DreamCam : A modular FPGA-Based Smart Camera Architecture

Merwan BIREM, François BERRY

Institut Pascal - UMR 6602 UBP/CNRS - Campus des Cézeaux
24, Avenue des Landais
63177 AUBIERE Cedex France

Abstract

DreamCam is a modular smart camera constructed with the use of an FPGA like main processing board. The core of the camera is an Altera Cyclone-III associated with a CMOS imager and six private Ram blocks. The main novel feature of our work consists in proposing a new smart camera architecture and several modules (IP) to efficiently extract and sort the visual features in real time. In this paper, extraction is performed by a Harris and Stephen filtering associated with customized modules. These modules extract, select and sort visual features in real-time. As a result, DreamCam (with such a configuration) provides a description of each visual feature in the form of its position and the grey-level template around it.

Keywords: Smart Camera, Image Processing, Interest Points, VHDL, Harris & Stephen Algorithm, Field Programmable Gate Array (FPGA), Hardware Implementation, Real-Time System

1. Introduction

Intelligent robots are becoming increasingly important. One of the key components of an intelligent robot is its ability to understand its environment and recognize its position. In the robot community, most researchers use information from sources such as odometry, laser-range-finders, and sonar sensors. In contrast, in the vision community, new methods using information from camera sequence are being developed, see (Davison and Murray, 2002; Davison, 2003).

Using an entire image as an observation is difficult or impossible owing to the high resolution, typically of the order of a hundred thousand pixels. Thus, to identify interest points, feature extraction which is the first crucial step, should be used (Förstner, 1994).

The algorithms that extract features are time-consuming, which is a huge drawback when developing real-time applications. One solution to this problem is the use of dedicated hardware for the algorithms, such as Field Programmable

Gate Array (FPGA), which can provide dedicated functional blocks that perform complex image processing operations in parallel. In addition to the parallel properties of FPGA, which lead to a high-throughput, FPGA has a small footprint system and low power consumption, which makes it ideal for mobile applications.

FPGAs have achieved rapid acceptance and growth over the past decade because they can be used in a very wide range of applications (DeHon, 2000). Although they are slower than the traditional Application Specific Integrated Circuit (ASIC), their design flexibility is a major advantage. Users can change program as desired at any stage of the experiment thereby saving time and cost.

Throughout this paper, we propose a customized smart sensor based on a CMOS imager. The main original feature is system management by a System-On-Chip integrated in an FPGA. Our approach allows most early perception processes to be performed in the main sensing unit (FPGA), and sends just the main sensing features to the host computer so as to reduce a classic communication bottleneck. Another advantage of this method is the real time feedback on the sensor. The different parameters can be actively tuned to optimize perception to render it similar to primate perception. For instance, in strong light the pupil contracts and becomes small, but still allows light to be cast over a large part of the retina (Chalimbaud et al., 2007). This embedded sensor can be considered as a reactive architecture, and above all, as a research platform for the smart sensor.

To highlight the novel elements in our work, we present in the following section previous research carried out in this field. In Section 3, we give a large overview of the smart sensor. An application for the extraction of visual features based on the Harris and Stephen algorithm is presented in section 4. In this work, we consider feature extraction as a combination of a feature detection followed by a description. Thus, feature detection consists in finding the interest points (features) in the image, whereas feature extraction consists in representing them. The final goal is to compare them with other interest points for applications such as navigation, object recognition,.... In section 5 we include results that support the relevance of our approach and in section 6 we give a conclusion.

2. Previous work

Comparison of our work with previous works can be done on two levels :

- System-level: At this level, we propose to study the most popular smart cameras developed in the last decade. As a reminder, a smart camera is defined as a vision system in which the fundamental function is the production of a high level understanding of the imaged scene. A camera is called a "smart camera" when it performs application specific information processing (ASIP). The output of such cameras is either the features extracted from the captured images or a high-level description of the scene.

More details about this system can be found in articles by (Wolf, 2009) and (Shi and Lichman, 2011).

Table 1 presents an overview of the most common smart camera platforms found in the literature.

System	Platform Capabilities			Application
Author(s)	Sensor	CPU	Power	
CMUcam A. Rowe and Nourbakhsh (2007)	CMOS Omnivision	Proc. ARM7	battery	Robotic applications
MeshEye Hengstler et al. (2007)	ADNS-3060 optical mouse sensor + CMOS VGA	Microcontroller AT91SAM7S	battery	Distributed imaging applications
SeeMOS Chalimbaud and Berry (2007)	CMOS Cypress Lupa 4000	FPGA Stratix 60	mains	Tracking
LE2I-Cam Mosqueron et al. (2007)	CMOS Micron (MTM9M413)	FPGA Virtex II	mains	Hight speed imaging
WiCa mote Kleihorst et al. (2007)	VGA CMOS	Xetal IC3D	battery	Vehicle detection and speed estimation
ITI Bramberger et al. (2004)	LM-9618 CMOS	DSP TMS320C6415	mains	Traffic control

Table 1: Classification of smart camera systems

Others works have been based on Silicon-integrated smart cameras. In these systems, the authors propose an approach in which image sensing and processing are integrated on a single silicon die. This kind of device is called "vision chips". The interested reader can find a detailed description in Moini (2000). Among these works, the project scamp by P. Dudeck is one of the most well-known Carey et al. (2013). Other contributions in the same vein can be found in Moorhead and Binnie (1999) and Albani et al. (2002).

- Algorithm-level: As explained below, we implemented a "Harris and Stephen"-based algorithm to extract visual features in our camera. At the output of this extraction step, several modules (filtering, sorting, description) were added to provide high-level features from the image. Thus, in this part, we propose a short overview of published papers about the implementation of the Harris and Stephen detector on FPGA. To our knowledge, there is no work proposing a full approach with detection, filtering, sorting and description steps. Consequently, the works presented above, represent a fragmented bibliography mainly focused on the Harris detection.

  The work presented in(Ekstrand et al., 2008) implements a Harris detection on a FPGA connected to a stereo rig. The FPGA provides a simple extraction on 320×480 pixels at 27 fr/s. Another "stereo camera-based" work was proposed by (Dietrich, 2009). In this work, the author used a Spartan-3E to rectify the stereo images and to detect the Harris points. Most of the authors had the same basic approach on how to implement the Harris and Stephen detector. In Goshorn et al. (2010), the authors used a Harris Filter to define theregion of interest. With these windows, a classifier is used to detect and identify some objects in the scene.

System	Platform Capabilities		
Author(s)	Sensor	Processor	Resol. @ Fps
Dietrich (2009)	EyeBot M6 OV6630	Xilinx Spartan-3E FPGA	352 × 288 @ 7.37
Ekstrand et al. (2008)	MT9P031 CMOS Micron	Xilinx Virtex-II FPGA	320 × 480 @ 27
Goshorn et al. (2010)	Video frame from a memory	Xilinx Virtex-5 FPGA	640 × 480 @ 266

Table 2: Hardware implementation of H&S algorithm

However, these works propose only architectures to detect the corners by the Harris and Stephen method. In our work, we propose to filter the detected points, to sort the most robust ones and to describe each feature by a grey-level template. These last steps are fundamental in computer vision, in which the input is not images but semantic features.

3. Hardware Description of the "DreamCam"

The goal of artificial vision research is to exploit images or image sequences generated by sensors, in order to effectively translate an environment. From this translation, different processes can be used to identify or inspect an object, control robots, etc. The first way to treat the vision problem is to carry out passive vision approaches, which is the classic way to analyze images. In contrast, another approach exists known as "*active vision*", which is the result of an attempt to simulate the human visual system.

Based on this concept, our approach consists in integrating the control of the imager in the perception loop, especially in the early perception processes. By integration of early processing, close to the imager, a reactive sensor can be designed. With such a smart sensor, it is possible to perform basic processing and the selection of relevant features (DeHon, 2000). For example, FPGAs have already been used to accelerate real-time point tracking (Benedetti and Perona, 1998), stereo-vision (Woodfill and Von Herzen, 1997), color-based object detection (Benitez and Cabrera, 1999), and video and image compression (Böhm et al., 2002). In our case, the notion of a system on programmable chip (SOPC) describes the whole system.

Most vision applications are focused on several small image areas and consequently acquisition of the whole image is not always necessary. It is evident, therefore, that one of the main goals of an efficient vision sensor is to select regions of interest (ROI) in the image and concentrate on processing resources on these. The notion of local study is then predominant and the choice of imaging technology becomes crucial. This explains why the CMOS imager was chosen. It is generally accepted that the CMOS technology, due to its capabilities will replace the traditional CCD technique in many applications :

- to allow accessing only parts of the image (ROI)
- to allow higher speeds (up to 60 MHz per output channel)

- to allow functionality on the chip (camera-on-the-chip)
- to provide a much higher dynamic range (up to 120 dB)
- to be based on standard manufacturing processes

The global processing system is composed of a SOPC (System On Programmable Chip), by which an entire system of components is put on a single chip (FPGA). The fine-grained structure of the FPGA allows the development of extremely optimized implementations. Image processing is well known to be algorithmically simple but computationally costly. Moreover, FPGA is the best candidate for a wide group of peripheral devices. DreamCam is a modular smart camera in which image sensors or communication boards can be easily changed.

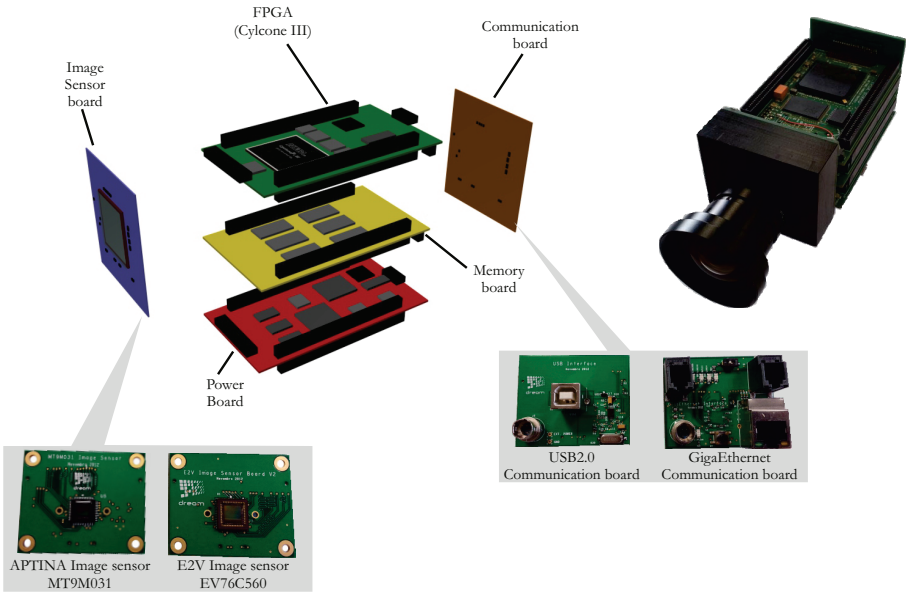

Figure 1: Overview of the camera and synoptic of the assembly

3.1. Global Hardware Architecture

The architecture of the camera is constructed with five interconnected boards as shown in figure 1. The core of this system is a FPGA which allows a high versatility. Thus, the image sensor board and the communication board can be easily replaced or updated in order to change the type of imager or the communication layer. Currently, we can propose two different image sensors and the ability to use a USB2.0 or Giga-Ethernet communication link. Each board is described in detail below.

3.1.1. Image sensor board

Both developed image sensor boards are based on a similar electronic architecture. This architecture can accept parallel differential or single-ended outputs from different kinds of image sensors. The image sensors used in this work are:

- MT9M031 imager: This 1.2-mega pixel (1280×960)CMOS image sensor is manufactured by Aptina. It can operate at 45 fps at full 1280x960 pixel resolution or at 60fps speed at 720pHD resolution (reduced FOV). The power consumption is 270mW in 720p60 mode. The dynamic range is 83.5dB - quite big for a global shutter sensor.

- EV76C560 imager: This is a 1.3-mega pixel (1280×1024) CMOS active pixel sensor dedicated to industrial vision features both rolling and global shutters. The pixel design offers excellent performance in low-light conditions with a high-readout speed of 60 fps in full resolution. Novel pixel integration/readout modes and embedded image pre-processing deliver superior performance parameters, including a bi-frame wide dynamic range ($>$ 100dB). Other on-chip pre-processing are included such as Image Histograms, Multi-ROI, Defective pixel correction,...

3.1.2. Processing board

This is the main part of the system using an Altera Cyclone-III EP3C120 FPGA (Fig-2). The need for strong parallelization led us to connect 6×1MBytes SRAM asynchronous memory blocks to the FPGA. Each memory has a private data and a private address bus. Consequently, six processes (using 1MB each) can access all the memories at the same time. We chose a low-power Cyclone III FPGA family. The reasons for this choice for Cyclone are given below:

- Firstly, its architecture consists of 120K vertically arranged logic elements (LEs), 4 Mbits of embedded memory arranged as 9-Kbit (M9K) blocks, and 288 18x18 embedded multipliers.

- Secondly, Cylcone integrates DSP Blocks. These embedded DSP Blocks have been optimized to implement several DSP functions with maximum performance and minimum use of logic resource. In addition, these embedded DSP Blocks can be used to create DSP algorithms and complex math routines in high-performance hardware DSP Blocks and they can be viewed as custom instructions to the NIOS CPU.

- Lastly, Cyclone is optimized to maximize the performance benefits of SOPC integration based on a NIOS embedded Processor. A NIOS processor is a user configurable soft core processor, allowing many implementations and optimization options.

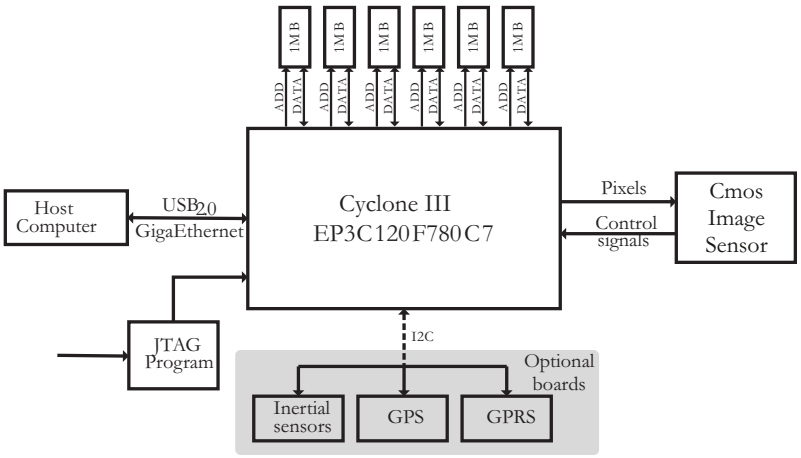

Figure 2: Synoptic of DreamCam

3.1.3. Communication board

This board is connected to the main board and manages all communications with the host computer. The communication layer is currently either high-speed USB 2.0 or Giga-Ethernet.

- USB2.0 is managed by the Cypress cy7c68013 microcontroller. It incorporates an enhanced processor based on a 8051 core and the instruction set is compatible with standard 8051, and in many ways improved. For example: The maximum operating frequency up to 48MHz, an instruction cycle is four clock cycles , two UART interfaces , and three time counters, an I2C interface.

- The Giga-Ethernet protocol is taken in charge by the Marvel 88E1111 transceiver. It is a physical layer device containing a single Gigabit Ethernet (GbE) transceiver. The transceiver implements the Ethernet physical layer portion of the 1000BASE-T, 100BASE-TX, and 10BASE-T standards.

3.1.4. Memory Board

The bank of memories contains six SRAM asynchronous memories, each of which has a size of 1MWords. These memories are high-speed, 16M-bit static RAMs organized as 1024K words by 16 bits. They have a high-speed access times 8ns under 3.3V and can be easily controlled. For instance, the read cycle consists only in accessing an address and after an output hold time the data can be read.

3.1.5. Power Board

The different boards need different kinds of voltage according to the respective devices. The initial input voltage is 6.5V and a set of regulators generates the different voltages. The global no-programmed power consumption is approximately 1.4W. Of course, this consumption widely varies with the configuration of the FPGA. This board provides 18 different voltages from 1.2V to 5V. In addition, a JTAG programmer (USB Blaster-like) has been integrated to configure the Cyclone III FPGA.

3.2. Internal FPGA Design

The aim of the proposed design is to create a flexible interface between the sensing device board and the host computer. This is how the whole system is separated into two main parts: a software part which is basically a **C++** code that retrieves the data that are in the USB packets sent from DreamCam; and a hardware part, which is developed in this paper.

In this approach, two blocks are very important and must be used for each design : the first one controls the CMOS image sensor which is the **Image sensor IP** block, and the second manages communication between the host computer and the DreamCam. The **Mem IP** block is used when external memories are needed. Theses blocks control each memory by generating the appropriate input signal of the memory such as (Address Bus : A0-A19, Chip Enable signal : CE, Write Enable signal : WE, Output Enable signal : OE) and by receiving the data. Finally the **Image processing algorithm** block will contain the algorithm that we want to implement on FPGA (in the present work the Harris algorithm was chosen). The diagram of the system is shown in Fig-3.

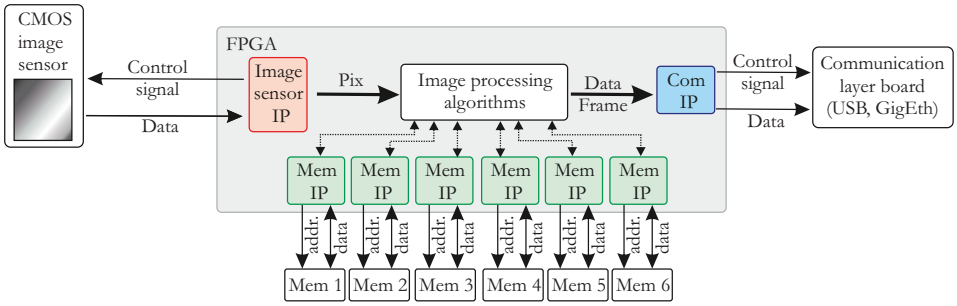

Figure 3: Internal FPGA Design

These different blocks will work with each other as follows. After powering the system, the CMOS imager starts to work under the control of the **Image sensor IP** inside the FPGA by sending the pixels of the image one by one and line by line (Flow noted `Pix` in figure 3. These pixels are sent to the **Image processing algorithm** block, where they will be processed according

to the algorithm implemented on it. After that, the results are sent to the **Communication IP** block (Flow noted `Data Frame` in figure 3, where they will be packed and sent.

4. Harris corner extractor application

In an image, the corner is an important local feature which focuses on a great amount of important image information and is rarely affected by illumination change (Chen, 2005). In addition, it has rotation invariant properties (Harris and Stephens, 1988). Provided there is no data loss, the corner feature is the smallest piece of data to deal with so that it improves the speed of detection. Thus, corner detecting has many important applications in practice, especially in the real-time target tracking field and autonomous navigation of vehicles.

In this section, we propose implementation of a feature extractor on the Dream-Cam. The term feature extractor is used to describe the combination of a feature detector and a feature descriptor. Detectors are used to find interest points in an image, after which a descriptor is created that describes the local neighborhood around the points. Mikolajczyk and Schmid (2005) have written a state-of-the-art overview of feature extractors.

Many extractors of features from an image have been reported in the literature. They differ in the method used to detect and describe the features, which implies a difference in algorithm complexity, processing time and resources needed. However, if the complexity of the algorithm increases, computation becomes heavier.

The feature extractor used in our work is a combination of the Harris corner detector (Harris and Stephens, 1988) and a simple descriptor which gives for each interest point an intensity patch from the image. The Harris corner detector (also known as Harris-Stephens or Plessy detector) is one of the most widely used interest point detectors, owing to its improved detection rate over the Moravec (Moravec, 1980) corner detector and to its high repeatability rate.

In the Harris corner detector the main operations used are the first derivative and convolution. These operations are single instruction multiple data (SIMD) and therefore highly parallelizable, which means they are suitable for implementation on FPGAs, which are low cost and high density gate arrays capable of performing many complex computations in parallel while hosted by conventional computer hardware.

A pixel is considered to be an interest point when its interest value R_i is higher than the predefined threshold, and the higher this value the more accurate is the detected interest point. The value is computed for each pixel according to (Harris and Stephens, 1988) using the following formula :

$$R_i = Det(M_i) - k^1 * (Trace(M_i))^2$$

[1] $k \in [0.04, 0.06]$ is an empirical value(Harris and Stephens, 1988)

where :

$$M_i = \begin{pmatrix} A & C \\ C & B \end{pmatrix}$$

which means that :

$$R_i = (A \times B - C^2) - k * (A + B)^2$$

- $A = (\frac{\delta I_i}{\delta x})^2 \otimes W$
- $B = (\frac{\delta I_i}{\delta y})^2 \otimes W$
- $C = (\frac{\delta I_i}{\delta x}\frac{\delta I_i}{\delta y}) \otimes w$

W is the 3x3 Gaussian filter :

$$w(u, v) = \exp\left(-\frac{(u^2 + v^2)}{2\sigma^2}\right)$$

and $\frac{\delta}{\delta x}$, $\frac{\delta}{\delta y}$ are the x and y derivatives operators. I_i is the 3x3 window surrounding the i^{th} pixel, and $\otimes$ represents the convolution operator.

The feature extraction system proposed in this paper detects first the interest points on an image, sorts them, and then describes them using a patch of pixels from the image. The system is composed of several modules (see Fig-4) that have been developed in VHDL (VHSIC Hardware Description Language), and are fully compatible with a FPGA implementation. The main modules of the system are as follows.

1. The Harris corner detector module : detects the interest points and filters them, in order to obtain only one point (pixel) for each corner,
2. The sort module : sorts the interest points in decreasing order according to their interest value R_i,
3. The swap memories module : retrieves the patch of each interest points, and constructs the data frame containing the interest point coordinates (X_i, Y_i) and their patches. More details about the data frame are given in section 4.3.

In the first module the results of the detection are filtered, because when an image is treated using the Harris corner detector, several pixels around a corner will be considered as interest points. The desired outcome is having one interest point, which means one pixel for each corner (see Fig-6). In previous works (Yiran, 2006; Dietrich, 2009; Brandon and Arin, 2010; Goshorn et al., 2010), this problem was solved by the non-maximum suppression method. To perform such a suppression, a window of a specific odd size is moved over the picture after treating all the pixels. If the center value of the window is the maximum interest value within the whole window the filter response is one, otherwise the filter response will be zero. In terms of hardware considerations, this method has several disadvantages such as the use of more memories and FIFO queue. In addition, it induces latency due to the buffering of three lines at minimum when a 3×3 window is used to perform the non-maximum suppression.

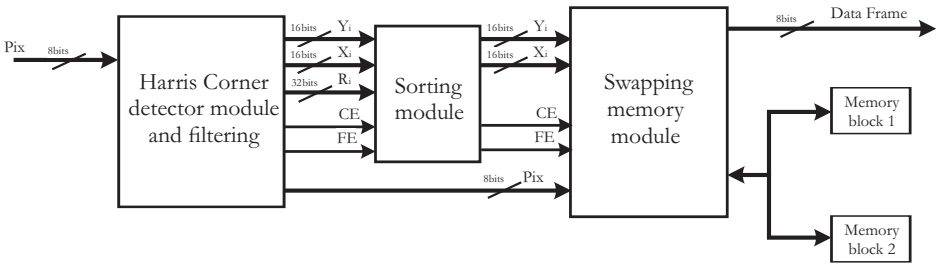

Figure 4: Block diagram of the implemented system, composed of three hardware blocks, used to extract features from the images

4.1. Harris corner detector module

This module represents the feature detector (the first part of the feature extractor). Fig-5 gives an overview of the architecture used to implement the Harris detector algorithm on FPGA.

To achieve a higher frequency, the system has to be parallelized to the maximum degree possible allowed by the architecture of the smart camera presented in this article. As shown in Fig-5, all operations that are independent of one another were implemented separately. The performance of the system can also be increased by using DSP-blocks for all the multiplications and summations that are in the Harris corner detector algorithm.

The system receives the stream of pixels and places them one by one in a FIFO queue. The calculation of the interest value R_i will start when the FIFO queue is almost full, more precisely when the second pixel of the fifth line is reached. After calculation of the interest value R_i, a simple comparator is used to determine if the treated pixel is an interest point or not.

This module contains a submodule that has nearly the same function as non-maximum suppression. The main difference between the two is in the pixel that will be kept. In non-maximum suppression the pixel kept is the one with the highest interest value R_i in an odd-size window. In the submodule implemented on FPGA the pixel kept is the first one to appear, which means there is no need for more memory or latency to obtain the results.

This submodule is based on two notions. When an interest point is detected the system will check if there is no interest point near it in the m previous lines. If this is the case, the pixel will be set as an interest point, and the following n pixels will not be treated at all. If there is an interest point near it in one of the m previous lines the system passes to the next pixel and so on. Fig-6 shows an example of the results obtained with and without this module.

The most important output signals of this module are CE, FE, X_i and Y_i. The first and second ones are set to "1" (for one clock cycle) when an interest point is detected and when the last pixel of an image is treated, respectively. The last two signals, X_i and Y_i, represent the coordinates of the detected interest point.

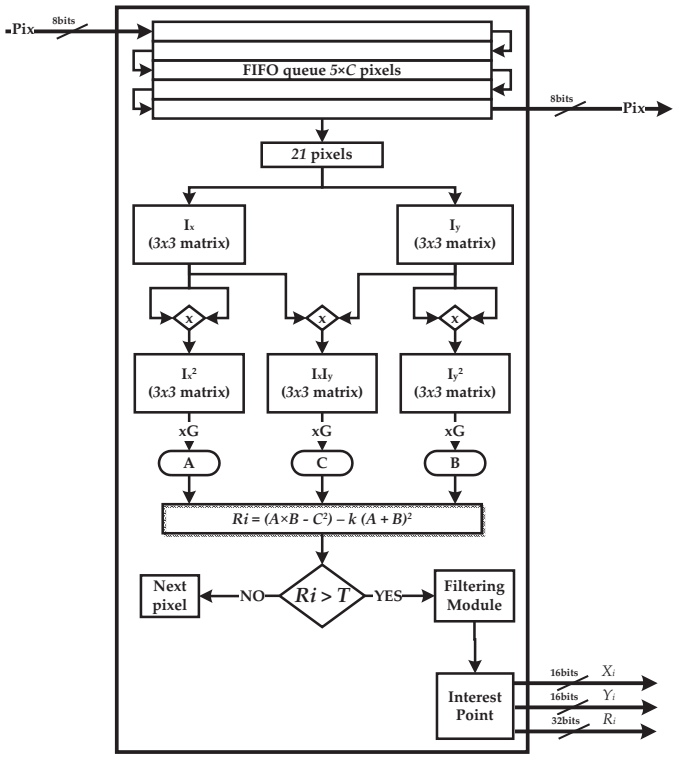

Figure 5: Block diagram of the implemented Harris corner detector module

4.2. Sort module

The robustness and accuracy of an interest point depends on the value of R_i and the higher the value of R_i, the more robust and accurate the point.

The technique used to sort the interest points is that described in the paper of Yasuura et al. (1982). It is done in two steps :

- **Step 1 :** The ordering process, in which the order of the detected interest points is found,
- **Step 2 :** The rearranging process of the points, which places them in a memory according to their order.

In this method, the sorting of the detected interest points requires the presence of all points. This is why the sorting is done only after image processing. It means that sorting the points detected in the "i^{th} image" will be done while the "$i + 1^{th}$ image" is being processed.

The principle of this sorting method is as follows. For a given set of interest points $SetIP = \{IP_1, IP_2, \ldots, IP_n\}$, the order C_i (of each interest point) is easily calculated by counting results of comparisons between R_i (the interest

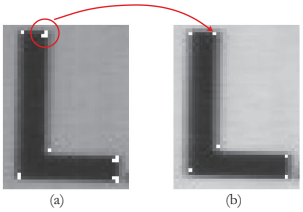

Figure 6: An example of the results obtained with the filtering module (real-time images) : (a) Detection without filtering, (b) Detection with filtering

value of the i^{th} point) and all the other values. Each time a value higher than R_i is found, C_i is incremented by 1. C_i represents the number of items in the set having a value higher than the i^{th} point, and represents the order of the point. The rearranging process uses the different values of C_i as addresses to put the points in decreasing order in a memory.

The basic algorithm to compute the order C_i is as follows.

Algorithm 1 Compute the order : C_i

```
C_i = 0
while j ≤ n do
  if R_i < R_j then
    C_i = C_i + 1
  end if
  j = j + 1
end while
```

4.3. Swap memory module

This module represents the feature descriptor (the second part of the feature extractor). As mentioned previously, the descriptor chosen to be implemented on FPGA is a simple one, which gives an intensity patch for each interest point. This module receives signals CE, FE, X_i and Y_i from the sort module. The signals are used to construct the data frame shown in Fig-7.

The first two elements and all the (X_i, Y_i) coordinates of the data frame are obtained from the Harris corner detector and sort modules. The pixels that are in each patch are obtained from one of two memories, $M1$ or $M2$ (see Fig-4). These memories contain the previous treated image (put on read mode), and the actual image under treatment (put on write mode), respectively. At the end of each image treatment the two memories change their operating mode i.e. swap mode.

This module is composed of two processes. The first one constructs the table that will contain the memory addresses of the detected interest points. The second process constructs the data frame. To do this, the process uses the memory put on read mode and the table constructed by the first process. This

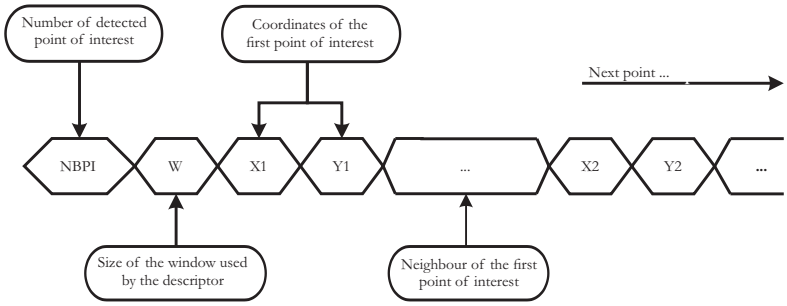

Figure 7: The data frame constructed by the swap memory module

module controls the two asynchronous memories by controlling their *WE* signal, and their *DATA* and *ADD* buses.

The combination of this module with the Harris corner detector and the sort module will give us a full feature extractor that detects, sorts, and describes the interest points. In other words, the extractor takes images as input and provides semantic information as output. This information can be used for navigation, 3D reconstruction or other applications.

5. Experimental results

The proposed algorithm (presented in the previous section) was implemented on the DreamCam. For all results given in this section, the DreamCam was equipped with an E2V imager and a USB2.0 communication layer. The image resolution was of 800×1024 and the size of each interest point patch was set at 15×15. The software used for these experiments was Quartus II V13.0 in setting the default options (no optimization for performance or particular effort level).

5.1. Consumption of FPGA resources

A first result concerns the consumption of resources in the FPGA. The desired number of interest points directly impacts the consumption of logic elements in the FPGA. This is due to the feature descriptor module (named Swap memory module in the architecture), whose role is to prepare data output. To do this it has to store each point descriptor with its grey-level template, its coordinates and Ri value, and it is this storage process that draws on internal resources of the FPGA.

Figure 8 shows the linear consumption of Logic Elements for a linear increase of desired interest points.

For information purposes, table 3 gives all resources used for 200, 400 and 600 points of interest.

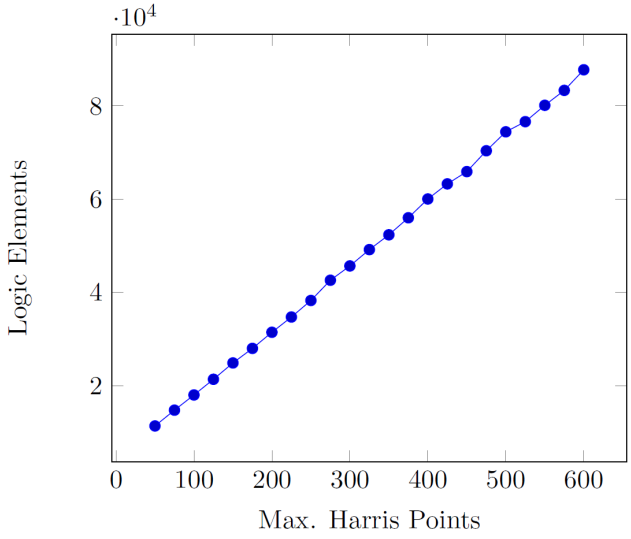

Figure 8: Used Logic Element according to the number of desired interest points. For this experiment, a set of synthesis was done from 50 points to 600 points in steps of 25 points. The resolution of the image was 800×1024.

Number of interest points	200	400	600
Total Logic Element	31433	60034	87754
Combinatorial	16601	29121	41636
Registers	23014	43814	64615

Table 3: FPGA resources used

5.2. Maximum frequency

A second result consists of the maximum frequency of work according to the maximum number of Harris points. The maximum frequency decreases from 105 MHz to 80 MHz (Fig. 9). This decrease is explained by the increase in the length of the critical path in the Swap memory module.

5.3. Comparison with others works

As explained in the section "Previous works", others authors focused only on the Harris and Stephen implementation. We propose therefore to compare the performance of the existing works with our implementation of Harris and Stephen module in table 4.

The best result is given by Goshorn et al. (2010), but they used a Virtex 5. Virtex 5 offers a clock tree up to 550 MHz, whereas the clock tree specification for Cyclone III (in 7 speed grade) is only 430MHz.

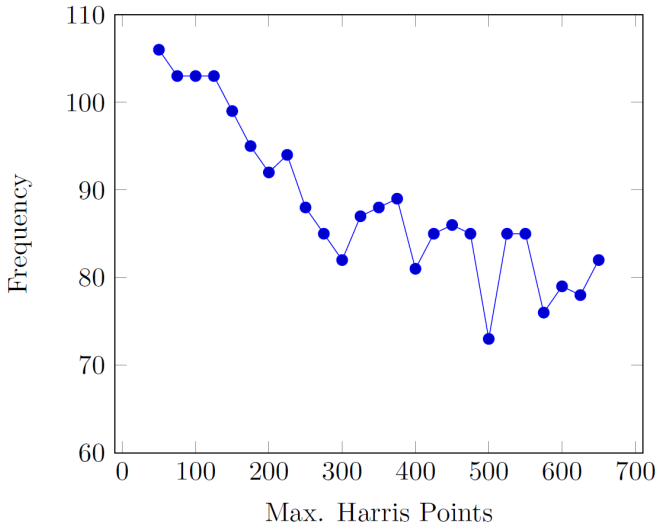

Figure 9: Maximum frequency of the system according to NPI (maximum number of Harris points). For this experiment, a set of synthesis was done from 50 points to 600 points in steps of 25 points. The resolution of the image was 800×1024.

5.4. Experiments

Fig-10 shows the results obtained when processing images captured from a mobile robot in experimental conditions. Images on the left are obtained without the feature descriptor, and images on the right are obtained with the feature descriptor. The size of images used for this were 256×256 pixels. The latter are constructed using the data frame that the Harris corner extractor sends to the host PC.

The Harris corner extractor implemented on FPGA works on the stream of pixels coming from the CMOS imager and because of this the size of the data frame must be smaller than or equal to that of the image treated to allow the data frame to be sent entirely without the loss of any information. In general, if the images treated have $L \times Cpixels$ each, and the patch chosen to describe the interest points has $W \times Wpixels$, then the maximum number of interest points that the system can detect is $n < \frac{L \times C - 2}{W \times W + 4}$, where -2 is for the first two elements of the data frame (the number of interest points detected, and the size of the patch), and $+4$ is for the two coordinates X_i, Y_i of each interest point. $+4$ means that the two coordinates are encoded on *two bytes* each, which will allow the system to obtain the coordinate of interest points detected in images that have a size greater than $256 \times 256 pixels$.

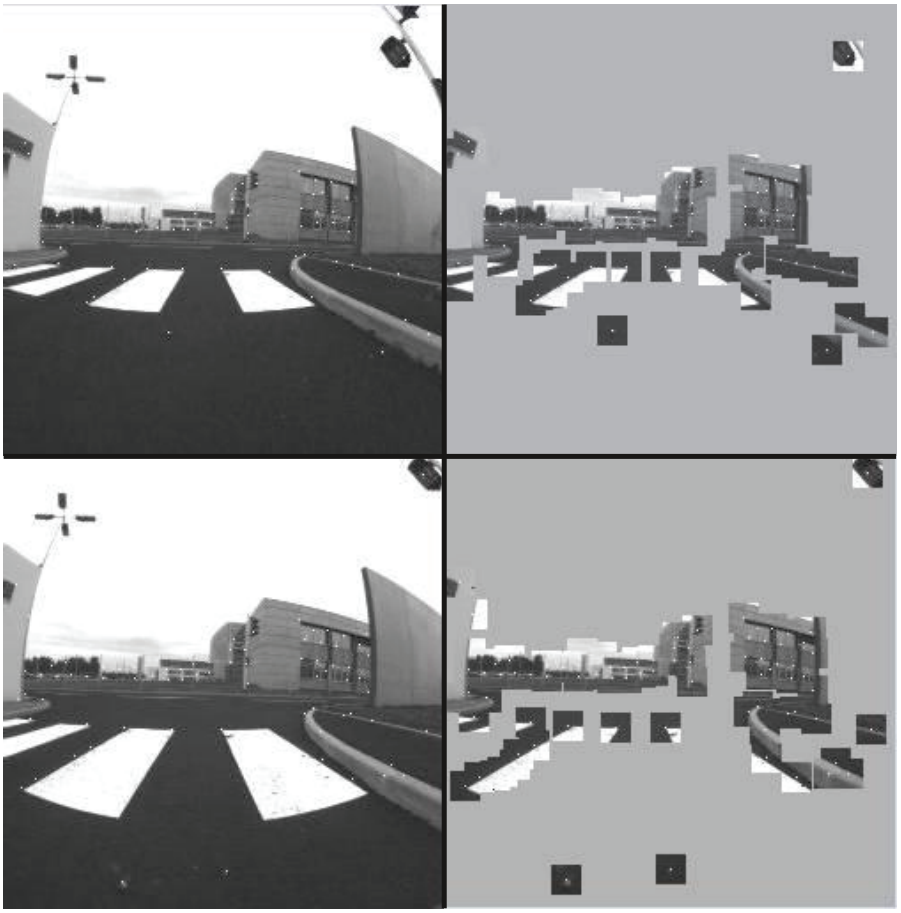

Figure 10: The results obtained when processing images were captured from a mobile robot. On the left the whole image with the interest points in white. On the right the point of interest and its neighbor

System	Platform Capabilities	
Author(s)	Processor	Resol. @ Fps
Our method	Altera Cylcone III	1024×800 @ 76 $F_{max} = 62MHz$
Dietrich (2009)	Xilinx Spartan-3E	352×288 @ 8 $F_{max} = 8MHz$
Ekstrand et al. (2008)	Xilinx Virtex-II	320×480 @ 27 $F_{max} = 41MHz$
Goshorn et al. (2010)	Xilinx Virtex-5	640×480 @ 266 $F_{max} = 81MHz$

Table 4: Comparison with others hardware implementations of H&S algorithm

6. Conclusion and future works

This paper describes the construction of a sensor for real-time visual applications. Its main originality consists in using CMOS imager and FPGA architecture to create a versatile smart system. The approach, based on FPGA Technology and CMOS imager, reduces the classic bottleneck between sensor and processing unit.

The system can acquire images of superior quality using the 1.3-mega pixel (1280*1024) CMOS image sensor IBIS5. Precise timing control guarantees the accuracy of image data. ROI readout guarantees the high frame rate of the system (more than $100fps$ for 640×480 pixels). The average transmission speed with USB is 48MB/s, which will meet the demands of real-time data transmission. The system can be used in many applications with demands of high resolution, high frame rate and real-time requirements.

The feature extractor application was implemented with success on the Dream-Cam, which can process up to $42fps$ (800×1024 pixels) and gives good quality results, as seen in section 5. The blocks of this application were developed in generic mode, which means the user can change the size of the image, the number of points needed, the lowest threshold allowed, and other parameters, and compile and synthesize the project to obtain a new system.

Two further steps could be implemented to improve the project. First, the development of an entire controller of the system from the PC, so that with a simple click on the mouse or the keyboard the DreamCam is reconfigured, Global or Rolling shutter mode, size of the image, integration time, threshold of the algorithm and other parameters are chosen or set to a particular value without recompilation of the HDL project. Second, the addition of blocks to do feature tracking or matching.

7. Acknowledgements

The work reported in this paper was supported by the Euripides European Program (Eureka), Seamoves Project, and the Altera Corporation under an

equipment grant.

A. Rowe, A.G. Goode, D. G., Nourbakhsh, I., may 2007. Cmucam3: An open programmable embedded vision sensor. tech. report CMU-RI-TR-07-13, Carnegie Mellon University.

Albani, L., Chiesa, P., Covi, D., Pedegani, G., Sartori, A., Vatteroni, M., 2002. Visoc : a smart camera soc. 28th European Solid-State Circuits Conference, Florence, Italy, p. 367370.

Benedetti, A., Perona, P., 1998. Real-time 2-d feature detection on a reconfigurable computer. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR '98. IEEE Computer Society, Washington, DC, USA, pp. 586–.
URL http://dl.acm.org/citation.cfm?id=794191.794768

Benitez, D., Cabrera, J., 1999. Reactive computer vision system with reconfigurable architecture. In: Proceedings of the First International Conference on Computer Vision Systems. ICVS '99. Springer-Verlag, London, UK, UK, pp. 348–360.
URL http://dl.acm.org/citation.cfm?id=645549.659164

Böhm, W., Hammes, J., Draper, B., Chawathe, M., Ross, C., Rinker, R., Najjar, W., Feb. 2002. Mapping a single assignment programming language to reconfigurable systems. J. Supercomput. 21 (2), 117–130.
URL http://dx.doi.org/10.1023/A:1013623303037

Bramberger, M., Brunner, J., Rinner, B., 2004. Real-time video analysis on an embedded smart camera for traffic surveillance. In: In Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium. pp. 174–181.

Brandon, M., Arin, M., December 2010. Rapid corner detection using fpgas. National Aeronautics and Space Administration.

Carey, S. J., Barr, D. R., Dudek, P., 2013. Low power high-performance smart camera system based on {SCAMP} vision sensor. Journal of Systems Architecture.
URL http://www.sciencedirect.com/science/article/pii/S1383762113000490

Chalimbaud, P., Berry, F., January 2007. Embedded active vision system based on an fpga architecture. EURASIP J. Embedded Syst. 2007, 26–26.
URL http://dx.doi.org/10.1155/2007/35010

Chalimbaud, P., Maromoiton, F., Berry, F., 2007. Towards an embedded visuo-inertial smart sensor. Int. Journal in Robotics Research.

Chen, L., 2005. A survey of corner detection algorithms. Techniques of Automation and Applications 24, 55.

Davison, A. J., 2003. Real-time simultaneous localisation and mapping with a single camera. In: Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2. ICCV '03. IEEE Computer Society, Washington, DC, USA, pp. 1403–.
URL http://dl.acm.org/citation.cfm?id=946247.946734

Davison, A. J., Murray, D. W., July 2002. Simultaneous localization and map-building using active vision. IEEE Trans. Pattern Anal. Mach. Intell. 24, 865–880.
URL http://dl.acm.org/citation.cfm?id=628329.628800

DeHon, A., Apr. 2000. The density advantage of configurable computing. Computer 33 (4), 41–49.
URL http://dx.doi.org/10.1109/2.839320

Dietrich, B., 2009. Design and implementation of an fpga-based stereo vision system for the eyebot m6. Ph.D. thesis, University of Western Australia.

Ekstrand, F., Jorgen, L., Lars, A., October 2008. Robotics for smes - 3d vision in real -time for navigation and object recognition. In: 39th International Symposium on Robotics (ISR 2008). IDR 2008, pp. 70–75.

Förstner, W., 1994. A framework for low level feature extraction. In: Proceedings of the third European conference on Computer Vision (Vol. II). Springer-Verlag New York, Inc., Secaucus, NJ, USA, pp. 383–394.
URL http://dl.acm.org/citation.cfm?id=200241.200283

Goshorn, D., Cho, J., Kastner, R., Mirzaei, S., 2010. Field programmable gate array implementation of parts-based object detection for real time video applications. In: Proceedings of the 2010 International Conference on Field Programmable Logic and Applications. FPL '10. IEEE Computer Society, Washington, DC, USA, pp. 582–587.
URL http://dx.doi.org/10.1109/FPL.2010.114

Harris, C., Stephens, M., 1988. A combined corner and edge detector. In: Proceedings of the 4th Alvey Vision Conference. pp. 147–151.

Hengstler, S., Prashanth, D., Fong, S., Aghajan, H., 2007. Mesheye: a hybrid-resolution smart camera mote for applications in distributed intelligent surveillance. In: In IPSN 07: Proceedings of the 6th international conference on Information processing in sensor networks. ACM Press, pp. 360–369.

Kleihorst, R., Abbo, A., Schueler, B., Danilin, A., 2007. Camera mote with a high-performance parallel processor for real-time frame-based video processing. In: Proceedings of the 2007 IEEE Conference on Advanced Video and Signal Based Surveillance. AVSS '07. IEEE Computer Society, Washington, DC, USA, pp. 69–74.
URL http://dx.doi.org/10.1109/AVSS.2007.4425288

Mikolajczyk, K., Schmid, C., October 2005. A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Mach. Intell. 27, 1615–1630.
URL http://dl.acm.org/citation.cfm?id=1083822.1083989

Moini, A., 2000. Vision chips / by Alireza Moini. Kluwer Academic Boston ; London.

Moorhead, T., Binnie, D., 1999. Smart cmos camera for machine vision applications. IEE Conference on Image Processing and its Applications, Manchaster, UK, p. 865869.

Moravec, H., September 1980. Obstacle avoidance and navigation in the real world by a seeing robot rover. In: tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & doctoral dissertation, Stanford University.

Mosqueron, R., Dubois, J., Paindavoine, M., 2007. High-speed smart camera with high resolution. EURASIP J. Emb. Sys. 2007.

Shi, Y., Lichman, S., 2011. Smart cameras: A review.

Wolf, W., July 2009. Towards pervasive smart camera networks.

Woodfill, J., Von Herzen, B., 1997. Real-time stereo vision on the parts reconfigurable computer. In: Proceedings of the 5th IEEE Symposium on FPGA-Based Custom Computing Machines. FCCM '97. IEEE Computer Society, Washington, DC, USA, pp. 201–.
URL http://dl.acm.org/citation.cfm?id=549928.795743

Yasuura, H., Takagi, N., Yajima, S., December 1982. The parallel enumeration sorting scheme for vlsi. IEEE Trans. Comput. 31, 1192–1201.
URL http://dx.doi.org/10.1109/TC.1982.1675943

Yiran, L., December 2006. Fpga implementation for image processing algorithms. Digital Signal Processing.