtered images and disparity map.



(a) Left image (b) Bilateral filtered (c) Right image



(d) Disparity WF (e) Disparity BF

Fig. 4: Images from OV7 camera with right image filtered and corresponding disparity maps from scan-line segment based stereo vision algorithm without filter (WF) and with bilateral filter (BF).

The bilateral filter blurred out irregularities of the balls while retaining sharp edges, which is good for the scan-line segment based stereo vision approach. The disparity map of the bilateral filtered image shows lesser noise, it does however possess larger black stripes.

Since our system is dedicated to a fast and easy-to-use development, reasonable computational speed has been achieved by round about 6 fps for the scan-line segment based stereo vision algorithm and 4 fps with an additional bilateral filtering.

7 Conclusion and Future Work

The prototyping system allows the development of algorithms without deeper knowledge of the underlying hardware platform and HDL experience. Furthermore, it is flexible and allows a time saving development and testing of algorithms on a FPGA platform.

We proposed a new approach since we implemented the system without an integrated processor, which is often present in Impulse C based applications. The advantage of this approach is that costly FPGAs with integrated processors are not required and a huge slice consumption for synthesized processors can be prevented. Also, the possibility of using unaltered vision algorithms on other hardware platforms by only adapting the framework is offered.

Rectified input images are essential for stereo vision algorithms to obtain satisfying results. We initially addressed this problem by a mechanical solution, i. e. we tried to orientate the cameras correctly and a low camera resolution of 320×240 was used. An improved version of the vision framework would take advantage of an additional component, which rectifies the images in a preprocessing step before the vision core component.