

---

# Parallel Early Vision Algorithms for Mobile Robots

Maissa Ali<sup>1</sup>, Joaquin Sitte<sup>2</sup>, and Ulf Witkowski<sup>3</sup>

<sup>1</sup> University of Paderborn, Germany [maissa@mail.uni-paderborn.de](mailto:maissa@mail.uni-paderborn.de)

<sup>2</sup> Faculty of Information Technology, Queensland University of Technology,  
Australia [j.sitte@qut.edu.au](mailto:j.sitte@qut.edu.au)

<sup>3</sup> Heinz Nixdorf Institute, University of Paderborn, Fuerstenallee 11, 33102  
Paderborn, Germany [witkowski@hni.upb.de](mailto:witkowski@hni.upb.de)

**Summary.** Real time image processing requires processing huge amounts of data in a very short time. Having this data processed serially, as it is the case using a von Neumann architecture, meeting a real time constraint usually needs a high clock frequency leading to bulky designs and high power consumption. FPGAs on the other hand can process data in parallel making them a suitable platform for real time image processing algorithms especially for mobile robots. This paper presents an implementation of a real time algorithm that extracts colour histograms from an image stream connected to an FPGA extension module that was designed for the Khepera robot. The colour histograms provide data being characteristic for a specific place and offer a useful technique for vision based navigation.

## 1 Introduction

Localization and navigation are essential tasks that a mobile robot needs to perform in order to gain the ability to interact autonomously with its environment. Image sensors provide rich information of their surroundings resulting in large volumes of data. Extracting special features of the images is necessary to reduce this amount of data. Werner et al. [8] have shown that extracting colour histograms from images provides highly specific place characterization. In computer vision and robotics these histograms are used for image retrieval [10], object localization [3], and vision based navigation [8]. Colour histograms are easy, fast to calculate and provide salient colour information of images in a very compact manner.

For the implementation of such algorithms on an autonomous mobile minirobot a processing unit that is small, fast and power efficient is required. Microcontrollers can perform a wide range of tasks, but for image processing, the huge amount of data leads to delayed results and can therefore often not meet real

time constraints. FPGAs (Field Programmable Gate Arrays) are a good compromise between the amount of data that can be processed, the time needed for the process and the power consumption of the device.

Primarily for the purpose of image retrieval, some work on calculating colour histograms on FPGAs has been done. Kotoulas and Andreadis [5] have implemented colour histograms for content-based image retrieval. Leaser et al [6] have developed an algorithm that uses 3-D colour histograms combined with pixel location aimed for FPGA implementation. Their algorithm is applied on still images for searching images in a database.

This paper shows the implementation of an early vision algorithm on an FPGA extension module designed for the Khepera robot [7]. The purpose of this algorithm is to extract colour features from the image into colour histograms in real time. The proposed design is developed using VHDL and extracts colour histograms from a 480 x 640 pixel video stream.

This paper is structured as follows: in Sec. 2, two early vision image processing techniques, Bayer interpolation and colour histograms are briefly described. Sec. 3 describes the hardware platform and the structure of the implementation, and states the implemented algorithms. The simulation and implementation results obtained are stated in Sec. 4, followed by a discussion and conclusion of these results in Sec. 5.

## 2 Image Processing

The image data is captured by the image sensor using a Bayer pattern. This pattern results from the architecture of the CMOS image sensor, which contains an array of photodetectors that are sensitive to light intensity. To get a coloured image, each photodetector is covered with a colour filter. In most CMOS image sensors, these filters are arranged in a Bayer pattern.

In a Bayer pattern (shown in figure 1) each photodetector is covered with a red, green or blue filter, resulting in pixels being sensitive for red, green, and blue. The array containing these pixels, also called the raw data array, represents the input values for the image processing algorithm.

### 2.1 Bayer Interpolation

An interpolation is necessary to convert the Bayer pattern into an RGB image of the same size, having each pixel represented by a tuple of a red, green and blue component. There are several algorithms that perform this interpolation. The idea behind these algorithms is that usually the value of one pixel and its neighbouring pixels is very similar or even equal, so that the missing two colour components can be calculated taking the average of their neighbouring pixels or by simply copying their values [2].

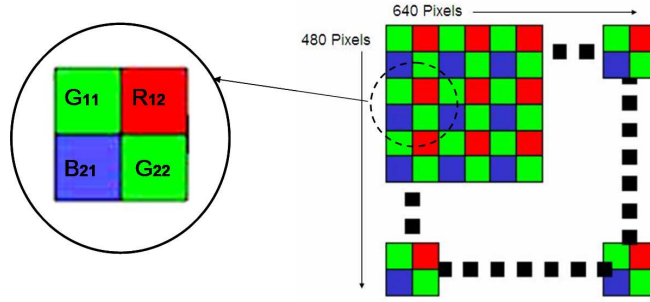


Fig. 1. Bayer Pattern: image format from the CMOS image sensor

### 2.2 Colour Histograms

Once each pixel is represented by a red, green and blue value, further image processing can start, in our case the calculation of the colour histograms. A colour histogram is a representation of an image by classifying a specific colour or colour range to the number of pixels containing this colour. A colour range is called a bin. The size of the histogram depends on the number of bins. Based on the colour space used to represent the image, several types of histograms can be build. Two types for the RGB colour space are described below:

**Separate RGB Histograms:** Three  $N$ -bin histograms are calculated. Each colour component is considered separately, resulting in three colour histograms (Fig. 2 shows an example). This way the combination of the components of a colour is lost [8]. However, the size of the histograms is very small with the total number of bins equals  $3N$ .

**3-Dimensional Histogram:** Here the components of a colour are not separated, the pixels are represented by a vector with 3 components. The histogram bins are 3-dimensional, with equally sized colour ranges for each component. This increases the number of bins needed to represent an image. If each component is divided into  $N$  bins, the resulting total number of bins equals to  $N^3$ . But, the 3D spatial information of the RGB tuples in the colour space is retained [8].

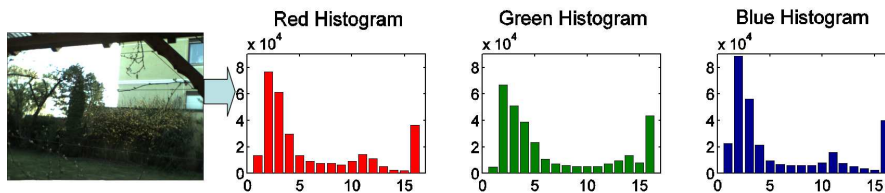


Fig. 2. Example for a histogram calculation

### 3 Hardware System and Implementation

The robot for our experiment is the mini robot Khepera. In its basic configuration the robot's processing capabilities are very limited. Therefore, the department of System and Circuit Technology at the University of Paderborn has developed an FPGA extension module for the Khepera robot [7]. This module is connected to a CMOS image sensor and provides a platform for the implementation of image processing algorithms to expand the visual capabilities of the Khepera robot. Figure 3 shows a picture of the FPGA extension module and a diagram of the module with its interfaces.

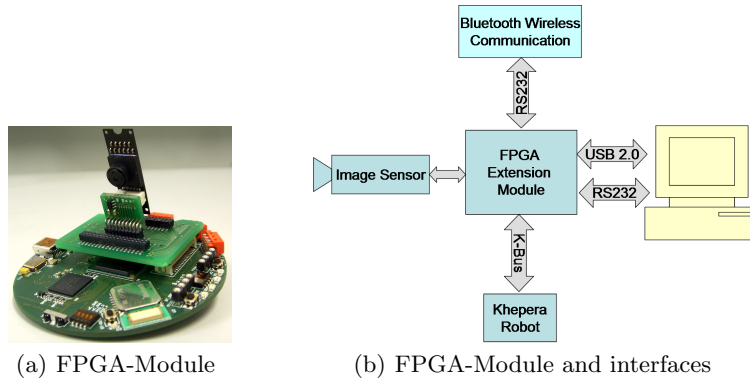


Fig. 3. FPGA-Extension-Module

#### 3.1 Field Programmable Gate Array

The main component of the FPGA Extension Module is the FPGA, a Virtex-E XCV300E FG256 from XILINX [9]. An FPGA is a semiconductor device that contains programmable logic. Its main components are an array of logical gates and a network of programmable interconnections. Advantages of an FPGA are mainly the reconfigurability, the ability of running many processes in parallel and its low power consumption. Whereas microprocessors are not suitable for instantaneous computation of simple functions [4], the parallelism of the FPGA offers a suitable platform for real time processes especially for early vision algorithms.

#### 3.2 CMOS Image Sensor

The CMOS Image Sensor transmits the image data with a frame rate of 5 frames/sec to the FPGA. The image is transmitted in serial, one pixel (one byte) at a time, one line after the other. The image is represented through

a Bayer pattern, as shown in figure 1, with 8 bits per pixel. The size of the image frame is 480 x 640 pixels.

### 3.3 Implementation

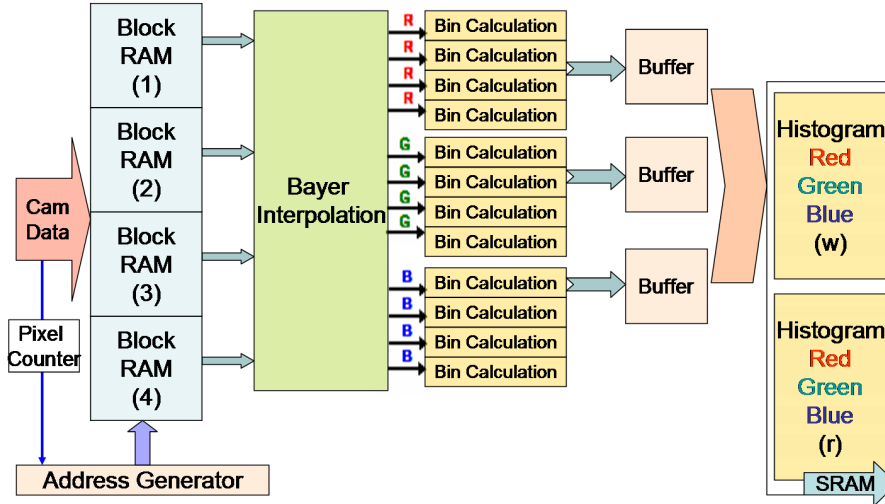


Fig. 4. Block diagram of the design

Figure 4 shows the architecture of the implemented design. The XILINX Virtex-E 300 offers 32 Block RAMs, each with a capacity of 512 Bytes. The address generator distributes the received data from the camera into four blocks, making it possible to access four pixels simultaneously, as four pixels are needed to perform a Bayer interpolation.

For the Bayer interpolation, a simple interpolation is adequate. In this method a window of 2x2 pixels provides the input for the interpolation algorithm. The red and blue pixels are quadrupled and the green pixels are averaged as shown in (1) to (4). By choosing an algorithm that requires a 2x2 window, less data must be saved which is important to keep the resource usage of the FPGA low and less calculations must be performed which effects the speed of the overall algorithm. As for the accuracy of these values it is sufficient, for the histogram calculation would revoke the extra accuracy that might be gained using a 3x3 window. In figure 1 the encircled region indicates the 4 pixels needed for the interpolation. These pixels are located on two lines, whereas for a 3x3 window the pixels are located on three consecutive lines.

$$(R_{11}, G_{11}, B_{11}) = (R_{12}, G_{11}, B_{21}) \quad (1)$$

$$(R_{12}, G_{12}, B_{12}) = (R_{12}, \frac{G_{11}+G_{22}}{2}, B_{21}) \quad (2)$$

$$(R_{21}, G_{21}, B_{21}) = (R_{12}, \frac{G_{11}+G_{22}}{2}, B_{21}) \quad (3)$$

$$(R_{22}, G_{22}, B_{22}) = (R_{12}, G_{22}, B_{21}) \quad (4)$$

For a  $2^n$ -bin histogram with equal sized bins, the  $n$  most significant bits of a pixel form the bin number. Eq. 5 shows the equation that calculates the bin number  $BN$  for a value  $\{x_72^7 \dots x_02^0\}$ .

$$BN = \sum_{i=8-n}^7 x_i 2^{i-(8-n)} \quad (5)$$

In the case of 3-dimensional histograms, the corresponding bin number of a pixel  $p = \{x_72^7 \dots x_02^0\}Red + \{y_72^7 \dots y_02^0\}Green + \{z_72^7 \dots z_02^0\}Blue$  is a combination of its red, green and blue components as shown in eq. 6.

$$BN = \sum_{i=8-n}^7 x_i 2^{3n-(8-i)} + \sum_{i=8-n}^7 y_i 2^{2n-(8-i)} + \sum_{i=8-n}^7 z_i 2^{n-(8-i)} \quad (6)$$

The bin number  $BN$  points at the address of the corresponding bin in the on chip RAM. A counter increments the value saved in that location. Two RAMs are required to save the histogram data. While one Block RAM is written with the histogram data of the currently processed frame, the histogram data from the previously processed frame can be read out into the SRAM of the FPGA extension module. From there it can be transferred via the serial port to a connected computer or via the K-Bus to the microcontroller of the Khepera.

## 4 Simulation and Implementation Results

The design was simulated using ModelSim XE 6.1e III [1], a simulation and debug environment. Table 1 shows the parameters that were chosen for the simulation. These values are very close to the physical values that are used to run the FPGA camera module. Table 2 shows the results that were gained from this simulation.

**Table 1.** Parameter setup used for the simulation

FPGA CLK Frequency:	50 MHz
Pixel rate:	12.5 MHz
Frame rate:	5 frames/sec
Frame size:	VGA (480 x 640 Pixels)
Picture format:	Bayer Pattern

**Table 2.** Simulation Results

---

Total time to process on frame:	26.6 ms
Time to process a 2x2 window:	14 clock cycles
Histogram fully calculated:	14 clock cycles (280ns) after receiving the full image

---

14 clock cycles after receiving the last pixel of the frame from the camera the colour histograms are calculated and can be transferred for further processing. In the case of using a 50 MHz clock frequency the histogram would be ready 280ns after the full image is received. Only the extracted information (histogram) is transmitted, reducing the amount of data significantly. For example, 16-bin RGB histograms need 38 bytes/histogram (19 bit/bin), reducing the image data from 307200 bytes (480x640 pixels) to 114 bytes, with a reduction rate of 2695%. This makes it possible to run further processing of the histogram data on the microprocessor of the Khepera, like matching algorithms needed to use the histograms for navigation. In order to have a rough reference of the speed-up gained through using an FPGA, the vision algorithms were run on an MC68331, the microprocessor available on the Khepera. For 8-bin RGB histograms, the time needed to read the image data of one frame and process the data amounts to 14616 ms, and 13555 ms for an 8-bin/colour 3-D histogram .

Table 3 shows the device utilization of the Virtex-E300. These numbers refer to the maximum bin numbers that can be implemented, 256 bins/colour for the RGB histograms resulting in a total of 768 bins, and 8 bins/colour for the 3-D histograms resulting in a total of 512 bins.

**Table 3.** Logic utilization of FPGA for RGB and 3-D histograms

---

	RGB	3-D	Available
Occupied Slices	1926 (62%)	1448 (47%)	3072
Bonded IOBs	94 (53%)	94 (53%)	176
Block RAMs	22 (68%)	19(59%)	32

---

## 5 Discussion and Conclusion

This paper presents a design for implementing early vision algorithms in an FPGA directly connected to a camera. This design assists the minirobot in the performance of vision based navigation. Advantageous features are the real time image processing, having the colour histograms of one image constructed after a delay of only 280ns and the data reduction gained with the extraction of colour histograms. Only the extracted data (the histogram data) is transferred

for further computations. The microcontroller of the Khepera is released from these tasks and will only process the information needed.

As mentioned, one advantage of using FPGAs is their ability to process data in parallel. The number of pixels that can be processed at the same time is limited due to the serial pixel stream from the camera. This creates a bottleneck in some way, but does not prevent a parallel process. The block diagram shown in Figure 4 describes the units that the data flows through. These units do not operate in a serial order but perform their calculations simultaneously. A microprocessor that consists of one ALU would not be able to perform the bayer interpolation of 4 pixels and determine the corresponding histogram bins of the resulting 12 values at the same time as it is the case using the FPGA.

## References

1. ModelSim Xilinx Edition-III. [www.xilinx.com/ise/verification/mxe\\_details.html](http://www.xilinx.com/ise/verification/mxe_details.html).
2. Uwe Furtner. Color processing with bayer mosaic sensors. [Online]:[www.matrix-vision.com/info/articles/pdf/art\\_bayermosaic.e.pdf](http://www.matrix-vision.com/info/articles/pdf/art_bayermosaic.e.pdf), August 2003.
3. Krzysztof Horecki, Dietrich Paulus, and Konrad Wojciechowski. Object Localization Using Color Histograms. In Karl-Heinz Franke, editor, *5. Workshop Farbbildverarbeitung*, pages 59 – 66, Ilmenau, 1999. Zentrum für Bild- und Signalverarbeitung e.V.
4. Richard C. Dorf John V. Oldfield. *Field Programmable Gate Arrays: Reconfigurable Logic for Rapid Prototyping and Implementation of Digital Systems*. Wiley-Interscience publication, 1995.
5. L. Kotoulas and I. Andreadis. Colour histogram content-based image retrieval and hardware implementation. In *IEE Proceedings Curcuits, Devices and Systems*, 2003.
6. Miriam Leeser, Natasha Kitaryeva, and Jill Crisman. Spatial and color clustering on an FPGA-based computer system. In *Configurable Computing: Technology and Applications, Proc. SPIE 3526*, pages 25–33. SPIE – The International Society for Optical Engineering, 1998.
7. Ulf Witkowski Teerapat Chinapirom and Ulrich Rueckert. Universal FPGA-microcontroller-module for autonomous minirobots. In *In Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE 2005), Awara-Spa, Fukui, JAPAN, 20. - 22. Sep. 2005*.
8. Felix Werner, Joaquin Sitte, and Frederic Maire. Automatic place determination using colour histograms and self-organising maps. In *Proceedings of the 13th International Conference on Advanced Robotics (ICAR 07)*, 2007.
9. XILINX. [http://www.xilinx.com/ise/logic\\_design\\_prod/webpack.htm](http://www.xilinx.com/ise/logic_design_prod/webpack.htm).
10. H. Zhang, Y. Gong, C. Y. Low, and S. W. Smoliar. Image retrieval based on color features: an evaluation study. In C.-C. J. Kuo, editor, *Proc. SPIE Vol. 2606, p. 212-220, Digital Image Storage and Archiving Systems, C.-C. J. Kuo; Ed.*, volume 2606 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 212–220, November 1995.