GIMME2 - an embedded system for stereo vision and processing of megapixel images with FPGA-acceleration

Carl Ahlberg, Fredrik Ekstrand, Mikael Ekström, Giacomo Spampinato and Lars Asplund
School of Innovation, Desing and Engineering
Mälardalen University

***Abstract*—This paper presents GIMME2, an embedded stereo-vision system, designed to be compact, power efficient, cost effective, and high performing in the area of image processing. GIMME2 features two 10 megapixel image sensors and a Xilinx Zynq, which combines FPGA-fabric with a dual-core ARM CPU on a single chip. This enables GIMME2 to process video-rate megapixel image streams at real-time, exploiting the benefits of heterogeneous processing.**

I. Introduction

Autonomous robots rely on sensor information, or stimuli, to act and react in a dynamic environment. There are several types of sensors, for different sensory modalities, such as optical, audio, pressure, thermal, inertial, etc. Suitable sensor(s) depend on the environment and application.

Vision is possibly the most versatile sense, allowing for different modalities in one sensor. Information can be extracted from colour, luminance, shape, or a combination thereof. Image sensors are passive, low-cost, of low complexity and power efficient.

In order to enable depth sensing, in an unknown dynamic environment, stereo vision is most versatile. However, the technology suffers from limited range, high computational complexity, poor performance in low-light conditions and with non-salient areas. The later two are related and can be overcome by inferring structured light, usually in the IR-spectrum, as for the first version of the Microsoft Kinect. However, this technology struggles with natural light. Another, approach is time-of-flight (ToF) cameras, which produces a high depth-accuracy but low image resolution. A workaround is to fuse ToF data with that of a high resolution camera through image registration. Both structured light and ToF are active technologies, and hence may encounter interference problems if several units operate within the same area.

Stereo vision is a well established research topic. The Middlebury Stereo Vision Pages[1] provide synthetic benchmark datasets and an evaluation tool for comparison of algorithm accuracy. A later addition the Middlebury evaluation is megapixel datasets with up to 800 levels of disparity, imperfect rectification and timing metrics [1]. KITTI is another benchmark based on real-world data where *"Preliminary experiments show that methods ranking high on established benchmarks such as Middlebury perform below average when being moved outside the laboratory to the real world"*[2].

Fig. 1. The GIMME2 system with F1.2/6mm lenses from Edmund Optics.

To perform real-time stereo vision of megapixel images, that implies video rate with low latency, hardware acceleration is required. FPGAs (Field Programmable Gate Arrays) combine high processing performance with low power consumption compared to GPUs, DSPs and CPUs but suffers from arduous implementation and a general lack of off-the-shelf systems [2]. There are different algorithmic approaches for stereo matching on FPGAs [3]. Extensive reviews on different algorithms, their accuracy and performance and the processing platforms are available [2], [4].

Current research on stereo vision on FPGAs favours algorithms based on Semi-Global Matching (SGM) [5] as it performs well in the evaluation. The first implementation on an FPGA, using a Zero-mean Sum of Absolute Differences (ZSAD) distance metric, achieved 25 fps on 680x400 images and 128 disparities [6]. A setup of a complete embedded vision-system, including rectification, capable of 30 fps on 640x480 images and 128 disparities, claimed a performance increase of factor 11 to previous work [7].

The stereo matching pipeline seen in the work of Jin et al. [8] manages matching of 640x480 images with 64 disparity levels at 230 fps, however only verified with a 60 fps camera, using a Census matching cost, i.e. the Hamming distance between Census transformed images [9], LR consistency check, uniqueness tests and filters.

A complete vision pipeline implemented on an FPGA for ZSAD stereo-matching of full-HD video-streams, i.e. 2

[1]http://vision.middlebury.edu/stereo/

[2]http://www.cvlibs.net/datasets/kitti/

978-1-4673-9406-2/15/$31.00

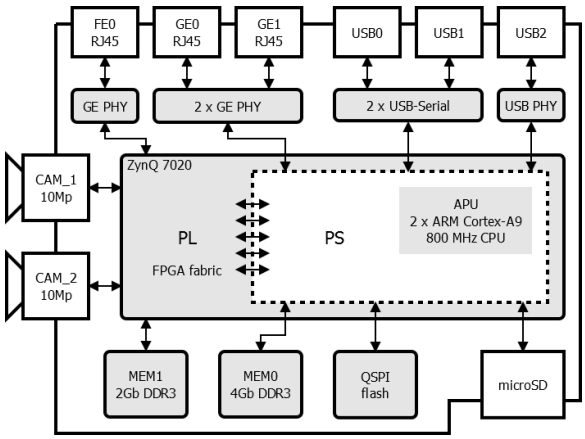


Fig. 2. GIMME2 hardware block diagram

megapixel, capable of 60 fps (system interface limit 30 fps) and 256 disparity levels was presented by Greisen et al. [10].

An FPGA implementation of full HD real-time depth estimation is capable of processing 240 disparities at 125 fps fusing SAD and Census information [11].

The original GIMME (General Image Multiview Manipulation Engine) featured an Qseven extension slot so that the FPGA could be completed by a CPU [12]. However, the communication between the two was too slow to fully integrate the heterogeneous processing. Hybrid System-on-Chips (SoC), featuring FPGA and CPU, is an emerging technology with Xilinx Zynq-7000 EPP, Altera SoC FPGAs, Microsemi SmartFusion2 as leading examples. With high-speed interconnects between the processing units vision algorithms may benefit from both technologies.

There are ongoing research on the Zynq platform for stereo vision. A small and power efficient 3D camera for mobile and embedded applications running SGM 30 fps on 640x480 images, limited to 32 disparities, is presented by Mattoccia et al. [13]. This, coupled to an embedded ARM co-processor completes a navigation system weighting less than 150g with an overall power consumption of 5W. The author also mentions a future Zynq-based design[3].

3DV[4] is small-size embedded depth-sensor targeted towards automotive applications, capable of 128 disparities at 30 fps on 640x480 images using Census-based SGM [14]. However, the monochrome sensors compromises generality for autonomous robots. Another contribution is the FPGA box[5] from SCS Super Computing System which is capable of 25 fps on 1024x400 with 128 disparities [15]. This system features a more powerful and expensive Zynq, is bulkier and the external camera interface does only support video applications up to XGA (1280x768). A later addition claims 22Hz of 2 megapixel images with up to 255 disparities, this however from 64 distributed hypothesis [16]. The SLAM sensor supports up to 4 cameras and an IMU [17]. It is used to generate synchronous and rectified FAST and Harris features (Section VI-B) from the cameras, fused with IMU data to perform Simultaneous Localization and Mapping (SLAM) on a host computer. The camera resolution of 752x480 is too low for our research purposes.

[3]http://vision.deis.unibo.it/~smatt/Site/Home.html

[4]http://vislab.it/products/

[5]https://www.scs.ch/en/fpgabox.html

TABLE I
GIMME2 SPECIFICATION

Image sensor	Aptina MT9J003
Resolution	10Mp
Frame rate	15fps@10MP, 60fps@1080p
Processing Unit	Xilinx Zynq 7020
Programmable Logic	Artix-7 85K Logic Cells
Processing System	Dual ARM Cortex-A9 (766MHz)
Memory	4Gb DDR3, 2Gb DDR3, QSPI, SD-card
Communication	2xGBE, 1xFE, 3xUSB 2.0 (one host)
Physical dimensions	130x82mm
Power input	12-24V
Power consumption	<18W@24V

This paper presents GIMME2, as pictured in Figure 1, an embedded stereo-vision and general image processing system for megapixel images, based on the Zynq, designed to provide a high processing power-to-cost ratio. A relatively small physical size, low power consumption and standardized interfaces facilitate implementation in larger context. GIMME2 can provide a solution where vision previously has been infeasible due to high costs, poor performance, size restrictions, cumbersome system integration, etc., whether it be for object detection/recognition/inspection, localization, in an autonomous robot or in automated industrial production lines.

The continuation of the paper is outlined as follows. First the hardware of GIMME2 will be described. Then, as the system is Zynq-based, follows three sections, in which the base framework is explained, covering the hardware accelerated processing performed by the FPGA, referred to as Programmable Logic (PL), the interface between the FPGA and ARM CPU(s), and the software application running on the ARM, or Processing System (PS). This is followed by two example designs extending the framework showcasing the functionality and diversity of GIMME2. The results are presented followed by a discussion and future works.

II. HARDWARE

The GIMME2 hardware is designed by AF Inventions[6]. The block diagram in Figure 2 provides an overview of the design, displaying key features and connections. The specification in terms of figures is summarized in Table I. Following is a description of design key features.

Image Sensor - Aptina MT9J003: To enable stereo vision GIMME2 is equipped with two image sensors, placed on a common base-line. The Aptina MT9J003 CMOS image sensor is capable of 15fps@10MP (3856 x 2764 bayer pattern array) or 60fps@1080p, with 12-bit colour resolution. Each sensor can produce data at a rate of 2.8 Gb/s over four 240 MHz LVDS-lanes.

[6]http://www.af-inventions.de

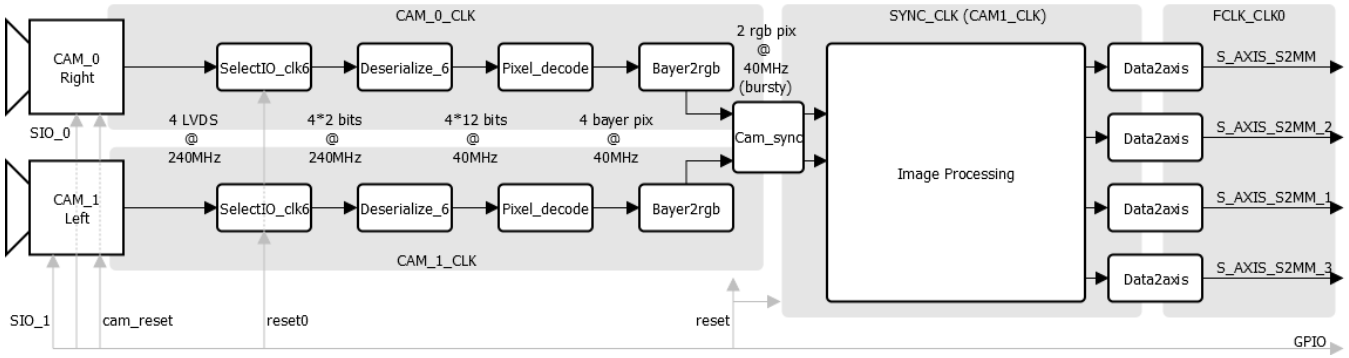


Fig. 3. PL1: VHDL framework. The image sensor data output of 4 parallel LVDS-streams of 240MHz is de-serialized, synchronized and decoded into 4 parallel Bayer-pixels at 40MHz. Two rows are combined into one row of RGB-pixels. The two streams are pixel-synchronized and transferred to the same clock domain (optional). This is followed by application specific image processing. The resulting output is converted into AXI4-stream master format and connected to up to 4 streams.

Processing Unit - Xilinx Zynq 7020 All Programmable SoC: In order to process the raw image data GIMME2 features, as previously mentioned, a Zynq SoC from Xilinx. The Z-7020 combines an 85K Logic Cell Artix-7 FPGA with a 766MHz Dual-Core ARM Cortex-A9 on a single chip, with high speed interconnects. The FPGA fabric - PL - is used for hardware acceleration of image processing/filter, stereo matching, signal processing, etc. On the CPU-side - PS) - the operating system and high level applications resides. Thanks to the high-bandwidth AXI-interconnects, data can be directed back and forth between the PL and PS, enabling FPGA acceleration in any stage/level of a processing chain. The Zynq makes for an excellent platform for image processing, stereo vision and high performance computing.

Memory: There are two DDR3 SDRAM memories, one QSPI flash memory and micro-SD slot on GIMME2. One SDRAM (4Gb), referred to as the PS-memory, is used by the operating system and software applications running on the ARM, and/or by the PL via direct memory access (DMA) over high-speed interconnects. An additional SDRAM (2Gb) memory is connected directly to, and thus dedicated to, the PL.

The non-volatile QSPI flash memory can hold a boot image (including FPGA-configuration, Linux or standalone applications), and per system specific data, such as identification numbers, camera calibration matrices, etc. Data storage is possible through a microSD connector.

IO: For communication GIMME2 features two Gigabit Ethernet (GE), one Fast Ethernet (FE), three USB 2.0, out of which one can be configured as host. There are 14 GPIO-pins (6 2.5V and 8 1.5V).

Boot mode: Boot mode jumpers select the Zynq PS boot mode. 3 different boot modes are supported; QSPI, SD-card and JTAG. In this example GIMME2 is setup to load a Linux boot-image and FPGA-configuration from the QSPI-flash.

Physical characteristics: The physical dimensions of GIMME2 is 130x82mm. The depth of the system depends on lens selection. The design features mounting holes for a CS-mount lens holder (or C-mount with adapter ring).

Power input is specified as 12-24V and power consumption is less than 18W@24V.

III. Programmable Logic

The three following sections will describe a general data framework for GIMME2 onto which application-specific configuration can be built. Following the work flow of the Xilinx Vivado tool-suite, which was used for implementation, specifically 2014.2, this section will focus on the PL, where logic is implemented on the FPGA, comprising of proprietary components (programmed in hardware description language (HDL) - in this case VHDL) and general IP-cores from the tool-suite library or 3rd party libraries. The next section, Section IV, will go into the PS-PL-interface, which is configured in a Block Design. The final part of the framework is the software application running on the PS, Section V.

The base framework consist of the following components and is shown in Figure 3:

1) CAM_0/1: The image sensors output data over 4 parallel LVDS-lanes, with a corresponding clock lane, running at 240MHz DDR, i.e. data on rising and falling clock edge.

2) SelectIO_clk6: The LogiCORE IP SelectIO Wizard core creates I/O circuits, which invoke FPGA I/O primitives. A wizard allows for quick and easy configuration to the specified LVDS-bus. It is also possible to specify the serialization factor, albeit not 12, so de-serialization is performed separately. As this component handles the clock primitives it has been customized to output a divided clock (by 6) used to clock the output of the de-serialization component.

3) Deserialize_6: 12 consecutive bits on one LVDS-lane represent one LVDS-frame (12-bit colour depth). The data is de-serialized into chunks of 12-bits, still 4 in parallel. The de-serialization needs to be synchronized (bit-slipped) to known patterns encoded into the LVDS-stream. Output is now 4*12-bits clocked by the divided clock of 40 MHz provided by SelectIO_clk6.

4) Pixel_decode: The LVDS-stream is always active. At each clock edge there is a valid data. How much of it is image data depends on resolution, horizontal and vertical blanking periods, binning and skipping, etc. Synchronization messages denoting Start-of-Frame (SOF), Start-of-Line (SOL), End-of-Frame (EOF) and End-of-Line (EOL), precede respectively succeed image data. Pixel data is extracted and complemented

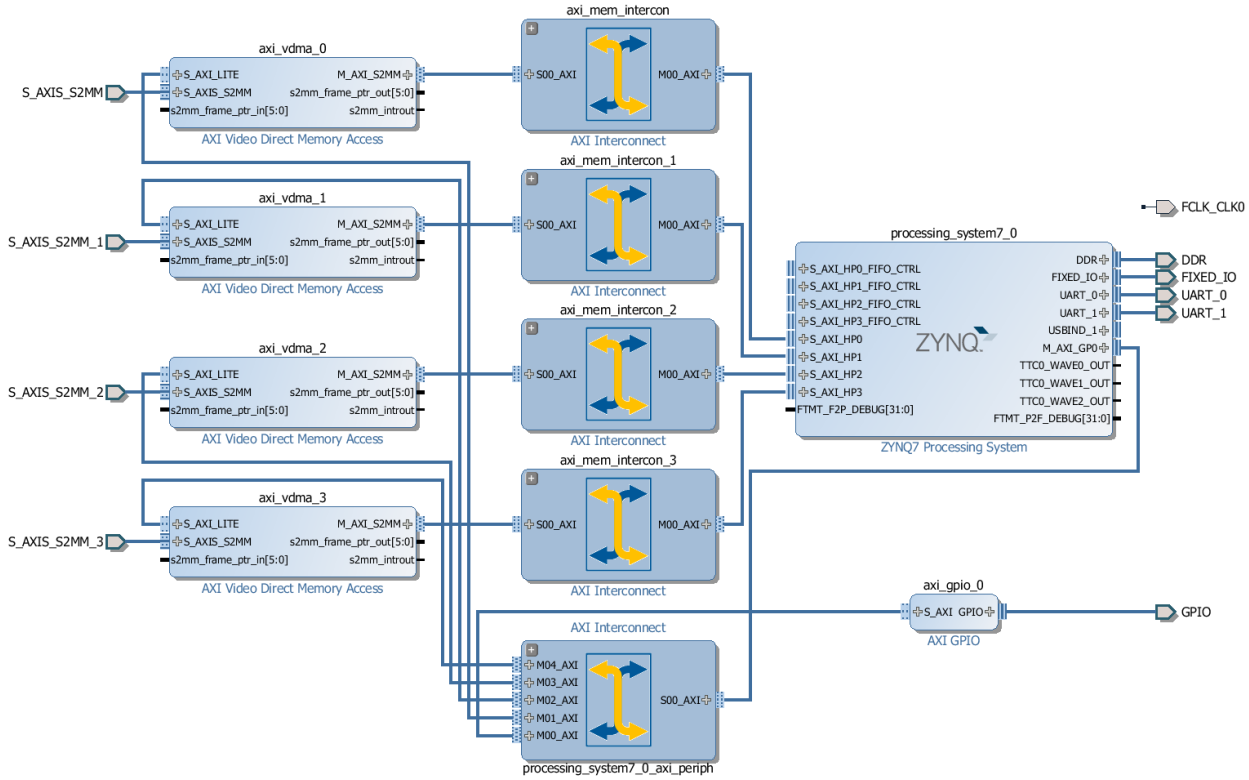


Fig. 4. PL2: Block design. Four VDMA-blocks map the video streams to the PS-memory via separate high performance interfaces. A peripheral AXI-bus for GPIO-signals and VDMA-block register access is connected to a slower general purpose port.

with SOF, EOL and data_valid control signals.

5) Bayer2rgb: The image sensor outputs a Bayer-pattern, that is every second row the color alternates between green and red and every other row blue and green. The bayer2rgb component combines the input from four neighbouring pixels into a RGB-pixel (36-bit long vector). The output is bursty as two Bayer rows are required to complete an RGB row.

This is to be updated with an interpolation technique [18] to retain the resolution of the image. Obviously this will impose higher timing requirements on the following stages in the framework.

6) Cam_sync: The optional Cam_sync-component synchronizes the pixel-streams and shifts the output to a common clock domain. Two FIFOs, one for each pipe, with separate read and write clocks, buffer the data. A read process synchronizes and verifies FIFO output with respect to the control signals.

7) Image processing: This is where the FPGA-accelerated image processing is performed to suit the specific application. Listed below are components that exist as of today. These can be combined to different results and used as blocks in new algorithms. Some of these are used in the example designs, described in Section VI.

- RGB2HSI
- RGB2HSV
- RGB to R,G,B or I - 12/8-bit gray
- Threshold to binary images
- Col_Disc (Section VI-D)
- Sliding windows
  - Generic window size (3x3, 5x5, MxN) - Larger sizes require insertion of pipeline stages to meet timing requirements.
  - Window operations: Sum, Sub, Mult, SAD
  - Linear filters: Sobel, Derivatives, Gaussian, Mean
  - Non-linear filters: Max, Min, Median, Mode, Non-maxima/minima suppression
  - Morphological operations: Erode, Dilate (binary)
  - Census transform [9]
- Harris Corner and Edge detection [19]
- Stereo, SAD block matching

8) Data2axis: The final step is to convert the processed data stream into AXI4-stream master format to interface the block design. A double clocked FIFO is used to shift into the AXI clock domain (142MHz). To avoid pre-emption availability of a full row is ensured before data valid is issued.

Although not shown in the figure it is possible to enable a read channel, to transfer images from the PS, either processed in a way that is infeasible in the PL or test images, such as the Middlebury data sets.

IV. Block Design

Block Design is a part of the Vivado tool-suite. It is a graphical programming interface. Signal mapping between components is done by selecting a port of a block and drag to the corresponding port on the next block. IP-block can either be selected from the Vivado IP-catalogue, created - encapsulating HDL-code) or provided by a 3rd party vendor. The idea is that the block design should significantly simplify

and reduce time for development and, at the same time, reduce the risk of errors. Potentially, developers with a limited knowledge of HDL-programming, can realize FPGA designs.

The block design of the framework is inspired by Xilinx XAPP792 [20] and is based on three main IP-blocks; ZYNQ7 Processing System, AXI Video Direct Memory Access and AXI GPIO, and can be seen in Figure 4.

In the ZYNQ7 Processing System block the PS is configured with respect to clock frequencies, PS-memory configuration, peripheral I/O (Ethernet, USB, SD-card, etc.), PL-interrupts, interface ports, etc. These settings will not be covered in detail but worth pointing out is that all four High Performance (HP) ports were enabled, one for each video stream, along with a General Purpose (GP) port for configuration access to VDMA and GPIO blocks.

The AXI Video Direct Memory Access block provides high-bandwidth direct memory access for a video stream. It can be used for both reading and writing. For this application only the write channel, stream-to-memory-map (S2MM), was enabled. The four VDMA-blocks connect the HDL-components to the HP-ports of the PS.The VDMA-block also has an AXI-Lite interface for register access from the PS. The control and configuration registers are mapped to a memory address in the PS-memory. These registers can be accessed/updated by both the PL and the PS during runtime.

The AXI GPIO-block allows for bidirectional data signalling through its register space. It provides a bridge for general control signals between the PS and HDL-components. More on this in the following section.

V. Processing System

The software application, developed in Xilinx SDK 2014.2, running under Xilinx Linux 2014.2, initializes the hardware and PL-cores, reads raw or processed image data from the PS-memory. This data is then to be analysed/processed to provide the most useful information possible to the next device in the chain, whether that be a host computer, hardware system, etc. The output can be sent over Ethernet, USB, CAN, stored to the SD-card or sent back to the PL for further processing.

Data transfer/communication between the PL and PS is done through the PS-memory, in core-specific registers. The AXI GPIO-block connects the following signals to the HDL-components, as shown in Figure 3:

- `cam_reset` - image sensor reset
- `SIO_0` - configuration interface for sensor CAM_0
- `SIO_1` - configuration interface for sensor CAM_1
- `reset0` - FPGA internal reset0
- `reset` - FPGA internal reset

The initialization of the hardware and PL can be described by the following sequence:

1) Setup AXI_GPIO with respect to direction and start values, i.e resets enabled and SIO idle
2) Setup and enable AXI_VDMAs
3) Release `cam_reset`
4) Setup image sensor registers and enable output
5) Release `reset0` - enable internal clk

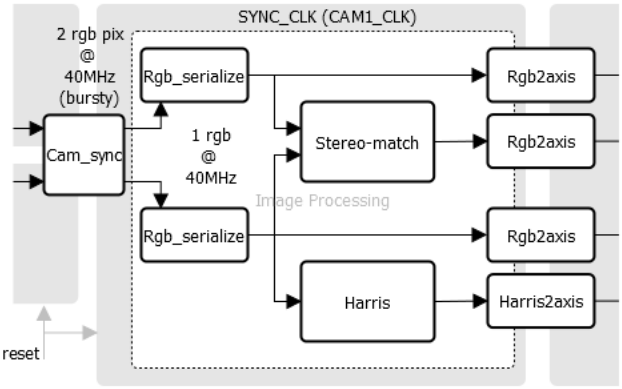


Fig. 5. Ex1 image processing block: The pixel-synchronized parallel bursty input is serialized and either connected directly to external AXI4-streams of the block design or processed by the stereo matching or Harris component.

6) Release `reset` - enable FPGA-pipe

Now data will continuously refresh the frame buffers. Now follows application specific steps.

7) (loop)
   a) Read frame buffers
   b) Process data (OpenCV available)
   c) Forward output
   d) (Optional) Check AXI_VDMA error and status registers
   e) (Optional) Every Nth frame check brightness and adjust exposure
8) Clean-up and exit

VI. Example designs

In this section image processing blocks are realized into the framework in two example designs, referred to as Ex1 and Ex2. Ex1 combines stereo matching, Harris corner and edge detection, and image streaming to highlight the functionality of GIMME2. The corresponding image processing blocks can be seen in Figure 5. The second example design, Ex2, changes colour representation two times, first to HSV followed by a colour discretization (Section VI-D). This design was developed for the Naiad[7] autonomous underwater robot for RoboSub[8] 2015. As the robot is quite slow, the objects are stationary with the cameras running at 60 fps there was no need to synchronize the cameras. This lead to a design with two separate pipes, as shown in Figure 6. The resulting discretized images were further processed in OpenCV, using one core for each image, to detect gates, buoys and markers in order for Naiad to autonomously navigate and complete tasks.

A. Stereo Matching

A simple SAD-based block-matching was implemented in order to verify the stereo capability of the system. The matching cost at a given pixel, $P_{(x,y)}$, for a disparity d is

[7]http://naiad.se/

[8]http://www.auvsifoundation.org/foundation/competitions/robosub/

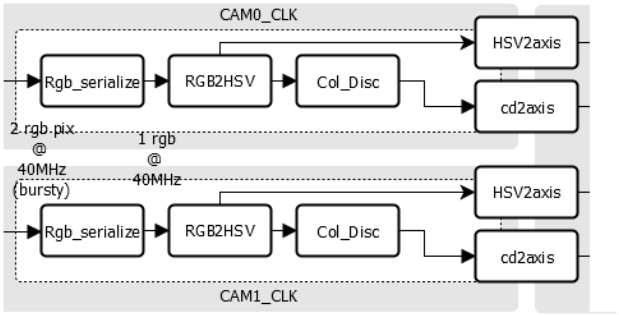


Fig. 6. Ex2 image processing block: Two separate and unsynchronised pipes, one for each camera, performing RGB2HSV and colour discretization.

given by:

$$SAD(P_{(x,y)}, d) = \sum_{(x,y)\in W} |I_r(x,y) - I_l(x+d,y)| \quad (1)$$

where W defines a window (area) and I_l and I_r the intensity representation of the left and right image. Over a range of disparities,

$$D = \{d | 0 \leq d \leq D_{max}\} \quad (2)$$

the cost function is minimized,

$$SAD(P_{(x,y)}, d) = \min_{\delta \in D} SAD(P_{(x,y)}, \delta) \quad (3)$$

so that the disparity d associated with the minimum cost, i.e., the highest correlation between the left and right windows, is selected.

As indicated, the calculations described above are performed for each position in the image(s) (invalid result along left edge (disparity) and edges (window)). For megapixel images at video-rate or faster this becomes computationally intense. Adding further to this is poor scalability with respect to window size and disparity range. However, for smaller windows and limited ranges implementation is apt for FPGAs.

For the implementation on GIMME2 the window size was set to 5x5 and the disparity range to 128 (0 to 127). The SADs were calculated in parallel.

Since this approach is quite crude a 9-pixel median filter, operating along the row, is applied in order to reduce noise. To filter in one dimension reduces buffering and thus resources but also preserves contours through less foreground fattening [21].

B. Harris Corner and Edge Detector

The Harris detector identifies corners and/or edges in an image [19]. As the algorithm is region based and offers possibilities for parallelism and pipelining, it is suitable to be implemented on an FPGA. The Harris response is calculated by the following equations:

$$H(x,y) = \begin{bmatrix} S_{x^2}(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_{y^2}(x,y) \end{bmatrix} \quad (4)$$

where S denotes the products of intensity derivatives in the x, y and xy-direction smoothed by a Gaussian bell.

$$R = Det(H) - k(Tr(H))^2 \quad (5)$$

where R is the response. A positive value indicates a corner and a negative an edge. To select sensible features the response is subjected to neighbourhood suppression (corners) and thresholded (positive for corners and negative for edges).

The implementation is based on 5x5 windows (for derivatives, Gaussian filtering, neighbourhood suppression) and the constant k is set to $1/16$. The output is updated in accordance with the AXIS-interface. The implementation is described in previous work [12].

C. RGB2HSV

To calculate the HSV value of an RGB-pixel effectively in the PL the following approximative equations were used:

$$H = \begin{cases} (G-B)*42/R & \text{if } R > G > B \\ 6*42-(B-G)*42/R & \text{if } R > B > G \\ 4*42+(R-G)*42/B & \text{if } B > R > G \\ 2*42-(R-B)*42/G & \text{if } G > R > B \\ 2*42+(B-R)*42/G & \text{if } G > B > R \\ 4*42-(G-R)*42/B & \text{if } B > G > R \end{cases}$$

$$S = MAX(R,G,B) - MIN(R,G,B)$$

$$V = MAX(R+G+B)$$

Calculation of hue involves an expensive division which is performed in a 7 state process. As this process is not fully pipelined a wrapper has been added to cycle through 7 instances of the component and is thus able to process new data every clock cycle.

D. Col_Disc

The idea behind the Col_Disc component is when operating in an environment with large non-salient areas, area-based stereo-matching suffers. In such a case Col_Disc may provide a close to segmented image and favourable conditions for object-based stereo-matching in the PS.

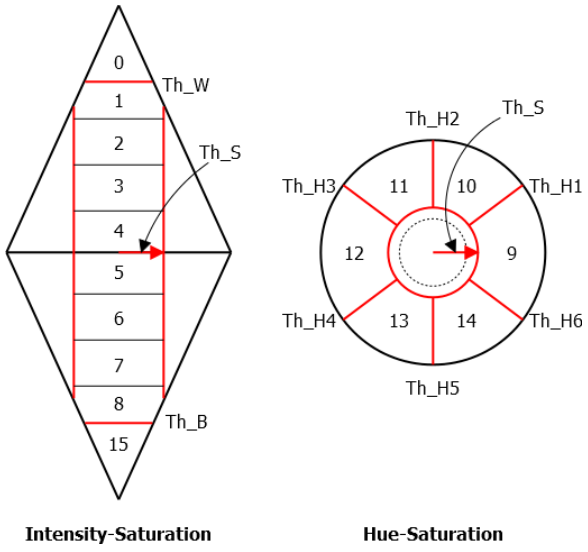
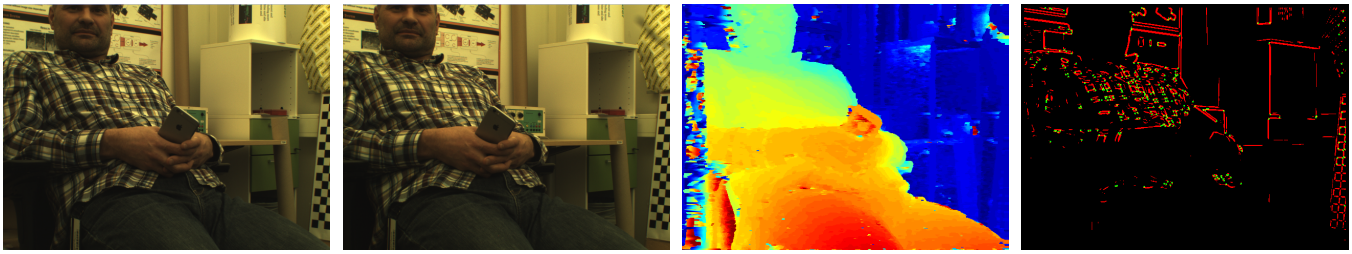


Fig. 7. The colour discretization component - the HSI/HSV colour space is divided into 16 colours using 9 thresholds. The thresholds are setup as registers in the PS-memory.



(a) Left image (b) Right image (c) Disparity map (d) Harris corners (G) and edges (R)

Fig. 8. Output from the first example design - snapshot images from the four video streams.

The component discretizes an HSI/HSV image into 16 colors using 9 thresholds as shown in Figure 7. The first two, `TH_W` and `TH_B` act on the intensity and set all pixels above respectively below to 0 and 15. The third threshold, `TH_S`, separates colour pixels from gray. The gray are evenly distributed inte 8 different shades in between `TH_W` and `TH_B`. The final 6 thresholds divides colour pixels into 6 colour buckets.

The output is treated with a noise-reducing mode-filter (the most frequent occurrence within the window) before sent to the PS.

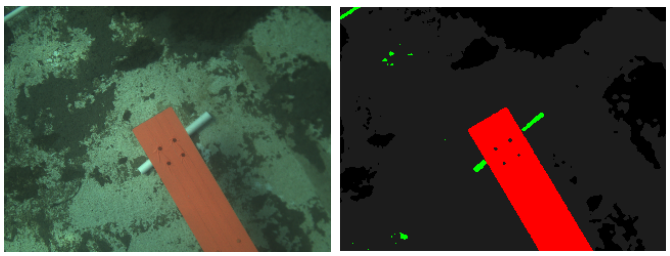
The thresholds are setup as registers and are hence available from the PS and can be updated during runtime.

VII. Results

In first example the FPGA was configured to output four video streams, out of which the first two were unprocessed. A snapshot from these streams can be seen in Figure 8a and 8b. Raw image streams can be processed and/or stereo-matched in the PS by users unaccustomed with FPGA-programming. This, of course, without hardware acceleration. Raw streams can also be used for proof-of-concept of algorithms running on the PS, before taking the step to adapt and implement for the FPGA.

An area-based stereo-matching was implemented in the PL to draw advantage from the FPGA. The accuracy does not compare with the state of the art algorithms but basic block matching is still used and suitable for FPGAs. However, it shows that stereo-matching can run on GIMME2 and meet the performance requirements from the rest of the framework and image sensors. This, without platform specific optimizations. The resulting disparity map is shown in Figure 8c.

By adding the Harris combined corner and edge detector the parallel nature of the FPGA was highlighted. Additional processing blocks can be added without limiting the performance as long as there are available logic resources. Of course, if the number of parallel image-processing pipes, outputs must be multiplexed to the available DMA-channels. The resulting image can be seen in Figure 8d.



(a) RGB image (b) Color Discretized Image

Fig. 9. Example of Col_Disc output.

TABLE II
FPGA resource utilization for the two designs, Ex1 and Ex2.

Resource	Available	Utilization		Utilization (%)	
		Ex1	Ex2	Ex1	Ex2
FF	106400	39054	24907	36.70	23.41
LUT	53400	46954	19924	88.26	37.45
Memory LUT	17400	743	1049	4.27	6.03
I/O	200	43	35	21.50	17.50
BRAM	140	70	37.5	50.00	26.79
DSP48	220	23	0	10.45	0
BUFG	32	3	5	9.38	15.62

The left image, the disparity map and the Harris features are aligned and may be processed to synergistic effects in the PS, possibly using OpenCV. For example, the original image can be segmented, and act as support, for improving the disparity map.

The second example was implemented for the Naiad underwater robot. Figure 9 shows an original image and the corresponding discretized image for a bottom marker captured by Naiad's downward facing camera.

The resource utilization for the two example designs is shown in Table II. A setup, where only the two image streams were configured, resulted in 10% LUT and 20% BRAM. As the components were not optimized for a certain image size, window size, bus width, and use excessive buffering, these figures can be lowered.

Constraints ensure that the implemented FPGA-logic performs at least on par with the image sensors, i.e. the frame rate of the video streams to the PS-memory is the same as image sensors.

VIII. Conclusion and Future Work

In this paper the GIMME2 embedded stereo-vision system have been presented. It combines FPGA-based hardware acceleration with a dual-core ARM and can process megapixel video-streams in real-time. Designed to provide a good cost-to-performance ratio, have limited physical dimensions and

low power consumption, it is suitable for implementation in robotic systems, that rely on vision to operate.

The processing performed on GIMME2 is ultimately application dependent. The SoC solution of the Zynq makes it possible to accelerate selected parts of the processing pipe exploiting the benefits of heterogeneous processing. Tasks with high-speed requirements, that are parallelizable and have a local footprint are preferably accelerated by the FPGA, while global, iterative and dynamic behaviour is more suitable for the CPU.

For GIMME2 a number of basic image processing components have been developed benefiting from FPGA acceleration. These can be combined to suite the intended application or used as building blocks for more advanced algorithms. In combination with OpenCV on the PS side this makes for a highly adaptable package. To add to that flexibility several FPGA-configurations can be developed and used for different tasks. That can be taken one step further through partial reconfiguration in the context of image processing and adaptable algorithms.

The future work focuses both on algorithmic development and application in context of robotic systems. Next, to be implemented for GIMME2 is video-rate stereo-matching for megapixel images in real-time posing little or no constraint on disparity range. Stereo-matching algorithms, implemented in hardware, are typically limited with respect to disparity range. An increase in sensor resolution translates into a greater resolution in disparity but, at the same time, a shorter valid depth range, if the disparity range remains the same. This is reflected by the Middlebury evaluation and dataset that now feature up to 800 pixels of disparity. The future research is to address this problem.

GIMME2 will be implemented in the SVAHLA (Stereo Vision Assisted Hauler and Loader Alignment) project, carried out in collaboration with Volvo CE.

Acknowledgment

Thanks to AF Inventions for the GIMME2 design, Xilinx for supporting this project, the Knowledge foundation, the Ralf3 partners and Volvo CE.

References

[1] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nesic, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *Pattern Recognition - 36th German Conference, GCPR 2014, Münster, Germany, September 2-5, 2014, Proceedings*, 2014, pp. 31–42. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-11752-2_3

[2] B. Tippetts, D. Lee, K. Lillywhite, and J. Archibald, "Review of stereo vision algorithms and their suitability for resource-limited systems," *Journal of Real-Time Image Processing*, pp. 1–21, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11554-012-0313-2

[3] S. Mattoccia, "Stereo vision algorithms suited to constrained FPGA cameras," in *Advances in Embedded Computer Vision*, ser. Advances in Computer Vision and Pattern Recognition, B. Kisaanin and M. Gelautz, Eds. Springer International Publishing, 2014, pp. 109–134. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-09387-1_5

[4] C. Ttofis, C. Kyrkou, and T. Theocharides, "A hardware-efficient architecture for accurate real-time disparity map estimation," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 14, no. 2, p. 36, 2015.

[5] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 328–341, Feb. 2008. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2007.1166

[6] S. K. Gehrig, F. Eberli, and T. Meyer, "A real-time low-power stereo vision engine using semi-global matching," in *Proceedings of the 7th International Conference on Computer Vision Systems: Computer Vision Systems*, ser. Lecture Notes in Computer Science, M. Fritz, B. Schiele, and J. Piater, Eds., vol. 5815. Springer, 2009, pp. 134–143.

[7] C. Banz, S. Hesselbarth, H. Flatt, H. Blume, and P. Pirsch, "Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation," in *Embedded Computer Systems (SAMOS), 2010 International Conference on*, July 2010, pp. 93–101.

[8] S. Jin, J. Cho, X. D. Pham, K. M. Lee, S.-K. Park, M. Kim, and J. Jeon, "FPGA Design and implementation of a real-time stereo vision system," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, no. 1, pp. 15–26, Jan 2010.

[9] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence." Springer-Verlag, 1994, pp. 151–158.

[10] P. Greisen, S. Heinzle, M. Gross, and A. Burg, "An FPGA-based processing pipeline for high-definition stereo video," *EURASIP Journal on Image and Video Processing*, vol. 2011, no. 1, 2011. [Online]. Available: http://dx.doi.org/10.1186/1687-5281-2011-18

[11] H. Li, P. An, G. Teng, and Z. Zhang, "FPGA implementation of full HD real-time depth estimation," in *Consumer Electronics ??? Berlin (ICCE-Berlin), 2014 IEEE Fourth International Conference on*, Sept 2014, pp. 249–253.

[12] C. Ahlberg, J. Lidholm, F. Ekstrand, G. Spampinato, M. Ekström, and L. Asplund, "GIMME - A general image multiview manipulation engine," in *Proceedings of the International Conference on ReConFigurable Computing and FPGAs (ReConFig 2011)*, November 2011. [Online]. Available: http://www.es.mdh.se/publications/1076-

[13] S. Mattoccia and M. Poggi, "A passive RGBD sensor for accurate and real-time depth sensing self-contained into an FPGA," in *Proceedings of the 9th International Conference on Distributed Smart Cameras*, ser. ICDSC '15. New York, NY, USA: ACM, 2015, pp. 146–151. [Online]. Available: http://doi.acm.org/10.1145/2789116.2789148

[14] G. Camellini, M. Felisa, P. Medici, P. Zani, F. Gregoretti, C. Passerone, and R. Passerone, "3DV - An embedded, dense stereovision-based depth mapping system," in *Intelligent Vehicles Symposium'14*, 2014, pp. 1435–1440.

[15] F. Eberli, "Next generation FPGAs and SOCs - How embedded systems can profit," in *CVPR Workshops'13*, 2013, pp. 610–613.

[16] S. Gehrig, R. Stalder, and N. Schneider, "A flexible high-resolution real-time low-power stereo vision engine," in *Computer Vision Systems*, ser. Lecture Notes in Computer Science, L. Nalpantidis, V. Krger, J.-O. Eklundh, and A. Gasteratos, Eds. Springer International Publishing, 2015, vol. 9163, pp. 69–79. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-20904-3_7

[17] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 431–437.

[18] H. S. Malvar, L.-W. He, and R. Cutler, "High-quality linear interpolation for demosaicing of bayer-patterned color images," in *International Conference of Acoustic, Speech and Signal Processing*. Institute of Electrical and Electronics Engineers, Inc., May 2004. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=102068

[19] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the Alvey Vision Conference*. Alvety Vision Club, 1988, pp. 23.1–23.6, doi:10.5244/C.2.23.

[20] J. Lucero and Y. Arbel. (2012, Oct) Designing high-performance video systems with the Zynq-7000 all programmable SoC - XAPP792 (v1.0.1). Xilinx Inc. [Online]. Available: http://www.xilinx.com/support/documentation/application_notes/xapp792-high-performance-video-zynq.pdf

[21] F. Ekstrand, C. Ahlberg, M. Ekström, L. Asplund, and G. Spampinato, "Utilization and performance considerations in resource optimized stereo matching for real-time reconfigurable hardware," in *VISAPP 2012 - International Conference on Computer Vision Theory and Applications*, February 2012. [Online]. Available: http://www.es.mdh.se/publications/2638-