DEGREE OF DOCTOR OF PHILOSOPHY IN
COMPUTER ENGINEERING AND SCIENCE

DOCTORATE SCHOOL IN
INFORMATION AND COMMUNICATION TECHNOLOGIES

XXIV Cycle

UNIVERSITY OF MODENA AND REGGIO EMILIA

INFORMATION ENGINEERING DEPARTMENT

Ph.D. DISSERTATION

Green Vision and Embedded Systems

Candidate:
PAOLO SANTINELLI
Advisor:
PROF. RITA CUCCHIARA
The Director of the School:
PROF. GIORGIO M. VITETTA

DOTTORATO DI RICERCA IN
COMPUTER ENGINEERING AND SCIENCE

SCUOLA DI DOTTORATO IN
INFORMATION AND COMMUNICATION TECHNOLOGIES

XXIV Ciclo

UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

TESI PER IL CONSEGUIMENTO DEL TITOLO DI DOTTORE DI RICERCA

Green Vision and Embedded Systems

Tesi di:
PAOLO SANTINELLI
Relatore:
PROF. RITA CUCCHIARA
Il Direttore:
PROF. GIORGIO M. VITETTA

Contents

List of Figures

List of Tables

Acknowledgments

After I have been spending 20 years teaching in the school, I decided to come back to the school, as a student rather than as a teacher. It would not have been possible to do this experience without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here. Above all, I would like to thank my wife Catia for her personal support and great patience at all times. My son and my daughter, they have given me their unequivocal support throughout, for which my mere expression of thanks likewise does not suffice. I want to acknowledge my advisor Prof. Rita Cucchiara and Prof. Andrea Prati. Without their help, support and patience this experience and this thesis would not have been possible. Thanks at all the researchers of the ImageLab group for the good advices, support and friendship they have shown, for which I am extremely grateful. I want to express all my gratitude to Mauricio Casares and Prof. Senem Velipasalar for the friendship proven and the support they provided during my three-months staying at the University of Nebraska Lincoln.

Chapter 1

Introduction

Embedded systems represent a major fraction of the digital systems market as indicated by the fact that embedded systems represent a key technology in the automotive, consumer electronics, industrial automation, military and aerospace applications, office automation, telecommunication and data communication industries. As much as 98% of all 32-bit microprocessors currently in use worldwide are used in embedded systems. These embedded processors provide special purpose functionality as opposed to the genera lpurpose applications familiar to desktop computer users. A recent report estimates that the typical household has 100 processors in its confines. The growth rate for embedded processors far exceeds that of traditional computers. Embedded systems span all aspects of modern life and there are many examples of their use.

Mobile devices are embedded system devices that derive the energy required for their operation from batteries. In the case of many consumer electronics devices, especially mobile phones, battery capacity is severely restricted due to constraints on size and weight of the device. This implies that energy efficiency of these devices is very important to their usability. Hence, optimal management of power consumption of these devices is critical. At the same time, device functionality is increasing rapidly. Modern high-end mobile phones combine the functionality of a pocket-sized communication device with PC-like capabilities, resulting in what are generally referred to as smartphones. These devices integrate such diverse functionality as voice communication, audio and video playback, web browsing, short-message and email communication, media downloads, gaming and more. The rich functionality increases the pressure on battery lifetime, and deepens the need for effective energy management.

As en example, video surveillance systems are a fast-growing class of networked embedded systems. The miniaturization of image sensors (cameras) combined with

powerful embedded processors allows engineers to build smart video nodes that can process a video stream and transmit it to a central gateway through an interconnection network.

Smart cameras capture high-level descriptions of a scene and perform real-time analysis of what they see. These low-cost, low-power systems push the design space in many dimensions, making them a leading-edge application for embedded system research. Smart cameras, characterized by performing on-board video analysis, start to be adopted for real-world applications like assisted living. Typically these systems rely on fixed infrastructure like Ethernet and main power supply. Visual sensor networks represent an emerging trend in research, combining aspects of smart cameras and wireless sensor networks. A key factor for cameras to become truly pervasive sensors embedded in our environment, is a reduction of infrastructure requirements. As power consumption is an important issue, visual sensor networking platforms often are specifically designed for a certain purpose, exposing only limited interface options as well as using low performance components.

Wireless embedded smart cameras are stand-alone units that combine sensing, processing and communication on a single embedded platform. Wireless embedded camera networks have e promising applications in surveillance, traffic analysis and wildlife monitoring. Unlike wired camera systems, these cameras have very limited energy, processing power and memory.

This thesis addresses a three-years research on embedded systems, embedded smart cameras, heterogeneous sensor network and their applications in the field of computer vision. The main focus is on the hardware and software architecture of embedded systems, battery powered embedded smart cameras, tackling the challenges of the limited resources such as computational power, memory and energy, and the issues concerning the containment of energy consumption and the implementation of light-weight computer-vision algorithms and methodologies that aim to increase the energy-efficiency of the embedded smart cameras. We will refer to all these challenging topics with the term **Green-Vision**. This thesis describes some cases of study concerning embedded systems and heterogeneous sensor networks. It is organized as follows:

- *Chapter two* provides a general overview on the embedded systems and on the embedded systems application fields. It also focuses on the development of en embedded system design and how the recent advances in semiconductor technology have increased the designers choices and on the new trend on embedded systems design flow.

- *Chapter three* is a case of study on the design and development of a low cost veiling luminance estimation system based on the use of a CMOS image sensor and a System on Chip (SoC), fully implemented on a FPGA. The soft processor is used to handle image acquisition and all computational tasks needed

to compute the veiling luminance value. The advantages of this single chip FPGA implementation include the reduction of the hardware requirements, power consumption, and system complexity and flexibility. The problem of the high dynamic range images have been addressed with multiple acquisitions at different exposure times. Vignetting, radial distortion and angular weighting, as required by veiling luminance definition, are handled through a single integer look-up table (LUT) access.

- *Chapter four* faces the comparison of the two most advanced algorithms for connected component labeling, highlighting how they perform on a soft core SoC architecture based on FPGA. In particular, a previously developed block-based connected components labeling algorithm, optimized with decision tables and decision trees has been tested. The results highlight the importance of caching and data cache sizes for high performance image processing tasks.

- *Chapter five* describes the embedded camera motes based on ARM processor employed in the case of study reported in chaper six on "Energy efficient software application on embedded smart cameras".

- *Chapter six* is a case of study that deals with the limited resources available on embedded systems, limited processing power, memory and energy. In this chapter, the methodologies that aim to increase the energy-efficiency and battery-life of an embedded smart camera have been introduced. These methodologies are based on hardware operations performed at the level of the image sensor. They are used to perform object detection and tracking on a battery powered embedded smart camera. The use of these techniques reduces the amount of data that is moved from the image sensor to the main memory at each frame. The better use of the memory resources results in a significant decrease in energy consumption and an increase in battery-life.

- *Chapter seven* is a case of study on the field of heterogeneous sensor network and sensor fusion; it has been presented a procedure for the mutual calibration of camera motes and Radio Frequency Identification Devices (RFIDs) for people localization and identification. This topic faces the problem of localizing and identifying objects with the final aim to perform intruder detection in wide open area.

Chapter 2

Embedded systems: a short introduction

2.1 Introduction

This chapter provides a general overview on the embedded systems and on the embedded systems application fields. It also focuses on the development of en embedded system design and how the recent advances in semiconductor technology have increased the designers choices and on the new trend on embedded systems design flow.

An embedded system is a microprocessor-based system that incorporates any electronic devices usually available in a computer. However it is built with the aim to control a range of functions and the end user cannot program it at the same way that is possible with a PC. In other words, the functionality of the system cannot be changed by the user and the software cannot be added or changed. Embedded systems are designed to perform one particular task although with choices and different options.

Embedded devices constitute most of the world production of microprocessors. There are over one hundred embedded devices for every PC in the world. In fact, embedded systems can be found in a wide array of products including:

- Military systems and aircrafts.
- Cars.
- Communications.
- Biomedical systems.

- Electronic instrumentation.
- Computer I/O devices.
- Home electronics.
- Robots.
- Industrial equipment.
- Office machines.
- Personal devices and smart toys.

Embedded systems designers often face challenging design goals. Many embedded systems have critical performance and power design constraints, and some devices may need to run on battery power for long periods of time. Real-time constraints occur in many applications and many devices have limited processing power and memory. Moreover, embedded systems must be reliable. Indeed, some of them may not be able to reboot, and cannot crash. In addition, consumer devices are very cost competitive and typically have a fast time to market on new products.

In terms of environmental impact, the use of embedded systems in the engine-management system have brought a reduction of the 90% of the automobile emissions in the last 20 years. To this aim, fuel-injected systems using multiple sensors in order to optimize performance and minimize emissions over a wide range of operating conditions are employed. These systems took the place of the carburetors, old open-loop fuel control systems.

Without the embedded systems as a control element, this type of performance improvement would have been impossible. A new luxury-class automobile might have more than 70 dedicated microprocessors, controlling tasks from the engine spark and transmission shift points to opening the window slightly when the door is being closed to avoid a pressure burst in the drivers ear.

Another example of the wide diffusion of the embedded systems is on the F-16 aircraft. It cannot be able to fly without on-board computers constantly making control surface adjustments to keep it in the air. The computer attempts to comply with the pilot request to change the planes flight profile to the extent that it can and still keep the plane in the air. A modern jetliner can have more than 200 on-board, dedicated microprocessors. Here is a short list of the differences between a PC and the typical embedded system:

- Embedded systems are dedicated to specific tasks, whereas PCs are generic computing platforms.
- Embedded systems are supported by a wide array of processors and processor architectures.

- Embedded systems are usually cost sensitive.
- Embedded systems have real-time[1] constraints.
- The implications of software failure is much more severe in embedded systems than in desktop systems.
- Embedded systems often have power constraints.
- Embedded systems often must operate under extreme environmental conditions.
- Embedded systems have far fewer system resources than desktop systems.
- Embedded systems often store all their object code in ROM.
- Embedded systems require specialized tools and methods to be efficiently designed.
- Embedded microprocessors often have dedicated debugging circuitry.

2.2 Digital Design Platforms

Until recently, designers have been limited to the choice of microprocessor versus microcontroller. Recent advances in semiconductor technology have increased the designers choices. Now, at least for mass-market products, it makes sense to consider a system-on-a-chip (SOC) implementation, either using a standard part or using a semi-custom design compiled from licensed intellectual property. Table 2.1 lists the major contemporary digital designs along with their relative merit.

2.2.1 Microprocessor-based Design

The microprocessor has changed digital design methodology like no other digital component. It started out in 1971 and still continues to be the digital controller of choice across several application areas. The microprocessor brought the concept of instruction set architecture (ISA), assembler and compiler. There are many real-time applications, with fast update rates that require programming the microprocessor in its native assembly language. This is usually done when the size of available memory is a constraint. Even though, most commercial microprocessors used today cater to data-centric applications, there are microprocessor cores embedded in microcontrollers for real-time control applications.

[1] Real-time events are external (to the embedded system) events that must be dealt with when they occur (in real time). If an embedded system is using an operating system at all, it is most likely using a real-time operating system (RTOS), rather than Windows 9X, Windows NT, Windows 2000, Unix, Solaris, or HP-UX.

Digital design platforms	Main feature
Microprocessors	Reconfigurable using software. Good for computations
Microcontrollers, digital signal controllers	Combination of peripherals and CPU
Application specific standard product (ASSP)	A specialized peripheral with the ability to communicate with a host processor
Field programmable gate array (FPGA)	Ability to combine the strengths of processor, controller and ASSP

Table 2.1: Digital design platforms

2.2.2 Single-chip Computer/Microcontroller-based Design

The microcontroller represents the next generation of controllers for embedded systems. It allows creating systems with fewer numbers of components by incorporating peripherals that were earlier externally interfaced with the general purpose processor. A block diagram of a typical single-chip controller is shown in Figure 2.1. Like the microprocessor, tasks in a microcontroller design environment are divided as per the update rates required. For tasks requiring low update rates, coding is accomplished using a software programming language such as C. Tasks that need to have high deterministic update rates are coded using the native assembly language for a particular microcontroller. It is difficult to port routines written in assembly language as they are tied to a specific CPU ISA. The other constraint with a microcontroller-based system is the fixed number of available peripherals. Though microcontroller vendors offer a wide range of devices with different numbers and types of peripherals, it is not always possible to find one that matches the application requirements perfectly.

2.2.3 Application Specific Standard Products (ASSPs)

An ASSP is a configurable logic component for a specific application. The functionality of an ASSP is tweaked by specifying its control word. ASSPs are made in volumes and cater to the generic requirements of the application. Most of the times, ASSP-based designs are used on a PCB. For example, in a robot control application, an ASSP can be used for controlling the motor for each axis of the robot. Based on the type of motor and control strategy used, a corresponding ASSP is chosen. Two examples of ASSPs for motor control include LM629 from National Semiconductor for control of a brushed DC motor and AC three-phase motor control (see Figure 2.2(a) and 2.2(b)) from Freescale Technology. Configurable ASSPs provide address, data and control bus connectivity for interfacing with the host processor.

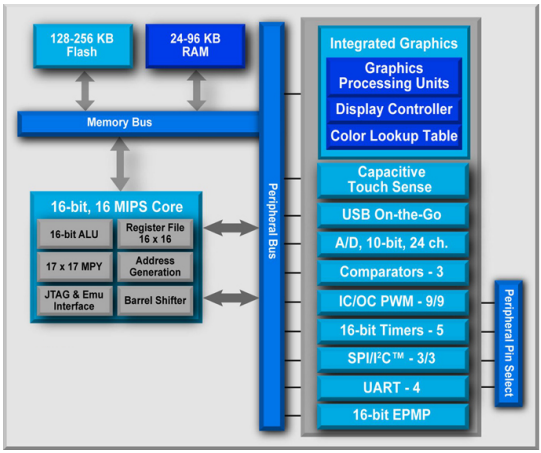


Figure 2.1: Block diagram of a typical single-chip controller

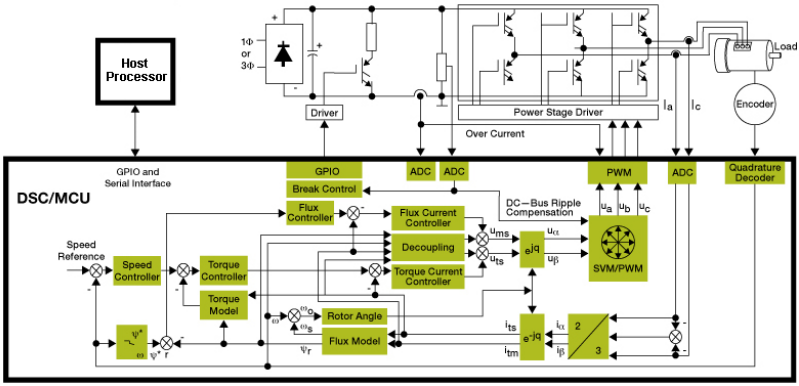
2.2.4 Design Using FPGA

The present-day FPGA provides a platform that supports both processor and custom logic requirements. The microcontrollers currently have an edge over the FPGA in terms of power and cost. But FPGAs are catching up by offering portability of code across various FPGA vendors, libraries of re-usable code and availability of low-cost programming tools. Programmable devices that were traditionally low gate count devices are now in a position to support large parts of digital system logic. The digital designer today has a viable option of using only the FPGA device as the embedded system controller. The availability of high density, low-cost FPGA devices have given digital designers lots of flexibility to design custom digital architectures using FPGA and HDLs. FPGA devices have evolved from their glue logic predecessor to a device that now contains a large variety of built-in digital components (memory, multipliers, transceivers and many more). FPGA device density has risen over the years and at the same time its cost has made it economically viable for use in several applications. Contemporary FPGAs contain thousands of look up tables (LUTs) and flip flops (FFs) for implementing complex digital logic.

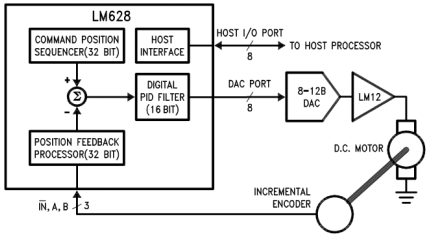
Contemporary FPGAs offer:

- Reconfigurability: Field programmable devices can be reconfigured at any time. Designers can integrate modifications or do complete personality changes.

- Software-defined design: The hardware is defined by software-like languages (HDL). Designers can develop, fully simulate and test a circuit before "running" it on a field programmable device.



(a)



(b)

Figure 2.2: ASSP chip for control of a three-phase AC Induction Motor (a) and ASSP chip for control of a DC motor.

- High speed: Because an FPGA is a hardware implementation running with fast clock rates, designers can achieve very high speeds. Coupled with parallelism, FPGA implementation can outperform processor-based systems.

- IP (Intellectual Property) protection and re-use: Once compiled and downloaded to a FPGA, hardware implementation is difficult to reverse engineer. A tested hardware design can be re-used multiple times by instantiating.

FPGA-based systems are gaining acceptance because these systems integrate digital logic design, processors and communication interface on a single chip. The front end design flow of a FPGA is very similar to that of a custom logic design. Almost all FPGA vendors offer a suite of software tools that allows a designer to simulate, synthesize, place and route and program the FPGA. Table 2.2 shows the different design tools offered by two leading vendors. Once a designer feels comfortable in a particular design suite, it is easy to migrate to another vendors design

tools because they work in a similar fashion.

Functionality	XILINX	ALTERA
Design synthesis, mapping, place and route	Integrated Software Environment (ISE)	Quartus II
FPGA embedded processor design tool	Embedded Design Kit (EDK)	System on Programmable Chip (SoPC) builder
Custom peripheral Support	Yes	Yes
On-Chip signal logic Analyzer	ChipScope Pro	SignalTap
MATLAB cosimulation and IP cores Library	System Generator	DSP Builder

Table 2.2: Design tools provided by two leading FPGA Vendors

Software-based design flows are suited for applications which are data centric while hardware design flow is suited for fast real-time applications. Table 2.3 provides a transition path for migrating from microprocessor/controller to FPGA-based design. The FPGA design process consists of design entry, which is accomplished by using either schematic or HDL. Following the design phase, digital logic is synthesized, mapped and placed on a FPGA.

Existing microprocessor/microcontroller code	Field programmable device
Target independent C Code	Embedded processor within the FPGAdevice
Target dependent assembly constructs for routines requiring fast update rates	Target independent HDL-based coding for routines requiring very fast update rates

Table 2.3: Migration design-flow from a microcontroller-based system to a FPGA system

2.2.5 Development of an Embedded Systems Design

The primary focus of the software and engineering design effort occurs in the development phase which will be described now in more detail. Early on, the designers must select a processor and an operating system. The selection of a processor for an embedded device involves many considerations such as price, performance, power consumption, and software support.

Given that most embedded devices now require an operating system, the availability of an appropriate OS, device drivers, application programs, and the required compilers and software development tools are a major factor to consider in any new design, perhaps even more important than the processor choice. Manufacturers provide data manuals with their processors and typically provide designers with complete reference board designs that can be used as a starting point when developing a new computer design using their processor.

After an embedded systems hardware designer has selected a processor and its associated memory devices, the next step is to add the I/O hardware devices and the associated bus structures nedeed to interconect to the required to the processor. Since processors are already designed by the manufacturer and memory interfaces are largely dictated by the processor, a significant portion of the hardware design effort in embedded devices is dedicated to selecting and connecting the hardware needed for the various I/O devices required in the new design.

After carefully entering a schematic for the design, a printed circuit board (PCB) is designed for the embedded device using a PCB computer-aided design (CAD) tool. This tool imports the pin connection information from the schematic and uses it to design and verify the copper traces used to connect the integrated circuits (ICs) on the PCB. Several PCBs are manufactured, stuffed with parts, and then used to run extensive software tests on the new design. Any hardware design errors detected during testing will require changes to the schematic, modification of the PCB design, and a new round of PCB fabrication and testing that will add to the development time.

The software development tools are typically provided with the OS. Since the OS is written in C/C++ a compiler, linker, debugger, and binary image tools are needed to generate a new OS. These same tools are typically used for application development. Software development occurs in parallel with hardware development to reduce the total product development time. This has become more important given the ever shortening product life cycles of current embedded devices. Emulation tools and embedded computer boards with similar hardware running the same OS can be used to develop and test software before new hardware platform is available. Since the majority of the code is written in C/C++, a large portion of the software can even be developed and tested on a different processor or emulator. Code is then recompiled to target a new processor for the final round of development and testing ones the new hardware becomes available.

2.2.6 FPGA Devices

FPGA devices have grown in density from a few thousand gates in the 1980s to approximately 2 million logic cells, which is equivalent to 20 million ASIC gates in 2011[2]. There are proven advantages in choosing a FPGA-based implementation of digital logic over a fixed custom implementation of digital logic. The advantages

[2] Virtex-7 2000T FPGAs, www.xilinx.com

include cost economies when a product is produced in low volume, in-system reprogrammability and a shorter design cycle from concept to silicon. Most of the contemporary FPGAs from various vendors have common on-chip resources. These device hardware primitives are comparable to assembly language constructs of a general purpose processor, invoked by the compiler. Though one may rarely use programming at the primitive device level using HDLs, it is good to know the underlying hardware, used by the synthesis tool for realising digital logic.

Architecture of a FPGA

The contemporary FPGA is changing very fast. It is following Moore's law in speed and density and also incorporating lots of functionality along the way. Though FPGAs from different leading vendors such as Xilinx, Altera, Actel, and Lattice differ in some aspects, they all share some common architectural attributes. These architectural attributes can be thought of as device specific primitives. For illustration, a sample FPGA device, Xilinx SPARTAN-3E is shown in in Figure2.3. Following are the typical features of a contemporary FPGA:

- Logic cell resources, consisting of LUT and FFs
- Hard intellectual property(IP), comprising of dedicated multipliers and embedded memory
- Clock distribution resources, digital clock manager (DCM) providing frequency synthesis and phase shift
- I/O features number of user available I/Os and I/O standards
- Hardware immersed and software configurable processors, along with logic fabric in a single FPGA device.

FPGA-based Embedded Processor

With rising gate densities of FPGA devices, many FPGA vendors now offer a processor that either exists in silicon as a hard IP or can be incorporated within the programmable device as a soft IP. The purpose of having a processor co-exist with conventional digital logic components is to provide flexilibility of combining software and hardware based control in one chip. Many algorithms that are difficult to code in HDL and have update time requirements in milliseconds can use the processor inside the FPGA. A whole suite of tools, consisting of compilers and assemblers help the designer code in C or C++.

The ability to support processor logic has brought a new dimension to the use of FPGA devices. It has provided designers the freedom to partition their designs either for single-threaded software flow or to use concurrent digital logic. A quick search

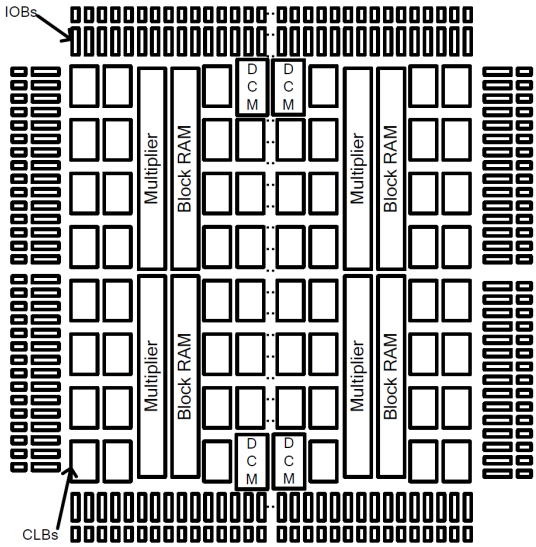


Figure 2.3: Architecture of a Spartan 3E FPGA

on the internet shows that several 8-bit and 32-bit proprietary processors are offered by leading FPGA vendors along with established processors. The motivation for using the time tested, established processor is to shorten the learning curve of designers and build confidence in their use. Almost all major vendors of field programmable devices provide processors, for use with their respective devices. Table 2.4 shows an incomplete list of vendors and the processors they offer.

Soft Processors

Soft processors exist as synthesized netlists incorporated in the FPGA using logic block resources of a particular FPGA. FPGA vendors offer soft processors catering to 8-bit and 32-bit applications. The 8-bit processor occupies a small footprint on the FPGA device and it uses the FPGA embedded memory for program and data memory storage. Figures 2.4(a) and 2.4(b) show the block diagram of PicoBlaze and its memory interface. Picoblaze is an 8-bit soft controller from Xilinx. The PicoBlaze controller consists of an 8-bit input and an 8-bit output port. It also supports interrupt. The embedded block RAM of the FPGA serves as a location for program and data memory for the PicoBlaze. The PicoBlaze assembler takes the program file and creates the coefficient (.coe) file, loaded in the embedded memory of the FPGA. Among the 32-bit soft processors, two of the leading 32-bit proprietary soft processors are NIOS from Altera and MicroBlaze from Xilinx. These processors use a portion of the FPGA resources. The remaining part of the FPGA can be used for

Processor name	Type/Bits	Interface bus	FPGA vendor
MicroBlaze	Soft/32	IBM Coreconnect	Xilinx
NIOS	Soft/32	Avalon	Altera
LatticeMico32	Soft/32	Wishbone	Lattice
CoreMP7	Soft/32	APB	Actel
ARM Cortex-M1	Soft/32	AHB	Vendor independent
LatticeMico8	Soft/8	Input/Output	ports Lattice
Core8051	Soft/8	Nil	Actel
Core8051s	Soft/8	APB	Actel
PicoBlazeTM	Soft/8	Input/Output ports	Xilinx
PowerPC	Hard/32	IBM Coreconnect	Xilinx
AVR	Hard/8	Input/Output ports	Atmel

Table 2.4: Incomplete list of contemporary FPGA-based processors

incorporating other digital logic.

The MicroBlaze 32-bit soft processor, shown in Figure 2.5(a) is a reduced instruction set computer (RISC) based engine with a 32 * 32-bit LUT RAM-based register file with separate instructions for data and memory access. It supports both on-chip block RAM and external memory for program/data memory. All peripherals are implemented on the FPGA fabric and interface to the MicroBlaze using the onchip peripheral bus (OPB) or processor local bus (PLB) as shown in figure 2.5(b). The MicroBlaze processor options include instantiation of additional hardware to implement IEEE 754 single precision floating point standards. With this option included, it can support floating point addition, subtraction, multiplication, division and comparison. Soft processors listed in Table 2.4 can be customized by adding a barrel shifter or modifying the size of the data and instruction cache. Additional processors can also be added to provide a multi-processing option. Based on the results of software profiling, certain resource intensive software algorithms can be moved to the hardware fabric as coprocessors or as custom peripherals.

2.3 New trend on embedded systems design flow

Over the last three decades, as the complexity of circuits being designed has increased by several orders of magnitude, the front end design flows have seen a paradigm shift from Capture and Simulate methodology to Describe and Synthesize methodology to finally Specify, Explore and Refine methodology [1].

The Capture and Simulate methodology was based on design entry in the form of schematic diagrams and simulation dominates the flow. To cope up with increasing size of the circuits, there was a shift from structural design descriptions in the form of schematics to behavioral descriptions using Hardware Description Languages such

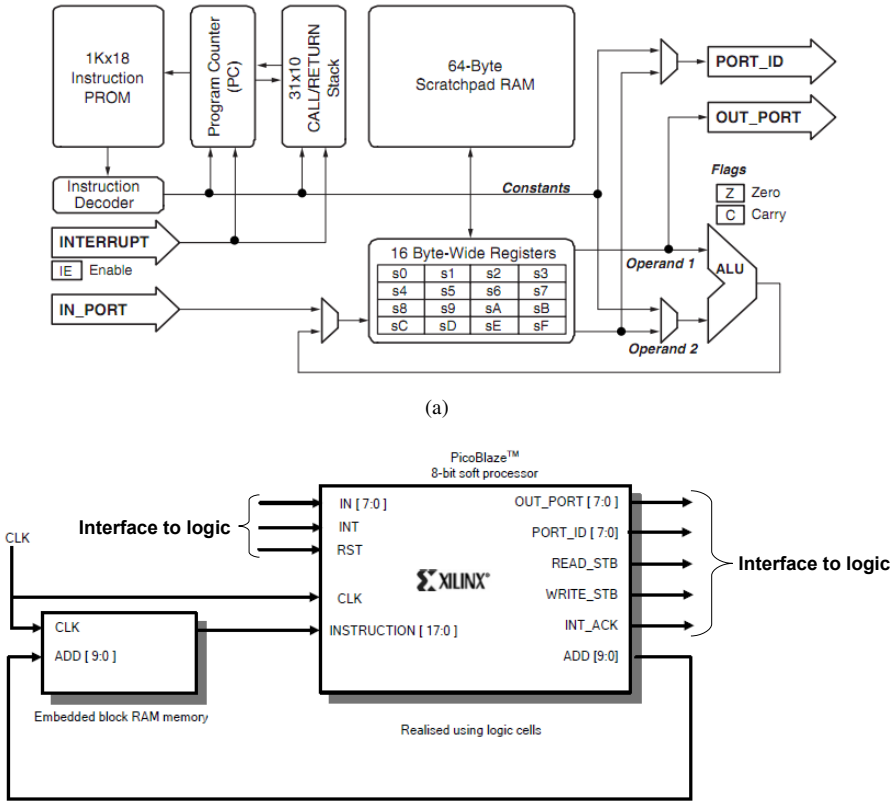
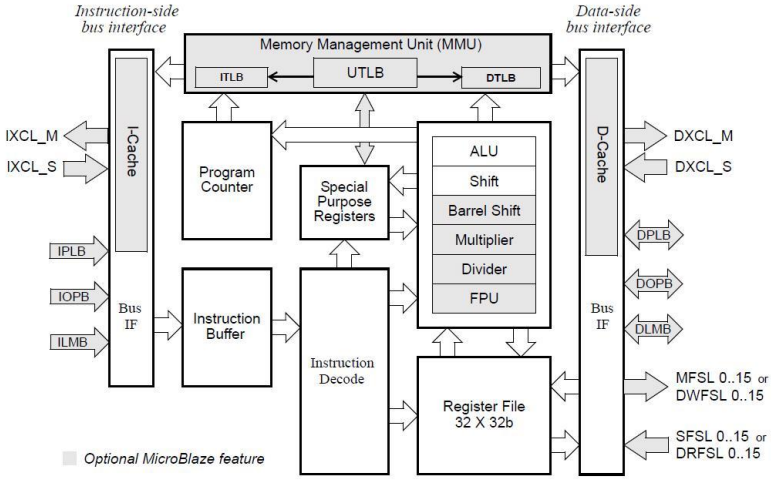


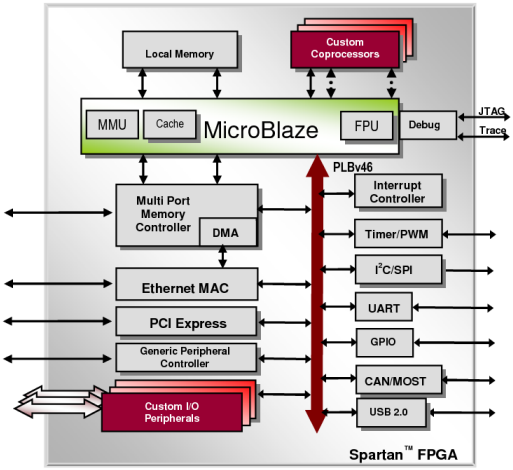
Figure 2.4: Picoblaze 8-bit soft controller architecture (a) and Picoblaze memory interface

as VHDL and Verilog. The most common abstraction level of design entry in Describe and Synthesize methodology has been register transfer level (RTL). Synthesis tools are used to bridge the gap between the design abstraction levels at entry and exit points.

The design of an IC chip is no longer a matter of simply designing a piece of hardware. With the continued scaling of CMOS technologies, more and more complex systems are being put on single chips, invariably including software components. This requires Specify, Explore and Refine methodology, in which exploration of a vast design space is a major activity. The designer starts with a loose specification of the system and refines it as the design decisions are made. Languages such as SystemC, System Verilog, SpecC etc. are used to specify system level behavior at a



(a)



(b)

Figure 2.5: Microblaze block diagram (a) and Microblaze processor system architecture

comparatively higher level of abstraction.

Earlier systems on chip (SoCs) were built around a single embedded processor. Increasing demand for performance is driving the SoC architecture towards integra-

tion of more and more processors on a single chip, giving rise to Multi Processor SOCs or MPSoCs. A high end MPSoC today includes, apart from multiple processors, multiple memory modules, high performance interconnection network, custom hardware blocks to perform some critical functions, and a variety of peripheral controllers and interfaces. The processors could be standard off the shelf modules or custom designed for the given application. An MPSoC requires a design flow that is a combination of several sub-flows, each taking care of a different requirement.

SoC Platforms and IPs

As design entry moves towards higher levels of abstraction, the basic building blocks evolve towards more complex components such as processors, on-chip memory, and custom designed cores. Design reuse oriented paradigms drive the integration of such thirdparty intellectual property (IP) cores into Systems on chip; such cores would be pre-verified and lead to a tremendous productivity boost and reduced design times. The designer may still include custom-designed and synthesized functional blocks as a differentiating factor. However, the overall design methodology needs to be aware of the requirement that much of the SoC will consist of instantiations of reused blocks. The hardware and software architecture and interfaces of the SoC need to be appropriately designed in order to enable a clean composition of design elements that may have their origin in disparate vendors and design groups.

SoC Architecture and Task Mapping/Scheduling

The general principle of SoC design thrust has been the need for programmability and customization. This makes the processor core an important reusable block. For the processor class of cores, the instruction set forms a natural interface to the external world. The architectural instance could be that of a general purpose processor, a DSP processor, or a microcontroller core. The component chosen here will depend on the instruction set implemented, the associated software development environment, the communication interfaces available, etc. Models of the processor at different levels of abstraction (instruction set level, cycle accurate level, etc.) are required for performing system simulations. In addition to the processor core, the SoC architecture platform may consist of several types of memory cores including instruction and data caches, SRAM cores serving as scratch pad memory, and FIFO structures. The personalization of such modules is usually performed through a memory-compiler tool that can generate the layout of the required configuration. Programmable components present in modern SoC fabrics may also include some reconfigurable logic of the type present in FPGAs (Figure 1). This allows for minor updates to the hardware functionality post-deployment as well as for future versions of the SoC. Finally, the architecture platform may accommodate custom hardware designed with a traditional HDL synthesis and physical design flow. Appropriate interfaces need to be implemented for such hardware to communicate with other IP

blocks. General-purpose platforms need to be tailored for specific application scenarios for maximum impact. Application modeling, efficient representation of the data and communication requirements of system tasks, and mapping of these tasks into PEs are key steps in SoC architecture exploration. The task allocation problem can be viewed as follows: given an SoC architecture and a graph capturing the computation and communication requirements of the application, assign tasks to PEs so that a given objective function (such as overall throughput) is optimized subject to constraints imposed on the system (such as power) [2]. Similarly, the task scheduling problem can be formulated as the problem of assigning start times for the tasks executing on the selected PEs as well as for the initiation of data transfer on the communication network.

Software Development

To cut on the development time, software development needs to be carried out concurrently while the hardware architecture is being explored. This can be done using a simulated model of the architecture. Different simulation models are required at different stages. Requirements of software developers are different from those of system architects and hardware implementers. Hardware designers need to simulate precise timings in the design to verify its behavior. System architects also need timing information, though not very precise, for design space exploration and selection of suitable architecture. In contrast, software developers need only functionality of the hardware to be simulated, ignoring the timing details. However, they require a faster speed of simulation since the software running over simulated hardware adds a layer of interpretation. Since timing details are not relevant at this stage, simulation speed can be increased by excluding these details from the simulation models, thus raising the level of abstraction. This means that while functionality of the instructions is modeled, the micro-architectural and cycle level details are omitted. Based on this principle, Virtual Prototyping Environments (VPEs) are created to facilitate software development concurrent with hardware development. VPEs include high level executable models of various subsystems such as processors, memories, peripheral controllers, interconnecting buses, networks, etc. The processors are modeled using instruction set simulators (ISS), enclosed in appropriate wrappers to communicate with other subsystems.

Chapter 3

Computer vision application on a FPGA based SoC

3.1 Introduction

This chapter describes the design and development of a Veiling Luminance estimation system based on the use of a CMOS image sensor, fully implemented on FPGA.

Veiling Luminance is the veiling effect produced by bright sources or areas in the visual field that results in decreased visual performance and visibility. This phenomenon is particularly important for a vehicle's driver approaching to the entrance of a tunnel, and typically produces a reduction in driver's ability to perceive the presence of an obstacle or a slowdown caused by the traffic. The European Standard CIE 88/2004 (Italian UNI 11095 [2]) defines the threshold luminance in terms of the minimum amount of luminance necessary to let a driver to see an obstacle within its safety stop distance. This value can be computed as the veiling luminance, defined as the luminance that creates a disturbance on the driver visual within its safety stop distance.

The veiling luminance estimation system proposed in this chapter is composed of the CMOS Image sensor, FPGA, DDR SDRAM, USB controller and SPI (Serial Peripheral Interface) Flash. The FPGA is used to build a system-on-chip integrating a soft processor (Xilinx MicroBlaze) and all the hardware blocks needed to handle the external peripherals and memory. The soft processor is used to handle image acquisition and all computational tasks need to compute the Veiling Luminance value. The advantages of this single chip FPGA implementation include the reduction of the hardware rquirements, power consumption, and system complexity. The problem of the high dynamic range images have been addressed with multiple acquisitions at

different exposure times. Vignetting, radial distortion and angular weighting, as required by veiling luminance definition, are handled by a single integer look-up table (LUT) access. Results are compared with a state of the art certified instrument.

3.2 Optics: Basic Concepts

3.2.1 General definitions

Radiometry

Radiometry is the science of measuring light in any portion of the electromagnetic spectrum. It describes the measurement of optical radiation from a physical point of view. In practice, the term is usually limited to the measurement of infrared, visible, and ultraviolet light using optical instruments. Typical units are Watt and Joule.

Photometry

Photometry is the science of measuring visible light in units that are weighted according to the sensitivity of the human eye. It is a quantitative science based on a statistical model of the human visual response to light, that is the perception of light, under carefully controlled conditions. Typical photometric units are Lumens, Lux, Candelas.

The human visual system is a complex and highly nonlinear detector of electromagnetic radiation with wavelengths ranging from 380 to 770 nanometers (nm). The sensitivity of the human eye to light varies with the wavelength. In photometry, it is not measured watts of radiant energy. Rather, it is attempted to measure the subjective impression produced by stimulating the human eye-brain visual system with radiant energy. Figure 3.1 shows the photopic luminous efficiency of the human visual system as a function of wavelength. It provides a weighting function that can be used to convert radiometric into photometric measurements.

3.2.2 Radiometric Units

Radiant Energy $Q_e, [J], Joule$

Radiant energy is the energy of electromagnetic waves, its SI (abbreviated SI from French: Système international d'unités) unit is the joule. It expresses the radiation emitted by a source into the surrounding environment.

Radiant flux $\phi_e, [W], Watt$

Radiant flux (or radiant power) is the radiant energy transferred per unit time. It measures the total power of electromagnetic radiation (including infrared, ultraviolet,

and visible light). The power may be the total emitted from a source, or the total hitting on a particular surface.

$$\phi_e = \frac{dQ_e}{dt}$$

Spectral Radiant Power $\phi_{e\lambda}, [W/m], Watt\, per\, meter$

It is the radiant power per unit wavelength interval at wavelength λ

Radiant Intensity $I_e, [W/sr], Watt\, per\, steradian$

Radiant intensity is defined as the radiant flux (or radiant power) proceeding from a point source per unit solid angle in the direction considered,

$$I_e = \partial\phi_e/\partial\omega$$

Solid angles are measured in units of steradian, sr. A unit solid angle, or 1 sr, is defined as the solid angle subtended at the center of a sphere with a radius R = 1 meter by an area of one square meter on its surface.

Radiance $L_e, [Wm^{-2}sr^{-1}], Watt\, per\, square\, meter\, and\, steradian$

Radiance is defined as the radiant power that leaves a surface per unit solid angle and unit projected area of that surface,

$$L_e = (\partial^2\phi_e)/(\partial A\, \partial\omega \cos\theta) = \partial I_e/(\partial A \cos\theta)$$

The term $\cos\theta$, in which θ is the angle between the normal to ∂A and the direction of view, transforms area ∂A to projected area. If the amount of flux $\partial^2\phi_e$ that leaves an element of area ∂A in solid angle $\partial\omega$ in all directions is proportional to $\cos\theta$, the surface is said to obey Lambert's law and is often referred to as a Lambertian, or perfectly diffuse radiating (or reflecting or transmitting), surface. Because the projected area is also proportional to $\cos\theta$, the radiance of such a perfectly diffuse surface is independent of the direction of view.

Radiance can be applied to both an extended source and a point source. In practice, the finite extent of a source is sometimes neglected if its diameter is less than about $1/20$ of the distance to the irradiated surface.

3.2.3 Photometric Units

Luminous Flux $\phi_V, [lm], lumen$

Luminous flux, also called luminous power, is the photometric counterpart of the radiant power. The unit of luminous flux is the lumen, lm. One lumen is defined as

the luminous flux emitted (within a unit solid angle) by a point source of luminous intensity (defined below) of one candela. In other words, a point source that radiates uniformly in all directions with a luminous intensity of 1 candela emits a total of 4π lm.

For example, at 555 nm, 1 lm is equivalent to 0.00147 (1/680) W, or 1 W at 555 nm is equal to 680 lm. In order to use this conversion anywhere else in the visible spectrum, the proper luminous efficiency must be included, see Figure 3.1. For example, the luminous efficiency at 600 nm is 0.63, thus, 1 W of monochromatic light at that wavelength equals $0.63 \cdot 680 = 428$ lm, or 1 lm is equal to $0.00147/0.63 = 0.00233$ W. If the source is not monochromatic, integration is needed.

Luminous Energy $Q_v, [lms],\ Lumen\, second$

In analogy with radiant energy that is the product of radiant power and time, luminous energy Q_v is the product of luminous power and time. This unit is the talbot (1 talbot = 1 lumen-second). One lumen, hence, is the luminous power of 1 talbot per second.

Luminous Intensity $I_v, [cd = lm/sr],\ Luminous\, flux\, per\, unit\, solid\, angle$

Luminous intensity is the photometric equivalent of the radiant intensity, it is a measure of the wavelength-weighted power proceeding from a point source per unit solid angle in the direction considered.

$$I_v = \partial\phi_v/\partial\omega$$

The candela is the unit of luminous intensity, it is equal to 1 lumen of luminous power per unit solid angle; thus, 1 candela = 1 lumen/steradian, lm/sr.

Luminance $L_v, [cd/m^2],\ Candela\, per\, square\, meter$

Luminance is the photometric equivalent to radiance, it is a measure of the luminous intensity per unit area of light travelling in a given direction. It describes the amount of light that passes through or is emitted from a particular area, and falls within a given solid angle.

$$L_v = (\partial^2\phi_v)/(\partial A\, \partial\omega \cos\theta) = \partial I_v/(\partial A \cos\theta)$$

Luminance is often used to characterize emission or reflection from flat, diffuse surfaces. The luminance indicates how much luminous power will be detected by an eye looking at the surface from a particular angle of view. Luminance is thus an indicator of how bright the surface will appear. In this case, the solid angle of interest is the solid angle subtended by the eye's pupil

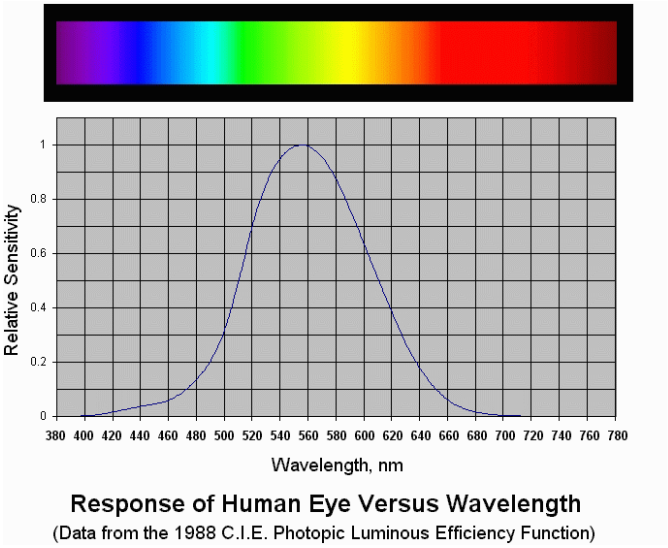


Figure 3.1: Photopic luminous efficiency of the human visual system. This curve provides a weighting function that can be used to convert radiometric into photometric measurements

Illuminance $E_v, [lux = lm/m^2],\ Lumen\, per\, square\, meter$

Illuminance is the luminous flux (or luminous power) per unit area incident on a surface.

Spectral Illuminance $E_\lambda, [lux/nm],\ Lux\, per\, nanometer$

Spectral illuminance is the luminous flux (or luminous power) per unit area and per unit wavelength interval incident on a surface.

3.3 Veiling Luminance estimation system: problem definition

Veiling Luminance is the veiling effect produced by bright sources or areas in the visual field that results in decreased visual performance and visibility. This phenomenon is particularly important for a vehicle's driver approaching to the entrance of a tunnel, and typically produces a reduction in driver's ability to perceive the presence of an obstacle or a slowdown caused by the traffic, especially in the first part of the tunnel (called *threshold zone*). The correct estimation of the veiling luminance

can be exploited to design a measuring system combined with a power controller, in order to directly control the tunnel lighting, adapting automatically to the environmental conditions outside the tunnel by increasing or decreasing accordingly the lighting intensity inside a tunnel. This plays a key role in traffic safety: a correct adaptation can avoid drivers to experience the hazardous "black hole" effect, being a very precious addition to its safety. The European Standard CIE 88/2004 (Italian UNI 11095 [2]) defines the threshold luminance in terms of the minimum amount of luminance necessary to let a driver to see an obstacle within its safety stop distance. This value can be computed as the veiling luminance, defined as the luminance that creates a disturbance on the driver visual within its safety stop distance.

3.4 The veiling luminance

The veiling luminance is composed by three different contributions:

$$L_v = L_{seq} + L_{atm} + L_{winds} \ . \tag{3.1}$$

where L_{seq} is the equivalent veiling luminance, L_{atm} is the atmospheric luminance and L_{winds} is the windscreen luminance.

The windscreen luminance L_{winds} can be directly estimated from the L_{seq}:

$$L_{wind} = 0.4 \cdot L_{seq} \tag{3.2}$$

whereas the atmospheric veiling luminance L_{atm} is usually estimated from tabulated data or measured before the tunnel launching using a bulky procedure as defined by the UNI 11095 standard [2].

The key quantity to compute for the correct L_v estimation is the equivalent veiling luminance L_{seq}.

3.4.1 Approaches to equivalent veiling luminance computation

The equivalent veiling luminance L_{seq} is directly caused by bright glare sources in the periphery of the point of view of the driver that reduces contrast visibility since light scattered in the lens obscures the fovea.

By law, two alternative specifications are available for the equivalent veiling luminance computation (or equivalent straylight luminance), L_{Seq}.

- the summation of the luminance contributions of 108 sector of a polar diagram, centered in the middle of the tunnel at 1.5 meters high

- the summation of the luminance contributions of every light source, weighted by the angle with the driver's observation direction

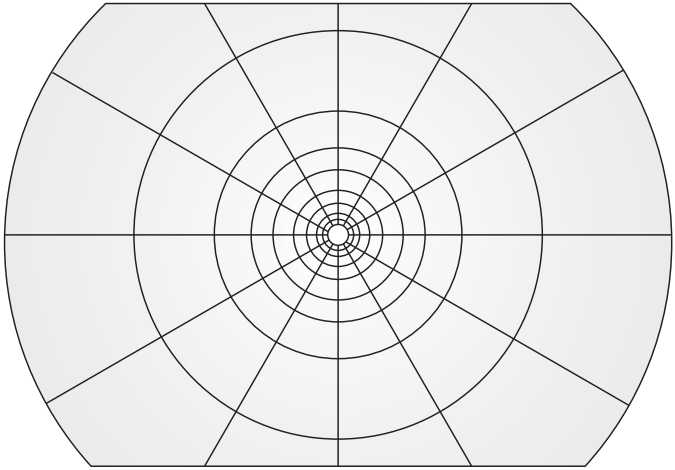


Figure 3.2: Polar diagram similar to the one provided by UNI 11095 standard for the L_{seq} estimation process. Such as the UNI 11095 standard polar diagram, it is composed of 9 rings each one divided in 12 sectors. The diagram is pruned in order to take into account the stop effect provided by the vehicle windscreen

The first approach obtains L_{Seq} using a graph (similar to the one depicted in Figure 3.2) superimposed and centered to the tunnel scene, at a distance equal to the stopping distance. In each area the amount of luminance is considered equal to the straylight in the corresponding center. To compute the resulting value, the formulation proposed is the following:

$$L_{Seq} = 0.51 \cdot 10^{-3} \cdot \sum_{i=1}^{9} \sum_{j=1}^{12} L_{ij} \tag{3.3}$$

where L_{ij} is the mean luminance emitted by the i-th ring at the j-th sector of the polar diagram.

Following the indications provided by UNI 11095, the luminances L_{ij} of these regions can be computed manually using some printouts: the tabulated values of luminance for various typical surfaces are provided by empirical studies (CIE 88-1990 [3]), and are considered to represent with a sufficient frequency of time during the year for most common tunnel conditions.

The second approach considers more specifically every light source in the scene. The formulation proposed is the following:

$$L_{Seq} = 9.2 \cdot \sum_{i=1}^{N} \frac{E_i}{\theta_i^2} \tag{3.4}$$

where E_i is the illuminance (i.e. light incident on a surface) produced on the human eye weighted by the angle θ_i between the direction of origin of the i-th light source and the observation direction of the driver.

In order to capture the photometric measures necessary for the second formulation, some specific tools should be required, for example a lux meter. The lux meter gathers illuminance information about its entire solid angle, it is expensive and it is not so stable especially in a real world scenario. Instead of the lux meter, a low cost and standard CMOS-based vision system can be used, which capture in every acquired pixel the illuminance information in the scene in the form of brightness (grayscale value of the acquired image). For this reason, a CMOS camera can be considered a "low cost" lux meter.

Given the current properties of the camera (such as focal length, exposure time, ...) and supposing some kind of linear relation between the grayscale value of the acquired image and the real illuminance of the scene, it is possible to come out with an automatic procedure to compute the equivalent veiling luminance in real time.

3.4.2 Illuminance Estimation

In the acquisition process, the illuminance information $E(y, z)$ is captured by the CMOS sensor and converted into electrical signals at discrete positions, representing the pixels of the acquired image. In this approximation, every pixel of the image is considered a light source whose gray value represents the amount of illuminance of the relative region of the real world. This approximation should be balanced by the fact that a CMOS camera is not an ideal lux meter, and that the illuminance we estimate from the grayscale value is not obtained using an ideal reflectance reference target.

In literature, the relation between $E(y, z)$ and the gray level $g(i, j)$ in the image plane is well know, as presented in [4]:

$$g = t \int_{y-\frac{1}{2}\Delta y}^{y+\frac{1}{2}\Delta y} \int_{z-\frac{1}{2}\Delta z}^{z+\frac{1}{2}\Delta z} \int_{\lambda_1}^{\lambda_2} R_\lambda(\lambda) \cdot E_\lambda(y, z)\, \mathrm{d}\lambda \mathrm{d}z \mathrm{d}y \tag{3.5}$$

where, Δy and Δz are the pixel horizontal and vertical dimensions respectively, i and j are the indexes of the considered pixel in the image plane, $[\lambda_1, \lambda_2]$ is wavelength range of the image illuminance, E_λ is the image spectral illuminance, $R_\lambda(\lambda)$ is the digital sensor spectral responsivity and t is the exposure time. Equation 3.5 has been obtained considering an array composed of identical photosensors, neglecting any noise and supposing that the image illuminance $E(y, z)$ at a given λ did not change during the exposure time. Simplifying the formulation provided in Equation 3.5, we can highlight a linear relationship that links the grayscale value g to the illuminance E and the exposure time t, with a negligible error and an (unknown) multiplier parameter R depending on the sensor spectral responsivity and pixel size.

$$g = R \cdot E \cdot t \tag{3.6}$$

3.4.3 Overview of the algorithm

The first step of the algorithm is a necessary preprocessing aimed at the correction of the distortions introduced by the vision system. As it will be detailed in a later Section, a uniform region is acquired using the CMOS camera, then the obtained image is used to estimate and correct radial distortions and vignetting.

The second step is the determination of the parameters of the linear relation between the grayscale value of the image and the illuminance of the acquired scene, with a given exposure time. As shown in the Figures 3.3 and 3.4, the light source is varied in intensity, for each condition an average reading of the illuminance values of a lux meter (positioned near the camera lens) is plotted with the relative average grayscale value of the acquired (and corrected) image, and the same procedure is adopted at different exposure times. The experimental results confirm a linear relationship between these two dimensions: for each useful exposure time provided by the camera, the relative parameters are saved.

Finally the third step computes the final luminance value. Since the aforementioned relation holds for different exposure times, Equation 3.6 can be used rewritten as illuminance in function of grayscale value:

$$E = R \cdot \frac{g'}{t} \tag{3.7}$$

where R is a constant parameter, g' is the grayscale value corrected of the distortions and t is the exposure time. Following the formulation proposed by the UNI 11095 standard [2], all the illuminance values computed considering pixels as light sources, weighted by their angle with the driver's point of view can be summed up, obtaining an estimation of the equivalent veiling luminance. In particular, since the device must adapt to different light conditions, a multi-esposures (HDR) procedure is employed: for each pixel in the scene, the illuminance value is computed at the minimum exposure time before saturation (either towards the dark or the bright).

3.5 Distortions correction

In computer vision, the basic camera model for image formation is the pin-hole camera model. It assumes that each image point is generated as a direct projection of a world point through the optical center. In practice, the model of real cameras is not so ideal, and many factors contribute to introduce different types of distortion that can slightly change the object's appearance. This problem can be sometimes neglected, but not in this case since the estimation of the L_{seq} depends directly on the grayscale

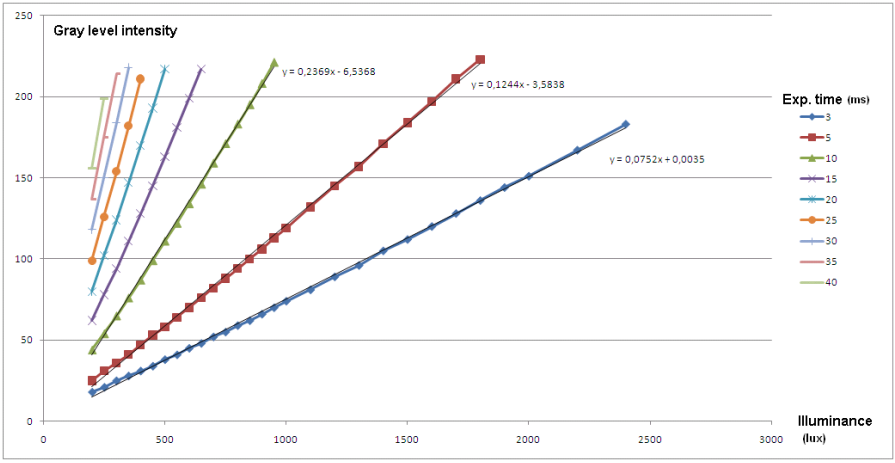


Figure 3.3: Linear relationship between grayscale value of the image and the illuminance of the acquired scene for different exposure time values.

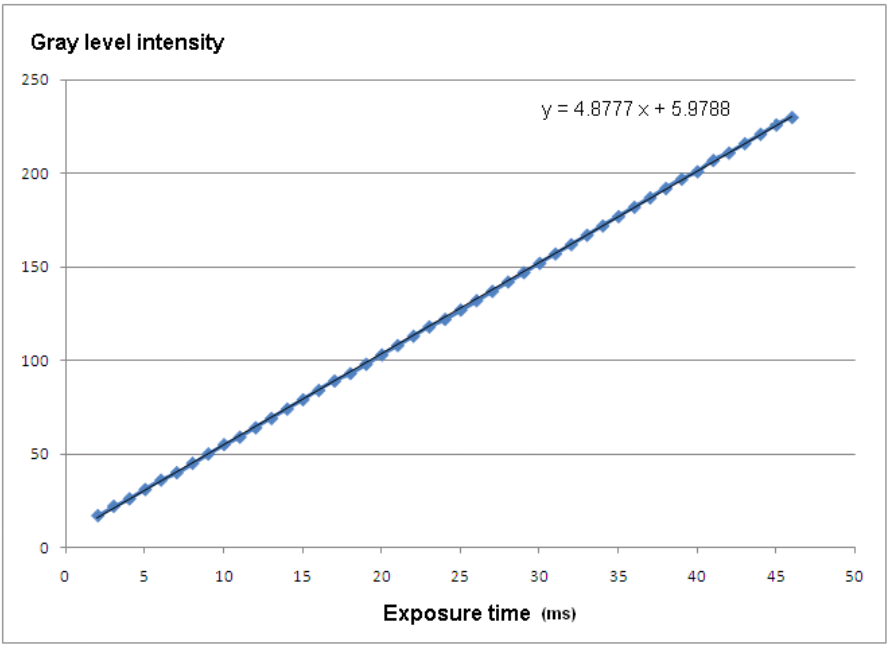


Figure 3.4: Linear relationship between gray level intensity and exposure time.

values. For this reason, a preprocessing stage over the acquired image is necessary for a correct estimate.

3.5.1 Radial distortion

The radial distortion is a common non-linear distortion due to the lens (in particular wide-angle or low-cost lenses) and causes rectilinear (straight) lines that do not pass through the optical center to be represented as curved lines. As reported in [5], given a distorted image point $p_d = (x_d, y_d)$, we can obtain the undistorted image point $p_u = (x_u, y_u)$ as follows:

$$\begin{aligned} x_u = \;& c_x + (x_d - c_x)\left(1 + k_1 r_d^2 + k_2 r_d^4 + \ldots\right) + \\ & p_1\left[2\,(x_d - c_x) + r_d^2\right] + 2p_2\,(x_d - c_x)\,(y_d - c_y) \\ y_u = \;& c_y + (y_d - c_y)\left(1 + k_1 r_d^2 + k_2 r_d^4 + \ldots\right) + \\ & p_2\left[2\,(y_d - c_y) + r_d^2\right] + 2p_1\,(x_d - c_x)\,(y_d - c_y) \end{aligned} \tag{3.8}$$

where (c_x, c_y) are the coordinates of the center of distortion and $r_d = \sqrt{x_d^2 + y_d^2}$. As mentioned in [6], this formulation can be approximated and simplified by neglecting the tangential components and by considering only the lower-order components: in this way, more than 90% of the radial distortion can be corrected. The final formulation employed in this work is the following:

$$\begin{aligned} x_u = \;& x_d + (x_d - c_x)\left(k_1 r_d^2\right) \\ y_u = \;& y_d + (y_d - c_y)\left(k_1 r_d^2\right) \end{aligned} \tag{3.9}$$

The parameters of the radial distortion are computed in a semi-automatic fashion using the method described in [6]. Briefly, the method exploit the distortion itself to evaluate its magnitude: one or more straight lines (candidate lines) are automatically extracted from the image by means of the Hough Transform, then an iterative variation of the estimated distortion parameters is performed in order to maximize the straightness of the candidate lines. This method proved to be sufficiently robust to correct the radial distortion with the used setup.

Once the points have been mapped from the distorted to the corrected image, they can be used to compute the correct angular weight (as proposed by UNI 11095 standard [2]).

3.5.2 Vignetting correction

The lens attached to the camera front introduce a strong darkening effect at the edges of the image, known as *vignetting*. Vignetting effect refers to a position-dependent loss of light in the output of an optical system, due mainly to the blocking of a part of the incident ray bundle by the effective size of the aperture stop; thus gradual fading-out of an image at points near its periphery results [7].

Vignetting correction begins by setting up a uniform white illumination source with a known input intensity level over a reference object with low specular reflection [8]. To accomplish this task, an integration sphere should be necessary, in order to have an uniform light source without distortions. This tool, very common in many photometric activities, is also very expensive, so it has been decided to approximate the sphere with an handmade *integration cube* (Figure 3.5), internally covered by mirrors and with two opal glass faces. The light of a source controlled in voltage passes through the input opal glass face, and the internal mirrors contribute to make the light look uniform all over the surface of the output opal glass face. The experiment for measuring illumination intensity response is carried out in a dark room to avoid ambient light interference. The camera is pointed toward the reference surface and intensity response at each pixel position is recorded.

Subsequently, a correction factor at each pixel position is calculated by the following form [9]:

$$F(x,y) = \frac{\max(I_{ref})}{I_{ref}(x,y)} \tag{3.10}$$

where $I_{ref}(x,y)$ is an intensity value at (x,y) pixel position, $\max(I_{ref})$ is the maximum value of $I_{ref}(x,y)$, and $F(x,y)$ is a correction factor to be stored in a LUT at (x,y) position. Thereafter, an image captured with the same camera can be corrected by multiplying pixel values of the image with corresponding correction factors stored in the LUT:

$$I'(x,y) = I(x,y) \cdot F(x,y) \tag{3.11}$$

where $I(x,y)$ and $I'(x,y)$ denote intensity values at (x,y) pixel position before and after vignetting correction.

The correction procedure is inspired to the technique propose by [9]. In their approach, an hyperbolic cosine function is considered the most suitable to mimic the physical manifestation of vignetting effect:

$$f(x,y) = cosh(r_x(x - x_0)) \cdot cosh(r_y(y - y_0)) + c \tag{3.12}$$

where r_x and r_y denote falloff rates along x and y axis of the image, while x_0 and y_0 denotes the image center.

Instead, in this implementation we have preferred to use a paraboloid function:

$$f(x,y) = a(x - x_0)^2 \cdot b(y - y_0)^2 + c \tag{3.13}$$

where a, b, c, x_0 and y_0 are the parameter to determine in order to model the intensity plateau. Analogously to [9], a nonlinear model fitting using the Levenberg-Marquardt optimization algorithm has been employed to find out the model parameters, and thus the image $F(x,y)$ of correction factors .

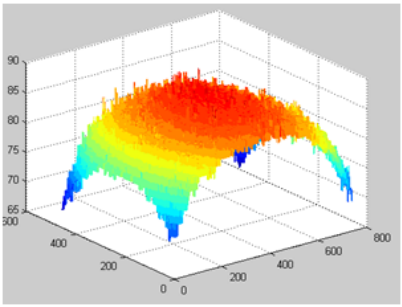
The results of the vignetting correction are shown in Figure 3.6

Figure 3.5: Integration cube

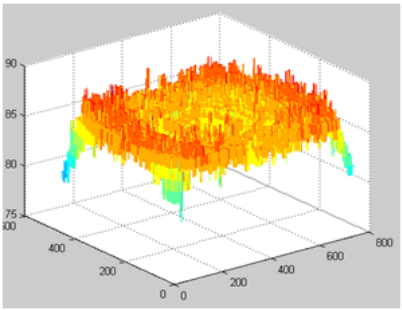
3.6 Architecture and Design

3.6.1 Hardware organization

The hardware employed to realize the whole system is the Cameleon camera, from OptoMotive [10]. This is an innovative USB camera based on FPGA system-on-chip. The Cameleon camera is a ready-to-use solution together with the full source code of the complete system and the powerful HSDK (hardware-software development kit). The open architecture nature of this camera allows easy customization and integration of new features and functions. The hardware system (Figure 3.7) is composed of 2 boards: the base board and the head board. The base board includes power supply, FPGA, DDR SDRAM, USB controller and SPI Flash. The



(a)



(b)

Figure 3.6: In a and b are shown the gray level spatial distributions of an image obtained from a uniform source with a known intensity level before and after vignetting correction.

whole hardware system is USB powered with triple DC-DC converters for maximal power efficiency (SMPS - Switching Mode Power Supply). The core is a 1.6 Mgates Spartan-3E FPGA, with 64 MB DDR SDRAM. The 16-bit wide DDR data bus running at 100 MHz offers a 400 MB/s peak bandwidth. The USB controller is the well proven Cypress CY7C68013A (also known as FX2) which offers up to 36 MB/s of bandwidth in bulk mode. The link between FPGA and FX2 is 8-bit wide and runs at 48 MHz. The FX2 is also connected to FPGA through I2C which is used as a command interface. The board also includes non-volatile SPI FLASH memory. The SPI bus is shared by FPGA and FX2 which is necessary to enable USB firmware upgrade and FPGA booting. This board is even equipped with two high-density shockproof B2B (board-to-board) 80 pin expansion connectors, a 6 pin JTAG header for FPGA programming and debug, a 6 pin SPI header for direct Flash programming.

The head sensor board houses the image sensor. It is the Aptina MT9V034 1/3"'CMOS monochrome sensor, 752 x 480 pixels, 64 frames per second at full

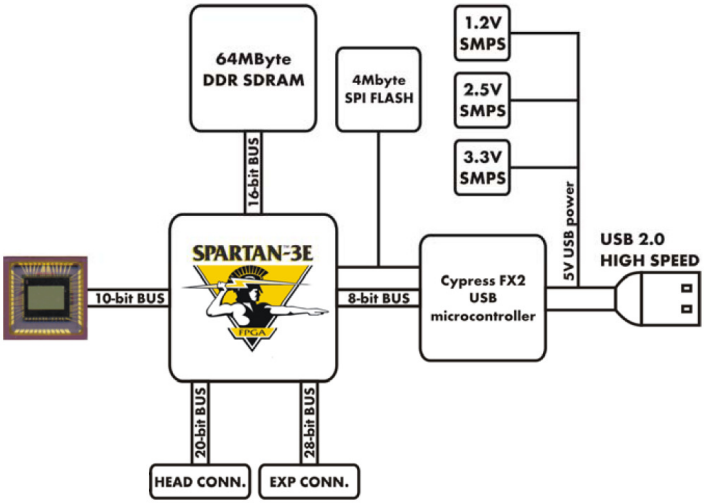


Figure 3.7: Camera structure block scheme

resolution, with a global shutter. Global shutter means that exposure of all pixels starts and stops at the same time preventing motion blur for fast moving objects. The pixel clock is set to 27 MHz. The sensor uses 10-bit parallel interface for reduced system power consumption and electromagnetic noise.

3.6.2 The System-on-Chip architecture

The system-on-Chip has been created by means of the Xilinx Embedded Development Kit (EDK). It is an embedded system based on the Xilinx 32 bit MicroBlaze soft core microprocessor and includes all the hardware blocks needed to handle the external peripherals and memory, as depicted in Figure 3.8.

The system architecture is characterized by a high level of data transfer parallelism. The system is equipped with DMA engine and FIFOs and it is able to deal with up to three image sensors (in this application is employed only one image sensor). This architecture enables high speed data streaming to and from the external memory with minimal CPU activity. It is possible to stream data from multiple image sensors to external RAM and at the same time to draw out data from the memory intended for software or hardware processing. The raw image data transfer is performed by the DMA engine moving data from memory to USB interface.

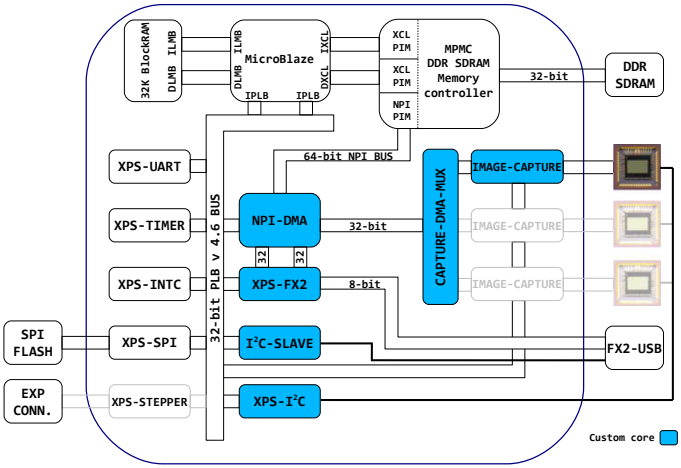


Figure 3.8: Soft core architecture

CPU and Memory subsystem

This hardware section employs the external 64MB DDR DRAM and a set of Xilinx standard IP Modules: the MicroBlaze CPU, 32 Kbytes Blocked RAM, Multi Ports Memory Controller. The MicroBlaze embedded processor soft core is a highly configurable reduced instruction set computer (RISC) optimized for implementation in Xilinx FPGAs. In this application it runs at 50 MHz clock frequency with data and instruction caches, integer divider and 64-bit multiplier units enabled. The Blocked RAM is a fast synchronous static RAM buried in to FPGA. This memory is initialized with the bootloader program when the FPGA load the bitstream configuration file and it is used to boot the embedded processor at the power up time. The external DDR SDRAM is handled by the MPMC (Multi Port Memory Controller). MPMC provides access to external DDR SDRAM memory for one to eight ports, each port can be chosen from a set of Personality Interface Modules (PIMs). Using a custom-built DMA engine it can stream data from multiple imaging sensors to external RAM. The MicroBlaze CPU can access at data and code located in the external DDR SDRAM by means of MPMC using IXCL and DXCL (Instructions Xilinx CacheLink, Data Xilinx CacheLink) interfaces ports. The DDR has 16-bit wide data bus running at 100 MHz offering 400 MB/s peak bandwidth. The overview of the entire system is proposed in Figure 3.9.

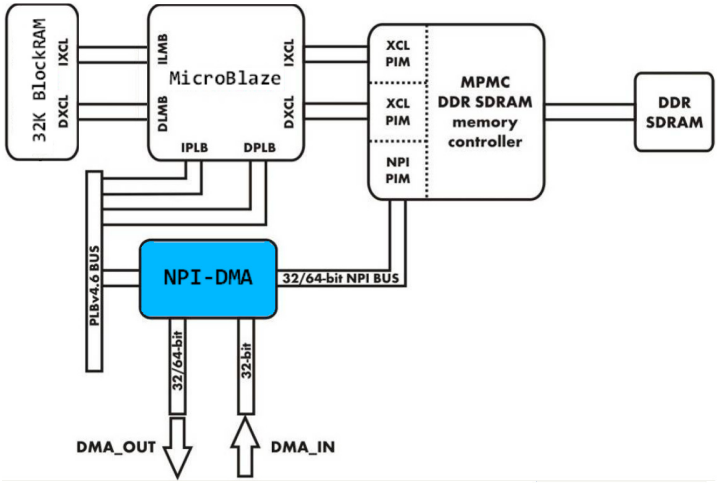


Figure 3.9: CPU-Bram-MPMC

High speed data transfer subsystem

Figure 3.10 depicts the subsystem elected to handle the high speed data streaming between external memory, image sensor and USB controller. This subsystem is composed by the following blocks: NPI-DMA, CAPTURE-DMA-MUX, IMAGE-CAPTURE. All these blocks implements functionality not available in the EDK peripherals library. These blocks have been created as custom VHDL IP modules using the "Create and Import Peripheral" function available from EDK. The image sensor is an Aptina MT9V034 1/3"' CMOS monochrome sensor, 752 x 480 pixels, capable to deliver up to 64 frames per second at full resolution and provides 10 bits per pixel. The pixel clock is set to 27 MHz. It streams image data using 10 bit parallel interface. The Cameleon camera is able to deal with up to three image sensors. Each of these is directly connected to the FPGA and sends its data stream to a IMAGE-CAPTURE module. This module handles the image sensor control signals and packs three pixels bits in one 32 bit word to send at the CAPTURE-DMA-MUX module. This module is a mux and provides to route the image capture data stream from the selected image sensor to the external memory by DMA engine. NPI-DMA is high performance direct memory access (DMA) engine. It is highly flexible, it enables high speed data streaming to and out from the external memory attached to the Multiport Memory Controller (MPMC). The input and output port can run simultaneously. The NPI-DMA has 4 interfaces:

- Xilinx PLBv4.6 for access to 9 32-bit registers. These registers control the whole peripheral;

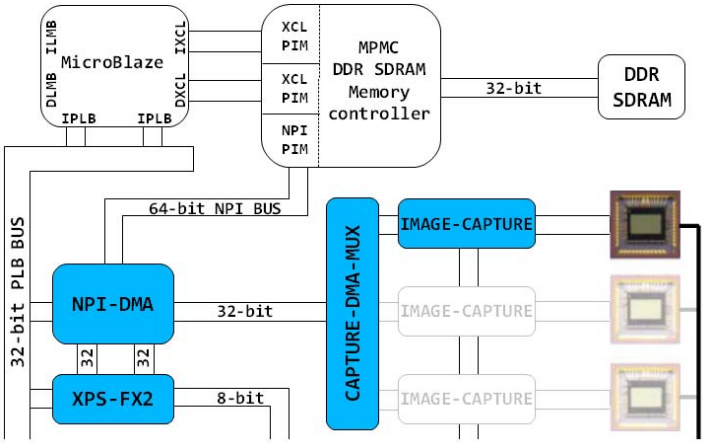


Figure 3.10: High speed data transfer subsystem

- MPMC Native Port Interface (NPI) bus supporting 32 or 64-bit width. This bus is used for high speed access to external memory;
- Proprietary asynchronous 32-bit wide DMA-IN bus used for data streaming to external memory. This port can stream data in blocks up to 64 word burst. Maximal sustainable bandwidth at 100MHz NPI-Clk is 200MB/s (50MHz Capture-valid);
- Proprietary synchronous 32 or 64-bit wide DMA-OUT bus used for data streaming from external memory. The bus width is dependent on NPI bus width. This port can stream data in single beat transfers word (32-bits), double word (64-bit), 16, 32 and 64-word per one transaction. Maximal sustainable bandwidth at 100MHz NPI-Clk is approx. 300MB/s at 32-bits and 600MB/s at 64-bits.

USB communication subsystem

USB port is provided by USB High Speed microcontroller (Cypress CY7C68013A, also known as EzUSB FX2), depicted in Figure 3.11. The FPGA side of the USB microcontroller is handled by two custom VHDL IP modules, XPS-FX2 and I2C-SLAVE, both registers accessible by MicroBlaze soft processor. I2C-SLAVE is a low speed communication core used to send commands from the PC to the MicroBlaze soft processor. XPS-FX2 is a high speed communication core used to transfer data

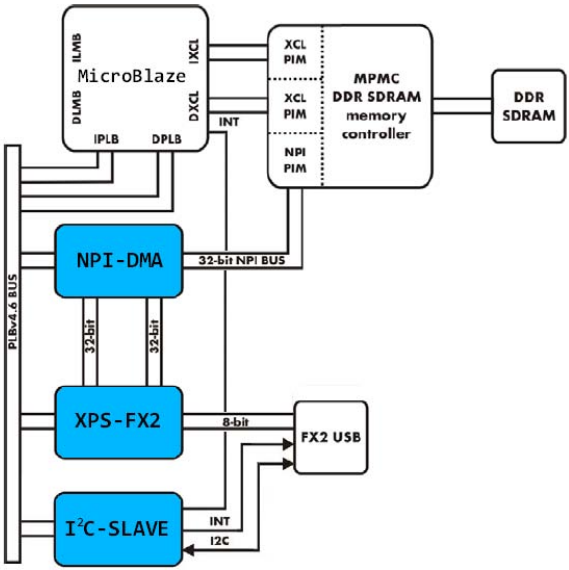


Figure 3.11: USB communication subsystem

between PC and MicroBlaze soft processor. It is directly connected to the DMA engine to provide high bandwidth data streaming. When transferring raw image to USB, data are transferred directly from DMA engine to XPS-FX2. The USB microcontroller (FX2) firmware was designed for maximal supported bandwidth of 36 MB/s for TX and 24 MB/s for RX.

Auxiliary subsystem

This subsystem is made up of services blocks providing standard functions parts of the EDK peripherals library. These blocks are:

- Uart (Universal Asynchronous Receiver Transmitter), useful as easy communication channel to quickly connect external equipments or PC; software driven Timer;
- SPI (Synchronous Serial Peripheral Interface) used to get access at the external SPI Flash where are stored the FPGA Bitstream loaded at the powerup time by the FPGA, the soft core application program loaded in to DDR SDRAM by the bootloader (from the BlockRAM) at boot time and the look-up table useful to corrects the Vignetting and radial distortions;
- I^2C to configure and control the Image capture sensor;

- 20 Inputs/Outputs general purpose digital lines.

Device utilization summary

The digital architecture described has been implemented and tested on the Xilinx FPGA available on the Camelon board, the Spartan-3E FPGA. It is characterized by 1.6 million system gates, an array of 76x58 configuration logic blocks (CLBs) providing 14,752 Slices (1 CLB = 4 Slices) and a maximum of 231 Kbits distibuted RAM. Furthermore, it provides 36 18 x18 bits multiplier blocks and 36 18-Kbit selected-RAM blocks for an amount of 648 Kbits RAM. Here is reported the synthesis report, that is, the number of instantiated components for the whole design.

```
Device utilization summary:
---------------------------

Selected Device : 3s1600efg320-4

 Number of Slices:                     8868  out of  14752    60%
 Number of Slice Flip Flops:          10777  out of  29504    36%
 Number of 4 input LUTs:              16185  out of  29504    54%
    Number used as logic:             11099
    Number used as Shift registers:     694
    Number used as RAMs:               4392
 Number of IOs:                         111
 Number of bonded IOBs:                  67  out of    250    26%
    IOB Flip Flops:                      39
 Number of BRAMs:                        34  out of     36    94%
 Number of GCLKs:                        15  out of     24    62%
 Number of DCMs:                          3  out of      8    37%
```

3.7 Experimental results

In order to evaluate the performance of the developed system, a comparison with a third part reference equipment has been done. The reference equipment provides the evaluation the veiling luminance of the observed scene. To obtaine the series of comparative data both the equipments have been palced in order to align them as much as possible on the same target. The observed scene was a street in an average sunny day with some moments of cloud cover, which were reported by both systems. In particular, in the reported graph (see Figure 3.12) is shown the substantial correspondence between the estimation values of the veiling luminance reported by both the equipments. The average deviation between the two data set is 4.51%. To scale

this deviation it is necessary to take into account two sources of error. An observed 3% of error from the CMOS image sensor that indicates the fluctation of the average gray level obtained by observing a stable scene, and 2% of error due to the employed light meter.

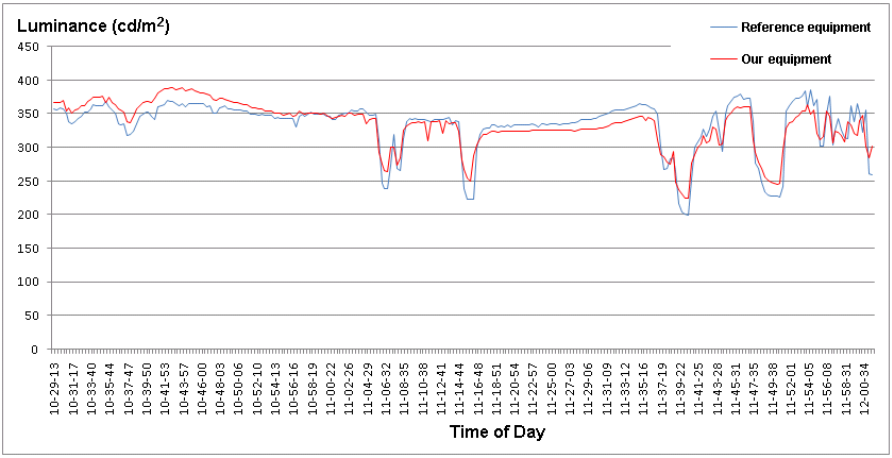


Figure 3.12: The graph shows the estimation of the veiling luminance at different time of day. The comparison data have been acquired using the developed equipment and a reference equipment. The average deviation between the two sets of data is less then 4.52 %.

Chapter 4

High Performance Connected Components Labelling on FPGA

4.1 Introduction

In this chapter a comparison of the two most advanced algorithms for connected components labeling is proposed by highlighting how they perform on a soft core SoC architecture based on FPGA. In particular, the block based connected components labeling algorithm optimized with decision tables and decision trees developed by us is tested. The embedded system is composed of the CMOS image sensor, FPGA, DDR SDRAM, USB controller and SPI Flash. Results highlight the importance of caching, and instruction and data cache sizes for high performance image processing tasks.

4.2 Labeling

Connected component labeling is a fundamental task in several image processing and computer vision applications, e.g. for identifying segmented visual objects or image regions.

The research efforts in labeling techniques have a very long history ([11], [12], [13], [14], [15]), full of different strategies, improvements and results. Even if current computer architectures do not suffer anymore of many of the resource limitations which motivate many proposals, embedded systems generally have limited processing power, memory and cache. So, a fast and efficient algorithm able to minimize its impact in terms of hardware requirements is undoubtedly very advantageous.

Although the first algorithms for connected components labeling were proposed more than fifty years ago, only in the last years new strategies provided significant performance improvements, in particular with the introduction of the *Union-Find* approach for label equivalences resolution, array based data structures and smarter neighborhood management. In [16], a new step forward has been made. It has been resumed the "old" condition-action paradigm which can be effectively described as a *single entry decision table*, then it has been proposed to further enhance this tool introducing the *OR-decision tables*, which enclose the possibility to represent more than one (*equivalent*) action for each set of conditions. Then, a boolean optimization algorithm has been adopted to automatically produce the optimal decision tree in terms of number of evaluations, thus access costs. Thus, raster-scan labeling has been approached in a block-wise (instead of pixel-wise) manner: this paradigm, applied as-it-is without any modifications, granted significant performance improvements of the neighborhood exploration in terms of memory access times, compared to the state of the art.

In this chapter, the effectiveness of the proposed approach even when implemented on a FPGA hardware, using a configurable soft core RISC microprocessor will be shown by detailing the hardware parameters affecting the algorithm performance.

4.3 High performance connected component labeling

The procedure of collecting labels and solving equivalences may be described by a *command execution metaphor*: the current and neighboring pixels provide a binary command word, interpreting foreground pixels as 1s and background pixels as 0s. A different action must be taken based on the command received.

Four different types of actions can be identified: *no action* is performed if the current pixel does not belong to the foreground, a *new label* is created when the neighborhood is only composed of background pixels, an *assign* action gives the current pixel the label of a neighbor when no conflict occurs (either only one pixel is foreground or all pixels share the same label), and finally a *merge* action is performed to solve an equivalence between two or more classes and a representative is assigned to the current pixel. The relation between the commands and the corresponding actions may be conveniently described by means of a *decision table* [17], that is a tabular form that presents a set of conditions and their corresponding actions: actions corresponding to *true* (1s) entries should be performed if the relative set of conditions is verified. Classically, the execution of **all** the actions in the set is required, defining the so called *AND*-decision tables. Instead, for this problem an *OR*-decision table has been defined, in which **any** of the actions in the set may be performed in order to satisfy the corresponding condition.

The majority of images are stored in raster scan order, so the most common technique for connected components labeling applies sequential local operations in

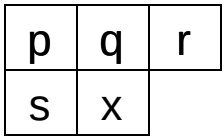


Figure 4.1: The pixel mask $\mathcal{M}(x)$ used to compute the label of pixel x, and to evaluate possible equivalences in raster scan techniques.

that order, as firstly introduced in [11]. This is classically performed in 3 steps:

1. First image scan (provisional labels assignment and collection of label equivalences);
2. Equivalences resolution (equivalence classes creation);
3. Second image scan (final label assignment).

In order to describe the behavior of a labeling algorithm with a decision table. It is necessary to define the conditions to be checked and the corresponding actions to take. In the more classical approach, the conditions are given by the fact that the current pixel x and the four neighboring ones in mask $\mathcal{M}(x)$ belong to the foreground $\mathcal{F}$, as shown in Figure4.1. The conditions outcomes are given by all possible combinations of 5 boolean variables, leading to a decision table with 32 rules. The actions belong to four classes: *no action*, *new label*, *assign* and *merge*.

The action entries are obtained applying the following considerations:

1. *no action* if $x \in \mathcal{B}$;
2. $L(x) =$ *new label* if $x \in \mathcal{F} \wedge \mathcal{M}(x) \subset \mathcal{B}$;
3. $L(x) = L(y)$ if $x \in \mathcal{F} \wedge \exists!\ y \in \mathcal{M}(x) \mid y \in \mathcal{F}$;
4. $L(x) = merge(\{L(y) \mid y \in \mathcal{M}(x) \cap \mathcal{F}\})$ otherwise

Using these considerations the equivalences are solved and a representative (provisional) label L is associated to the current pixel x. The process then moves ahead to the next pixel and the next neighborhood accordingly. The algorithm used to solve equivalences (namely a *disjoint set union* problem) is the *Union-Find* which allows the inclusion of the equivalences resolution step directly into the first image scan, removing the need of collecting equivalences. This approach constitutes the basic structure of all modern labeling algorithms, which perform online equivalences resolution.

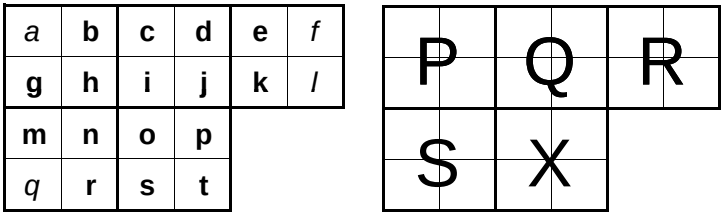
Firstly, *merge* operations have a higher computational cost with respect to an assign, so their number should be reduced at the minimum in order to improve the

							assign				merge	
x	p	q	r	s	no action	new label	x = p	x = q	x = r	x = s	x = p+r	x = r+s
0	-	-	-	-	**1**							
1	0	0	0	0		**1**						
1	1	0	0	0			**1**					
1	0	1	0	0				**1**				
1	0	0	1	0					**1**			
1	0	0	0	1						**1**		
1	1	1	0	0			*1*	**1**				
1	1	0	1	0							**1**	
1	1	0	0	1			*1*			**1**		
1	0	1	1	0				**1**	*1*			
1	0	1	0	1				**1**		*1*		
1	0	0	1	1								**1**
1	1	1	1	0			*1*	**1**	*1*			
1	1	1	0	1			*1*	**1**		*1*		
1	1	0	1	1							*1*	**1**
1	0	1	1	1				**1**	*1*	*1*		
1	1	1	1	1			*1*	**1**	*1*	*1*		

Figure 4.2: The resulting *OR*-decision table for labeling. Bold 1's are selected with the described procedure

performance of labeling. Similarly a *merge* between two labels is computationally cheaper than a *merge* between three labels. Extending the same considerations throughout the whole rule set, an effective "compression" of the table has been obtained, as shown in Figure 4.2: whenever it is possible, all costly operations are substituted with cheaper ones, obtaining an *OR*-decision table with multiple alternatives between *assign* operations, and only in a single case between *merge* operations. Moreover the reduction leads also to the exclusion of many unnecessary actions (for example, the merge between p and q) without affecting the algorithm outcome.

The second step to further reduce memory accesses is getting the optimal ordering of conditions tests. This is well represented by an optimal *decision tree*: the sequence requiring the minimum number of tests corresponds to the decision tree with the minimum number of nodes. The transformation of the decision table in an optimal decision tree has been deeply studied in the past and the Dynamic Programming technique proposed by Schumacher [18] has been used, which guarantees to obtain an optimal solution. Since this algorithm works on *single entry decision tables*, in [16] Grana proposed a greedy algorithm to find out which action to choose among all the alternatives: the rationale behind this approach is that the more rules



(a) (b)

Figure 4.3: The mask used for 2×2 block based labeling is shown. (a) gives the identifiers of the single pixels employed in the algorithm (a, f, l and q are not used), while (b) provides the blocks identifiers.

require the execution of the same action, the more likely it will be to find large grouping covering that action.

The availability of the previously described technique allows to enlarge the neighborhood exploration window, with the aim to further speed up the connected components labeling process. As previously reported in [16], the key idea of this proposal starts from two very straightforward observations: i) when it is used 8-connection, the pixels of a 2×2 square are all connected to each other and ii) a 2×2 square is the largest set of pixels in which this property holds. This implies that all foreground pixels in a block will share the same label at the end of the computation. For this reason it has been proposed to scan the image moving on a 2×2 pixel grid applying, instead of the classical neighborhood of Figure 4.1, an extended mask of five 2×2 blocks, as shown in Figure 4.3.

Scanning the image with this larger grain has the advantage to allow the labeling of four pixels at the same time. The number of provisional labels created during the first scan is roughly reduced by a factor of four, and it is necessary to apply much less unions, since labels equivalence is implicitly solved within the blocks. Moreover a single label is stored for the whole block.

On the other hand, the neighborhood to consider now is much larger. The standard procedure (that is to consider all the pixels in the neighborhood) greatly increases computational time due to the number of memory accesses and merge operations required. Likewise a manual approach for an effective neighborhood exploration is unfeasible since it brings to deal with much more than 5 pixels for each labeling operation, and the amount of combinations to explore becomes enormous. But the general procedure described in the previous section is designed to provide an effective way to face the optimization in this situation.

The new scanning procedure may require also the same pixel to be checked multiple times, but the impact of this problem is greatly reduced by the proposed optimized pixel access scheme. Finally, a second scan requires to access again the

original image to check which pixels in the block require their label to be set. Overall the advantages will be shown to largely overcome the additional work required in the following stage.

Employing all pixels in the new mask of Figure 4.3, it would need to work with 20 pixels: for this reason, the decision table would have $L = 20$ conditions, and $N = 2^{20}$ possible configurations of condition outcomes. However, it can be noticed that not all those pixels are necessary to compute labeling information. In particular pixels a, f, l, q do not provide 8-connection between blocks of the mask and can be ignored. It is necessary to deal with $L = 16$ pixels (thus conditions), for a total amount of 2^{16} possible combinations.

Since manually specifying the action entries for all 65 536 combinations is impractical, it has been chosen not to directly deal with the condition outcomes but abstracting the relations between blocks, and then obtaining back the pixel combinations accordingly.

After the application of the described method, the 65 536 rules can be produced. An empirical algorithm (as detailed in [16]) has been used to select a single action among the alternative for each rule. The Schumacher's algorithm is finally applied to this single entry decision table, producing an optimal tree containing 210 nodes, with 211 leaves sparse over 14 levels. The C/C++ code implementing the sequence of these conditions was automatically generated.

4.4 Architecture and Design

The hardware employed to realize the whole system is the Cameleon camera, from OptoMotive [10]. This is an innovative USB camera based on FPGA system-on-chip. It is described in section 3.6.

4.5 Experimental Results

To test the effectiveness of the proposed approach for embedded processors, two aspects have been stressed: the data access and the code access, since this approach allows to reduce the number of memory accesses, but requires to access farther memory cells (up to two lines before current pixel) and has a huge code size (with respect to the traditional approaches). Another aspect which could impact performance is the fact that a decision tree reduces basically to a large number of nested *if-then-goto* statements, possibly influencing the pipeline effectiveness.

Working with a configurable IP-core microprocessor, allows personalizations on many hardware specifications related to the aforementioned problems. The main question is how the memory access time impacts on performance. The instruction and data cache sizes can be configured in order to specifically analyze algorithms behavior.

To provide a valuable comparison, it has been chosen to also test one of the best labeling techniques available, that is the proposal by He *et al.* [15]. This algorithm performs an optimal neighborhood search on the mask in Figure 4.1, making it the best competitor for the proposed approach. Specifically, the machine code is smaller and the data access is limited to one image line before current pixel.

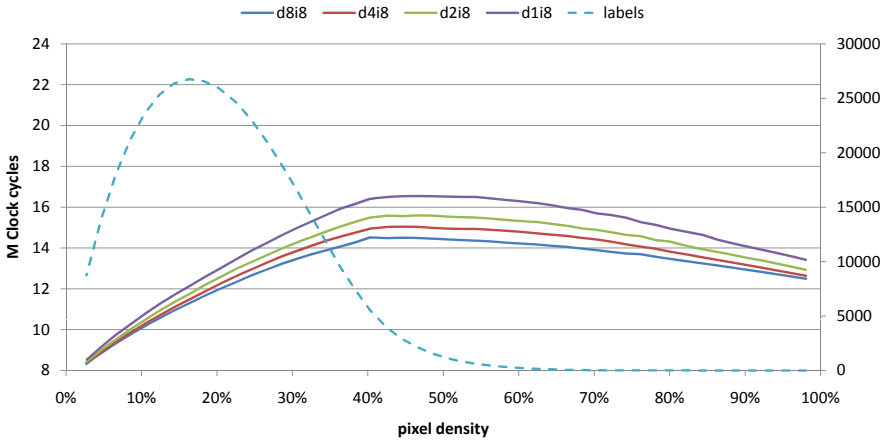


Figure 4.4: Performance of the proposed approach, expressed in clock cycles.

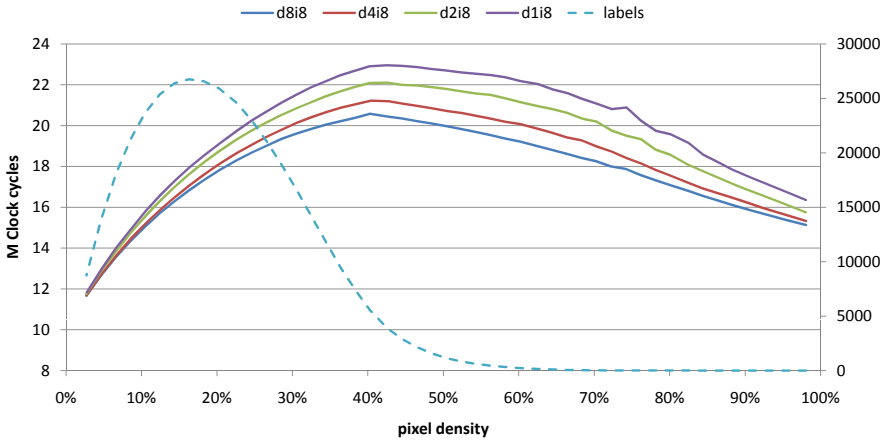


Figure 4.5: Performance of He's approach, expressed in clock cycles.

The dataset used for the test is a set of randomly generated images, thresholded at 50 different levels (50 different objects pixel densities), to allow measuring algo-

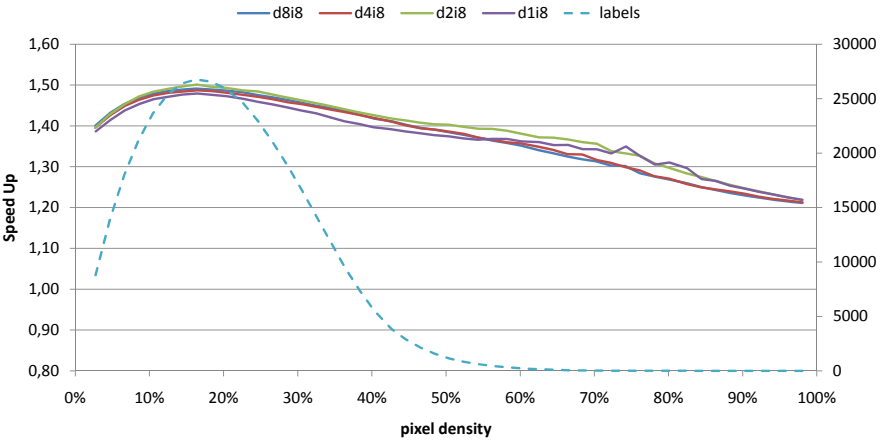


Figure 4.6: Chart showing how the speedup of the proposed algorithm with respect to He's approach varies along with a decreasing size of data cache.

rithms performance in different conditions (varying number of connected objects and different objects structures). The reported times are averaged over the whole set. The reported times are espressed in number of clock cycles which correspond to 20 ns in the used architecture, while the comparison between algorithm is given in speedup

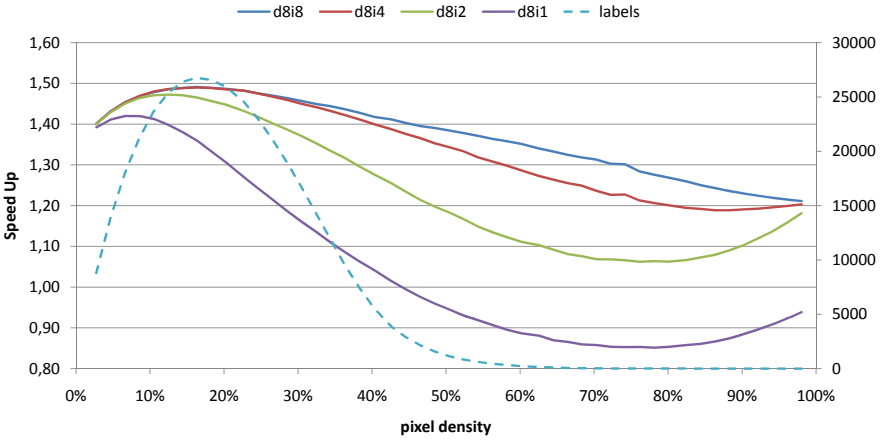


Figure 4.7: Chart showing how the speedup of the proposed algorithm with respect to He's approach varies along with a decreasing size of instructions cache.

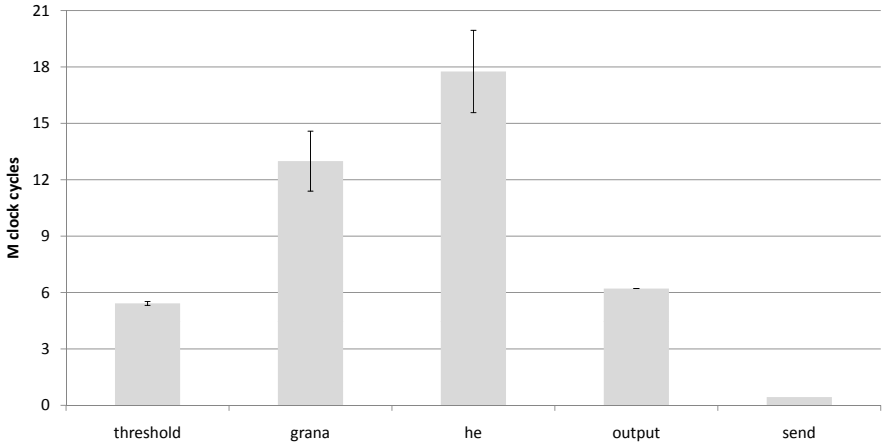


Figure 4.8: Overview of average computational time of every function called during the tests, using an architecture with data and instructions cache of 8kB.

(SU), which is the proposed algorithm time over He's algorithm time. Moreover, in all reported charts a data series describing the trend of the number of labels has been included. In Figure 4.4 and 4.5 is reported a first test obtained by varying the data cache of the architecture, starting from 8kB down to 1kB. It is possible to observe that the larger the cache, the faster the algorithms are. Figure 4.6 shows that the speedup provided by the proposed algorithm is between 1.5 and 1.2 depending on the objects pixel density, meaning that He's algorithm is 20% to 50% slower than the proposed approach. This chart shows that the data cache size does not clearly influence the speedup. Things are radically different when moving to instruction cache size. The results of the test are shown in Figure 4.7. As expected, the proposed algorithm performance is heavily influenced by the instructions cache size: the algorithm performance decreases along with the cache size, since the main code loop (which contains the decision tree) is very long. In particular, it is possible to note that only when instructions cache is under 2kB the algorithm performs slower than He's at higher densities. Globally, as shown in Figure 4.8, when data and instructions cache sizes are set to 8kB, the proposed approach can provide a significant speedup with respect to the state of the art. Given a very basic threshold procedure as baseline, the proposed connected components labeling requires slightly more than twice the time.

Chapter 5

Embedded camera motes based on ARM processor

5.1 Introduction

This chapter describes the embedded camera motes based on ARM processor employed in the case of study reported in chaper six on "Energy efficient software application on embedded smart cameras".

The diffusion of embedded systems has increased enormously worldwide in recent years due to their completeness, relatively-low cost and power efficiency. Embedded systems equipped with smart camera sensors are employed in many different applications including environmental monitoring [19], stereo matching [20], as well as a plethora of applications related to video surveillance, such as foreground object detection [21–23], face detection [24], people detection [25] etc. When a camera sensor is added to an embedded system, several problems arise due to the large amount of image data to be stored and processed. This impacts the overall efficiency, the memory requirements, the communication bandwidth and the energy consumption of the system.

Most of the works related to camera-equipped embedded systems mainly focus on maximizing computational efficiency by optimizing the video processing algorithms. However, as the focus moves towards mobile applications [24], limited resources mandate consideration of energy consumption. With the advances in hardware technology, embedded devices are becoming more sophisticated. Embedded smart cameras are being equipped with general purpose processing units that allow implementing sophisticated vision algorithms on these platforms. Battery-powered embedded smart cameras provide a lot of flexibility in terms of camera quantities

and placement, however they have limited resources, such as computational power, memory and energy. Since battery-life is limited, and video processing tasks consume considerable amount of energy, it is essential to have lightweight algorithms and methodologies to increase the energy-efficiency of each camera. Even though methods have been proposed for object detection and tracking with embedded systems, much less effort has been spent on developing applications that decrease the energy consumption of the embedded cameras. Chen et al. [26] introduced the CITRIC camera mote that provides more computing power and tighter integration of physical components while still consuming relatively little power. An Omnivision sensor OV9655 is employed to capture frames. The down-sampling of the images is performed by software using the CITRIC API libraries. Rinner et al. [27] present a comparison of various smart camera platforms. Casares et al. [28] introduced an algorithm for resource efficient foreground object detection, and presented the savings in processing time and energy consumption on CITRIC cameras. They obtained the savings at software-level. Also, it was recently shown by Casares et al. [29] that performing hardware-level operations at the image sensor level significantly reduces the energy consumption of the embedded smart cameras.

5.2 The CITRIC camera

The CITRIC camera is depicted in Figure 5.1, it is a full programmable embedded smart camera ready to be connected to the TelosB, a wireless mote from Crossbow Technology. The block diagram of the camera board is shown in Figure 5.2. The camera is equipped with a general-purpose processor running embedded Linux (see Section 5.3), an image sensor (see Section 5.4), external memories and others supporting circuits. The ARM PXA270 microprocessor is a fixed-point processor from Marvell with a maximum speed of 624 MHz, 256 KB of internal SRAM and a wireless MMX co-processor to accelerate multimedia operations. The typical CPU frequencies that the CITRIC platform supports are from 208 to 520 MHz. The available external memory is composed of 64 MB of SDRAM and 16 MB of NOR FLASH. The SDRAM has significantly-high density and speed, and very little power consumption. The NOR FLASH has the capability to execute code directly out of the non-volatile memory on bootstrap (eXecution-In-Place, XIP) and is natively supported by the PXA270 processor.

5.3 The Microprocessor

The CITRIC camera platform is equipped with a general purpose processor running embedded Linux, the Intel PXA270 processor [30]. The choice to employ a general purpose processor running embedded Linux aims to facilitate the development of software applications using C/C++ language and standard libraries of well studied

Figure 5.1: The picture of the CITRIC camera mote.

image processing and computer vision algorithms. The PXA27x processor is a fixed point integrated system-on-a-chip microprocessor for high-performance, low-power, portable, handheld and handset devices. It incorporates the Intel XScale technology with on-the-fly voltage and frequency scaling and sophisticated power management to provide industry-leading MIPs/mW performance. The PXA27x processor complies with the ARM Architecture V5TE instruction set (excluding floating point instructions). It also supports Intel Wireless MMX integer instructions in applications such as those that accelerate audio and video processing. The PXA27x processor memory interface supports a variety of external memory types to allow design flexibility. Support for the connection of two companion chips permits a glueless interface to external devices. The processor also provides four 64-Kbyte banks of on-chip SRAM, which can be used for program code or multimedia data. Each bank can be configured to retain its contents when the processor enters a low-power mode. An integrated LCD panel controller provides support for displays up to 800 x 600 pixels. It permits 1, 2, and 4-bit gray scale and 8- or 16-bit color pixels. A 256-entry palette RAM provides flexibility in color mapping. A set of serial devices and general system resources provides computational and connectivity capability for a variety of applications. The PXA27x processor incorporates a comprehensive set of system and peripheral functions that makes it useful in a variety of low-power applications. Figure 5.3 illustrates the system-on-a-chip processor. The diagram shows a primary system bus with the Intel XScale core attached, along with an LCD controller, USB

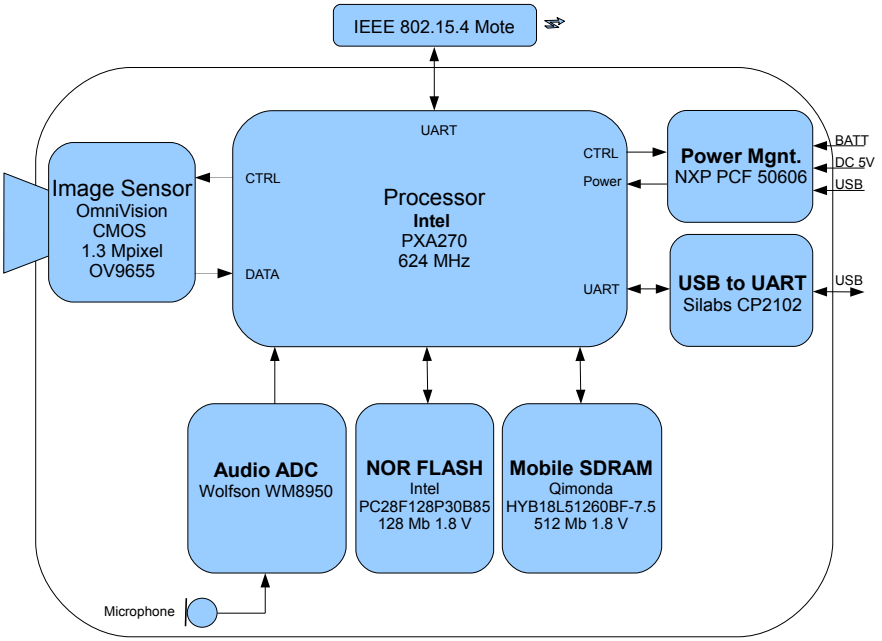


Figure 5.2: The block diagram of the CITRIC camera.

host controller, and 256 Kbytes of internal memory. The system bus is connected to a memory controller to allow communication with a variety of external memory or companion-chip devices, and it is also connected to a DMA controller/bridge to allow communication with the on-chip peripherals. Some of these peripheral functions provide the ability to handle directly the image sensor. In particular, the Quick Capture Interface (QCI) provides a connection between the processor and the image sensor (as shown in Figure 5.4). The QCI is able to acquire data and control signals and performs the appropriate data formatting before routing the data to the memory using direct memory access (DMA). The I2C interface is directly connected to the SCCB interface of the image sensor and is used to access the configuration register set.

5.4 The Image sensor

The image sensor on the CITRIC camera is a OmniVision OV9655 [31], which is a low voltage SXGA CMOS image sensor with an image micro-controller on board. It supports image sizes SXGA (1280 x 1024), VGA (640 x 480), CIF (352 x 288),

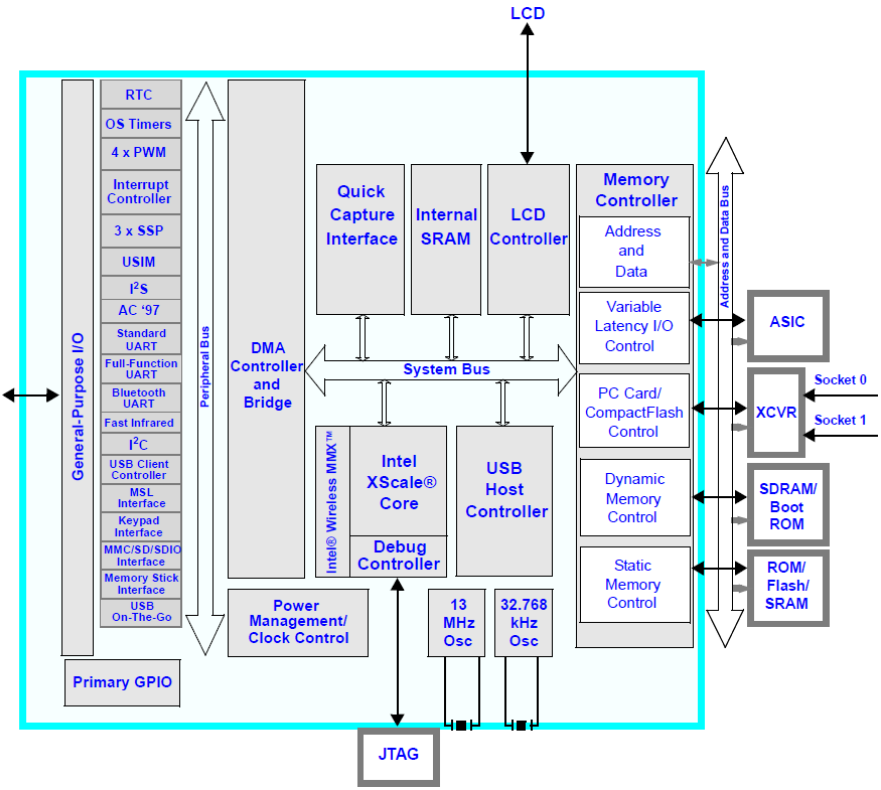


Figure 5.3: Intel PXA270 block diagram.

and any size scaling down from CIF to 40 x 30, and provides 8-bit/10-bit data formats [26]. It can operate up to 15 frames per second (fps) in SXGA mode and up to 30 fps working in VarioPixel(R)[1] mode when performing sub-sampling. Figure 5.4 shows the interconnection of OV9655 and the Intel Quick Capture Interface on ARM PXA270. The image sensor offers the full functionality of a camera and image micro-controller on a single chip. There is a complete control over image quality, formatting and output data transfer and all required image processing functions are also programmable. The Serial Camera Control Bus (SCCB) interface is used to program the sensor behavior by setting all the control registers in the device. It is an Inter-Integrated Circuit (I2C) compatible hardware interface. The Digital Video Port provides a connection between the sensor and the CITRIC camera main pro-

[1] VarioPixel is a newly developed technology that uses multiple pixels to act as a single pixel in order to improve the chips performance. This significantly improve low light performance and enhance the video capture.

cessor PXA270. It is used to capture the image data. It is a unidirectional communication bus transferring 10-bit data signals and the line and frame synchronization signals [30].

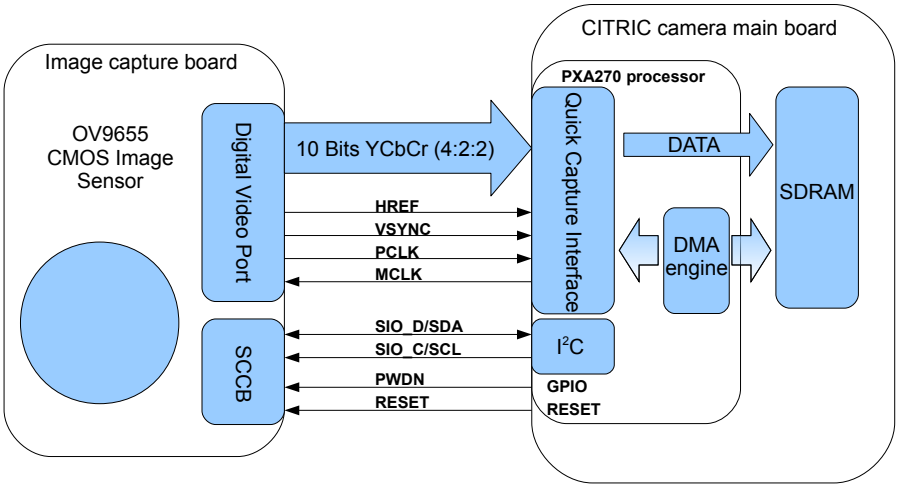


Figure 5.4: Interconnection of OV9655 and the Intel Quick Capture Interface on ARM PXA 270.

5.5 The Mote: TelosB

The CITRIC camera platform is ready to be connected to the TelosB, a wireless mote from Crossbow Technology. The camera communicates with the mote by dedicated asynchronous serial interface. TelosB is an ultra low power wireless sensor module (mote) for research and experimentation. It is developed by UC Berkeley to enable wireless sensor network (WSN) research. The main features of the mote are: minimal power consumption, easy to use, and software and hardware robustness. TelosB is based on the Texas Instruments MSP430 microcontroller, Chipcon CC2420, IEEE 802.15.4-compliant radio, and USB, its power profile is almost one-tenth the consumption of previous mote platforms while providing greater performance and throughput. It enables experimentation with WSNs in both lab, testbed, and deployment settings eliminating programming and support boards. The maximum data rate of 802.15.4 is 250 kbps per frequency channel (16 channels available in the 2.4 GHz band), too low to perform real-time image streaming from the camera back to the server at hight enough quality and frame rate. Even with this limitation the data speed permits to to push computing out to the edge of the network sending only post-processed data (eg, low-dimensional features from an image) in real-time

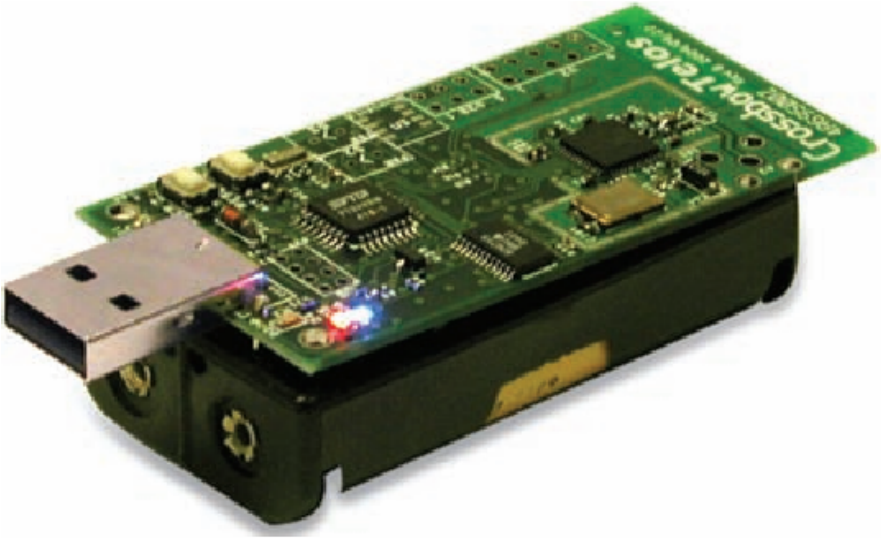


Figure 5.5: The TelosB mote

back to the centralized server and clients for further processing. If an event of interest occurs in the network, a query can be sent for the relevant image sequence to be compressed and sent back to the server over a slightly longer period of time. Since TinyOS [32] is the operating system running on the mote, it is relatively easy to substitute different standard routing protocols to suite the particular needs of an application. For instance, the real-time requirements of video-surveillance imply that typical communication does not need to run over a reliable transport protocol. TinyOS is a componentized operating system suitable for research in wireless embedded systems for sensor networks where a lot of tiny, low-power nodes, each of which execute concurrent, reactive programs must operate with severe memory and power constraints. TinyOS meets the sensor network challenges of limited resources, low-power operation and event-centric concurrent applications and has become the platform of choice for sensor network research. TinyOS has a component-based programming model, codified by the NesC language [33], a dialect of C. It is not an OS in the traditional sense; it is a programming framework for embedded systems and a set of components that enable building an application-specific OS into each application. The composition of components and whole program analysis allows researchers to work on the system at any level (e.g., the details of link protocols up to the application semantics).

In TinyOS the hardware primitives, such as register access and module flags, are

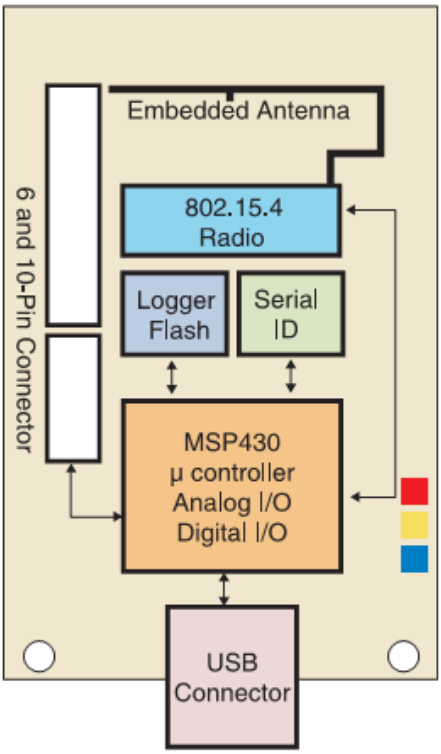


Figure 5.6: The TelosB architecture

exposed through a hardware presentation layer (HPL). A platform-dependent hardware abstraction layer (HAL) exposes hardware module functionality so that the full power of the hardware may be used. The HAL includes getting data from the ADC, setting a hardware Timer, or writing to external flash memory. The hardware independent layer (HIL) exposes a subset of platform capabilities that are available to system services. The HPL/HAL/HIL is implemented in TinyOS for the TI MSP430 microcontrollers. On top of the HAL/HPL/HIL abstraction there is a platform independent radio stack (link protocol and physical layer access) for the CC2420 transceiver that provides register access to the radio device; the radio stack then acts as a library that uses these primitives to the control the radio.

5.6 The application development

The CITRIC camera permits to develop a new set of in-network information processing and networking techniques for heterogeneous sensor networks. Since wireless camera networks performing in-network processing are relatively new, it is important to balance performance with ease of development of in-network computer vision algorithms to enable a wider base of applications. Modularity is a key tenet of this design, reflected in the separation of the image processing and networking hardware on the CITRIC mote and in the separation of functions in client/server back-end software architecture for the entire CITRIC system. The CITRIC camera platform is equipped with a general purpose processor running embedded Linux, an open source operative system wide-employed on a lot of embedded system platforms; this permits rapid prototyping and ease of programming to code algorithms and application program in C/C++ and the use of standard libraries like OpenCV [34]. With the camera platform is available an essential Software Development Kit (SDK) that includes the standard gcc C/C++ compiler [35] and an API library with a set of functions for handling the image sensor, the wireless communication and the jpeg image compression and decompression.

5.7 The Energy consumption

The camera mote is designed to be powered by direct connection with four AA batteries or through the USB port when the camera is connected to a personal computer for programming or data retrieval operations. The reported power consumptions [26] concern the four AA batteries power supply case. Three different test results are reported; IDLE test: power consumption of the camera platform board alone running Linux with no active processes. In this case, on average the adsorbed power is 428 - 478 mW; IDLE+MOTE test: the same IDLE measurement but with the mote attached, although no data sent to the mote. In this test the mote was running an application that waits to receive any packets from the camera board and transmits over the radio. In this operating condition, on average, the adsorbed power is 527 - 594 mW; FULL+MOTE test: power consumption of the camera mote running a typical background subtraction function. The test utilizes all the components of the camera mote by both running the CPU and using the mote to transmit the image coordinates of the foreground. At the processor speed 520MHz, the power consumption is 970mW. Assuming that the camera mote consumes about 1W and runs on batteries with 2700mAh capacity, the expected battery life is 16 hours with the camera mote working under continuous operation.

Chapter 6

Energy efficient software application on embedded smart cameras

6.1 Introduction

In this chapter, the methodologies that aim to increase the energy-efficiency and battery-life of an embedded smart camera have been introduced.

The research on wireless sensor networks (WSNs) has witnessed an enormous growth in the last years, due to the everdecreasing costs of equipments, such as sensors, communication devices, batteries and microprocessors. Moreover, the undoubted advances in digital design and microelectronics made the goal of a powerful yet small-sized embedded device suitable. Among them, embedded smart cameras are devices which include in a stand-alone unit, an image capturing device, a processor, memory and communication interface. Despite the clear advantages of a mobile, networked solution for several emerging applications (including, but not limited to, security, environmental monitoring, urban control and planning, e.g. for traffic monitoring, etc.), wireless and battery-powered devices introduce many additional challenges since they have limited resources, such as energy, computational power, storage and communication bandwidth.

For this reason, differently from PC-based solutions, where the accuracy of algorithms is, most of the times, the driving force, here attention must be paid also to efficiency and power consumption. Embedded smart cameras provide a lot of flexibility in terms of camera quantities and placement, however video processing algorithms consume considerable amount of energy, and since battery life is limited, it is

essential to have lightweight algorithms and methodologies to increase the energy-efficiency of each camera. The requirements and constraints to be considered when designing video processing algorithms on battery-powered embedded smart cameras can be summarized as follows:

- *best trade-off between efficiency and accuracy*: this is a very common requirement, but in the case of embedded systems, efficiency also translates to low battery consumption, which not always coincide; the development of an accurate algorithm which also reduces the amount of consumed energy on the selected device is the main objective of this chapter;
- *saving of memory resources*: the best efficient and accurate algorithm may make use of an extensive storage structure for its data, which can be unfeasible in the embedded device due to limited memory resources; therefore, the design of the algorithm should also try to limit memory allocation and usage (which is often related to energy consumption as well);
- *adaptiveness*: even more than in the case of standard PC-based solutions, attention must be paid to develop an algorithm that adapts to different conditions by, for instance, switching off certain computational tasks whenever they are not necessary, to both save power and reduce the required bandwidth.

Among the many different video processing algorithms to be implemented on an embedded smart camera, moving object detection and tracking are relevant tasks, often related to surveillance and security applications. In this chapter an object tracking algorithm is proposed which combines the feedback-based tracking method presented in [28] with hardware-level operations (specifically hardware-level down-sampling and cropping), and is presented a comprehensive tracking analysis. Differently from our previous proposal [36], in this paper is also considered the case of multiple objects to be tracked. This circumstance poses the additional challenge of dealing with a way to crop (and track) more than an object at a time. Two basic approaches will be compared: one in which the cropping will try to incorporate into a single cropped window all the objects to be tracked; the other in which object cropping will alternate among objects. Regardless of the chosen approach, cropping will be performed at hardware-level with varying window sizes. The proposed solution modifies the low-level kernel-driver structure of the camera, which resets the circular FIFO buffer of the camera. The goal is to further increase the energy-efficiency and the battery-life by hardware-level operations when performing object detection and tracking.

6.2 Object detection and tracking

The goal of this chapter is to demonstrate how to increase the energy-efficiency and the battery-life when performing object detection and tracking, by hardware-level operations. Traditional tracking systems, as depicted in Figure 6.1, perform foreground object detection and tracking at each frame independently, and in a sequential manner. This will henceforth be referred to as the *sequential* method. Cesares and Velipasalar have presented a software-based feedback method [28], which is a lightweight foreground object detection and tracking algorithm suitable for embedded platforms. This will henceforth be referred to as the *software-based feedback* method. As shown in Figure 6.2, in this method, feedback from the tracking stage is used to determine search regions for the following frame, and to perform detection and tracking only in those regions instead of the whole frame. They have shown that this provides significant savings in processing time, and thus increases idle state durations of cameras to improve the battery-life.

1. *hardware level operations*: rather than performing down-sampling and image cropping at the main microprocessor on the camera board, these operations are performed at the micro-controller of the OV9655;

2. *hardware-based feedback*: it is first estimated the location of the target in the next frame, form a search region around it and then the next frame is cropped by hardware using the HREF and VSYNC signals at the micro-controller of the OV9655, and detection and tracking are performed only in the cropped search region.

Performing these functions at hardware level provides multiple advantages including savings in processing time and significant decrease in energy consumption. The amount of data, which is moved from the image sensor to the main memory at each frame, is greatly reduced. This, in turn, leads to significant savings in energy consumption thanks to the better use of the memory controller and the memory resources and not occupying the main microprocessor with performing image down-sampling and cropping at software-level. The comparison is done by performing down-sampling and cropping by software using the CITRIC API libraries (see section 5.6), and by hardware using the micro-controller of the image sensor. The presented experimental results will show the savings in processing time and energy consumption, and the gain in the battery-life of the CITRIC camera when using the feedback from the tracking stage, and performing down-sampling and cropping operations at hardware-level. This new method is depicted in Figure 6.3 and is referred to as the *feedback hardware-level based* method.

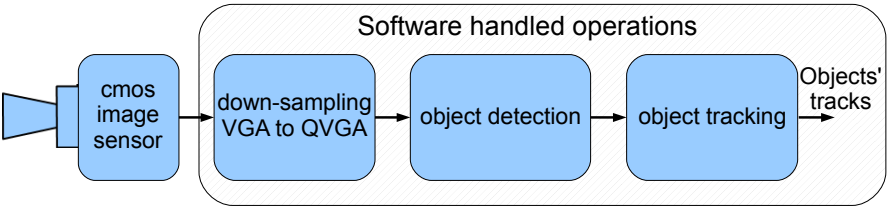


Figure 6.1: Sequential software-level based method: traditional tracking system performs foreground object detection and tracking at each frame independently, and in a sequential manner. All the computational tasks are performed by software at the main microprocessor camera board level, down-sampling is implemented using the CTRIC API library and whole frames are processed each time.

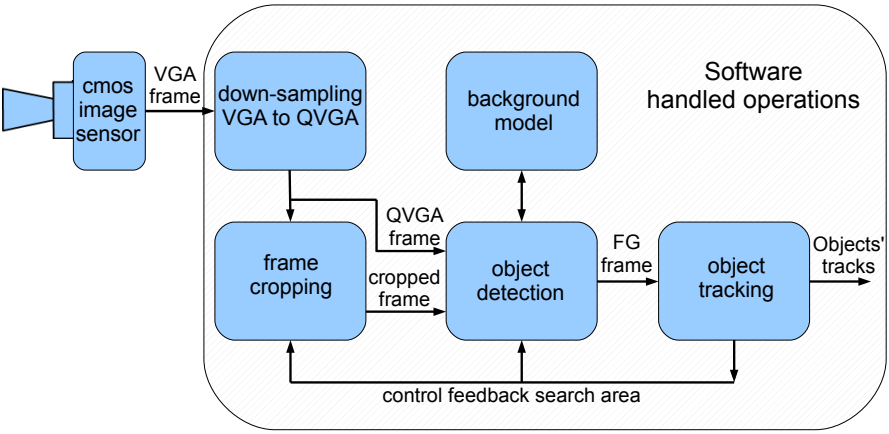


Figure 6.2: Feedback software-level based method: this method uses feedback from the tracking stage of the algorithm in the foreground object detection, instead of performing detection and tracking independently.

6.3 Frame Capture Operation

On the CITRIC camera platform, the QCI operates in 8-bit YCbCr 4:2:2 Master Parallel mode. It operates in a master mode in which the image sensor provides the line and frame synchronization signals. The line synchronization signal is commonly referred to as HREF (Horizontal REFerence) or line valid and the frame synchronization signal is commonly referred to as VSYNC (Vertical SYNChronization) or frame valid. The sensor can be programmed for exposure, frame rate, and additional parameters. The MCLK signal output for the sensor is programmable, it is the master

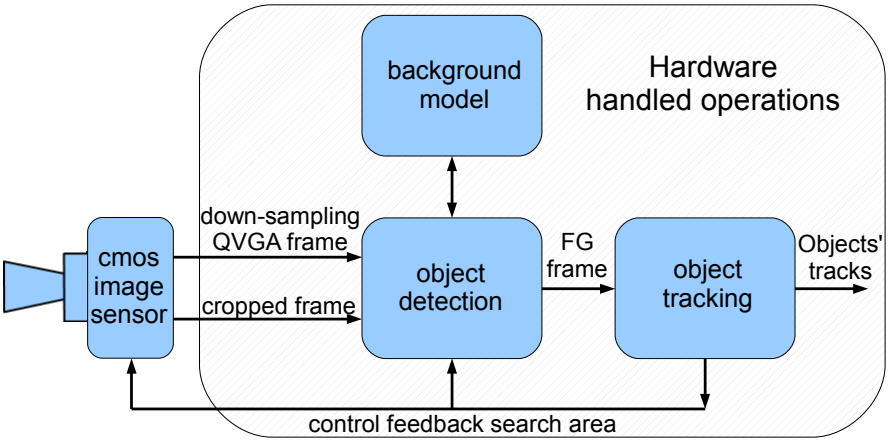


Figure 6.3: Feedback hardware-level based method: the feedback hardware-level based method combines the feedback-based tracking with hardware level operations. Rather than performing image down-sampling and cropping at the main microprocessor on the camera board, these operations are performed at the microcontroller of the image sensor.

clock provided by the Quick Capture Interface to the image sensor. The Pixel Clock signal, PCLK, provided by the image sensor, is derived from the MCLK and is used to perform all the frame transfer operations. Among the different 8-bit output for-

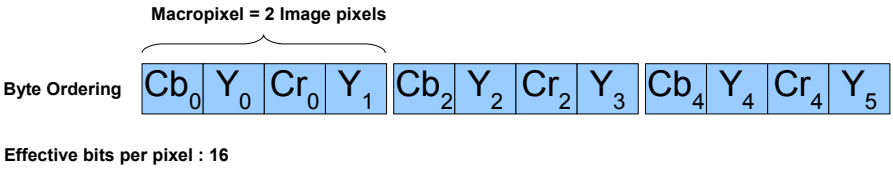


Figure 6.4: Image sub-sampling schema.

mats supported by the image sensor, the CITRIC camera uses YCbCr 4:2:2 format, where 16 bits per pixel are moved in two clock cycles. In the YCbCr 4:2:2 format the signal is divided into a luminance component (Y) and two color difference components (Cb and Cr, called chroma). On the CITRIC camera, as reported in Figure 6.4, the YCbCr 4:2:2 image is sampled using the CbYCrY schema. The behavior of the OV9655 sensor is controlled by a set of 208, 8-bit registers which are accessible by the I2C interface of the main microprocessor at the camera board level. This register set takes factory default configuration values at reset or at power-up. When required,

these registers can be singly loaded with the appropriate configuration values.

The CITRIC camera is equipped with the embedded Linux operative system in which a custom device driver provide some I/O Control (IOCTL) services in order to handle the image sensor, especially to control the activities of the QCI and the DMA engine. Despite the availability of the device driver, the image sensor configuration task is in charge at the API library of the SDK provided with the CITRIC camera. The API functions work in user-space and they are used to access the single configuration register even if they are not integral part of the image sensor device driver. These functions have to be used at the application program level in order to control the image sensor. It is important to remark how, to have the full control of both the capture device configuration and the frame capture operation, the IOCTLs services provided by the custom device driver and the appropriates API library functions have to be used. Tab. 6.1 reports two sets of values for the configuration register set of the image sensor: (i) the original register configuration-values used for the VarioPixel, 30 fps, VGA operating mode available in the API library of original CITRIC camera, and (ii) the new register configuration-values used to perform the size scaling down to QVGA by hardware at the image sensor level. Only the registers whose value are different from the factory default configuration are reported in Tab. 6.1. The API function used to initialize the camera board loads the configuration register set with the appropriate values reported in Tab. 6.1.

6.3.1 Data transfer

The frame synchronization signals VSYNC and HREF and the pixel clock PCLK plays a crucial role in the image data transfer from the image sensor to the camera platform. They control the finite state machine (FSM) described in this section. This FSM has in charge the acquisition of the data from the sensor. The Figures 6.5, and 6.6 show the frame synchronization signals timing for the original VGA frame. They have been obtained by reverse engineering operation [1] on the image sensor.

The Figure 6.7 is a reconstruction of the timing diagram for grabbing a frame. The image sensor activates the signal VSYNC to indicate that a new frame is about to occur and then activate the HREF signal each time a data line is available. The assertion of this signal indicates that pixel data are available at rising edges of the pixel clock. The HREF signal is active during the whole active line for a number of clock cycles double the number of pixels per line provided by the image sensor. It is important to remark that the pixel clock signal, provided by the image sensor, is always available even when no pixel data are ready on the digital video port. The Figure 6.8 reports the details of the data acquisition. There are shown the timing of

[1] The goal of the reverse engineering was to gain a deeper understanding of the fareme transfer operation. This has been done using a dual channels digital oscilloscope in order to collect the timing diagrams of the frame synchronization signals, VSYNC, HREF, PCLK. The timing information have been collected by connecting the oscilloscope to the aforementioned signals and performing the signals acquisition during the frame grabbing operations.

OV9655 Register values								
Addr	**Value (VGA)**	**Value (QVGA)**	**Addr**	**Value (VGA)**	**Value (QVGA)**	**Addr**	**Value (VGA)**	**Value (QVGA)**
0x00	0x00	0x00	0x4C	0x7F	0x7F	0x84	0x95	0xAC
0x01	0x80	0x80	0x4D	0x7F	0x7F	0x85	0xA6	0xB8
0x02	0x80	0x80	0x4E	0x7F	0x7F	0x86	0xB5	0xC3
0x03	0x12	0x02	0x4F	0x90	0x98	0x87	0xCC	0xD6
0x04	0x03	0x03	0x50	0x89	0x98	0x88	0xDD	0xE6
0x0B	0x57	0x57	0x51	0x07	0x00	0x89	0xED	0xF2
0x0E	0x61	0x61	0x52	0x0F	0x28	0x8A	0x03	0x24
0x0F	0x42	0x40	0x53	0x93	0x70	0x8C	0x89	0x80
0x10	0xF0	0xF0	0x54	0xA1	0x98	0x90	0x7D	0x7D
0x11	0x01	0x01	0x55	0x00	0x00	0x91	0x7B	0x7B
0x12	0x62	0x62	0x56	0x40	0x40	0x9D	0x04	0x02
0x13	0xE7	0xC7	0x57	0x80	0x80	0x9E	0x04	0x02
0x14	0x2A	0x3A	0x58	0x9E	0x1A	0x9F	0x7A	0x7A
0x16	0x24	0x24	0x59	0xCD	0x85	0xA0	0x79	0x79
0x17	0x16	0x18	0x5A	0xB9	0xA9	0xA1	0x40	0x40
0x18	0x02	0x04	0x5B	0xDF	0x64	0xA2	0x96	0x96
0x19	0x01	0x01	0x5C	0x76	0x84	0xA3	0x7D	0x7D
0x1A	0x3D	0x81	0x5D	0x4D	0x53	0xA4	0x50	0x50
0x1E	0x04	0x04	0x5E	0x11	0x0E	0xA5	0x68	0x68
0x24	0x40	0x3C	0x5F	0xF0	0xF0	0xA6	0x4A	0x4A
0x25	0x30	0x36	0x60	0xF0	0xF0	0xA8	0xC1	0xC1
0x26	0x82	0x72	0x61	0xF0	0xF0	0xA9	0xEF	0xEF
0x27	0x08	0x08	0x62	0x00	0x00	0xAA	0x92	0x92
0x28	0x08	0x08	0x63	0x1A	0x00	0xAB	0x04	0x04
0x29	0x15	0x15	0x64	0x04	0x02	0xAC	0x80	0x80
0x2A	0x00	0x00	0x65	0x00	0x20	0xAD	0x80	0x80
0x2B	0x00	0x00	0x66	0x05	0x00	0xAE	0x80	0x80
0x2C	0x08	0x08	0x69	0x0A	0x0A	0xAF	0x80	0x80
0x32	0xFF	0x24	0x6A	0x02	0x02	0xB2	0xF2	0xF2
0x33	0x00	0x00	0x6B	0x4A	0x5A	0xB3	0x20	0x20
0x34	0x3F	0x3F	0x6C	0x00	0x04	0xB4	0x26	0x20
0x35	0x00	0x00	0x6D	0x55	0x55	0xB5	0x00	0x00
0x36	0xFA	0x3A	0x6E	0x11	0x00	0xB6	0xAF	0xAF
0x38	0x72	0x72	0x6F	0x9E	0x9D	0xBB	0xAE	0xAE
0x39	0x57	0x57	0x70	0x21	0x21	0xBC	0x7F	0x7F
0x3A	0x0C	0x8C	0x71	0x78	0x78	0xBD	0x7F	0x7F
0x3B	0x44	0x04	0x72	0x00	0x11	0xBE	0x7F	0x7F
0x3C	0x0C	0x0C	0x73	0x00	0x01	0xBF	0x7F	0x7F
0x3D	0x99	0x99	0x74	0x3A	0x10	0xC0	0xAA	0xAA
0x3E	0x0C	0x0E	0x75	0x35	0x10	0xC1	0xC0	0xC0
0x3F	0xC1	0xC1	0x76	0x01	0x01	0xC2	0x01	0x01
0x40	0xC0	0xC0	0x77	0x02	0x02	0xC3	0x4E	0x4E
0x41	0x00	0x01	0x7A	0x19	0x12	0xC5	0x03	0x03
0x42	0xC0	0xC0	0x7B	0x06	0x08	0xC6	0x05	0x05
0x43	0x14	0x0A	0x7C	0x0E	0x16	0xC7	0x80	0x81
0x44	0xF0	0xF0	0x7D	0x20	0x30	0xC9	0xE0	0xE0
0x45	0x3E	0x46	0x7E	0x42	0x5E	0xCA	0xE8	0xE8
0x46	0x5F	0x62	0x7F	0x52	0x72	0xCB	0xF0	0xF0
0x47	0x2A	0x2A	0x80	0x62	0x82	0xCC	0xD8	0xD8
0x48	0x3F	0x3C	0x81	0x70	0x8E	0xCD	0x93	0x93
0x4A	0xFC	0xFC	0x82	0x7F	0x9A	-	-	-
0x4B	0xFC	0xFC	0x83	0x8B	0xA4	-	-	-

Table 6.1: VGA and QVGA image sensor register set values.

the signals HREF, and PCLK at the beginning of a line for the VGA image size. The lower side of this Figure is the zoomed portion of what is represented in the upper side. As it will be shown in the following section, changing the width of the active portion of the HREF signal (The amount of time the signal remains high) and its activation time are strictly related with the selective acquisition of a sub-region from each line of the image. In a similar way, the possibility to drop some HREF pulses is related with the selective acquisition of a sub-region of the image starting from a predefined row.

The image data transfer from the image sensor to the main memory of the CIT-

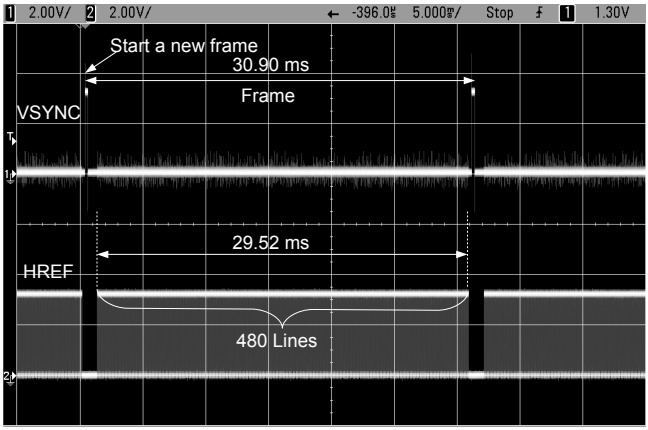


Figure 6.5: VGA frame timing.

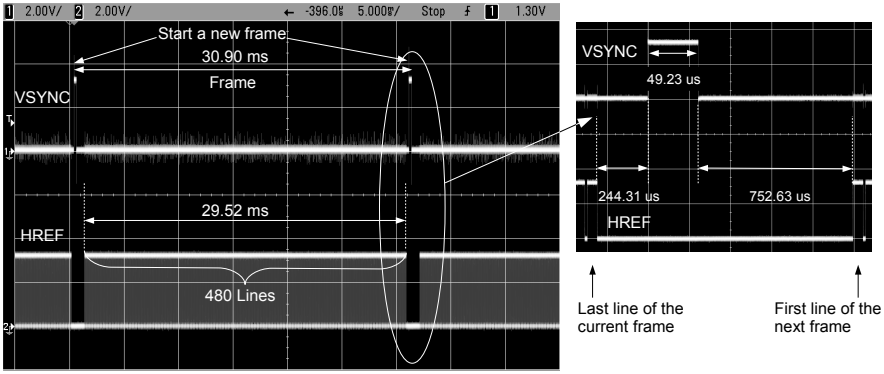


Figure 6.6: The beginning and the end of a VGA frame.

RIC camera board is handled by the Quick Capture interface. This interface operates in Master Parallel Mode and its behavior can be described by the Master Modes State Diagram reported in Figure 6.9 which rapresents the FSM that handles the data transfer. This FSM is full software-programmable by direct access at the QCI configuration register set. The acquisition of data from the sensor is initiated by transitions based on the state of the HREF and VSYNC signals, which are generated internally by the sensor. The assertion of the VSYNC signal indicates that a frame read-out is about to occur. The assertion of HREF causes the quick capture interface moving on to the active data capture state where a valid data line is captured under the control of the PCLK signal as depicted in Fig 6.7. To support any required varia-

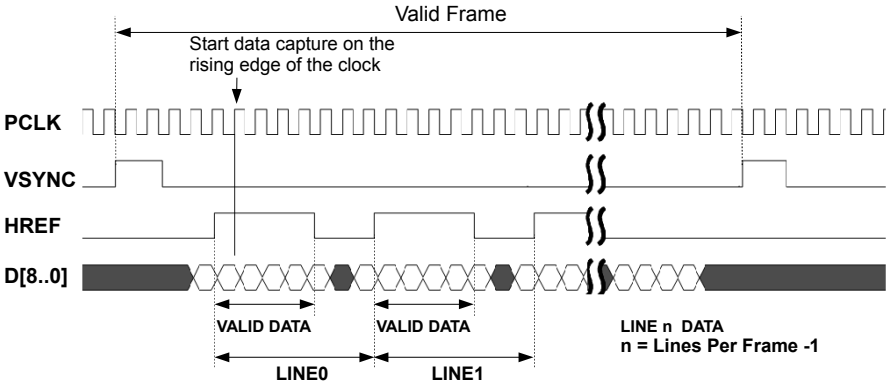


Figure 6.7: Timing diagram for grabbing a frame using the Quick Capture Interface.

tion in delay from signal transitions to valid data, several programmable wait states are introduced, as illustrated in the Figure 6.9. The assertion of VSYNC causes the state machines transition to the Begin Frame Wait state, where it then waits for the number of line valid transitions (HREF) configured by BFW. Once this BFW count has elapsed, another HREF assertion brings the state machine to the Begin Line Wait state. In this state, the master mode state machine waits for BLW pixel clock cycles before moving on to the active data capture state where (Pixels-Per-Line) PPL pixels are acquired. Counters are maintained for both the PPL and the Lines-Per-Frame (LPF).

The quick capture interface has to be programmed with the image sensor specific PPL and LPF values. Wait states are skipped if the corresponding delay values (BFW, BLW) are zero.

6.3.2 FIFO Operation

The capture interface has three separate FIFOs to act as temporary storage for the captured image data. The channel 0 FIFO has a storage capacity of 128 bytes, and each of the 16 FIFO entries is 8 bytes wide. The channel 1 and channel 2 FIFOs have each a storage capacity of 64 bytes. Each of their 8 entries is 8 bytes wide. The planarized YCbCr mode employed in the CITRIC camera uses all three channel input FIFOS. The channel 0 FIFO stores the Y data component, the channel 1 FIFO stores the Cb component, and the channel 2 FIFO stores the Cr component. Parallel data from the sensor are captured and re-arranged based on input data width and number of bits per pixel. The timing references for capture are provided by the HREF and VSYNC inputs. When an End of Frame is encountered during data capture, the remaining bits in the 8-byte wide FIFO location (after writing the last data) are padded with zeros. For example, if the last sample of a frame occupies byte 0 of the

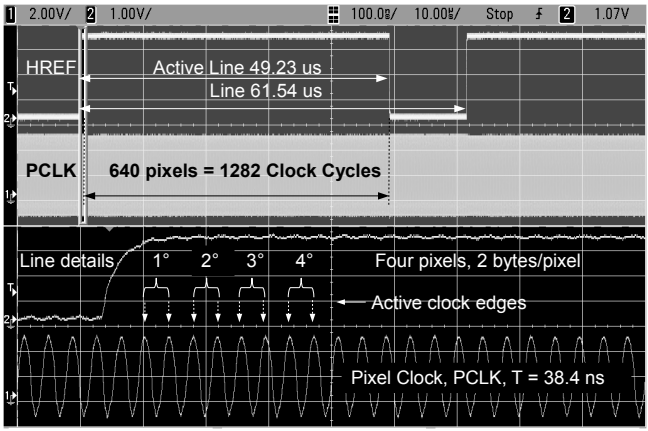


Figure 6.8: Details on a VGA row.

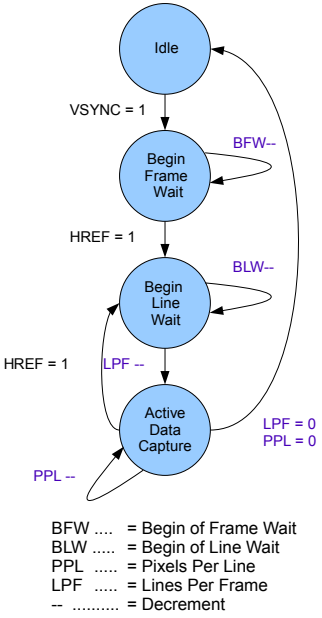


Figure 6.9: Master Modes State Diagram.

8-byte FIFO entry, bytes 1 through 7 are written with zeros. The first valid data of the next frame is loaded to the next 8-byte wide FIFO entry. The capture-interface data output FIFOs (eight bytes wide) are padded with zeros at the end of a frame if the

captured frame data does not end at an 8-byte boundary. The DMA controller must be programmed to read the entire data including any zero padding. The programmed DMA length must be a multiple of 8 bytes. Similarly, if the processor is reading data from the capture-interface output FIFOs (1,2 or 4 bytes at a time), it must read the entire FIFO entry including padded zeros (if any, at end of frame) before moving on to the next frame.

The data transfer is performed when the FIFO threshold levels are reached. The processor can perform data transfers from the quick capture interface FIFOs in response to an interrupt generated by the FIFO's threshold level logic, or more effectively, the quick capture interface can activate the DMA controller which empties the FIFOs by reading the corresponding Receive Buffer registers fully by hardware, involving the CPU only when a whole frame has been transferred. The channel 0 FIFO's threshold level is programmable. Threshold levels of the channel 1 and channel 2 FIFOs are fixed at 32 bytes. The number of bytes in the capture-interface FIFOs corresponding to a captured frame is always a multiple of eight bytes.

6.3.3 DMA Data Transfers from FIFOs

The quick capture interface has three DMA receive-data service requests, one for each of the three input-channel FIFOs. When a FIFO issues a DMA service request, the DMA controller responds by transferring a burst of data from the FIFO to the destination address specified in the DMA descriptor. The DMA Source Address register must point to the corresponding Receive Buffer registers x (x indicates the FIFO channel, 0, 1 or 2), depending on which, one of the three DMA services is being employed. The DMA descriptors must be programmed to transfer all bytes in a frame. In planarized YCbCr mode of operation, program individual DMA channels to transfer the total number of bytes per frame for each of the components.

6.4 Implementation of Down Sampling

Down-sampling task is depicted in Figure 6.10, and is achieved by obtaining the desired frame size (which will be QVGA, 320X240) through the setting of the appproppriate number of pixels for line and lines for frame at the image sensor level on the configuration parameters VSTART, VSTOP, and HSTART, HSTOP (see tables 6.2 and 6.3). Moreover, it is necessary to select the zoom and scaling functions and to set the horizontal and vertical scaling down coefficients by accessing the image sensor register set as shown in Tab. 6.4. By appropriate settings of the registers, in particular the Pixel Output Index (POIDX), the micro-controller in the OV9655 performs the scaling down by averaging the necessary pixels to get the desired frame size as depicted in the Figure 6.10. Further timing details regarding the QVGA frame are shown in the Figures 6.11, 6.12, 6.13.

Cropping related OV9655 registers			
Reg	Add	Bit	Description
HSTRT	0x17	Bit[7:0]	Horizontal Frame **column start** (HREF signal), bit[10:3]
HSTOP	0x18	Bit[7:0]	Horizontal Frame **column end** (HREF signal), bit[10:3]
HREF	0x32	Bit[5:3]	Horizontal Frame **column end** (HREF signal), bit[2:0]
		Bit[2:0]	Horizontal Frame **column start** (HREF signal), bit[2:0]

Table 6.2: The HSTART and HSTOP parameters control the active portion of the HREF signal. Their values are 11-bits wide and are stored in to three 8-bits configuration registers named HSTRT, HSTOP and HREF. The value of the HSTART parameter is obtained by the composition of the HSTRT register with the HREF[2:0] register. Parameter HSTART = [HSTRT:HREF[2:0]]. The value of the HSTOP parameter is obtained by the composition of the HSTOP register and HREF[5:3] register. Parameter HSTOP = [HSTOP:HREF[5:3]].

Cropping related OV9655 registers			
Reg	Add	Bit	Description
VSTRT	0x19	Bit[7:0]	Vertical Frame **row start** (VSYNC signal), bit[10:3]
VSTOP	0x1A	Bit[7:0]	Vertical Frame **row end** (VSYNC signal), bit[10:3]
VREF	0x03	Bit[5:3]	Vertical Frame **row end** (VSYNC signal), bit[2:0]
		Bit[2:0]	Vertical Frame **row start** (VSYNC signal), bit[2:0]

Table 6.3: The VSTART and VSTOP parameters control the picture rows interval output by the image sensor. Their values are 11-bits wide and are stored in to three 8-bits configuration registers named VSTRT, VSTOP and VREF. The value of the VSTART parameter is obtained by the composition of the VSTRT register with the VREF[2:0] register. Parameter VSTART = [VSTRT:VREF[2:0]]. The value of the VSTOP parameter is obtained by the composition of the VSTOP register and VREF[5:3] register. Parameter VSTOP = [VSTOP:VREF[5:3]].

Down-sampling related OV9655 registers					
Reg	Add.	Description	Bit	VGA	QVGA
COM14	0x3E	Zoom function ON/OFF selection	Bit[1]	OFF	ON
COM16	0x41	Scaling down ON/OFF selection	Bit[0]	OFF	ON
POIDX	0x72	Vertical pixel output option	Bit[6]	Use pixel average data	Use pixel average data
		Vertical pixel output index	Bit[5:4]	Normal	Output 1 line every 2 line
		Horizontal pixel output index	Bit[1:0]	Normal	Output 1 pixel every 2 pixel
PCKDV	0x73	Pixel clock output frequency adjust.	Bit[3:0]	0x00	Default, 0x01
XINDX	0x74	Horizontal scaling down coefficients	Bit[7:0]	Default, 0x3A	0x10
YINDX	0x75	Vertical scaling down coefficients	Bit[7:0]	Default, 0x35	0x10
COM24	0xC7	Pixel clock frequency selection	Bit[2:0]	Default, 0x80	0x81

Table 6.4: Down-sampling related OV9655 registers.

6.5 Implementation of Cropping

The image cropping is the selection of an area inside the whole image, as shown in Figure 6.10. This area is named “cropped window” and characterized by its position, width and height.

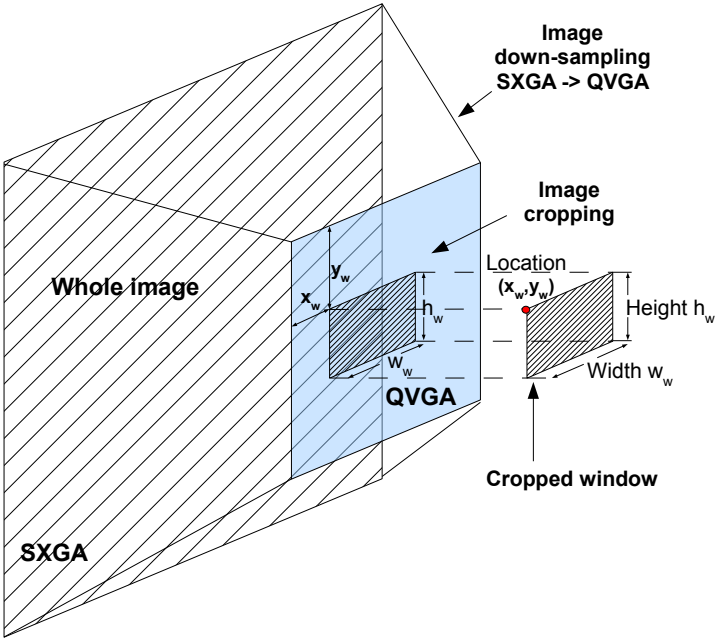


Figure 6.10: Image down-sampling and cropping.

The position is represented by the pixel coordinates of its upper left corner inside the whole image, it is controlled by working on the image sensor configuration while the size of the cropped window is controlled by reconfiguring the QCI. The synchronization signal VSYNC indicates that a new frame is about to occur, then the first cycle of the HREF signal that happens is related with the first data line that is acquired from the image sensor. So changing the Yw origin of the cropped window means to drop the right number of lines at the beginning of each frame. This can be achieved by careful reconfiguring the image sensor in order to carries out the dropping of the desired number of HREF cycles at the beginning of each frame after the activation (pulsing) of the VSYNC as shows in the Figure 6.14. In a similar manner, in order to change the Xw origin of the cropped window it is necessary to reshape the active portion of the HREF signal changing its activation time. Doing that, the right number of pixels at the beginning of each line will be dropped. Summarizing, setting the position of the cropped window means: for the Yw, the reconfiguration of

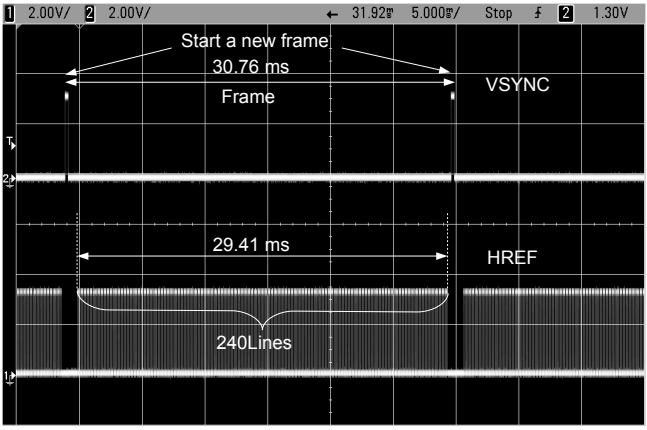


Figure 6.11: Timing details on the QVGA frame obtained performing downsampling by hardware operating at the image sensor level.

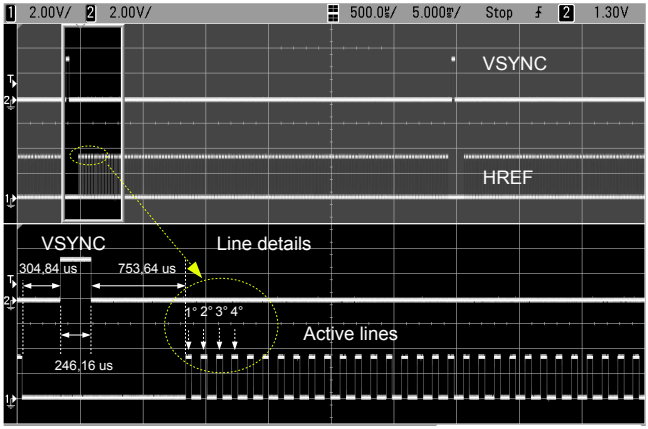


Figure 6.12: Details on the timing of the beginning of a QVGA frame. The lower side of the picture shows the zoomed portion of the signals highlighted in the upper side.

the sensor in order to drop the appropriate number of cycles of the HREF signal at the beginning of each frame so to skip the right number of lines; for the Xw origin, the reshape of the HREF signal so to drop the right number of pixels at the beginning of each line.

The reverse enginnering activities performed on the image sensor have permit-

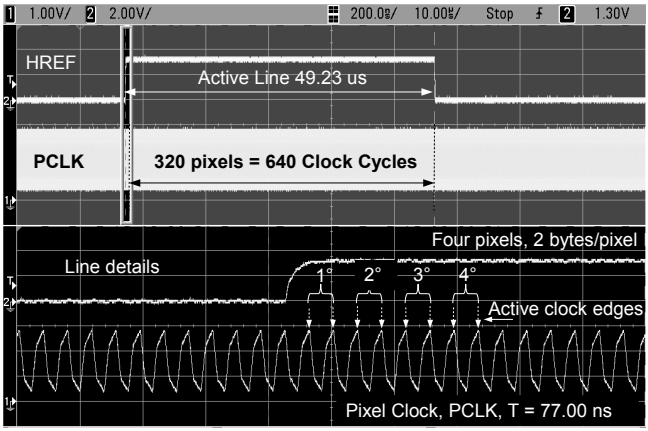


Figure 6.13: Timing of the pixel clock and HREF signal for a QVGA frame.

ted to show detailed information regarding the timing of the frame synchronization signals and the pixel clock signal PCLK. The Figure 6.15 shows the changes of the HREF signal when cropping on the x axis. In the upper side of the figure is shown the active portion of the HREF signal while in the lower side is reported the same signal when: a) no pixels, (b) one pixel, or (c) two pixels are cropped (dropped) at the beginning of the lines. In similar fashion the Figure 6.16 shows the frame synchronization signals for a whole QVGA frame in a), details on the lines with the PCLK and HREF signals are reported when no pixels b), ten pixels c), or twenty pixels d) are cropped along the x axis. Regarding the cropping along the x axis, the row cropping, the Figure 6.17 shows the cropping along the y axis when, a) no lines, b) ten lines, c) twenty lines, and d) thirty lines are dropped at the beginning of each frame.

The origin of the cropped window is set with the aforementioned operations. The setting of the size of the cropped window is achieved operating at the QCI level, changing the values of the FSM parameters PPL (Pixels Per Line), and LPF (Line Per Frame). As described in the section 6.3.1, when the FSM moves to the Active Data Capture State an amount of pixels equal to PPL are acquired, then it moves back to the Begin Line Wait State where it waits for a new line (new HREF assertion). Here, as depicted by the bottom side of the Figure 6.14, any extra pixel coming from the image sensor due to still-in-porgress line transfer, is dropped. As depicted in the top right side of the Figure 6.14, a similar behavior occurs when the LPF counter reachers zero; the FSM moves to the Idle state where it waits for the assertion of the VSYNC signal, which indicates that a new frame is almost occuring, any extra line coming from the still in porgress frame is dropped.

The access at the register set of the image sensor (as described in the section 6.3)

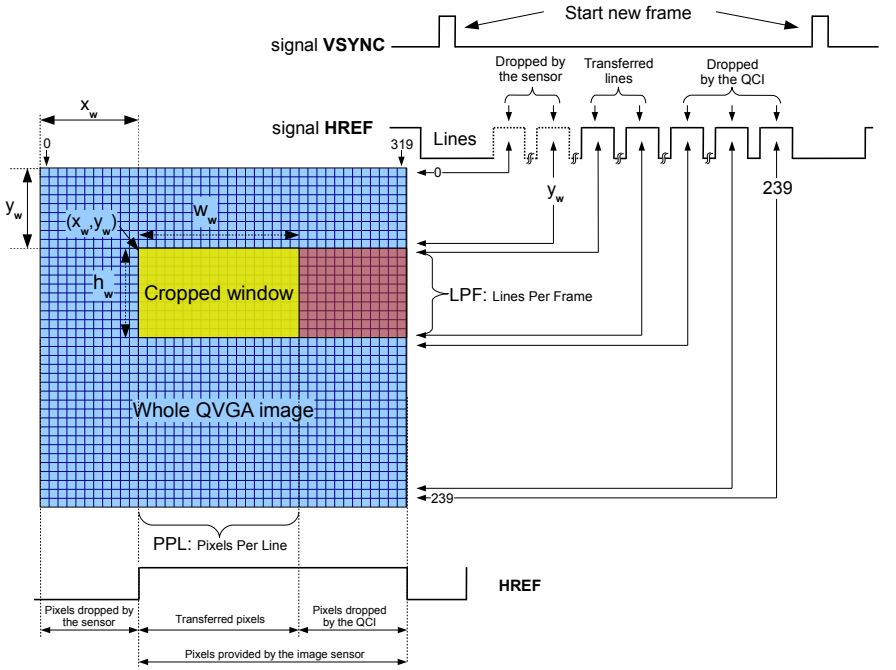


Figure 6.14: This picture summarize the operations to perform the image cropping implementation. In the right upper side are shown the frame synchronization signals and the relationship among the HREF cycles and the image rows. In the bottom side of the picture is shown the relation between one cycle of the HREF signal and the pixels of a line. The location of the cropped window is set by dropping pixels at the beginning of each line and lines at the beginning of each new frame. These operations are performed at the image sensor level. The window sizing is set at the QCI level by careful setting the values of the PPL (Pixel Per Line) and LPF (Line Per Frame) FSM parameters.

is in charge at the API library functions available with the CITRIC camera SDK. Instead the control of the quick capture interface and the DMA engine is more complex, since it requires a specific device driver. Typically the possibility to sample a subsection of an image and shrink it, is provided by Video4Linux API [37], but the newest kernel version (at the time of writing this thesis) [38], Linux-3.1.1, does not support the device driver for the CITRIC camera platform and for any ARM PXA270 based platforms equipped with the OV9655 image sensor.

The number of the dropped lines at the beginning and at the end of each frame, and the number of the dropped pixels at the beginning and at the end of each line

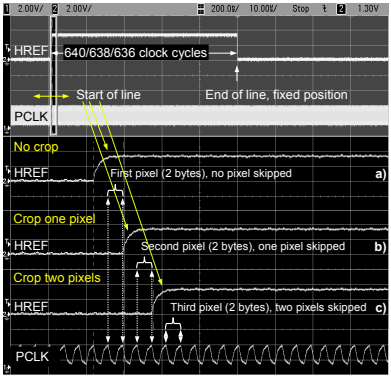


Figure 6.15: The picture shows the changes of the HREF signal when cropping is performed on the x axis. In the upper side of the figure is shown the active portion of the HREF signal, in the lower side is reported the same signal when: a) no pixels, (b) one pixel, or (c) two pixels are cropped (dropped) at the beginning of the lines.

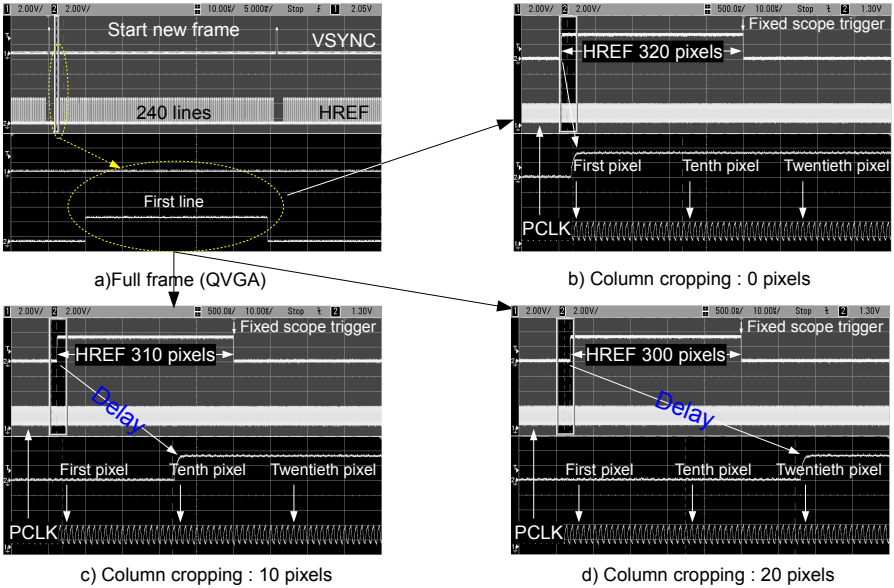
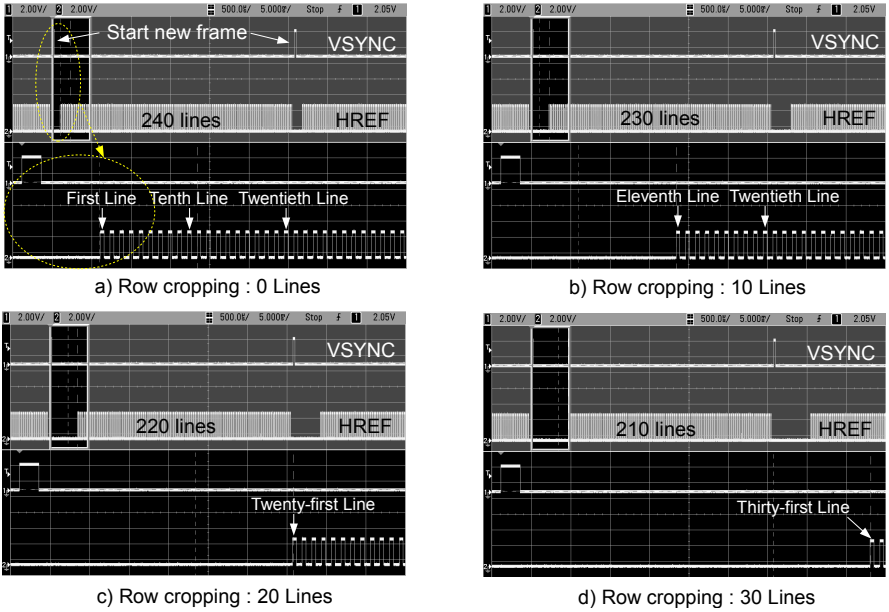


Figure 6.16: The picture shows the frame synchronization signals for a whole QVGA frame in a), details on the lines with the PCLK and HREF signals are reported when no pixels b), ten pixels c), or twenty pixels d) are cropped along the x axis.



a) Row cropping : 0 Lines

b) Row cropping : 10 Lines

c) Row cropping : 20 Lines

d) Row cropping : 30 Lines

Figure 6.17: In these pictures is shown the cropping along the y axis when, a) no lines, b) ten lines, c) twenty lines, and d) thirty lines are dropped at the beginning of each frame. In the upper side of each pictures are reported the timing for the frame synchronization signals for one frame, while in the lower side, the zoomed portion where details on the first lines of the frame are shown.

are controlled respectively by the 11-bit length parameters VSTART, VSTOP, and HSTART, HSTOP. These parameters are stored in a subset of six (8-bit long) registers of the image sensor register set as reported in the tables 6.2 and 6.3.

In the YCbCr 4:2:2 output format each pixel is represented with two 8-bit data bytes as depicted in Figure 6.4 and it needs two clock cycles (PCLK) to flow from the image sensor to the QCI. In order to set the location of the cropped window, (Xw, Yw), the values of the HSTART and VSTART parameters have to be changed. So, in order to shift the cropped window position one pixel to the right or one pixel to the left along the x direction, it is necessary to add or subtract two at the value of the parameter HSTART. In a similar way to get a movement along the y direction it is necessary to change the value of VSTART parameter. Increasing or decreasing the value of VSTART moves the position of the cropped window down or up. Vice versa, to set the width and height of the cropped window, it is necessary to reconfigure the QCI and the DMA engine using a service provided by the image sensor device driver. This is performed by an expressly created IOCTL so to perform this task quickly.

6.6 Esperimental Results

6.6.1 Savings in energy consumption

In order to show the advantages of the proposed appropach it is provided a quantitative comparison showing the advantages of performing hardware-level downsampling and cropping at the micro-controller of the OV9655 sensor for tracking purposes rather than processing whole frames and performing these tasks at software level on the main micro-processor of the camera board. It will present savings in energy consumption when performing hardware-level down-sampling and cropping, and using the feedback method for object detection and tracking. As stated in [28], the feedback method provides significant savings in processing time, and thus allows to increase idle state durations of cameras to increase the battery-life. As described in Section 6.2, in the feedback method, information from the tracking stage is used to determine search regions in the next frame so that detection and tracking can be performed only in these regions instead of the whole frame.

(a) (b)

Figure 6.18: Last QVGA frame captured while computing pixel displacement of tracked object (a), and Search regions cropped at hardware level (b).

Figures 6.18(a), and 6.18(b) show a sequence of frames in which a remote-controlled car is tracked. Figure 6.18(a) shows a QVGA frame grabbed during the tracking of the remote-controlled car. Whole frames are grabbed until the displacement of the target is computed from two consecutive frames. Then, the location of the target is estimated at the following frame. A search region of size 2w x 2h is formed around this location, where w and h are the width and height of the bounding box in the current frame, respectively. The details can be found in [28]. Then, the following frame is cropped to the search region at hardware level, and the detection and tracking are performed only in the cropped region as depicted in Figure 6.18(b). To show the movement of the car, and the changing cropped window, a small red circular reference point is highlighted on the cropped frame sequence. Before presenting the energy consumption analysis during feedback-based tracking combined

with hardware-level cropping, it will be first compared the following two scenarios on a single QVGA size frame to separately show the contribution of hardware-level down-sampling to the savings:

i. obtaining QVGA images with software-level down-sampling, and performing all processing (down-sampling, foreground object detection and tracking) on the main microprocessor of the camera board;

ii. performing down-sampling at hardwarelevel on the micro-controller of the OV9655 sensor, and performing foreground object detection and tracking at the main microprocessor.

The operating currents of the camera board while using these approaches are presented in Figure 6.19.

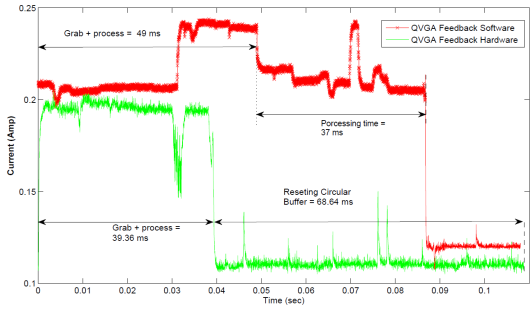


Figure 6.19: Operating currents when (i) obtaining QVGA images with software-level down-sampling, and performing all processing on the main microprocessor; (ii) performing down-sampling at hardwarelevel on the micro-controller of the OV9655 sensor, and performing foreground object detection and tracking at the main microprocessor.

Even though collaborating with the image sensor and hardware-level operations slightly prolong the processing time per frame by 22 ms, they considerably decrease the energy consumption of the camera by 27.38% as presented in Table 6.5.

	QVGA		
Method	Power (W)	Energy (mJ)	Gain (%)
Software-level down-sampling	1.0415	112.5	-
Hardware-level down-sampling	0.751	81.7	27.38%

Table 6.5: Energy Consumption when grabbing a QVGA frame at software-level versus hardware-level, and performing detection at the main microprocessor.

Now, savings in energy consumption when using the feedback method for object detection and tracking, and performimg hardware-level cropping are presented. The aforementioned scenarios analyzed for QVGA resolution, are now performed in a reduced search region cropped by software or hardware-level operations. The software-based feedback method [28] grabs a frame in VGA resolution, down-samples it and crops the search region all by software. On the other hand, the hardware-level method uses the capabilities of the OV9655 to down-sample, and then crop the search region. Having the search regions, foreground detection and tracking tasks are performed only in those regions at the main micro-processor of the CITRIC camera.

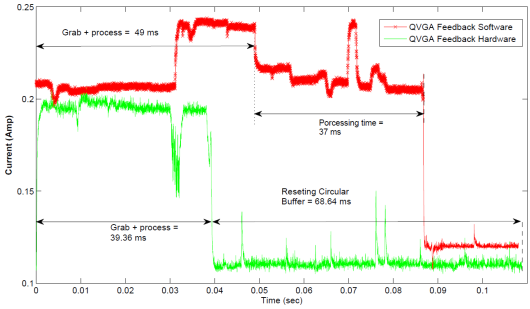


Figure 6.20: Operating currents when performing foreground object detection and tracking on cropped search regions obtained by softwareversus hardware-level cropping.

Figure 6.20 shows the operating current of the camera board when:

i. using the feedback method implemented entirely at software level;

ii. applying hardware-level operations for cropping and down-sampling.

Even though the processing time increases by 23 ms when cropping a search region of 100x100 pixels at hardware-level and processing it, using the hardware capabilities of the OV9655 provides 28.3% decrease in energy consumption of compared to software-level cropping and processing.

Different scenarios are summarized in Table 6.6 presenting the energy consumption and savings when processing a single frame. Figure 6.21 shows an experiment where a remote-controlled car is tracked continuously for 3 seconds, and the size of the cropped window is changed once every second. The energy consumption has been measured during this period. Figure 6.21 shows the operating current of the camera board for different scenarios during 1-second portion of this 3-second experiment. However, feedback method combined with hardware-level cropping provides 29.4% and 37% decrease in energy consumption compared to the software-based feedback method and sequential method, respectively.

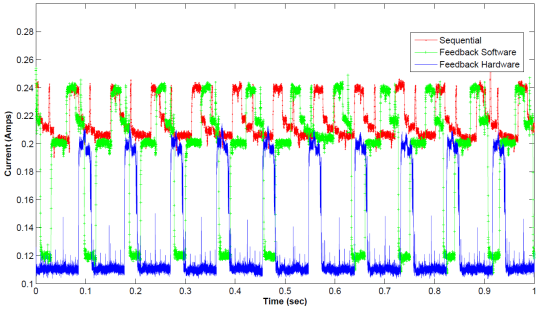


Figure 6.21: Operating currents when performing foreground object detection and tracking during 1-second time interval.

	100x100 Search Area		
Method	Power (W)	Energy (mJ)	Gain (%)
Sequential software-level	1.0415	112.5	-
Feedback software-level	1	92.23	18.02%
Feedback hardware-level	0.719	66.1	41.24%

Table 6.6: Energy Consumption when grabbing and cropping a search region (100x100) at software-versus hardware-level and performing detection at the main microprocessor.

Method	Power (W)	Energy (J)	Gain (%)
Sequential software-level	1.1422	3.4273	-
Feedback software-level	1.0203	3.0608	10.7%
Feedback hardware-level	0.7203	2.1609	37%

Table 6.7: Energy Consumption when performing detection and tracking during a 3-second time interval at software-versus hardware-level.

Table 6.7 summarizes the power and energy consumptions, and savings. When there are multiple objects in the scene, we need to form multiple search regions, and crop multiple windows. In this case, hardware-level cropping can still be performed for one window per frame, and different windows for different objects can be cropped at alternating frames.

6.6.2 Increase in battery-life

A projection of the battery-life of the camera has been done for the following scenarios:

i. Sequential method: performing downsampling at software-level, and detecting and tracking objects in the whole frame;

ii. Software-level feedback method: performing down-sampling and cropping at software-level, and detecting and tracking objects in smaller search regions;

iii. Hardware-level feedback method: performing down-sampling and cropping at hardware-level by exploiting the image sensor capabilities, and detecting and tracking objects in smaller search regions.

The projection has been done using the characteristics curves provided by the manufacturer of the batteries. When there is one car in the scene, the average currents drawn are 0.2162 A, 0.1926 A and 0.1345 A for scenarios (i), (ii) and (iii), respectively. The projected battery lifetimes and energy savings are summarized in Table 6.8.

Method	Battery Lifetime (hours)	Gain (%)
Sequential Method	7.48	-
Software-level Feedback Method	8.40	12.3%
Hardware-level Feedback Method	15.5	107.2%

Table 6.8: Battery Lifetime projection.

As can be seen, when feedback method is combined with hardware-level operations (scenario (iii)), the battery life increases to 15.5 hours, and it provides 84.52% and 107.2% increase in battery-life compared to scenarios (i) and (ii), respectively. It should be noted that the projected battery lifetimes are based on the scenario, where there will always be an object to track in the scene, i.e. the scene will never be empty.

Chapter 7

Heterogeneous sensor network

7.1 Introduction

This chapter proposes the joint use of cameras and RFID sensors with the final objective of detecting and localizing intruders where cameras will be only devoted to people localization while RFID sensors will be devoted to people identification. Achieving both localization and identification of people in a wide open area using only cameras can be a challenging task, which requires cross-cutting requirements: high resolution for identification, whereas low resolution for having a wide coverage of the localization. To ground the observations on a common coordinate system, a calibration procedure is here defined. This procedure only demands a training phase with a single person moving in the scene holding a RFID tag. Although preliminary, the results demonstrate that this calibration is sufficiently accurate to be applied whenever different scenarios, where area of overlap between the field of view (FoV) of a camera and the "field of sense" (FoS) of a (blind) sensor must be efficiently determined.

7.2 People localization and identification in surveillance applications

There are several surveillance applications where *localizing* and *identifying* objects (people, vehicles, etc.) are crucial objectives for successive tasks such as collecting statistics, detecting intruders, analyzing paths of movement, recognizing suspicious behaviors, etc.. Localization and identification are typically two competing tasks since localization needs to not focus too much on a single object to have a wide coverage of the scene, whereas identification often requires a close-up of the object.

This is particularly true in the case of cameras and when the objects are people: identification might require zooming on the person's face and localization needs an unzoomed view to catch more people simultaneously and position them with respect to the scene. Moreover, cameras are typically mounted on high poles in order to reduce their number to cover the entire surveilled scene and the resolution is insufficient to recognize people by their face, or other biometric features (height, dresses, hair color). Finally, robust identification based on face recognition needs frontal or not occluded views and requires a certain degree of collaboration from the people themselves, which is unlikely to happen when looking for intruders.

Alternative sensors can be used for localization and identification purposes. Among the many, RFID (Radio Frequency IDentification) sensors [39] gained much attention thanks to their ease of use, low cost and touchless way-of-reading. The identification with RFIDs is accurate, using RFIDs for people identification introduces a limitation: RFID system detects only true positives, i.e. people wearing a RFID tag, but does not detect true negatives, i.e. "intruders" that are people not authorized or without the tag. Regarding localization, there have been previous attempts [40–43] which use multiple RFID tags or readers to assess people's locations using triangulation and RFID signal power (RSSI - Received Signal Strength Indicator). These approaches, however, do not guarantee a sufficient accuracy in the localization.

With these premises, this work proposes to jointly use cameras and RFIDs to take the best from both of them: cameras are used to localize all the people in the scene (regardless if they are intruders or not), while RFIDs are used to identify allowed people only. A data fusion procedure could thus be employed to infer the number and location of people present in the scene, their IDs and the presence of intruders. The scenario here is that of a wide open area with no designated entrances such as a construction working site, which makes the use of standard devices, such as badge, fingerprint or iris readers, unfeasible.

In order to fuse the data coming from the camera and the RFID, camera calibration, at least partial, is indeed necessary to create a common coordinate system with the RFID sensors. To make the system desirable also in terms of costs, standard cameras are replaced with wireless sensors nodes (motes) equipped with low-cost CMOS cameras. These devices yield also the appreciable characteristics to be moveable and to not require wired power supplies, which are very useful for outdoor setups. Unfortunately, the use of a wireless network of sensors poses also several challenges, such as, for instance, the communication and synchronization issues. From this perspective, however, the main problem is that, being the motes not fixed, standard calibration procedures are either not applicable or too time consuming.

Thus, this paper addresses this last problem and presents a novel method to (semi-)automatically calibrate a joint RFID-camera system used for localization and identification of people in wide open areas. Despite the specific application scenario, this approach is extensible to different scenarios, where area of overlap between the field

of view (FoV) of a camera and the "field of sense" (FoS) of a blind RFID sensor must be efficiently determined. The approach extends similar methodologies previously applied to multiple camera systems, using a person walking around equipped with a RFID tag as a "probe". Data collected by heterogeneous sensors are processed together to find a precise and accurate area and allowing a mutual calibration.

Several methods for RFID tag localization have been proposed in the literature. They can be classified based on the operating environments, either indoor or outdoor. The Scout system proposed in [42] is an outdoor localization system based on off-the-shelf active RFID systems and a probabilistic localization algorithm. The system is capable to cover wide outdoor environments and it is organized in a hierarchical architecture comprising servers, RFID readers and RFID tags; object tags are used for object identification while so-called "reference tags" are used for the environmental parameter calibration, by measuring their RSSI information. The authors proved that in 90% of the localization estimation the system provides object location with an average error distance less than 7 meters. LANDMARC [43] is a prototypal system that uses RFID technology for locating objects inside buildings. In order to improve the accuracy, this system makes use of the reference tags as described above. The authors evaluated performance over a 35 square meters indoor lab using 4 readers and a grid of 16 reference tags. They have shown that the 50 percentile has an error distance of around 1 meter while the maximum error distance are less than 2 meters. The proposed approach has some advantages: using extra cheaper tags there is no need for a large number of expensive RFID readers and the environmental dynamics can be easily accommodated. An Euclidean distance of the signal strength between a tracking tag and a reference tag is defined. For m reference tags each tracking tag has its E vector composed of the m distances. The proposed algorithm finds the unknown tracking tags' nearest neighbours by comparing different E values.

In [44] a RFID-based localization system for indoor localization of mobile robots is proposed. This system localizes RFID readers mounted on mobile robots using distributed tags on the floor. The reported experiments using a 5 x 4 tag square grid pattern with 7 cm inter-tag distance result in error variance reduction of the position estimation when power control is applied. The square root of the mean of error variance in distance is 5.6 cm in power control case and 5.9 cm in non-power control case. In all these methods based on RFIDs only the problem of "intruder" is not taken into account. The work in [45] is focused on a situation where a robot simultaneously interacts with two or more people and has to identify them with a passive-type RFID reader and floor sensors. A method is reported that integrates a person identification function provided by RFIDs and a person tracking function provided by floor sensors. To solve the association problem, if two or more people are around the robot, hypotheses are modeled using Bayesian networks and validated using the observations. Experimental results revealed that the developed system based on the proposed method successfully identified interacting people with 79.2% of the accuracy.

In [46] a sensor fusion method for an heterogeneous sensor environment with visual and identification sensors is proposed. The proposed technique addresses the problem of the coverage uncertainty of the identification sensors. This problem is managed by grouping unassociated identifications. The comparison for association performance between the existing association methods and the proposed one in terms of number of identification sensors and tracking performance has shown the supremacy of the proposed approach in all the simulated cases. This work supposes to have a precise calibration between sensors available, without addressing the problem explicitly. The approach proposed in this chapter could be employed in this situation.

7.3 The Camera Motes-RFID Architecture

The general hardware architecture is depicted in Fig. 7.1. The system is an heteroge-

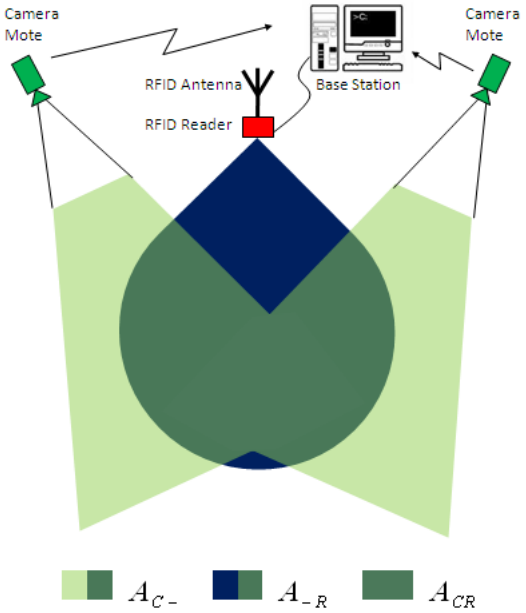


Figure 7.1: Sketch of the layout and the hardware architecture.

neous sensor network where the main elements are the cameras (which are supposed to be nodes of a wireless sensor network, i.e. camera motes), at least one RFID reader with its antenna, the active RFID tags and a server named “Base Station”. The camera motes are able to establish wireless communication with each others

and with the Base Station. The RFID reader is currently cable-connected to the Base Station server, but it can be equipped with a wireless interface becoming a node of the wireless network itself. The number of cameras that can be used is virtually unlimited, except for the inherent limit of the Base Station in terms of computational power and communication bandwidth.

The use of multiple cameras is crucial to enlarge the coverage of the scene. However, if the cameras are not coordinated the result of scene monitoring is much poorer. Conversely, if the label assigned by the video processing algorithms to a person can be kept consistent among the different camera views (procedure known as *consistent labeling*), the entire history of that person in the scene can be reconstructed: for instance, whenever the person is identified as an intruder, his/her movements in the whole scene can be back-traced. The FoVs of two close cameras can be either overlapped or disjoint. Often cameras' FoVs are disjoint, due to installation and cost constraints. In this case, the consistent labeling should be based on appearance only [47], i.e. matching two views of the same person only based on a more or less sophisticated model of the person's appearance.

If the cameras' FoVs are (partially) overlapped, consistent labeling can exploit geometry-based computer vision. This could be done with a precise system calibration, but this is not often feasible, in particular if the cameras are pre-installed and intrinsic and extrinsic parameters are not available. Thus, partial calibration or self-calibration methods can be adopted to extract only some of the geometrical constraints, e.g. to compute the ground-plane homography [48–50], and base inter-view matching on distances on a common coordinate system. Similarly, the number of RFID readers depends on the coverage for identification that needs to be reached. The only requirement for merging identification with localization based on video processing is that the FoS of each reader is partially overlapped with the FoV of at least one camera. Therefore, in this general scenario, the following areas can be defined (Fig. 7.1): A_{C-}, A_{-R}, A_{CR}, and two secondary areas $A_{\overline{C}R}$ and $A_{C\overline{R}}$. The area A_{C-} corresponds to the cameras' FoV resulting from the merge of multiple cameras. A_{-R} is the RFID FoS, i.e. the area where the RFID reader correctly detects tags. A_{CR} is the intersection of these two areas where both RFID-based identification and camera-based localization are possible. $A_{\overline{C}R}$ is the area where cameras cannot localize but the RFID can identify, that is $A_{\overline{C}R} = A_{-R} \setminus A_{C-}$, where $\setminus$ represents the set difference. Similarly, $A_{C\overline{R}}$ is the area where cameras can localize, but the RFID cannot identify, $A_{C\overline{R}} = A_{C-} \setminus A_{-R}$. It is clear that the correct knowledge of these areas and their limits is crucial for inferring the status of the people in the scene. While cameras' FoVs can be determined using known methods such as the EoFoV (Edges of Field of View) [48] or the E^2oFoV (Entry Edges of Field of View) [49], a method for calibrating the FoS of the RFID reader with respect to the camera is not available in the literature. Similarly to the E^2oFoV where a single person is required to move in the scene in the calibration phase in order to collect pairs of points in two different overlapped views, the proposed approach requires a

single person to move in the scene of overlap between the FoS and the FoV: the pair image coordinates-power of the RFID signal is used to determine the area in which the identification of a person holding a RFID tag must be unmistakable.

7.4 The Hardware Architecture

The following subsections detail the camera motes and the RFID technology used in the described system, namely CITRIC wireless camera motes and active RFIDs. The employed camera motes are the CITRIC camera. It is widely described in chaptetr 5.2.

RFID is a technology for automated identification of objects and people, and may be viewed as a means of explicitly labeling objects to facilitate their "perception" by computing devices. A RFID system has several basic components including RFID readers, antennas and tags, and the communication between them. The RFID reader can read data emitted from RFID tags through an antenna. Readers and tags use a defined radio frequency and a protocol to transmit and receive data. Moreover, the tags can be divided in passive or active. Passive tags operate without a own power supply source: all the energy they need for the operation is provided by the reader and is collected by the passive tag antenna. Passive tags are much lighter and less expensive than active tags, offering a virtually unlimited operational lifetime. However, their read range is very limited. Active tags contain both a radio transceiver and a battery to power the transceiver. Since there is an on-board radio on the tag, active tags have a wider range than passive tags. With the capability of providing RSSI information, current advanced RFID systems have become a potential candidate for mass localization, though they are not enough accurate for our scopes. The proposed system uses off-the-shelf long range active RFID systems from IDENTEC SOLUTIONS [51]. This system works at 868 MHz frequency range, with a read range up to 100 meters, and includes active tags i-B, reader i-PortM, and an antenna. The active tags continuously send their unique ID at regular intervals (programmable ping rate) without being requested by a reader.

7.5 Software Architecture

Considering the hardware setup described in the previous section, several algorithms are needed to reach the goal. First, the camera modules must process computer vision algorithms which detect and track moving objects in the scene. Specifically, in the case of camera motes, these algorithms should be computationally inexpensive due to the limited resources, but, at the same time, should guarantee a good robustness to noise, illumination changes, background camouflage, etc.. While these algorithms aim at computing people locations, the identities of allowed people are directly provided by the RFID technology. However, due to the unavoidable signal noise and

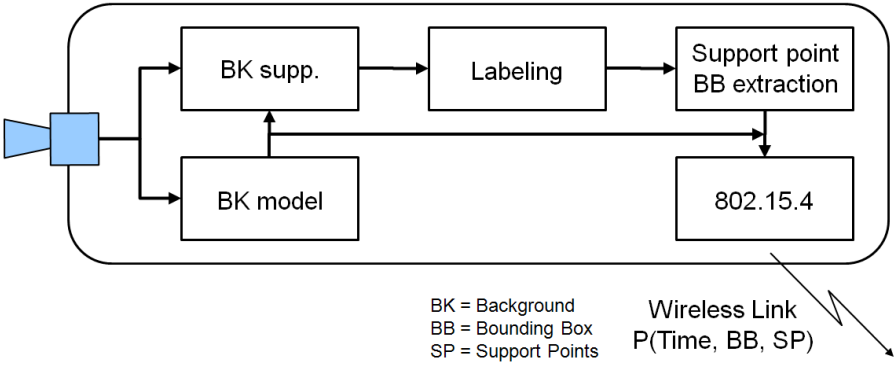


Figure 7.2: Data flow of the algorithms implemented in the camera mote.

to other sources of disturbance (e.g., presence of metal pieces, radio interferences, etc.), the RSSI obtained by the reader needs to be pre-processed to be cleaned. Finally, given both the set of people seen along the time by the camera motes and their corresponding locations, and the set of IDs sensed by the RFID reader, a data fusion strategy is required. It is exploited in the calibration step to automatically determine the area of overlap of the camera's FoV and the RFID's FoS; similar data will be used after calibration to infer precisely the list and locations of intruders.

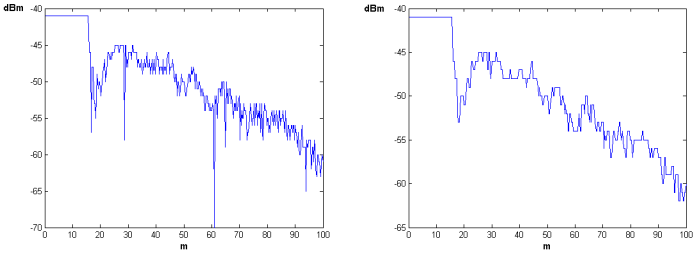
7.5.1 People Localization with Network of Cameras

The final prototype will make use of camera motes for the reasons reported in the introduction. However, some tests have been conducted using standard hard-wired cameras. In this case, cutting-edge algorithms for people detection and tracking can be used, such as the Sakbot approach [52], which is based on a sophisticated background suppression method and an appearance-based tracking. Unfortunately, in the case of camera motes, both the computational and the communication resources are limited and thus several operations must be performed on-board and only high-level post-processed data are sent to the Base Station. In order to reduce the amount of base station data processing each camera mote (Fig. 7.2) performs background subtraction and blob identification (by grouping all connected regions in the foreground image), and compute a set of boxes bounding the resulting blobs. Then, for each image the full set of geometrical information extracted from the bounding boxes, such as its dimensions and positions in the image plane, are sent to the Base Station to be used for tracking moving objects. Because of the reduced computing capabilities of the camera motes the computational complexity of the algorithms is a crucial factor which justifies the choice of the simple background suppression technique. In this approach, each incoming frame is compared with the background

model, which is updated by using median filtering as described in [26]. A reduction in the number of false alarms in outdoor environments is achieved by carefully tuning the blending factor value used to update the background model and discarding small bounding boxes (with area below the a given threshold). To enable data consistency and coordination, each camera motes send local time information. To keep consistency information, camera motes time synchronization is required [53]. On the Base Station a set of remote commands is available in order to aid the camera motes deployment. During the people localization activities, camera motes send to the base station geometrical information related to the moving objects detected on each frame. Simultaneous tracking of multiple objects using Kalman filter [54] and consistent labeling among different views (using the HECOL approach [49]) are performed by the Base Station.

7.5.2 People Identification with RFID Technology

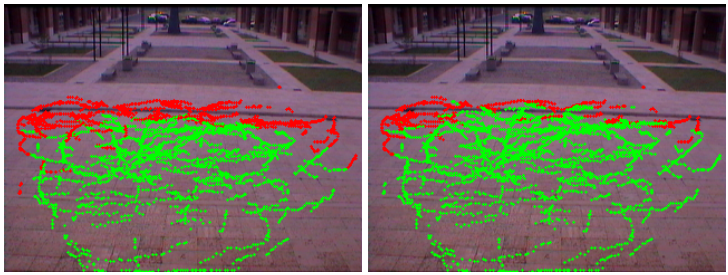
It is well known that the signal provided by RFID technology is very noisy, in particular when using active tags which have a very high reading range. Fig. 7.3(a) reports an example of the RSSI value (expressed in dBm, i.e. power ratio in decibels (dB) referenced to one milliWatt) for a person moving slowly away from the RFID antenna. It is evident from this plot that the RSSI generally decreases, but with a lot of noise and with a particular but not accidental drop around 20 meters from the antenna. Given these problems of not linearity between RSSI and distance, both the localization and the identification using directly the RSSI information are doomed to fail. Thus, first the signal is cleaned by applying a median filter of width W (in



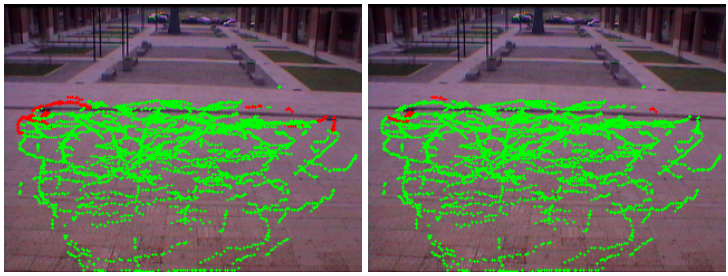
(a) (b)

Figure 7.3: RFID signal strength RSSI over distance: (a) before pre-processing, (b) after applying median filtering.

these experiments $W = 5$ - result in Fig. 7.3(b)) and a threshold must be selected to decide whether the RFID is sensed/identified or not. Hereinafter, the tag will be considered in the status "sensed" when the RSSI is above or equal to the threshold,



(a) -41 dBm (b) -43 dBm



(c) -45 dBm (d) -47 dBm

Figure 7.4: Distribution of below-threshold (red) and above-threshold points by changing the threshold (in dBm).

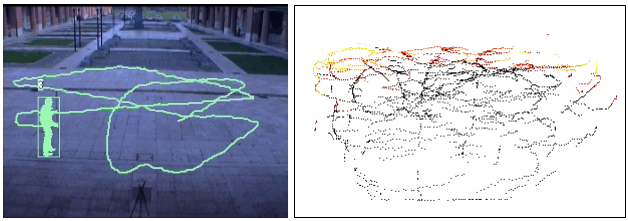
and "not sensed" otherwise[1]. In the case of a single person moving in the scene, the status of each reading can be automatically associated with the location of the single blob detected. Fig. 7.4 shows some experiments obtained changing the threshold. With higher threshold, as for instance Fig. 7.4(a) with respect to Fig. 7.4(d), the number of locations with a sensed RFID (in green) is lower and is higher the number of detected location where the RFID is not sensed (in red).

7.6 Calibration of Cameras and RFIDs

Similar to what has been done for calibrating pairs of overlapped cameras [49], this proposal for calibrating the RFID reader with a camera is to consider a procedure where a single person holding a RFID tag is moving in the scene. This simplification, aside from easing the detection and tracking by means of the camera, allows a straightforward association between the (x, y) coordinates of the lower support point of the person (extracted as described in Section 7.5.1) and the corresponding RSSI

[1]Physically the tag is sensed also under the threshold but with a low RSSI.

value of the tag. Fig. 7.5 shows an example of the tracking (Fig. 7.5(a)) and the corresponding plot of the collected points with the color indicating the RSSI value (Fig. 7.5(b)).



(a) (b)

Figure 7.5: Example of the calibration procedure: (a) example of tracking and localization with the camera, (b) plot of locations with color indicating the RSSI value.

(a) (b) (c)

Figure 7.6: Example of the calibration procedure: (a) DAT area for example of Fig. 7.5(b), (b) DBT, (c) final determination of the areas.

By starting from the distribution of points of Fig. 7.5(b) and applying the thresholding described in the previous section, two sets of points/locations can be obtained: above and below the threshold. Considering that the ping rate of the RFID reader is not so high (1 sample/second in this case) and that a person should not correspond to a single point but has a certain occupancy in the scene, on these two sets is applied a morphological dilation with a circular structuring element (with radius equal to 10), resulting in two sets called DAT (Dilated Above Threshold) and DBT (Dilated Below Threshold). These two sets for the example in Fig. 7.5(b) are reported in Figg. 7.6(a) and 7.6(b), respectively.

It is worth noting that after dilation these two sets are not disjoint and their intersection defines an area $U = DAT \cap DBT$ where the RFID reading can not be considered certain (i.e., it may happen or not that a tag is sensed). Similarly, the two areas $CS = DAT \setminus U$ and $CN = \Omega \setminus DAT$ represent the "certainly sensed" and "certainly not sensed" areas, respectively, where Ω represents the domain of the locations (i.e., the image). By applying morphological operations to these two areas

and using a convex hull to close polygons, three different areas can be obtained (Fig. 7.6(c)): the so-called area *NEAR* is defined by the pixels inside the convex hull applied on CS (inside yellow area in Fig. 7.6(c)); the area *FAR* by the pixels outside the convex hull on DAT (outside cyan area in Fig. 7.6(c)); finally, the area *UNCERTAIN* which is contained between the *NEAR* and the *FAR* areas. An evaluation of the accuracy in determining these areas will be reported in Section 7.7.

The calibration of these areas can be then exploited to assert, for any person detected and tracked by the cameras, whether he/she is in the *NEAR*, *UNCERTAIN* or *FAR* area, as an input of more or less sophisticated reasoning engines. As a possible way of proceeding, consider Fig. 7.7 where all the possible transitions Ci of a person (Fig. 7.7(a)) and all possible transitions Ri of a RFID tag (Fig. 7.7(b)) are shown. Thanks to the calibration, it is possible to use the area *NEAR* as FoS of the RFID, and, considering the movement of an allowed person (thus holding a tag), some of the combinations between Ci and Ri are not compatible. Table 7.1 summarizes these combinations. Note that some combinations with $R3$ are yet compatible but of no use for the inference (indicated with $C*$). These considerations can be a starting point for at least discarding not compatible observations of cameras and RFIDs. It is, however, not enough to infer the presence of intruders and more sophisticated approaches, based on statistical inference, such as Transferable Belief Model (TBM) [55] should be used.

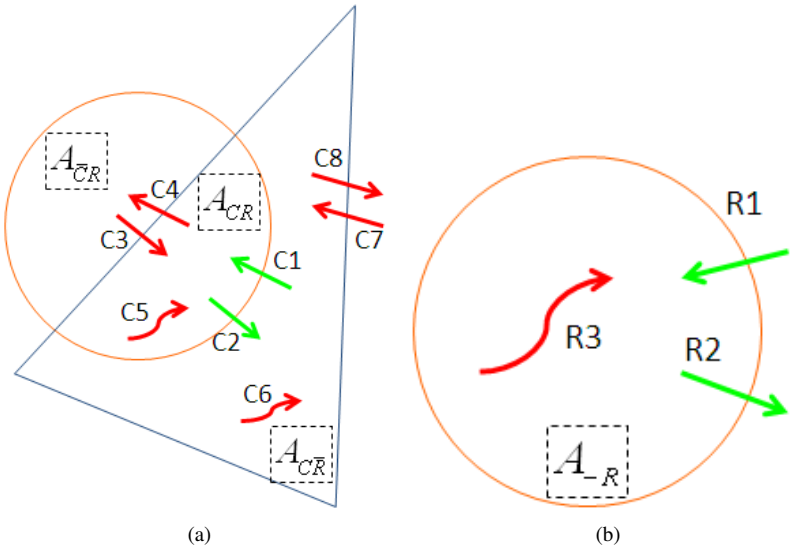


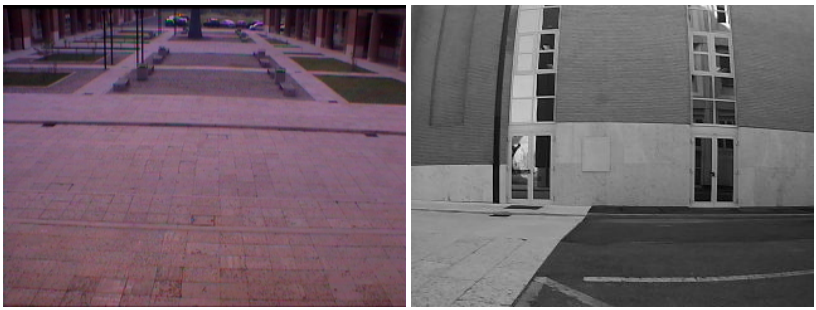
Figure 7.7: Example of transitions in the FoV of the camera (a) and in the FoS of the RFID (b).

		R1	R2	R3
		$A_{-\overline{R}} \rightarrow A_{-R}$	$A_{-R} \rightarrow A_{-\overline{R}}$	$A_{-R} \rightarrow A_{-R}$
C1	$A_{C\overline{R}} \rightarrow A_{CR}$	C	NC	NC
C2	$A_{CR} \rightarrow A_{C\overline{R}}$	NC	C	NC
C3	$A_{\overline{CR}} \rightarrow A_{CR}$	NC	NC	C*
C4	$A_{CR} \rightarrow A_{\overline{CR}}$	NC	NC	C*
C5	$A_{CR} \rightarrow A_{CR}$	NC	NC	C*
C6	$A_{C\overline{R}} \rightarrow A_{C\overline{R}}$	NC	NC	NC
C7	$A_{\overline{CR}} \rightarrow A_{C\overline{R}}$	NC	NC	NC
C8	$A_{C\overline{R}} \rightarrow A_{\overline{CR}}$	NC	NC	NC

Table 7.1: Possible combinations of transitions of camera tracks (row) and RFID tags (column). C=Compatible, NC=Not Compatible. Please note that $A_{-\overline{R}}$ represents the area outside the RFID's FoS, while $A_{\overline{CR}}$ is composed of points outside both the FoV and the FoS. C* means that the combination is yet compatible but of no use for the inference.

7.7 Experimental Results

The work reported in this chapter is part of a larger project for the monitoring and security of construction working sites, which includes the detection of intruders. At this early stage, since the inference engine has not been yet fully developed, the preliminary results only aim at evaluating the accuracy of the calibration process. To achieve this, two outdoor scenarios have been considered, one taken with a single normal camera, and one taken from a pair of camera motes (Fig. 7.8). In the case of



(a) Normal high-resolution camera (b) Camera mote

Figure 7.8: Snapshots of the two scenarios.

the normal high-resolution camera, it has been connected to the Base Station through a wired connection, the video feeds are transmitted to the Base Station as is, and the

Sakbot algorithms (Section 7.5.1) are run on the Base Station directly. In the case of the pair of camera motes, instead, the communication is wireless, the algorithms are much lighter (see again Section 7.5.1) and directly implemented on board. Then, only the locations of the moving people are sent to the Base Station which implements tracking, consistent labeling among the two camera motes and data fusion with the RFID. The layout used in our experiment with two camera motes is shown in Fig. 7.9.

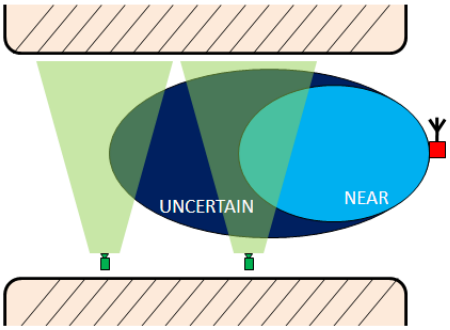
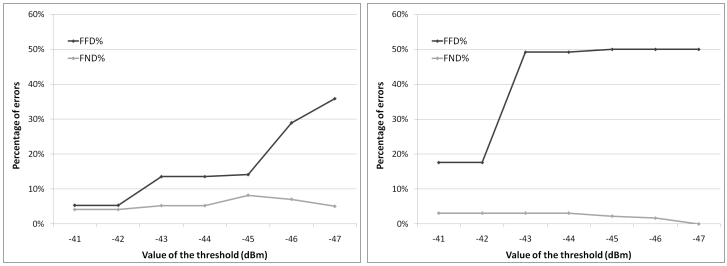


Figure 7.9: Layout of the scenario with two camera motes.



(a) (b)

Figure 7.10: False Near Detections (FND) and False Far Detections (FFD) in percentage for the scenario with a normal camera (a) and with a pair of camera motes (b).

The experimental methodology is as follows. A video with a single person holding a RFID tag is used as training for delimiting the *NEAR* and *FAR* areas as described above. Then, a certain number of videos with multiple people holding tags is used a testing set: in these experiments 4 videos for the first scenario and 12 videos for the second have been used. With respect to the testing videos, the number of *false*

near detections FND (i.e., the number of locations which are supposed to be in the *NEAR* area but in which the RSSI value is below the threshold - the same used during the training phase) and the number of *false far detections* FFD (i.e., the number of locations which are supposed to be in the *FAR* area but in which the RSSI is still above the threshold) are employed to evaluate the accuracy of the calibration procedure. These two values, averaged on the testing videos and with different values of the threshold, are reported for the two scenarios in Fig. 7.10. As expected, the FND parameter is low because of the high signal to noise ratio of the points in the NEAR area. Viceversa reducing the threshold value increases the number of points having a lower signal to noise ratio justifying the increase of the FFD value.

Chapter 8

Conclusions

This thesis has allowed to explore the potentiality of embedded devices for Green Vision by addressing some of the most challenging problems, namely the energy efficiency of devices, the implementation in flexible architectures such as FPGA of highly-demanding algorithms, and the fusion of heterogeneous sensor modalities for high level tasks.

In chapter three, the design and development of a low cost veiling luminance estimation system based on a CMOS image sensor, fully implemented on FPGA has been reported. The project is motivated by the fact that the veiling luminance is particularly important for a vehicle's driver approaching to the entrance of a tunnel, since it impacts on the driver's ability to perceive the presence of an obstacle or a slowdown caused by the traffic, and thus its safety. The formulation used to compute the luminance value follows the European Standard CIE 88/2004 (Italian UNI 11095 [2]). The proposed solution, based on an FPGA implementation, presents advantages in terms of size, energy efficiency and costs that make it suitable to develop stand alone device capable to operate on the field close the street tunnels. The device for the veiling luminance computation, once combined with a power controller, can be used to increase or decrease accordingly the lighting intensity inside a tunnel avoiding problems for the driver's safety.

In chapter four, a comparison of the two most advanced algorithms for connected components labeling has been proposed by highlighting how they perform on a soft core SoC architecture based on FPGA. The block-based connected components labeling algorithm from Grana et al. [16] has been summarized, optimized with decision tables and decision trees and implemented on a soft core SoC architecture which allows to perform different high level image processing tasks. The results show that

the proposed approach provide a significant speedup (up to 50%) with respect to the state of the art.

In chapter six, two methodologies that aim to increase the energy-efficiency and the battery-life of an embedded smart camera have been presented. The energy saving is obtained by hardware-level operations when performing object detection and tracking. First, instead of performing down-sampling at software-level at the main microprocessor of the camera board, this operation is performed at hardware-level on the micro-controller of the OV9655 image sensor of a CITRIC camera. Moreover, rather than performing object detection and tracking on the whole frame, the location of the target in the next frame is estimated from a search region around it, the next frame is cropped by using the HREF and VSYNC signals at the micro-controller of the OV9655, and object detection and tracking are performed only in the cropped search region. Reduced amount of data that is moved from the image sensor to the main memory at each frame, better use of the memory resources and not occupying the main microprocessor with image down-sampling and cropping tasks, provide significant savings in energy consumption and battery-life. Experimental results show that, compared to software-level cropping, performing hardware-level cropping when tracking one object provides 84.52% increase in battery-life prolonging the life of the camera up to 15.5 hours. In addition, hardware-level down-sampling and cropping, and performing detection and tracking in cropped regions provide 41.24% decrease in energy consumption, and 107.2% increase in battery-life compared to performing software-level down-sampling and processing whole frame.

In chapter seven an easy calibration procedure for (semi-)automatically determining the area of overlap between the cameras' FoV and the RFID's FoS has been presented. The system exploits camera motes to detect, track and localize moving people and active RFID technology for identifying allowed people. In order to move towards an inference engine capable to detect and localize intruders in wide open areas (with no designated entrances), a calibration procedure of the two subsystems is presented.

Bibliography

[1] D.D. Gajski, F. Vahid, S. Narayan, and Jie Gong, "SpecSyn: an environment supporting the specify-explore-refine paradigm for hardware/software system design," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 6, no. 1, pp. 84 –100, march 1998.

[2] Normativa UNI 11095, *Illuminazione delle gallerie*, Dec. 2003.

[3] Commision Internationale de lEclairage CIE, *Guide for the lighting of road tunnels and underpasses*, Vienna, Austria, USA, 1990.

[4] Bernd Jahne, *Practical Handbook on Image Processing for Scientific and Technical Applications, Second Edition*, CRC Press, Inc., Boca Raton, FL, USA, 2004.

[5] C.C. Slama, *Manual of Photogrammetry, Fourth Edition*, American Society of Photogrammetry and Remote Sensing, Falls Church, Virginia, USA, 1980.

[6] Rita Cucchiara, Costantino Grana, Andrea Prati, and Roberto Vezzani, "A computer vision system for in-house video surveillance," *IEE Proceedings - Vision, Image and Signal Processing*, vol. 152, no. 2, pp. 242–249, Apr. 2005.

[7] E. Hecht, *Optics*, Addison-Wesley, 4thl, 2001.

[8] G. Wyszecki and W.S. Stiles, *Color science: concepts and methods, quantitative data and formulae*, John Wiley & Sons, Inc., 2ndl, 2000.

[9] Wonpil Yu, "Practical anti-vignetting methods for digital cameras," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 4, pp. 975–983, Nov. 2004.

[10] (2010) Optomotive., "Optomotive, mehatronika d.o.o., V Murglah 229, SI-1000 Ljubljana, Slovenia," `http://optomotive.si`.

[11] Azriel Rosenfeld and John L. Pfaltz, "Sequential Operations in Digital Picture Processing," *J. ACM*, vol. 13, pp. 471–494, October 1966.

[12] R. Haralick, "Some neighborhood operations," *Real Time Parallel Computing: Image Analysis*, pp. 11–35, 1981.

[13] K Suzuki, "Linear-time connected-component labeling based on sequential local operations," *Computer Vision and Image Understanding*, vol. 89, no. 1, pp. 1–23, 2003.

[14] Fu Chang and Chun-Jen Chen, "A Component-Labeling Algorithm Using Contour Tracing Technique," in *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, Washington, DC, USA, 2003, ICDAR '03, pp. 741–, IEEE Computer Society.

[15] Lifeng He, Yuyan Chao, Kenji Suzuki, and Kesheng Wu, "Fast connected-component labeling," *Pattern Recogn.*, vol. 42, no. 9, pp. 1977–1987, September 2009.

[16] C. Grana, D. Borghesani, and R. Cucchiara, "Optimized Block-Based Connected Components Labeling With Decision Trees," *Image Processing, IEEE Transactions on*, vol. 19, no. 6, pp. 1596 –1609, june 2010.

[17] L. J. Schutte, "Survey of decision tables as a problem statement technique," *Computer Science Department, Purdue University, CSD-TR 80*, 1973.

[18] Helmut Schumacher and Kenneth C. Sevcik, "The synthetic approach to decision table conversion," *Commun. ACM*, vol. 19, pp. 343–351, June 1976.

[19] R. Pon, M.A. Batalin, J. Gordon, A. Kansal, D. Liu, M. Rahimi, L. Shirachi, Y. Yu, M. Hansen, W.J. Kaiser, M. Srivastava, G. Sukhatme, and D. Estrin, "Networked infomechanical systems: a mobile embedded networked sensor platform," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, april 2005, pp. 376 – 381.

[20] Stefan K. Gehrig, Felix Eberli, and Thomas Meyer, "A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching.," in *ICVS*, Mario Fritz, Bernt Schiele, and Justus H. Piater, Eds. 2009, vol. 5815 of *Lecture Notes in Computer Science*, pp. 134–143, Springer.

[21] L. Tessens, M. Morbee, W. Philips, R. Kleihorst, and H. Aghajan, "Efficient approximate foreground detection for low-resource devices," in *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, 30 2009-sept. 2 2009, pp. 1 –8.

[22] V. Reddy, C. Sanderson, B.C. Lovell, and A. Bigdeli, "An efficient background estimation algorithm for embedded smart cameras," in *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, 30 2009-sept. 2 2009, pp. 1 –7.

[23] Mauricio Casares, Senem Velipasalar, and Alvaro Pinto, "Light-weight salient foreground detection for embedded smart cameras," *Comput. Vis. Image Underst.*, vol. 114, pp. 1223–1237, November 2010.

[24] V. Kianzad, S. Saha, J. Schlessman, G. Aggarwal, S. S. Bhattacharyya, W. Wolf, and R. Chellappa, "An architectural level design methodology for embedded face detection," in *Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, New York, NY, USA, 2005, CODES+ISSS '05, pp. 136–141, ACM.

[25] Zaihong Shuai, Songhwai Oh, and Ming-Hsuan Yang, "Traffic modeling and prediction using camera sensor networks," in *Proceedings of the Fourth ACM/IEEE International Conference on Distributed Smart Cameras*, New York, NY, USA, 2010, ICDSC '10, pp. 49–56, ACM.

[26] P. Chen, P. Ahammad, C. Boyer, Shih-I Huang, Leon Lin, E. Lobaton, M. Meingast, Songhwai Oh, S. Wang, Posu Yan, A.Y. Yang, Chuohao Yeo, Lung-Chung Chang, J.D. Tygar, and S.S. Sastry, "CITRIC: A low-bandwidth wireless camera network platform," *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, pp. 1 –10, sept. 2008.

[27] B. Rinner, T. Winkler, W. Schriebl, M. Quaritsch, and W. Wolf, "The evolution from single to pervasive smart cameras," *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, pp. 1 –10, sept. 2008.

[28] M. Casares and S. Velipasalar, "Resource-Efficient Salient Foreground Detection for Embedded Smart Cameras br Tracking Feedback," in *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, 29 2010-sept. 1 2010, pp. 369 –375.

[29] M. Casares, P. Santinelli, S. Velipasalar, A. Prati, and R. Cucchiara, "Energy-efficient foreground object detection on embedded smart cameras by hardware-level operations," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, june 2011, pp. 150 –156.

[30] Intel, "Intel PXA27x Processor Family Developers Manual," 2004.

[31] Omnivision Technologies Inc., "OV9655 Color CMOS SXGA (1.3MegaPixel) CAMERACHIP with OmniPixel Technology Datasheet," 2006.

[32] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, David Culler, J. Rabaey W. Weber, and E. Aarts, *TinyOS: An Operating System for Wireless Sensor Networks*, Ambient Intelligence. Springer-Verlag, 2004.

[33] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric Brewer, and David Culler, "The nesC language: A holistic approach to networked embedded systems," in *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, New York, NY, USA, 2003, PLDI '03, pp. 1–11, ACM.

[34] Dr. Gary Rost Bradski and Adrian Kaehler, *Learning opencv, 1st edition*, O'Reilly Media, Inc., 2008.

[35] Free Software Foundation, Inc. , "GCC, the GNU Compiler Collection," .

[36] M. Casares, P. Santinelli, S. Velipasalar, A. Prati, and R. Cucchiara, "Energy-efficient Feedback Tracking on Embedded Smart Cameras by Hardware-level Optimization," in *Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras*, Ghent (Belgium), Aug. 2011, pp. 1–6.

[37] Bill Dirks, Michael H. Schimek, Hans Verkuil and Martin Rubli, "Video for Linux Two API Specification. Revision 0.24," .

[38] Linux Kernel Organization, Inc., "The Linux Kernel Archives," .

[39] Klaus Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*, John Wiley & Sons, Inc., New York, NY, USA, 2003.

[40] T. Sanpechuda and L. Kovavisaruch, "A review of RFID localization: Applications and techniques," in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference on*, may 2008, vol. 2, pp. 769 –772.

[41] M. Bouet and A.L. dos Santos, "RFID tags: Positioning principles and localization techniques," in *Wireless Days, 2008. WD '08. 1st IFIP*, nov. 2008, pp. 1 –5.

[42] Xin Huang, R. Janaswamy, and A. Ganz, "Scout: Outdoor Localization Using Active RFID Technology," *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on*, pp. 1 –10, oct. 2006.

[43] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil, "LANDMARC: indoor location sensing using active RFID," *Wirel. Netw.*, vol. 10, no. 6, pp. 701–710, 2004.

[44] Youngsu Park, Je Won Lee, and SangWoo Kim, "Improving position estimation on RFID tag floor localization using RFID reader transmission power control," in *ROBIO '09: Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics*, Washington, DC, USA, 2009, pp. 1716–1721, IEEE Computer Society.

[45] K. Nohara, T. Tajika, M. Shiomi, T. Kanda, H. Ishiguro, and N. Hagita, "Integrating passive RFID tag and person tracking for social interaction in daily life," *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, pp. 545 –552, aug. 2008.

[46] Shung Han Cho, Sangjin Hong, and Yunyoung Nam, "Association and Identification in Heterogeneous Sensors Environment with Coverage Uncertainty," in *AVSS '09: Proceedings of the 2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, Washington, DC, USA, 2009, pp. 553–558, IEEE Computer Society.

[47] Christopher Madden, Eric Dahai Cheng, and Massimo Piccardi, "Tracking people across disjoint camera views by an illumination-tolerant appearance representation," *Mach. Vis. Appl.*, vol. 18, no. 3-4, pp. 233–247, 2007.

[48] S. Khan and M. Shah, "Consistent labeling of tracked objects in multiple cameras with overlapping fields of view," vol. 25, no. 10, pp. 1355–1360, Oct. 2003.

[49] Simone Calderara, Andrea Prati, and Rita Cucchiara, "HECOL: Homography and epipolar-based consistent labeling for outdoor park surveillance," *Computer Vision and Image Understanding*, vol. 111, no. 1, pp. 21 – 42, 2008, Special Issue on Intelligent Visual Surveillance (IEEE).

[50] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A system for learning statistical motion patterns," vol. 28, no. 9, pp. 1450–1464, September 2006.

[51] IDENTEC SOLUTIONS AG, "www.identecsolutions.com," .

[52] Rita Cucchiara, Costantino Grana, Massimo Piccardi, and Andrea Prati, "Detecting Moving Objects, Ghosts and Shadows in Video Streams," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1337–1342, Oct. 2003.

[53] Miklos Maroti, Miklos Maroti, Branislav Kusy, Branislav Kusy, Gyula Simon, Gyula Simon, Akos Ledeczi, and Akos Ledeczi, "The flooding time synchronization protocol," 2004, pp. 39–49, ACM Press.

[54] R.S. Bucy and P.D. Joseph, *Filtering for Stochastic Processes with Applications to Guidance*, AMS Chelsea Publ., 2005.

[55] P. Smets, "Data fusion in the transferable belief model," in *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*, july 2000, vol. 1, pp. PS21 – PS33 vol.1.