Graz University of Technology

Institute for Computer Graphics and Vision

Dissertation

Visual Surveillance on DSP-Based Embedded Platforms

Clemens Arth

Graz, Austria, March 2008

Thesis supervisor
Vertr.-Prof. Dr. Horst Bischof

Thesis reviewer
Univ.-Prof. Dr. Bernhard Rinner

We must not forget that when radium was discovered no one knew that it would prove useful in hospitals. The work was one of pure science. And this is a proof that scientific work must not be considered from the point of view of the direct usefulness of it. It must be done for itself, for the beauty of science, and then there is always the chance that a scientific discovery may become like the radium a benefit for humanity.

Marie Curie

French chemist & physicist (1867 - 1934)

Abstract

Visual surveillance has become an important topic of research in the last few years due to the increased need for security in public places and the unbowed trend to use digital cameras for surveillance purposes, or for integration of visual sensors in other objects of personal use, like mobile phones or PDAs. To perform dedicated operations at camera site, embedded platforms, so-called *Smart Cameras*, have become popular recently. These platforms are equipped with one or multiple video sensors and enough computational power to process the data stream onboard. Moreover, these platforms have additional advantages, like robustness against environmental stress or low power consumption. The use of smart sensors facilitates the building of large networks, and the local processing paradigm allows for extracting valuable information at site and transmission over low-cost and low-bandwidth communication channels.

However, until now, little work has been done on the development and investigation of state-of-the-art algorithms for surveillance from the field of computer vision in respect of embedded systems. On this account, in this thesis we focus on the tasks of object detection and object recognition on smart cameras. We discuss several issues of algorithm development on embedded DSP-based platforms, especially the issues related to parallelism mechanisms and fixed-point arithmetic. Given an extensive overview about current work on object detection and object recognition, we investigate both fields of development in detail. After discussing the basic algorithm, we experimentally demonstrate the suitability of the approach, derived in this thesis, for performing object detection in real-time on a real-world traffic surveillance scenario, given a prototypical DSP-based hardware platform. Our results encourage the integration of our algorithm into a larger system for public surveillance. In the context of object recognition, we demonstrate, how state-of-the-art recognition technology can be used to deploy reasonable recognition capabilities on smart cameras. After proving the suitability on a moderate size object database, we show how our algorithm can be used in a traffic surveillance scenario for recognition and reacquisition of vehicles on public streets. The main advantages of our approach are the high accuracy and the minimization in communication necessary between adjacent camera motes. By investigating our approach in detail, we are able to draw general conclusions and statements about the suitability of different aspects and methods of software development on DSP-based embedded systems.

Acknowledgments

After three years of ups and downs, finally the time of writing my PhD thesis nearly comes to an end. I would like to thank many people, who helped and supported me during this period of time.

First, I want to thank my wife Tamara for being so patient, whenever I was working far into the night, whenever I was irritated and obnoxious. Thank you for supporting me and keeping me grounded, whenever I was doubting about my work.

Thanks to my parents Manfred and Renate, who gave me the opportunity to attend university and to become what I am now. Thanks to my parents-in-law Heli and Helmut, for bearing my moods, not only, but especially during writing this thesis. Greetings also go to the rest of my family for many nice talks and festivities, which helped in compensating much of the stressful and tedious working effort during this time.

Special greetings go to the three students I supervised in master thesis projects. Thanks to Surinder Ram, Florian Limberger and Christian Leistner, who helped me diving deeper into the topic of embedded systems and computer vision. Special thanks to Christian for helping me writing almost half of the publications and for helping me in developing much of the stuff presented here.

Big thanks to my fellows at the Institute for Computer Graphics and Vision at Graz University of Technology. In the three years of my staying many of them became more friends than colleagues. Our technical and non-technical discussions for sure influenced this thesis and my motivation. Thanks to Horst, Peter, Martin, Michael, Matthias, Jakob, Roman, Thomas ($\times 2$), Werner, Martina and Sandra. Very special thanks to Helmut and Michael Grabner, who helped me a lot and for our encouraging discussions. Also big thanks go to the secretaries of our institute Renate and Christina, and to our network administrator Andi. Thanks to Markus Quaritsch from the Institute for Technical Informatics for some nice discussions on implementation issues on embedded systems.

Thanks go to the former members of the FREQUENTIS research center, Andreas, Jörg and Dieter. Thank you for giving me the opportunity to start my work on computer vision for embedded systems. Thanks to FREQUENTIS, for paying only half of my thesis

and withdrawing all our results of 2 year research work.

Last, but not least, I would like to thank Horst Bischof for giving me the opportunity to write this thesis, for his support and the fruitful discussions on problems in computer vision throughout the last years. Finally, I want to thank Bernhard Rinner for becoming my second supervisor and reviewing my thesis, for his comments and hints on software development for embedded systems.

Clemens Arth, Graz, Austria
March 2008

Contents

List of Figures

List of Tables

Inspiration is wonderful when it happens, but the writer must develop an approach for the rest of the time... The wait is simply too long.

Leonard Bernstein
US composer & conductor, 1918 - 1990

Chapter 1

Introduction

In the last few years there is an unbowed trend to use digital cameras in all areas of our everydays life. The reasons for that are the increased need for security in public places and thus the more frequent use of cameras for surveillance purposes, but also the decline in the price of sensors in general and the deployment of cameras in other objects of personal use, like mobile phones or PDAs.

With the increase of additional video sources the field of computer vision has become more and more important as the amount of information has to be processed somehow automatically. One possible solution to this is the use of large data processing centers with hundreds of computational units and considerable spatial dimensions. However, the increase in computational power per unit and the use of video sensors for tasks to be fulfilled under adverse environmental conditions has led to the development of a special group of devices which are combinations of *embedded computers* and video cameras.

So-called *Smart Cameras* are embedded platforms with small form factor equipped with a single or multiple video sensors and enough computational resources to perform dedicated operations onboard and at site. Additional advantages of these platforms are their low production costs, their low power consumption and their robustness to environmental stress. There is a wide area of applicability for these type of devices: care in residential home for the elderly, industrial robotics, domestic home care, and clearly all types of surveillance of public places, like traffic surveillance or access control in public transport systems. There is a big number of applications and the list is steadily getting longer. However, in this work we are concerned about the development of algorithms for surveillance purposes on embedded platforms only, thus we will extend that in more detail in the following.

1.1 Computer Vision and Embedded Systems

Computer vision can be used in multiple applications like robotics, industrial engineering, monitoring or surveillance. Especially in the area of surveillance a lot of scientific research

and financial investment is necessary for additional precautions against terroristic activity and assaults. But also due to the increased amount of individual transport on public highways and the resulting traffic jams monitoring is necessary and automated or assisted event detection is simply indispensable for the human observer.

The main goal is performing the detection and recognition of objects and the following reasoning about action, events and behaviour fully automatically. The most important reason for this is that it is prohibitive to align the number of employees to the number of video sources. The costs for the surveillance of even a small area are simply too high. However, another reason is that it is anyway impossible for a human to draw his attention to more than one scene at the same time without missing changes or knowledge about valuable coherences. Even drastic changes in a single scene can simply go unnoticed which is also known as *change blindness* [192, 218]. Humans cannot keep up with their concentration over a long period of time, moreover man cannot draw his visual attention to all regions of images equally well and thus selects one coevally discarding others accidentally [98].

From this fact it is easy to see that consistent observation of multiple scenes is impossible by using human data processing only if it is not even working with one single source. Therefore automatic procedures for performing object detection, recognition and reasoning are necessary and the use of computer vision is justified.

Although the development of algorithms is done almost exclusively for the use on usual computers, since the invention of so-called *Smart Cameras* developers also focus on the deployment of different algorithms on these platforms. In principle a *Smart Camera* is a combination of an *embedded computer* and a video sensor. An *embedded* ***system*** or *embedded computer* is defined to be an interconnection of digital and/or analog electronical circuits for a special predefined purpose. It is the design for exactly that purpose that allows a particular functionality to be fulfilled in a way optimized in terms of power consumption, reliability, processing speed and production cost. In fact this is the main difference compared to the principles of a general-purpose computing system which is not able to perform in that way. Moreover in contrast to software dedicated for general-purpose computers, specially designed and compiled code is necessary to be run on embedded systems, which is emphasized by the commonly used name *firmware* for these programs.

The term ***embedded*** means that such a system forms the core of any machinery it is designed to control or monitor. This general definition applies to all kinds of electronic devices, from CD players in cars to alarm equipment in private houses. However, given the term *Smart Camera* and from the previous definition it becomes clear that we are talking about a device where the machinery controlled is a video sensor and the dedicated task to be fulfilled is computer vision. Note, that the nomenclature *smart camera* does not make any statement about the application area. Smart cameras can be deployed in buildings to observe entrance areas, for example. In this case, the motes are static and not moving, which is one field of operation. However, they can also be deployed in vehicles to help the driver to detect obstacles or haz-

ards. In this case, the units are mobile and a different type of functionality is necessary.

As already mentioned an embedded system is some type of device which is highly optimized for a given functionality. Embedded systems can be divided into several subgroups which vary in their flexibility from *close-to general-purpose computers* as *Digital Signal Processors* (DSPs) to maximally *application specific* circuitry like *Application Specific Integrated Circuits* (ASICs). Unfortunately the common advantages like low production costs, low power consumption and high throughput are dearly bought. The most limiting factor for the deployment of algorithms is the modality of memory management. It is easily possible to trade memory consumption versus computing resources on general-purpose computing systems. However, not the availability of memory, but the lack of a memory management unit and the necessity to manage memory consumption, memory mapping and swapping by hand, makes well established programming techniques hard to utilize on embedded systems. Moreover, the lack of memory is a minor problem in one-dimensional signal processing like speech compression, but it is even a more severe problem in two-dimensional signal processing, *i.e.* image processing, as images need a lot of memory a priori. It is even more important that the handling of images in algorithms has to be carefully organized to guarantee for good performance - or to get some algorithm to run at all not even thinking of real-time behaviour.

The vast majority of approaches designed for usual general-purpose computers cannot easily be deployed on the kind of hardware described. This is the consequential outcome of an algorithm design process where the most limiting factors and parameters of embedded systems are not taken into account. Though one is not hopefully lost when it comes to implementing state-of-the-art methods on embedded devices. By keeping an eye on the most important and impacting factors and analyzing the characteristics of algorithms many of them can be tailored and suited to the hardware platform of choice. Looking at the problem from a more philosophical point of view, it is facing and solving the true problems in the real-world application of algorithms that even make them state-of-the-art. Though embedded systems are commonly used especially in industrial environments it is essential that algorithms become practically usable in real-world. Not until implementation issues are tackled and removed, algorithms can finally make the step from being subject to scientific invention and research to their application in real-world engineering.

1.2 Object Detection and Recognition

For surveillance purposes the focus of research in computer vision is on three big topics mainly, *object detection*, *object recognition* and *object tracking*. An example is given in figure 1.1 where thee basic principles are illustrated. Note, that the *interpretation of events, action or context* is also a research topic closely related to those mentioned above.

Indisputably, object *detection* and object *recognition* are the most important tasks

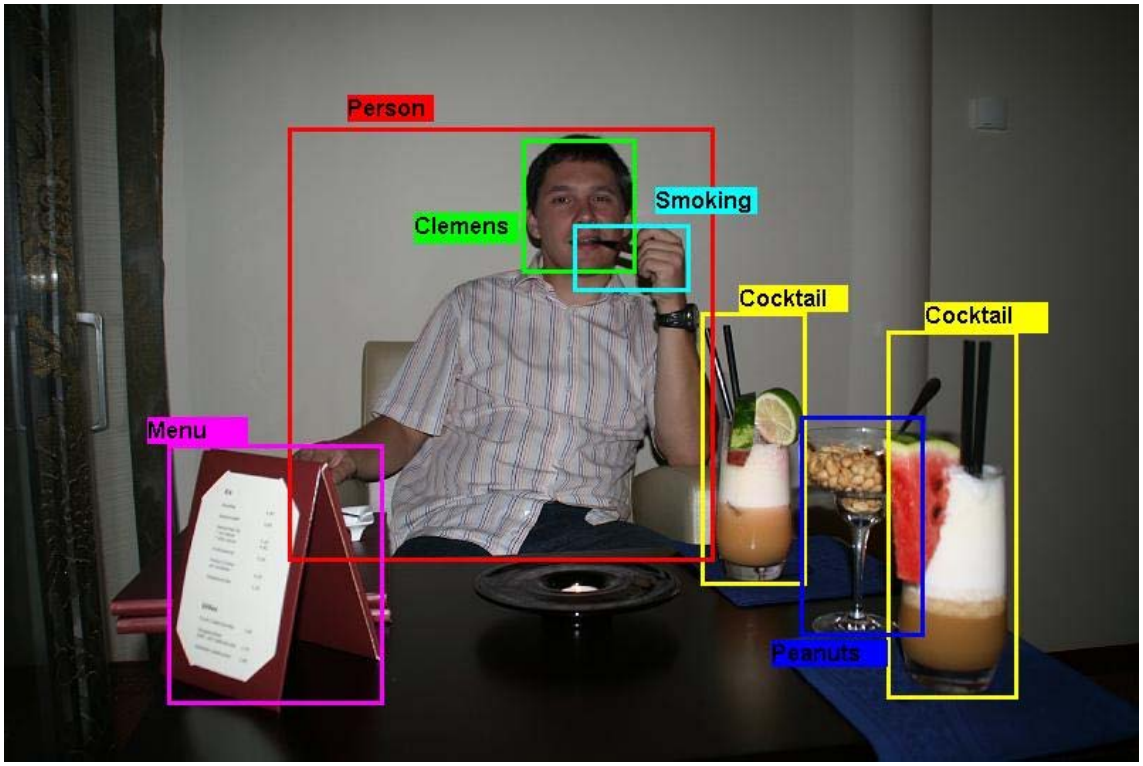


Figure 1.1: This figure illustrates the topics mentioned, *object detection* and *object recognition*. *There is an object* in this case means the detection of a person (or other types like the menu card, the glass with peanuts or the cocktails), *it's that type of object*, in this case this is a person and it is Clemens. The *interpretation of events, action or context* in this case is *what Clemens is doing*, he is smoking. Note that *tracking* would denote, for example, the tracing of Clemens' head in consecutive images when he is moving.

and are based on the processing of raw sensor input. For object *tracking*, it is necessary to incorporate some additional temporal information. Tracking can also be interpreted as pulling together several events on a fixed timeline, *i.e.* several consecutive detections of an object in consecutive frames. To explicate these in more detail,

- **object detection** is the task of becoming aware of the presence of a predefined object in an image and finding its location (typically without perfect segmentation),

- **object recognition** is the task of determining an object to be from a predefined category (*categorization* or *generic* recognition), or classifying an object to be exactly the same object as noticed previously (*specific* recognition), and

- **object tracking** is the pursuit of an object through consecutive image frames, collecting information about its position, its direction of movement and/or its orientation.

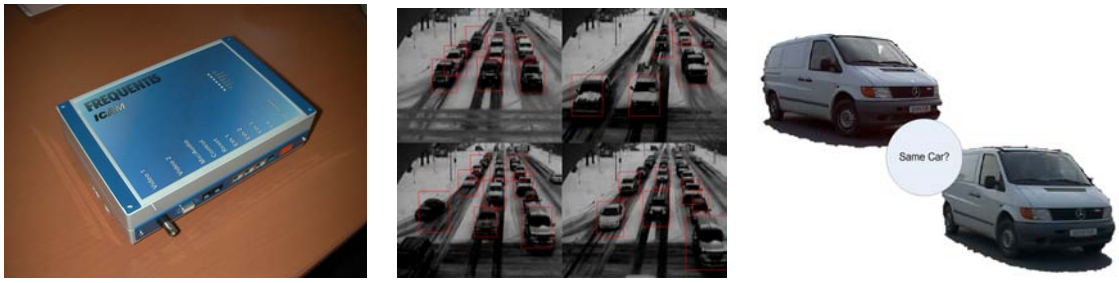
In this work we are concerned with the first two groups of methods only, object detection and object recognition. We assume that when an object is robustly and successfully detected and recognized in single images, computer vision has succeeded. In the following, we will not deal with the huge number of existing approaches to object tracking. For an overview, the reader is referred to the thesis of Michael Grabner [86].

1.3 Problem Statement

A lot of object detection and object recognition algorithms exist, however, varying in their number of advantages and disadvantages. Only a few approaches deliver promising results in terms of accuracy, and an even smaller group of approaches has been proven to be robust and applicable in real-time. Finally the smallest subset of algorithms fulfills the criteria of robustness, real-time applicability and precision, and additionally fits onto a platform, featuring only limited amounts of memory and computational resources.

This is the group of algorithms we are concerned with in this thesis. We focus on statically mounted devices only, which means that the camera is fixed and does not move around during operation. However, the applicability of the algorithms investigated and developed is not necessarily limited to this application domain. Smart cameras offer undisputable benefits in terms of power consumption, the possibility to co-locate the sensing and the processing task, and the robustness against environmental stress, amongst others. We want to investigate the current state of the art in object detection and object recognition from the area of computer vision in the context of smart cameras. Our goal is to take benefit of recent advances in computer vision research and smart camera development, and to overcome well-known drawbacks of embedded systems in algorithm design, such as reduced memory resources. We focus on the investigation of existing approaches and aim at discovering and declaring general rules for deploying state-of-the-art computer vision algorithms on existing smart camera setups. In other words, for object detection and object recognition on an embedded system, algorithms have to be adapted and tailored to meet real-time constraints and environmental restrictions without significant loss in robustness and performance. Our intention is to find and define general methods to achieve this goal, while we reemphasize, that the focus is on computer vision for smart cameras platforms for surveillance purposes.

Regarding the algorithms from both areas in detail, in the case of object detection the localization procedure should be fast enough to perform in real-time, taking advantage of architectural properties and advantages of the dedicated hardware platform. For object recognition the goal is to achieve high performance in terms of recognition accuracy. Even more important, operation should perform under real-time constraints to be applicable in different situations, like reacquisition of objects or information retrieval. In both cases, we consider our approaches to be *soft real-time* systems, which means that the completion of the task after its deadline is not critical, but should be



(a) (b) (c)

Figure 1.2: (a) Prototypical smart camera platform. (b) Example images of vehicle detection in traffic scenarios. (c) Object Recognition given two vehicle images.

avoided[1] [126, 141]. In the case of object detection, this means that the detection process should perform at frame rate, otherwise frames of the input stream are dropped. In the case of object recognition, we consider the system to perform in real-time, if the recognition result for a given input sample can be determined within a time span of several hundred milliseconds, allowing for recognizing several thousand objects per hour.

1.4 Contribution

The first major contribution is the investigation of the well-known Viola-Jones algorithm on DSP-based hardware platforms [250]. A prototypical DSP-based system was chosen, because DSPs form a large group of embedded processors, offering a great amount of flexibility and performance in equal shares. Moreover, we focus on the development of algorithms for usage on existing hardware platforms, rather than on hardware design, thus choosing a predefined DSP-based architecture as a hardware framework is reasonable. We are the first to analytically and experimentally investigate the properties of the Viola-Jones detection algorithm for usage on a prototypical embedded hardware platform in the context of surveillance. We propose special adaptations of the algorithm to meet real-time constraints. Moreover, we investigate and discuss the main problems in using Boosting based detectors on DSP-based systems, to find general rules for realizing and applying recent visual object detection technology on smart cameras.

The second major contribution is the analysis and realization of several state-of-the-art object recognition algorithms for deployment on our DSP-based hardware system. The DoG keypoint detection algorithm and the SIFT / PCASIFT descriptors [115, 142] are evaluated with respect to their suitability for DSP based

[1] In contrast, in a *hard real-time* system, the result of a task completing after the deadline is considered useless, and exceeding the time limit might lead to critical system failure.

hardware setups. Given the properties of these algorithms a number of different methods to circumvent the problems faced are proposed to better suit the approaches to the environmental restrictions of a given hardware setup and to allow for real-time performance. Furthermore, we investigate the properties of the individual algorithms and discuss some key properties to realize a powerful object recognition system. We discuss and analyze the tradeoffs, given limited computational and memory resources on smart cameras on the one hand, and the idea of deploying moderate-scale object recognition capabilities for autonomous systems on the other hand.

We show the suitability of our approach on the challenging task of vehicle detection and recognition on public streets. On the one hand, we will focus on detection accuracy and real-time capabilities, on the other hand we will also focus on high performance, specific object recognition given appearance information only. The advantages of our approach are described in detail, while our algorithm investigations and adaptations primarily focus on the meeting of real-time constraints. We treat memory consumption as an important issue during algorithm development and develop methods to get along with restricted resources. However, we do not discuss all other issues of memory management, like memory mapping or caching strategies in detail here. We proof our algorithms suitable in the context of scalability in a, possibly huge, network of smart cameras, rather than on a single unit. In this respect, we mainly focus on the communication requirements between individual smart cameras, beside the major aspects of calculation accuracy and real-time performance. We emphasize, that we do not treat other special issues of sensor networks in detail here, such as distributed computing or determining communication paths. However, we proof our approaches suitable to be part of a software framework for sensor networks in surveillance scenarios.

1.5 Thesis Outline

This thesis is organized as follows: in the next Chapter 2 an introduction to smart cameras is given. We will list the main aspects of smart camera development and, furthermore, discuss the main observations and criteria to qualify and classify algorithms suitable for embedded systems. In Chapter 3 an overview about related work in the area of object detection and object recognition is given. We will divide the individual approaches into several groups and discuss their properties, their advantages and drawbacks. The main two applications in respect of our embedded system are described in Chapters 4 and 5. In Chapter 4, a detailed discussion of the Viola-Jones object detection approach on DSP-based embedded platforms is given. In Chapter 5, we propose a powerful object recognition system for deployment on smart cameras and apply it to the task of vehicle reacquisition. Finally, some critical and concluding remarks are given in Chapter 6.

If the automobile had followed the same development cycle as the computer, a Rolls-Royce would today cost $100, get a million miles per gallon, and explode once a year, killing everyone inside.

Robert X. Cringely
InfoWorld magazine

Chapter 2

Smart Cameras

Smart Cameras are devices consisting of an embedded computer, a single or multiple image sensors, and some type of interface for external communication. However, the analog or digital video sensor must not necessarily reside on the same circuit board, but can also be connected over some analog-to-digital converter (A/D converter) or a frame grabber. Nevertheless for a typical smart camera the video sensor and the main board are residing in one common housing and a connection is often available through a dedicated video grabbing device or a special video port of the processing core. The main task of a smart camera is to perform a given task automatically and autonomously. Smart cameras offer undisputable advantages in terms of power consumption and physical size, compared to usual PCs. Moreover, they offer the possibility to co-locate the sensing and the processing task, they allow the extraction of valuable information at site and the transmission over low-cost, low-bandwidth communication channels. A qualitative overview of the most prominent embedded system technologies and their properties is shown in Figure 2.1[1]. In the following we give a short introduction to the current state-of-the-art in smart camera research in Section 2.1. An overview about the technology of embedded systems, followed by a set of smart cameras proposed in the literature is given in Sections 2.2 and 2.3. In Section 2.4, we describe general aspects of software development on DSPs. Finally, in Section 2.5 we introduce the TRICam, the DSP based hardware platform used for our software development throughout this thesis.

2.1 Smart Camera Research

The development topic of smart cameras was founded and formulated in the beginning of the new millennium as an important part of embedded system research [38, 121, 264]. In fact, smart cameras are pushing the design space in many dimensions and combine multiple interdisciplinary areas, like VLSI design and hardware/software co-design.

[1]Note, that this is just a rough map to illustrate the diversity of embedded system technology, and that the listing of different architectures makes no claim to be complete.

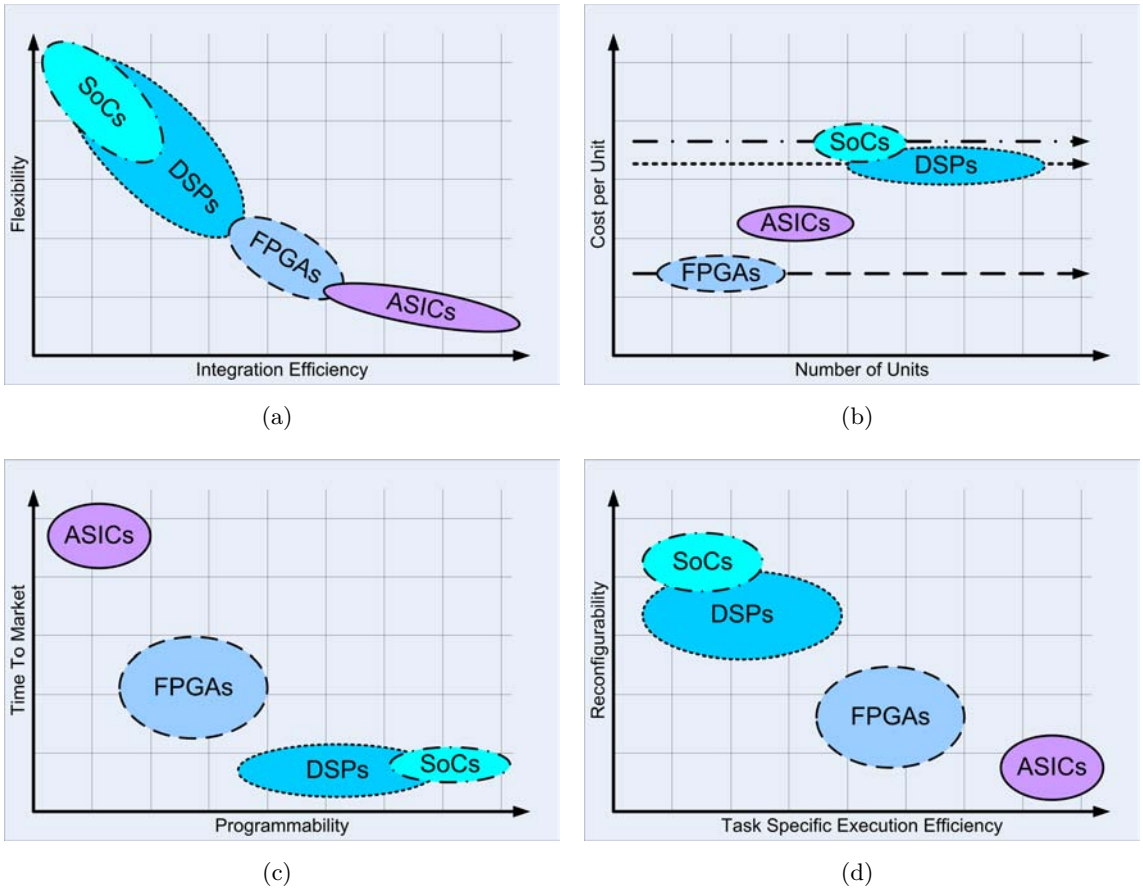


Figure 2.1: (a) Qualitative flexibility and integration efficiency tradeoff for different embedded system technologies. (b) Qualitative unit cost and production volume tradeoff for different embedded system technologies. (c) Qualitative programmability and time-to-market tradeoff for different embedded system technologies. (d) Qualitative platform reconfigurability and task specific execution efficiency tradeoff for different embedded system technologies.

There are many reasons for employing embedded technology in smart sensors. To summarize the most important ones, embedded computers allow for a cost-effective, large scale deployment of devices, coevally offering robustness against adverse environmental conditions, low power consumption and the possibility to co-locate the processing and the sensing task to process data at site. Moreover, special attention is given to the real-time capability of embedded systems and to an incremental miniaturization of devices. However, in former times, computational work, *i.e.* processing data, was done mainly on servers or computing engines, which were located in areas or rooms with restricted access. This circumvented the problem of physical attacks on these systems by offenders. Due to the distributed nature of smart cameras, they are physically accessible to attackers, which poses a big security issue. Moreover, additional attacks on smart cameras, or

entire camera networks, are now possible, such as battery or power attacks to drain or cut the power reservoir of a node, or attacks over wireless communication channels on all protocol layers to disturb or interrupt inter-node communication. Clearly, all of these advantages, drawbacks and security issues are subject to ongoing research, together with the development of suitable concepts for building high performance, small, yet robust and reliable smart camera platforms. Note, that in the following, we will mainly focus on embedded technology for usage in smart cameras for surveillance purposes, and will greatly omit issues of mobile devices, ubiquitous computing or self-configuration of communication and sensor networks.

The list of applications of smart cameras is diverse, such as the background of developers working with smart cameras. For a short time now, workshops and conferences are issued to band together engineers and researches from all different development areas, which mainly include the hardware community, the image processing and computer vision community, the sensor network and sensor fusion community and also the machine learning and artificial intelligence community [67, 68, 106]. A number of different prototypical smart camera setups have been developed, which will be discussed in detail in Section 2.3. To give a short and not exhaustive list of main research topics and dedicated applications, one can refer, for example, to gesture recognition [264, 267], smart home care [118, 260], and all sorts of surveillance applications, like people and vehicle detection, tracking and counting [11, 58, 184, 188]. Current, and also future trends clearly focus on issues of collaborative and distributed computing in smart sensor networks, on sensor fusion and on hardware architectures for solving complex algorithmic tasks on high performance embedded platforms.

2.2 Embedded System Technology

As already mentioned before, smart cameras are based on embedded system technology. The group of embedded systems is manifold and varies in its amount of flexibility between each category. The biggest sets are *Reduced Instruction Set Computers* (RISCs), *Digital Signal Processors* (DSPs), *Field Programmable Gate Arrays* (FPGAs), and *Application Specific Integrated Circuits* (ASICs). Two other groups exist which are sharing characteristics of embedded systems beside other special properties. These groups are *Microcontrollers*, which also include the set of *System-on-Chip* platforms (SoCs), and the recently emerging area of *Graphics Processing Units* (GPUs). A short characterization of each single domain is given hereafter, followed by notes on general selection criteria. For a more elaborate introduction to processors and embedded computing, the interested reader is referred to the book of Wolf [262]. Note that it is not possible to mention all types of methodologies, design issues, development groups and reference implementations here, thus we aim at giving a general overview and only focus on aspects relevant for our own work.

2.2.1 Reduced Instruction Set Computers (RISC)

The term RISC was introduced to differentiate from the *Complex Instruction Set Computer* (CISC) architectures. The RISC was introduced in the late 1970s based on several design principles to create a new architecture, originally to facilitate the use of optimized compilers for the generation of machine code. The instruction set should contain only simple instructions, being decodable by the CPU within one clock cycle. Furthermore, the use of register files and pipelining allows for execution at high frequencies, also factoring out slow memory accesses. For an early introduction and review of RISC processors, the reader is referred to the article of Patterson [181].

Main representatives of RISC processors nowadays are the ARM processor family [220], the MIPS architecture [234] and the PowerPCs. Manufacturers of RISC processors mainly include Freescale Semiconductors, IBM, AMCC and MIPS Technologies.

2.2.2 Digital Signal Processors (DSPs)

Another big and important group of embedded processors is formed by DSPs. Originally developed for one-dimensional signal processing tasks in the real-time computing domain for telecommunications, they are now more and more emerging into the image processing domain. To allow for fast filtering and folding operations, one important feature of DSPs was an onboard multiplier, also providing a multiply-accumulate (MAC) instruction. This is still a common feature on nowadays DSP architectures. Recent DSPs also often include specialized instructions for more evolved digital signal processing operations, such as Viterbi en-/decoding for example. Being related closely to general-purpose computing systems in respect of their programming flexibility, the newer series DSPs are featuring *Very Long Instruction Word* (VLIW) and *Single-Instruction, Multiple-Data* (SIMD) technology, which means that multiple functional units can be handled concurrently. Software for DSPs is usually written in a high-level language like C or C++ which makes development of applications relatively straight-forward and efficient. Optimization of programs is done during compilation which means that the developer is only able to influence optimization at a moderate level. While DSPs have been developed working mainly in the fixed point domain for almost two decades, recently DSPs equipped with a *Floating Point Unit* (FPU) have become more attractive. However, fixed-point calculation is still the predominant domain, as more evolved DSP architectures clearly come at a considerably higher price due to the increased hardware complexity. Needless to say, that the inclusion of FPUs in DSPs also comes at considerably higher energy consumption and increased chip area.

First prototypes of DSPs were proposed in the late 70s by Intel and AMI. Later the principles were refined and the first full-features DSPs were presented by AT&T and NEC in 1980. A big success was the introduction of the first DSPs from Texas Instruments which is still holding on. Today TI is the biggest manufacturer of DSPs beside other producers like Motorola and Analog Devices. The range of pricing for a DSP ranges

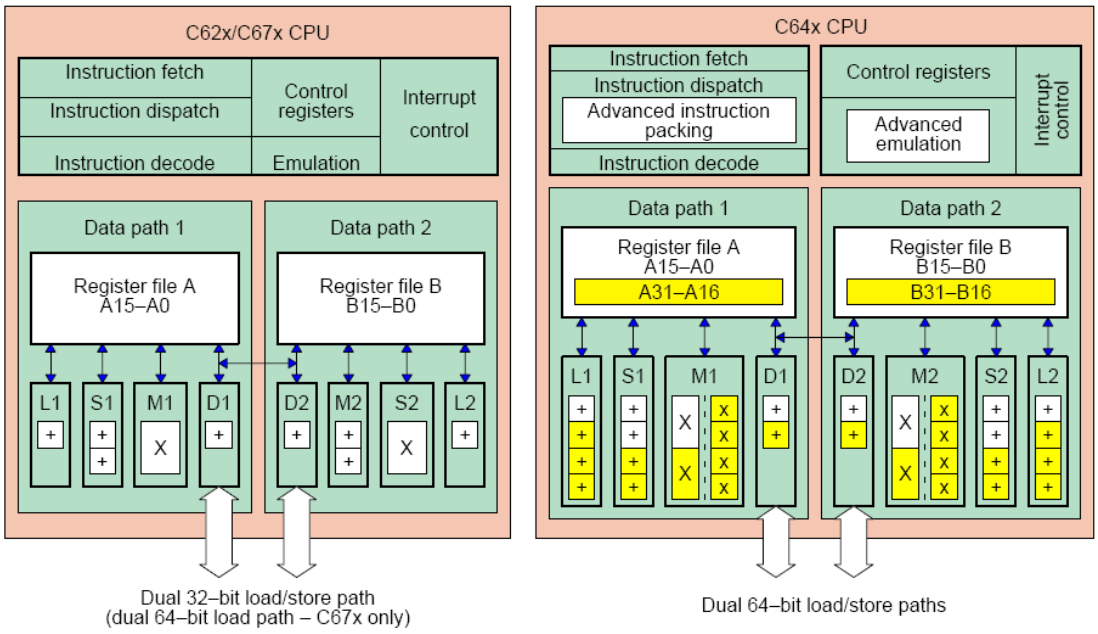


Figure 2.2: Block diagram of the C62x/C67x and the C64x DSP architecture from TI, taken from `TMS320C64x Technical Overview` [237].

from a few up to a few hundred dollars. Likewise the band of power consumption of DSPs ranges from $50mW$ up to $5W$. The variety of DSPs is manifold and special type processors are available for almost each specific application, thus it is up to the developer make the right selection. Concerning a very popular group of DSPs especially suitable for video processing, the TI TMS320C6xxx series a block diagram of the basic processor design is depicted in Figure 2.2.

2.2.3 Field Programmable Gate Arrays (FPGAs)

Field Programmable Gate Arrays are semiconductor devices which contain a huge number of logic elements and connections inbetween. The elements are called *logic blocks* and can be programmed to perform a logic operation with limited complexity, from simple AND gates up to full multipliers or even complexer functions. The name *field programmable* denotes the possibility to program interconnections and logic blocks after manufacturing *in the field*. Many different types of FPGAs exist, which can be chosen according to design and security issues. FPGAs based on *Static Random Access Memory* (SRAM) are programmed at power-on and are booted from some external functionality. *Erasable Programmable Read-Only Memory* (EPROM) devices can be programmed multiple times but is usually programmed during manufacturing. The program can be erased by exposing the device to ultra-violet light, and can be re-programmed afterwards. *Electrically Erasable Programmable Read-Only Memory* (EEPROM) technology based FPGAs can be erased

and re-programmed electrically which makes the need for ultra-violet light exposure obsolete. *Fuse* and *Anti-Fuse* based FPGAs are working on opposite electrical principles. In the first case programming is done by breaking conductive connections and in the latter case connections are established if the applied current is exceeding a specified limit. The latter technology is much more common in the world of *Integrated Circuits* (ICs), however, FPGAs based on both techniques can be programmed only once.

The programs for FPGAs are usually written in a *Hardware Description Language* (HDL) such as ABEL, VHDL or Verilog. Given the description of the desired functionality a number of steps have to be taken to finally place the functionality on the device. To speed up this process different development tools exist such as SystemC. An important feature are the libraries of function blocks and macros that can be used to further speed up the design of programs. The list of manufacturers of FPGAs contains companies like Altera, Atmel, Actel and Xilinx among others. FPGAs can be used to create complete devices such as DSPs or to perform computationally expensive tasks in hardware, such as *Fast-Fourier-Transform* (FFT) or video en- and decoding. However, FPGAs are also mostly used for prototyping ASICs, and as their capabilities and speed increases nowadays complete SoC outlines can be fabricated on FPGA technology.

2.2.4 Application Specific Integrated Circuits (ASICs)

A group of devices dedicated for high volume production are *Application Specific Integrated Circuits*. These devices only contain the components that are very specific and necessary for performing only one given task. Different design and manufacturing methodologies exist, ranging from full custom ASIC design to structured ASIC design. The major differences are in the usage of predefined macros, cell libraries and *intellectual property* (IP) cores. As in FPGA design these predefined modules can be used to speed up the development process trading against chip area and cost. While in custom ASIC development the granularity of development is to define the characteristics of metal layers in the semiconductor, in structured ASIC design a set of predefined characteristics and their semiconductor implementation are given in advance (which in turn reduces development time considerably). At the very high-end structured ASICs are also sometimes referred as SoCs if complete DSP cores and modules for interface functionality are included in the design.

The major benefits of ASICs are their asymptotically decreasing cost when manufactured in high numbers, a little increase in speed over FPGAs and a decrease in power consumption to a minimum. However, the biggest problem with ASICs is their design and production cycle and the non-recurring engineering costs, which can easily exceed 1 million dollars. Furthermore the static layout causes big problems as a redesign due to bug fixing is a costly exercise. However, as development tools get better also ASIC design and development becomes easier. As an example mobile phones are a representative application for ASICs as the worldwide sale in high volumes and the well-defined task to

perform easily balances the concerns in manufacturing and design. Producers of ASICs include Altera, Fujitsu, Infineon or NEC among others.

2.2.5 Graphics Processing Units (GPUs)

Graphics Processing Units are devices especially developed mainly for graphics generation and visualization. A big drive in this domain is the gaming industry forcing developers and manufacturers to build faster, more powerful engines for more realistic representations and higher throughput. The major goal is to generate output that is as realistic as possible. Clearly, in relation to computer vision the main striving behind this technology is entirely contrarian as the main task is information generation in the first case and information extraction in the latter one. However, the consistent progress of these devices and the introduction of a concept called *General Purpose Computation on Graphics Processing Unit* (GPGPU) in 2000 make these devices also usable for data processing and computer vision. The idea behind GPGPU is to use the highly parallel vector architecture of GPUs to perform highly parallel procedures on these devices. Clearly this leads to a high increase in performance coevally leaving more room for other tasks on the main processor. Since the introduction of the *Compute Unified Device Architecture* (CUDA^{TM}) by NVidia in 2006 and the release of Software Development Kits (SDKs) for writing procedures in C language, the development of programs for GPUs has gained even more attraction as before. Needless to say that as a consequence of this invention the development of application or porting has become a lot easier. For a survey of state-of-the-art in GPGPU the interested reader is referred to [175].

The currently market dominating manufacturers of GPUs are AMD and NVidia. AMD is producing the ATI Radeon series and NVidia is pushing his GeForce series of video cards. Less important for real visually realistic graphics processing are devices manufactured by Intel mainly integrated as onboard devices. In principle the development of GPUs is not subject to any restrictions of embedded systems, such as the necessity of low power consumption, the desired operation under a given range of temperature or the need for a small form factor. In contrary, GPUs are contradicting almost all characteristics of embedded technology. They waste a lot of energy, need big fans for cooling and the ongoing trend is against more powerful and voluminous sizes to deal with even higher performance. However, GPUs share one important property which is the principle of *pipelining*. Data processing is done similar as in DSPs, as pipelining is the magic word to achieve high throughput and efficient execution. Thus GPUs are called a member of the embedded system group, even if most of their characteristics state clear contradictions against embedded system philosophy and though the real membership of GPUs is still under debate in the embedded system community.

2.2.6 Microcontrollers and System-on-Chip platforms (SoCs)

Microcontrollers and *System-on-Chip* platforms form the biggest group of embedded systems nowadays, being part of almost any electronical device in our environment. Microcontrollers are some type of microprocessor, additionally including memory resources for program and data storage, timers and external, typically serial, interfaces. Microcontrollers emphasize the aspects of cost-effectiveness, high integration, low power consumption and self-sufficiency.

While microcontrollers are single physical packages aimed at performing small-sized tasks, the term System-on-Chip refers to the integration of an entire system - or all electronic circuits needed for performing a given application - into one single chip. Usually, System-on-Chip platforms are combinations of a core processor, a set of interfaces and a selection of external controllers in one physical package. Many SoCs consist of a *General Purpose Processor* (GPP), which can be a RISC processor like an ARM or a PowerPC, or can also be a x86-based processor like the Intel Celeron M. While the GPP is mainly included to perform operation system tasks, SoCs mostly contain one or more DSP units dedicated to the real signal processing tasks. These types of devices are also called *Media Processors*.

In the last few years, media processors are becoming more and more important as they are integrated for active video processing or streaming in many devices of everydays use. Target applications are present especially in mobile phones, PDAs, handheld video players, set-top boxes and in automotive engineering. In the context of image processing, these devices feature special properties and capabilities, such as video en- and decoding support in hardware and limited controlling mechanisms for external devices such as cameras. Though, on these special platforms only little image processing capabilities are usually available as they are tailored to meet a special application in a single or a set of appliances. However, the principles of SoCs make them also appealing to the designers of Smart Cameras as all necessary features are provided by a SoC solution. Peripherals for system integration, interfaces for connecting video sources and an extendable architecture to include more DSP units for even higher signal processing power make this set of devices a good choice as a base for building a video processing system.

Several manufacturers of Microcontrollers and SoC platforms exist. The biggest ones are TI, Analog Devices and Intel as a vendor of the XScale driven devices. Further vendors are AMD which is producer of the Geode processor, Atmel, STMicroelectronics, Freescale and Cirrus Maverick among others.

2.2.7 Technology Selection Criteria

The architectures described above, especially DSPs, FPGAs and ASICs, have their own field of application, their own roots and goals and obviously several benefits and disadvantages. The choice which technology to choose for a self-made prototypical setup are manifold, and most times, the final decision is based on a detailed planning process apply-

ing Hardware-Software Co-Design principles. General rules of thumb to select the right technology for a dedicated task were proposed in the work of Kisačanin [40]. The main aspects considered here, are

- the available time for development (Time-To-Market),
- the expected and required funds,
- the targeted area of application, and
- the expected and desired volume of production.

For a relatively small volume of less than 1.000 units per year general-purpose computers and FPGAs are a good choice considering their price and the reduced need for hardware dependent development. At the high end of production of more than 100.000 units per year, the usage of a tightly tailored ASIC is reasonable as the initial development costs and the need for special developing are notably high, but the deployment in high volumes justifies adequate investments. In the mid-range DSPs and media processors are good selections as they form a good trade-off between programming flexibility and the amount of necessary specific development.

Most smart cameras are based on a combination of various of these devices manufactured in SoC technology because a smart camera implicitly needs peripherals and interfaces but also signal processing power delivered by DSPs. Thus in our terminology smart cameras are SoC platforms as their properties are inherently linked to SoC principles.

2.3 Smart Camera Platforms

Recently a number of Microcontroller and SoC based vision systems have evolved especially dedicated to the task of image processing. The EyeQ vision platform from Mobileye is a prototypical smart camera platform with a GPU, a DSP and two CMOS camera interfaces. Another commercially available vision platform is a PCI board called "'Sarnoff Arcadia I"' by PyramidVision^{TM}. An additional indication for the suitability of SoC concepts for smart camera design is the variety of developments of smart camera platforms reported in the literature. Among others these include the VISoC [5], the CMUCam3 [197, 198, 199], the WiCa [108, 123], the Cyclops platform [190], the SmartCam platform [31, 32], the TRICam [10] or the MeshEye^{TM}system [94]. We will outline their design goals and capabilities in the following. Some of them are depicted in Figure 2.3.

2.3.1 The CMUCam

The ***C**arnegie **M**ellon **U**niversity **Cam**era* was originally presented in 2002 as a low cost embedded vision platform being commercially available right from the beginning for a little more than $100. The first version proposed in 2002 by Rowe, Rosenberg and Nourbakhsh

(a) (b) (c)

Figure 2.3: (a) The CMUCam3 platform from the Robotics Institute at the Carnegie Mellon University, taken from [197]. (b) The *Wireless Camera* (WiCa) from NXP, taken from [3]. (c) The MeshEye^{TM}platform from Stanford University, taken from [94].

was based on three chips mainly, a Omnivision OV6620 CMOS camera sensor, a Ubicom SX28 microcontroller and a simple level shifter for serial communication [198]. Without additional logical circuits, the camera sensor was directly connected to the microprocessor, which was running at 75 MHz and possessed 136 byte of SRAM (!). This was lowering the potential speed of the system as the read-out of every single pixel has to be synchronized to the microprocessor. The vision system still was able to run at 16.7 frames per second on 143x80 pixel images for a simple color blob detection application.

In 2005, the second generation of the platform was presented. One enhancement compared to the original version was the use of a Ubicom SX52 microcontroller still running at 75 MHz but now offering 262 bytes of SRAM [199]. The major advantage of this design was the use of an Averlogic AL422B frame buffer with 384k bytes of FIFO memory, which was used to buffer the incoming frames between the CMOS camera and the microprocessor. Thereby decoupled frame acquisition and processing could be achieved, coevally allowing for full-frame read-out at once instead of single-pixel read-out. The number of proposed algorithms was extended to include color statistics calculation and frame differencing beside color blob detection. The power consumption at runtime is about 850mW, while the price for the platform was about $199.

The third and actual version of the system is available since the beginning of 2007 for approximately $239 [197]. Now another microcontroller is used, namely a NXP LPC2106 which is a 32-bit 60 MHz ARM7TDMI processor with 64k bytes of RAM. Another big advantage is the use of a *Multimedia Card* (MMC) interface which can be used to read and write files. The overall system consumes between 300 and 500 mW of power, depending on the operation mode. The development of software can be done using open source programs and an open source compiler, while all libraries for image compression or convolution are

released on an open source policy. This allows for easy and cheap development without the need for expensive commercial compilers and development software. An implementation of the Viola-Jones algorithm exist, which is able to detect faces at a maximum size of 60x60 pixels in 176x144 pixel images at 1 Hz.

2.3.2 The WiCa

The ***Wi****reless* ***Ca****mera* was developed at NXP Semiconductors and is a smart camera platform featuring two connectors which can be used freely to capture from one single camera or to form a stereo camera setup [108, 123]. In detail, the hardware setup consists of one or two color cameras, a massively parallel SIMD processor, a microcontroller for host control, a block of dual-port RAM and a communication module. The cameras deliver images in VGA resolution of 640x480 pixels. The SIMD processor is a Philips Xetal IC3D device whose core is a linear array of 320 single RISC processors for low-level image processing tasks. The ATMEL 8051 microcontroller is dedicated to multiple tasks. It mainly controls the IC3D processor, communicates with the RISC processor array and takes care of program flow and video synchronization. The dual-port RAM available on the platform provides a total of 128k bytes of memory, which is separated into two banks with a block of 64k bytes each. For communication purposes a Aquis Grain ZigBee module is included, which is a very small radio system allowing wireless communication within about 5 m distance and up to 10k bytes per second transfer rate.

Programs to be run on the WiCa platform are written in C++, or rather using an extended C language (XTC). Depending on the level of the algorithm to perform the developer has to dedicate the operations to the suitable processor during program design. Low-level operations like contrast stretching or convolution are dedicated to the massively parallel Xetal processor, while high level reasoning is dedicated to the DSP processor. Several algorithms have been proposed for face detection or human gesture analysis where the main focus is to exploit algorithm level parallelism [108, 266].

2.3.3 The Cyclops

In respect of large sensor networks and energy-aware smart sensors, Rahimi *et al.* proposed a highly power-efficient smart vision platform called Cyclops [190]. The camera mote is mainly based on a CMOS camera unit, a microcontroller unit, a *Complex Programmable Logic Device* (CPLD), an external SRAM memory block and flash memory. The camera module is a 352x288 resolution ADCM-1700 CMOS camera from Agilent Technology and can be configured to deliver 8-bit grayscale, 16-bit color or 24-bit color images. The microcontroller unit is a ATMEL ATmega128L running at 7.37 MHz, which offers 4k bytes of internal SRAM memory and is extended to 64k bytes total memory on the external SRAM block. The processor is mainly used to time and coordinate external and internal events and interrupts. The Xilinx XC2C256 CoolRunner CPLD serves as a frame grabber which can fulfill the high demands on fast data transfer and address generation.

The CPLD can furthermore be used to perform restricted operations at capture time which are based on a set of available libraries. These libraries mainly include operations on images and matrices respectively, like addition, subtraction, scaling, thresholding, Sobel filtering, histogram and statistics calculation. More advanced libraries include functions for background modeling and a set of methods for coordinate conversion between world and an image coordinate systems.

Two applications are proposed to proof the suitability of the platform, object detection and hand gesture recognition. The object detection algorithm is based on background modeling and performs at about 4 frames per second on 128x128 pixel images. The gesture recognition algorithm is based on the matching of orientation histograms and can differentiate 5 different gestures from the American Sign Language (ASL) at about 2 frames per second and 92 % of recognition rate. The power dissipation of the platform is dependent on the operation mode. The platform is designed to work in larger sensor networks, thus it is highly power-aware and energy consumption is in the range of a few mW.

2.3.4 The SmartCam

The SmartCam is a low-power, high-performance embedded vision system mainly consisting of a set of individual components [31, 32]. The prototype is based on an Intel IXDP425 development board equipped with a XScale network processor running at 533 MHz. The board features 256M bytes of RAM and four PCI slots, an on-chip ethernet connection and multiple serial ports amongst others. The sensor module is a Kodak Eastman monochrome CMOS camera which delivers images up to VGA resolution at 30 frames per second and is connected via a FIFO ordered memory. For the main processing task, each PCI slot can host an ATEME Network Video Development Kit (NVDK) board which consists of 264MB of memory and TI TMS320C6416 DSPs running at 1 GHz. Communication channels with other smart cameras, external devices or hosts can be established using wired ethernet, IEEE 802.11 wireless LAN or wireless GPS/GPRS radio.

The plausibility of the concept was demonstrated on a vehicle detection and tracking application for tunnel safety [32, 33]. However, the focus of current research is on tracking in multi-camera networks [187], communication, distributed computing and distributed task allocation.

2.3.5 The MeshEye™

The MeshEye™platform was proposed as a single node for larger distributed smart camera networks [94]. Developed at Stanford University, the system is especially dedicated to the area of surveillance and consists of a low-resolution stereo camera setup and a single high-resolution color camera. The main platform can host up to 8 low-resolution imagers, however, the prototype contains only two 30x30 pixel optical mouse sensors and one 640x480 VGA resolution color sensor.

The main processing core is an Atmel AT91SAM7S microcontroller which is a ARM7TDMI 32-bit RISC processor running at 55 MHz. Similar to the CMUCam3 platform the system contains a MMC/SD card interface allowing for easy memory expansion. Another important feature is the use of a TI CC2420 2.4 GHz IEEE 802.15.4/ZigBee-ready RF transceiver to connect the mote to other notes in a larger network.

The main task of the camera mote is for object detection and tracking in surveillance applications. To perform this task both low-resolution imagers perform a motion estimation step based on a background model and producing difference images. The template of a moving area in one imager is sent to the other imager to estimate the distance of an object by stereo matching. This distance estimation is rather limited to a few meters due to the relatively small baseline in the current prototype, however, making the setup still suitable for indoor and limited outdoor usage. The detection of an object triggers the acquisition of a high quality color image using the high-resolution sensor, while the rough *Region of Interest* RoI information is already available from the former stereo-matching step. Additional recognition algorithms may finally be applied to the high resolution image of the object detected.

The major focus of the MeshEye^{TM}development is energy awareness and the intention to extend the principles to larger networks of smart cameras. To allow for autonomous battery powered functionality, parameters can be tuned for highly energy efficient operation and to allow for independent processing from a few hours up to a range of a few weeks. As there is currently no intention to include high-performance computing units like DSPs onto the platform, future development of vision algorithms will have to focus on tailoring the methods to meet the computational resources of the currently available RISC processor.

2.3.6 Summary

Smart cameras form a relatively young field of development. Hence it is not surprising, that there exist only a few platforms, which vary strongly in their properties and features. Some of the systems focus on low energy consumption, while others offer high computational resources, coevally requiring a lot of power. Finding the right and meaningful setup and configuration is matter to ongoing research. However, one can summarize that, in general, smart cameras are platforms with

- one or multiple video sensors, which are growing steadily in their resolution,
- a microcontroller, which is performing low-level operation system tasks,
- memory resources and a high performance computational engine, which is a DSP in the majority of cases, and
- some type of communication module, which is mostly wired or wireless ethernet.

Although a change in the dimension of the individual parts is to be expected, *e.g.* the overall size of platforms and the performance of the single components, we can state that the general issues of smart cameras will remain the same in the foreseeable future. Compared to usual computers, many embedded systems are already competitive - or even superior - in terms of computational power and memory resources. Coevally, they share the relevant properties of embedded hardware, like low power consumption or cost-effectiveness. Nevertheless, the main problems are the missing of memory management units and the lack of powerful tools, which allow for efficient programming and fully exploiting the benefits of embedded hardware. Thus it is of major importance to focus on hardware related development of algorithms, which allows for compensation of these drawbacks of embedded hardware compared to standard computers.

2.4 Aspects of Software Development on DSPs

Different design methodologies have been proposed in the literature for computer systems, from covering solely the software or the hardware design process, up to defining highly interrelated co-design processes [262, 263]. In simple terms, there are two ways of hardware related algorithm development to solve a given task. The first one is the classical way of development. Given a fixed hardware setup, a solution is acquired on common PCs, initially neglecting the special properties of the target platform. Then, the solution is tailored and modified, until it can be ported onto the target hardware, meeting all underlying restrictions and requirements. The second way of development is to focus on the suitability of the target platform for special algorithmic operations, coevally considering properties of software algorithms, and to compile an overall solution. This way of development is also known as *Hardware-Software Co-Design*, which has become a very important research topic in the last years [151, 263].

Clearly, both approaches have advantages and drawbacks, but are also aiming at different objectives. In the first case existing algorithms are ported onto existing hardware, while in the latter case, the goal is to iteratively design and develop an approach considering both hardware and software concurrently. Usually, approaches of the first kind deliver results with a higher accuracy, while they cannot take much benefit of the underlying hardware[2]. This mainly results from the lack of powerful compilers and tools, which are able to map the existing algorithm onto existing hardware efficiently. Approaches of the second kind are usually much more performant in terms of speed, as they are much more optimized for the advantages of the target hardware, but they are usually less competitive in terms of accuracy.

Clearly, for DSP based systems the usual way of development is the first one, as it is the easier and faster way due to advanced tools and helpers. The success of an algorithmic approach for a given application depends on two things, the principal suitability of

[2]In this context, "accuracy" means the algorithm precision or performance in terms of exactness.

an algorithm to perform a given task, and the suitability and portability of the chosen approach onto the DSP based platform. For developers the big challenge is to sort out those algorithms that deliver high performance in terms of accuracy, and are coevally suitable to run under restricted conditions prevalent on the platform. When this group of algorithms is found, the challenge for the developer is to adapt these algorithms that have proven to be successful on a usual computer such that they perform almost equally well on the platform of choice. Moreover, the adaptations should make the algorithms take benefit of special hardware features like pipelining or VLIW capabilities.

In the following we shortly outline the most important restrictions and also note some general ways to circumvent bottlenecks. We will place general DSP algorithm design rules and also comment on special properties of the TI TMS320C64xx family that are very useful for speeding up suitable algorithmic parts (a block diagram of the DSP unit is depicted in Figure 2.4). In this context, we address the topic of memory management and discuss the main aspects of parallel execution mechanisms and data processing, which are of special relevance due to the parallel architecture of DSPs. Hereafter, we will shortly address the topic of fixed point arithmetic and discuss advantages and drawbacks. Finally we will note some aspects, specific to software optimization for TI DSPs, and comment on possible modifications to source code, to facilitate software optimization.

2.4.1 Memory Management

The amount of available memory on embedded systems is a major concern. However, due to the use of flash memory and the advances in memory manufacturing, the limitations concerning "slow" memory have become less severe recently, but the lack of "fast" memory available is still prevalent. This mainly results from the big difference in manufacturing cost of fast SRAM memory, usually used as cache memory, compared to slow flash or DRAM memory chips installed on external banks. As a consequence, now it is possible to temporarily store a larger amount of data or intermediate results, which generally makes algorithm implementation easier. However, the major considerations on clever memory usage for efficient use of computational power remain the same.

Also memory management strategies for software development on DSP-based systems is a complex topic. Murthy and Bhattacharyya recently published a book dealing with efficient methods for software synthesis [164]. Considerations about memory consumption of algorithms are important aspects in multiple areas of software development for embedded systems. Here we will discuss only those more extensively that are somehow relevant to our work, and leave a detailed discussion of others to the appropriate literature.

2.4.1.1 Prerequisites and Platform Related Issues

The TI TMS320C6414 DSP shares 1024k bytes of internal SRAM memory. Usually a small part of this memory is reserved to hold the program code, which essentially means the algorithm executable. The remaining rest of the block is assigned as data memory.

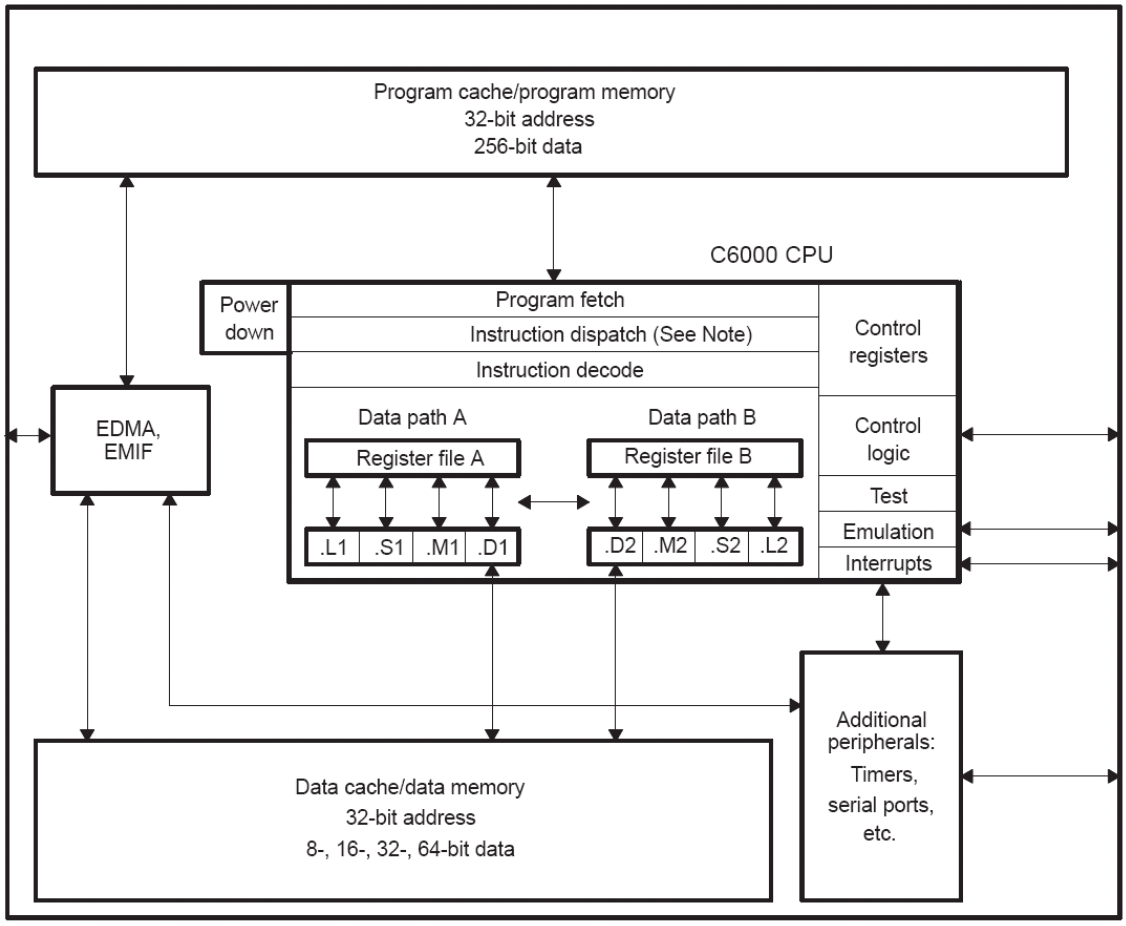


Figure 2.4: Texas Instruments TMS320C64x DSP block diagram. Taken from TI *TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide* (`spru732.pdf`). Note: The instruction dispatch unit has advanced instruction packing.

Access to this memory is very fast and can be done within one clock cycle. Furthermore, linear blocks of memory can be accessed using special data processing techniques that are discussed later. For now, note that it's advantageous to align data linearly in memory if possible because also the corresponding addresses increase linearly. The additional amount of external memory is a single block of 16M bytes of SDRAM on a single bank in our case. Fetching data from this source is considerably slower, typically it takes between 50 and 100 clock cycles for reading or writing a 32-bit word. Without loss of generality one can state that computation on data residing in external memory can not be done efficiently on DSPs using the current state-of-the-art in hardware architecture. Comparing the memory structure of a DSP and a usual computer in terms of access time, the equivalents are the internal memory of the DSP and the working memory of the computer, the external memory of the DSP and the hard disk drive of the computer respectively. Needless to say

that exchanging data between the working memory and a hard disk, or even a magnetic drive, is terribly slow.

2.4.1.2 Operating System Memory Management

Usually there is no operating system functionality available on the DSP except the developer incorporates one explicitly. For example, the *DSP/BIOS Real-Time Kernel* support from TI can be included for the TMS320C6x DSP series. The missing of an operating system essentially means, that noone is managing the memory for us concerning block allocation, fragmentation or swapping between higher-order and lower-order memory ranges. In fact, automatic memory management is one good reason to use a *Real-Time Operating System* (RTOS), such as "DSPnano RTOS" or "Unison DSP RTOS" for example [232]. However, since we are working without these goodies, we can control memory allocation on a very abstract level only. In fact, we can only decide upon placing a memory block in the internal or the external memory area, and if the block is aligned to a word or double-word boundary, or if it is not aligned at all. Needless to say, that we also have to manage the exchange of data between external memory banks and the internal memory resources of the DSP ourselves.

2.4.1.3 Caching and Memory Mapping

The term *cache* is mainly used to refer to any storage managed to take advantage of locality of access [182]. This is a general definition and not only applies to processors, but also to other computer hardware like hard disks, for example. In the context of processors, a cache is a small portion of memory which can be accessed by the CPU rapidly. It is used to store duplicate values which are stored elsewhere and are expensive to fetch, or were computed earlier and are expensive to recompute (compared to reading them from the cache). The CPU can store frequently used data and rapidly access it, if it is located in the cache, which is called a *cache hit*. If requested data is not located in the cache, a *cache miss* happens and the data has to be fetched from other memory areas and is inserted into the cache. Because memory stalls due to cache misses are an important issue and usually waste a large portion of CPU performance (*i.e.* stalling the CPU for a considerable number of cycles), it is advisable to use a clever caching strategy. A lot of attention was given to cache design in general-purpose computers, and many issues directly translate to embedded system design. Because of the special interaction of software and hardware in embedded system design, caching has also received extra attention to allow for optimized performance. The main influential factor for using a cache is the chosen size to hold temporary values. A too small, as well as a too large cache can have a negative effect on system performance [262]. However, in our work we did not investigate aspects of caching for increased system performance.

Memory mapping is a term used for different methods, but mainly refers the projection of the memory space and registers of one device into the address range of some other

device [262]. For example, memory-mapped I/O refers to the communication of CPUs and external devices by reading from and writing to the registers of a device, which are directly linked to areas of the CPU's addressable space. In contrast, a memory-mapped file refers to a piece of virtual memory, which is directly corresponding to some file or file-like resource, like a shared memory object for example. This is a mechanism especially used in operating systems for assigning a virtual memory area to individual applications, threads or tasks. Memory mapping is also used to make external memory banks accessible and treatable like internal memory resources of a CPU by simply assigning them to addressable space of a CPU. In our case, the external memory bank is mapped to a higher range of addresses of our DSP.

2.4.1.4 Direct Memory Access (DMA)

To overcome the bottleneck of slow memory access, the use of DMA data transfer is encouraged to keep the processor busy as steadily as possible. DMA is a technique whereby data can be transferred from or to the processor's memory without the involvement of the processor itself, and is typically used to provide improved performance for input/output devices. A separate DMA controller is necessary to do this, which can be an integrated part of a DSP chip or be implemented using external hardware [128]. Using DMA transfer essentially means, that the DMA unit is autonomously and continuously swapping blocks of data between the external and the internal memory, while the CPU is performing certain operations on the data. Note that DMA transfer only works on linear data blocks of a reasonable size as each transfer is preceded by a setup stage. The time for setting up a transfer is in the order of 20 to 40 clock cycles and has to be added to the amount of time needed for the effective data transmission, thus the use of DMA is only meaningful for data blocks of a certain size. Using this memory swapping technique, the CPU itself can be kept from stalling because a continuous input data stream is provided. This ensures a considerable improvement in computational efficiency and throughput.

DMA transfer is only possible between double-word aligned blocks. Without using a memory management unit, there is no support for automatic memory swapping. Consequently, we have to make sure that we schedule every memory allocation, deallocation and intermediate transfer ourselves. This also means that we have to place commands for any memory related operation in our program code wherever needed. Blocking statements are used to check if a transfer has completed and to synchronize the algorithm flow again. Although performing this type of memory management seems to be an easy task at first glance, it has a major impact on program design. Effectively, these type of memory exchange function introduce asynchronous behaviour into an otherwise serial and synchronous program flow. When handled without caution, this can introduce various types of hazards, like reading from or writing to uninitialized memory cells. It can at least cause undesired CPU stalls if data is not available in time, thus it is important to analyze these issues during programming.

2.4.1.5 Compiler Optimization for Code Size

Modern compilers allow to trade performance versus code size when building executable machine code. This is important in the aspect of pipelined architectures, because compilers usually aim at executing a maximum number of operations in parallel and try to organize code to be maximally efficient in terms of speed. However, issuing too many parallel instructions can swell code size, overrunning the available system memory. One example is the unrolling of inner loops to allow for increased parallelism, which is a feature of many compilers that can be allowed or disallowed at compile time. For example, the compiler from TI, which was also used for building our algorithms, gives the developer the possibility to profile code and optimize for speed, code size or both. In our development, we only instructed the compiler to optimize our code for speed. We did not investigate special compile options concerning loop unrolling and minimizing code size.

2.4.2 Parallel Execution Mechanisms

DSPs are devices highly optimized for repeated computations on large datablocks. Due to their composition of multiple data paths and the multiple ALUs and multiplier units respectively, it is possible to perform multiple operations within one clock cycle. As a consequence, a high level of parallelism can be achieved on all levels of operation, while it is on the programmer to exhaust these resources best possible. In the following, we discuss various ways that embedded processors, mainly referring to DSPs in our case, perform operations in parallel. For a more technical and in-depth discussion, refer to the book of Wolf [262]. Note that we follow a rather raw structure and most mechanisms are inherently related to each other in modern embedded processors.

2.4.2.1 Very Long Instruction Word (VLIW) Processors

The TI TMS320C64xx DSPs contain a total of six ALUs and two multipliers in two data paths, as can be seen in Figure 2.4. Up to eight 32-bit instructions per cycle can be performed, using the 64 general-purpose 32-bit registers available. This is possible through VLIW technology and a very efficient C/C++ compiler. On general-purpose, superscalar processors, operations are assigned dynamically to functional units at runtime. In contrast, a VLIW-aware compiler for DSPs determines if operations can (and should) be executed in parallel and packs these instructions into a *Very Long Instruction Word* at compile time. Thus the CPU fetches and decodes the VLIW and assigns the operations to the dedicated units. Special care has to be taken in program design to pass information about parallelizable operations to the compiler (and enable the compiler to see data hazards or other conflicts). In other words, VLIW are mainly used in applications with a great deal of data parallelism. However, the major benefit of using VLIW technology is rather obvious. A high degree of freedom is offered to the programmer already during writing of an algorithm. While the

compiler does its best to optimize for speed, the programmer can check at compile time if the algorithm is executed maximally efficient, or if additional optimization is necessary.

2.4.2.2 Single-Instruction, Multiple-Data (SIMD)

From the origin of DSP hardware development, special attention was directed at maximum performance of these devices for *Multiply-Accumulate* (MAC) operations, as these operations are the main content of filtering. The ALUs and multiplier units can be used separably within a single clock cycle, which means that additions and multiplications can be performed in parallel concurrently. Of special importance is the fact that also data of different bit-width can be used, which is also known as *subword parallelism*. The ALUs can be either used in normal mode, or can be split into smaller units by breaking the carry chain. In doing so, each subword can operate on independent data, and the same instruction is performed on several data values simultaneously. For example, for image processing it is important that each multiplier, as the functional unit with the highest complexity, can perform either two 16×16-bit multiplications, or even four 8×8-bit multiplications within one cycle. In other words, in 8-bit grayscale imagery, multiplications on up to 8 pixels can be performed simultaneously, which results in a maximum of performance. Needless to say that, vice versa, the algorithms especially suited for DSPs are filter-based ones, or algorithms that can be rewritten to behave like such.

2.4.2.3 Data Parallelism Issues

Apart from hardware related parallelism, considering possible parallelism on the data level has special relevance if it seems reasonable to the designer. Moreover, if memory usage is strictly limited, or if more than one DSP is available for processing, partitioning and restructuring an algorithm by hand is an option. The main reason to consider splitting algorithms into multiple blocks which can be executed in parallel, is the lack of powerful compilers, which are able to detect parallelizable operation on blocks of data at a larger scale. In general, compilers can already be configured to operate at different levels, from being very conservative concerning data dependencies to being very liberal. Detecting data hazards on the one hand, or data independence on the other hand, has special relevance to the design of modern compilers [4]. Nevertheless, compilers are usually working with a relatively small stack (compared to the total number of lines of code in an entire algorithm) and are not able to detect data independency in complexer algorithms at a larger scope. On this account, it might be favorable for the designer to balance this drawback of compilers. This idea becomes immediately clear when looking at two examples from the area of image processing:

- *Partitioning of large images* into smaller parts can be a very helpful possibility. For example, if two DSPs are available and the image is too large to fit into their internal

memory, the image can divided into small, overlapping parts, that can be dedicated to each DSP and might be processed independently. An inevitable prerequisite is that the operations to perform on one tile do not rely on any intermediate processing steps or results of the other. Moreover, solutions derived in the overlapping image areas have to be treated and combined in a special way.

- *Independent operations*, such as filtering or transformations, can be performed in parallel and independently, as long as their individual inputs and results are not related to each other. For example, for surface analysis in an industrial application, a set of Gabor filter responses might be calculated from one single image to capture the variation in texture. If multiple CPUs are available there is no need to line up all filtering operations sequentially, but the filter responses can be calculated in parallel concurrently. This can result in a speedup in the order of the number of computational units available

Applying the methodology of algorithm partitioning by hand is exceptionally dependent on the algorithm to perform and has to be considered occasionally. However, the direct need of such programming techniques only becomes important if the hardware is giving you the necessity and the possibility to draw a significant benefit through rewriting algorithms in such a way.

2.4.3 Pipelining

Pipelining is a special type of parallelism, but also relys on several other aspects of hardware design and is thus treated and described separately here. An extensive introduction to various sorts of pipelining can be found in the books of Hennessy and Patterson [95, 182].

Processing a machine operation on a CPU usually happens in three stages, namely instruction *fetching*, *decoding* and *executing*. Obviously the order of these stages can not be changed and has to be handled sequentially. However, pipelining is the key to remove the restriction that multiple machine operations also have to be processed fully serially, which is also known as *instruction pipelining* and is a key technology used today to build fast CPUs (irrespective of the number of functional units available in a CPU). On the DSP four phases for fetching an instruction have to be passed, namely *Address Generation*, *Address Sending*, *Access Ready Waiting* and *Fetch Packet Receiption*. Two decoding phases exist namely *Instruction Dispatching* and *Instruction Decoding*, and up to five executing phases exist, whose necessity is dependent on the instruction to perform, and which are just named *Execute 1-5* here for simplicity. Without pipelining, for each machine operation these 11 phases have to be performed sequentially and each phase is only occuring once in 11 cycles (and the associated hardware is idle for 10 cycles respectively). In simple terms, pipelining denotes the processing of multiple operations in parallel, such that the hardware affiliated with the 11 phases is used without stalling and interruption. An illustrative example is shown in Figure 2.5, where the stages and phases are depicted

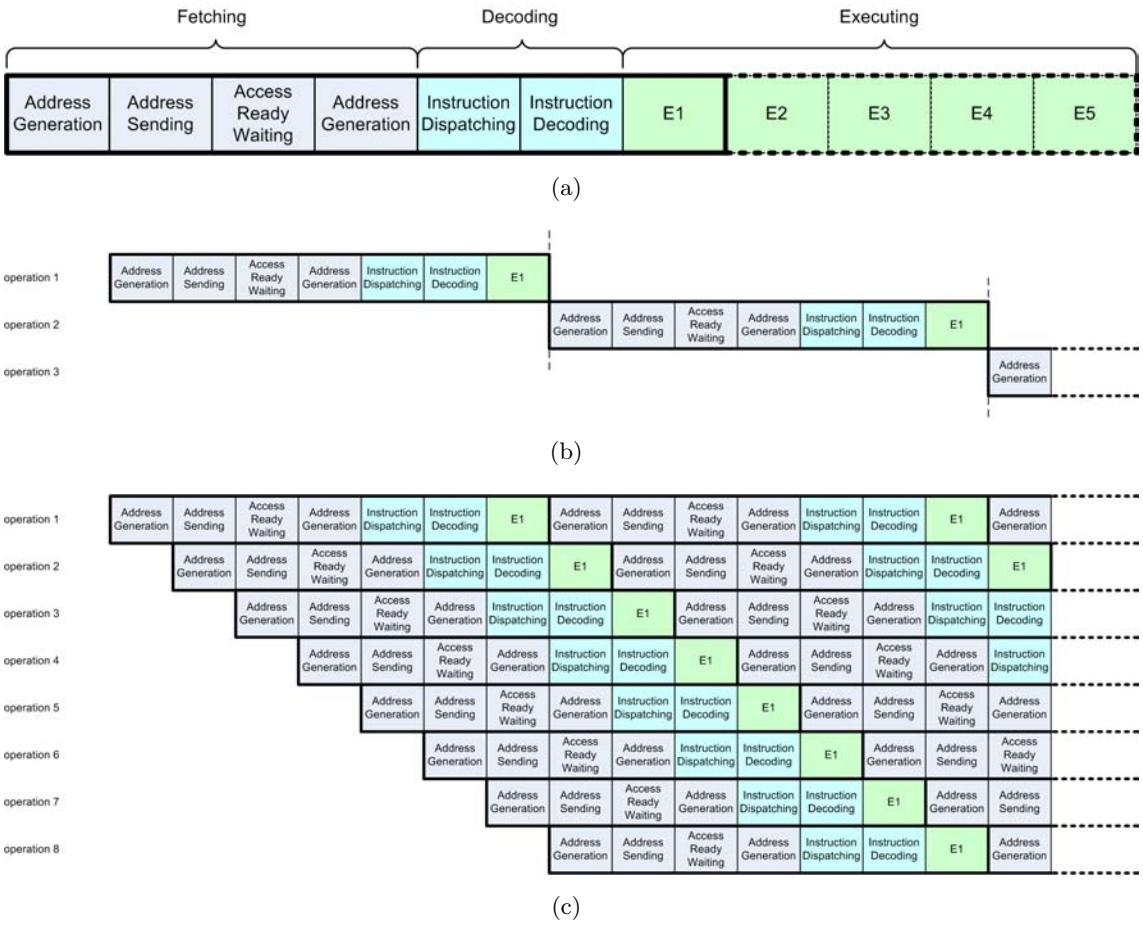


Figure 2.5: (a) The stages and phases of operation handling on the TI TMS320C64x DSP. The instruction dependent execution stages are framed with dashed lines. (b) Serial processing of stages and phases. Note that in this illustration an instruction needing only one execution phase is assumed. (c) Pipelined processing of multiple operations. Again an instruction taking only one execution phase is assumed.

in (a), together with an example of sequential and pipelined organization of multiple operations in (b) and (c) respectively. Intuitively speaking, in case (b) a result is generated every 7 cycles, while in case (c) calculating the first result also takes 7 cycles, but then a new result is available every subsequent cycle.

An elaborate discussion of pipelining is far beyond the scope of this section, but it is important to note that pipeline scheduling is done during compile time. This is relevant because it is, once again, up to the programmer to organize the program flow in such a way that the compiler can transform it into efficient code. From the programmers point of view, obviously a single loop, which contains a small block of operations to work on a larger amount of data, is suitable for pipelining, which is also known as *software pipelining* [7, 127]. Several problems are likely to occur which complicate the implementation of

algorithms, especially three types of hazards, namely *data hazards*, *structural hazards* and *control hazards*. In the first case data is needed as operands for a calculation, but depends on previous calculations that have not been performed yet. Structural hazards refer to conflicts, because more than one operation needs access to one and the same functional unit at the same time - which naturally is not possible. Lastly, control hazards are problems that are caused by events or unfavorable program flow which takes the current pipeline content ad absurdum. Usually the compiler is able to work around these issues, often finding a solution to the pipelining problem that is a good, but suboptimal one. However, further optimization is often possible if the programmer passes additional information about the data to the compiler using special keywords and macros. Note, that this is especially important for VLIW architectures for efficient scheduling of operations to multiple functional units [127]. Warnings are issued if it is not possible for the compiler to figure out, which order of operations is valid and is coevally the most efficient one. Moreover, pipelining will be simply omitted if it is not safe to use it, for example, if branching conditions within the program flow might lead to undesired behaviour. In fact, the compiler will always work conservatively and make sure, that a program can be executed safely, even if this means that the potential computational capabilities of the DSP are not fully exhausted.

To use pipelining capabilities of a DSP in an efficient way, the programmer may need to reconsider an existing implementation. In image processing often two nested loops are used, mostly to perform operations in horizontal and vertical direction respectively. The compiler can pipeline a single loop only, which is the inner one clearly. Assume, that the operations to be performed can be pipelined and that it is possible to choose between storing an image in column- or row-first order. It immediately becomes clear that the first step should be to place the loop working along the larger image dimension as the inner loop, because pipelining occurs less frequently, but each pipeline is performing longer and producing more results. Clearly this option reduces the amount of time needed to fill and empty pipelines in the first place. If it is possible, for example for filtering applications, the programmer may also consider collapsing two loops into a larger single loop. This ends up with a single pass over a linearly stored image, where a single pipeline is sufficient. If this is possible, this might be the most effective way. However, it is important that side effects of such an implementation are considered in subsequent calculations, as results might be invalid at the image boundaries, for example.

In general, it is sufficient for the programmer to write functional code which is using as little conditional statements as possible. Branches are the main source of pipeline stalls, though they should be avoided in loops if possible. Normally, if this aspect is considered, one can rely on the compiler to transform the program into efficient code without the need for additional optimization by hand.

2.4.4 Fixed Point Calculation

Historically, DSPs have been developed mainly for working on fixed point numbers. The inclusion of FPUs comes at a considerably higher price, a larger amount of chip area needed and considerably higher circuit complexity. As a result, the usage of floating point capable DSPs is highly constrained to applications where the use of FPUs is reasonable and necessary. On fixed point DSPs it is still possible to perform operations on floating point numbers, but they are all emulated in software and there is no dedicated hardware support for it. Thus program code written using floating point arithmetic is terribly slow and should only be used if it is really necessary.

In fact the invention of the FPU on general purpose computers in the 80s has generally led to a replacement of fixed point by floating point arithmetic during algorithm design. The main reason for this is the ease of algorithm development using floating point numbers, without the need for the designer to apply scaling factors and other techniques to prevent arithmetic overflow. On the one hand, this makes life a lot easier in algorithm design, implementation and testing on general purpose computers. On the other hand, it clearly makes some type of reverse engineering necessary if an algorithm is to be placed on hardware with such striking limitations.

Floating-point DSP processors are generally easy to program and use, but are usually more expensive and have higher power consumption. As a result, low-cost, high-volume embedded applications, such as cellular phones, hard disk drives, modems, appliance control, audio and video players, or digital cameras, use fixed-point processors. Usually, using fixed-point arithmetic in devices results in faster processing, is cheaper in terms of chip area and production cost, and is also more efficient in terms of power consumption. For a practical and in-depth discussion of applications and implementations on DSPs, including a review of the aspects of fixed-point and floating-point arithmetic, the interested reader is referred to the book of Kuo, Lee and Tian [126]. For a more detailed discussion on floating point to fixed point conversion rules, Randy Yates has written two helpful tutorials [273, 274].

Having some practice in working with fixed-point numbers, the special suitability of DSP hardware with SIMD support becomes self-evident. Using a 16-bit or 8-bit fixed point number representation, the ALUs and multiplier units can perform multiple operations within one CPU cycle. For example, for DSPs manufactured by TI, this is called *packed data processing*. It is rather obvious, that in this case the DSP has major advantages over general purpose CPUs, because the data throughput is considerably higher. This is also the major reason why algorithms for fixed point DSPs should be designed to take advantage of this feature, because tremendous speedups can be achieved in comparison to general purpose computing. Moreover, images are typically represented with 8-bit grayscale depth, or 16-bit or 24-bit color depth. In most cases, color is not necessary for performing a given task, thus the 8-bit grayscale format can be used, which is an ideal input format for this type of fixed-point DSPs.

2.4.5 Software Optimization

When writing computer programs, the transformation of source code into maximally efficient machine code for a given hardware platform is desired. However, after initial development, often additional optimization is necessary to meet environmental requirements. Considering the use of DMA, VLIW parallelism, pipelining and fixed-point math already during algorithm design, holds a big source of algorithm-level optimization and places general design rules independent of the specific DSP type. The major goal in software optimization is to improve the implementation in terms of memory usage or speed, without making further changes on the algorithmic level, but taking increased advantage of vendor specific optimization options. These ways of software optimization in the context of the TI DSPs are discussed in the following.

2.4.5.1 Programming Language

For programming on TI DSPs, the C and/or C++ programming language can be chosen. The decision, whether to use one or the other for programming, mainly depends on the scale of a specific application and has to be taken as the case arises. But a good deal more, the principal question poses, whether to use a language offering only very raw structuring capabilities, or to utilize a more elaborate program language, offering features like inheritance, virtual functions, additional datatypes, and so on. However, books exist dealing with this principle question and discussion boards are full of this topic, arguing about pros and contras of both. A comparison between C and C++ for DSP programming can be found in [256].

There are at least two good reasons to choose C over C++. First, compilers and optimizers on the TI DSPs are more advanced in building machine code from C source code, rather than from C++ sources. This mainly results from the, possibly unneeded, overhead of the C++ language itself, and from the longer tradition of the C language. Second, a lot of features of C++ are restricted, not allowed or not available anyway. Among those unsupported features are virtual and multiple inheritance, exception handling, complete standard library support and only limited support for templates. The main reason for these functionalities not being supported is their negative influence on either code size, runtime performance, or even on both. For example, exception handling, as well as polymorphism tasks, are mostly performed at run-time. In contrast, the assignment of a class does not differ significantly from allocating a struct, and determining private and public members of a class can be done at compile time. Nevertheless, many features allowed in C++ have more influence on programming style than they have on program flow or program behaviour. Furthermore, various things can also be implemented in C with some programming skills, such that possible advantages of C++ over C become even less influential.

2.4.5.2 Compiler Optimization

The C/C++ compiler from Texas Instruments for the TMS320C6x DSPs includes a very efficient optimizer. It mainly optimizes the source code for the register and pipeline usage of the underlying DSP. This is mainly accomplished by automatic allocation of variables to registers, rearrangement of statements, expressions and declarations, and by simplifying loop structures for pipelining. Special configuration can be set to optimize either for speed or code size. Moreover the compiler can be instructed to compile all sources as single large block of code, meaning that it can perform *program level optimization*, rather than *file level optimization*. This allows the compiler to see variables and data dependencies across file boundaries and allows for more effective software pipelining and loop scheduling. Several methods exist to further improve the efficiency of the compiler and the optimizer:

- **Inline functions**: Small functions can be marked using the *inline* keyword. In doing so, the function itself rather than its address is copied to each location it is used elsewhere in the program. On the one hand, this means that the compiler can optimize the code section, as it now sees the entire functionality rather than some call to a function. On the other hand the code is duplicated on the execution stack, which means that also the code size is increased.

- **Globals and Constants**: *Global*ly declared variables can be used to avoid the computationally intensive parameter passing process in function calls. Definitions of parameters or functions as *const* helps the compiler in optimization and increases code readability.

- **Register Variables**: Using the *register* keyword forces the compiler to prefer these variables for placement in a register rather than on the program stack. This can significantly increase performance, as priority to variables can be assigned manually and their intended frequent use is signalized to the compiler.

- **Loop Unrolling**: To help the compiler in optimization, the programmer can perform simple tasks like *loop fission* or *loop fusion*. Loop fission describes the splitting of single, large loops into multiple small loops. This mainly helps the compiler to increase instruction level parallelism. In contrast, loop fusion is the combination of multiple loops into a single, large loop, which decreases loop overhead and reduces the overhead of copying data to internal memory. By using special keywords and macros, the programmer can pass additional information about the intended number of runs through loops to the compiler. This can help the compiler to perform modulo-scheduling of loops, unrolling and optimize for code size and/or speed. One important requirement is that the loops are free of function calls, whereto also calls to memory allocation and deallocation belong.

- **Intrinsics and Packed Data Processing**: *Intrinsics* are highly optimized functions for C/C++, that are written in assembly language. They allow for maxi-

mally efficient use of DSP and vendor-specific hardware features on the programming language level. Closely related to intrinsics are *Packed Data Processing* methods. Packed data processing is a functionality that allows the programmer to take advantage of the SIMD ability of the underlying hardware by using special intrinsics, and to encourage parallelism. An explanation, how to use this functionality in the context of image processing is given in the book of Qureshi [189].

2.4.5.3 Lookup Tables, Iterative Methods and Memory Allocation

A lot of mathematical functions, such as the *sine*, *cosine* or *exponential* function, are computationally expensive to compute, especially if they are used frequently. Lookup tables for frequently needed function values can be used, if memory resources are available and the computational effort of calculating a single value is significant. An other possibility is the use of iterative methods to approximate the exact values. Iterating can be stopped, if the results seems to be accurate enough. In this respect, for example using *Newtons method* to calculate the square root of a number can be a reasonable choice, if taking the square root of numbers is a frequently used operation in an algorithm [36].

Another possibility to speed up algorithms is to avoid dynamic memory allocation. Static memory allocation should be preferred over dynamic allocation, if the amount of memory needed is known in advance, and is also available for the duration it is needed.

2.4.5.4 Linear Assembly and Hand-Written Assembly Code

Texas Instruments has used the synonym *linear assembly* for a more straight-forward form of writing assembly code for the C6x DSPs. Programming in linear assembly is simpler than programming in ordinary assembly, due to some simplifications. First, the programmer does not have to worry about parallelism and the assignment of functional units. The compiler is automatically dedicating operations, that can be executed in parallel, to multiple functional units and performs pipelining self-acting. Second, the assignment of registers and the insertion of necessary NOP instructions is performed automatically by the compiler. Because of these facilities, the source code is written in a much more straight-forward and *linear* way, and is also much easier to read, though this nomenclature is used. Usually, a direct translation of C code into linear assembly does not automatically lead to more efficient code. However, the compiler is able to better map the algorithm to the underlying architecture and its machine instructions, and can, thus, sometimes achieve more performant code.

Hand written assembly code is another option to improve program performance. The need for this type of optimization by hand sometimes arises, if the compiler is not able to efficiently translate critical code sections into machine code. Moreover, hand written assembly code can be very helpful, if some block of operations and statements form a very computationally intensive part of an algorithm and are used frequently. In this case, special optimization of these section simply pays off. Hand written assembly code can be

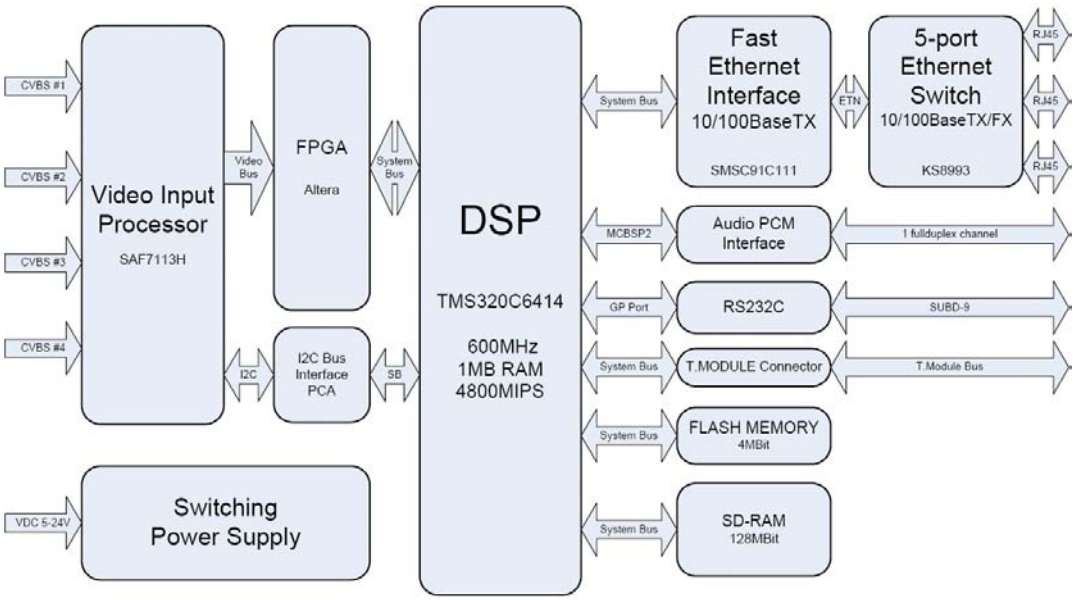


Figure 2.6: The overall block diagram of the prototypical hardware platform.

described as the most optimized way of programming, however, a lot of engineering power has to be done, hence detailed deliberation is necessary about the real requirement of this step.

2.5 TRICam - A prototypical embedded platform

A prototypical smart camera platform serving as a basis for image processing and computer vision applications was developed in the years 2004 to 2006 by FREQUENTIS, the so-called ***TR**affic **I**nformation **Cam**era* [10]. Initially designed as a high-quality analog-to-digital video converter and MPEG4 encoder, the extendable DSP-based design of the platform allows for the development and implementation of high-level computer vision applications. A overall hardware layout of the platform is shown in Figure 2.6, while the housing and the look of the system is depicted in Figure 2.7.

2.5.1 Main Features

The platform was developed to make it easily applicable as part of a larger surveillance system, eventually using existing infrastructure. The main features of the platform are:

- 1 TMS320C6414 DSP with 600 MHz and 1MB cache
- 128 MBit SDRAM for video compression, processing and storage of temporary data
- 4 MBit Flash Memory for firmware storage

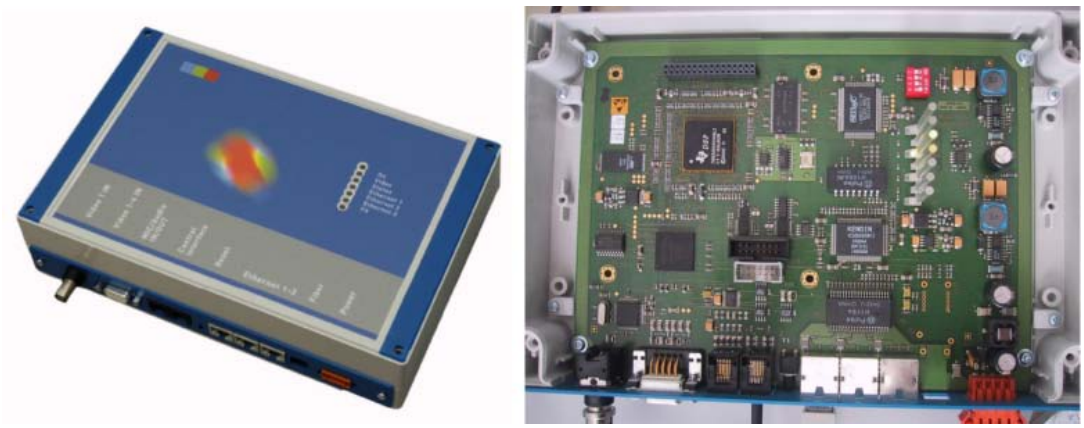


Figure 2.7: General look of the housing and the TRICam platform.

- a 5-port ethernet switch with three 10/100 BaseTX ports and one 100 BaseFx fibre port
- 1 video input processor, featuring 4 analogue CVBS or 2 SVHS TDM video channels per system (PAL/NTSC)
- 1 FPGA for buffering video frames/scanlines between the video input processor and the DSP
- a full-duplex PCM audio interface using 8 kHz for voice communication and sound processing
- 2 RS232 serial ports
- a T-module expansion connector for two additional DSPs

The use of an analogue video interface allows for fast integration of the video server into an already existing network of pre-assembled cameras. This is the main reason why the platform was not designed to include an own imaging device. However, this does not narrow the applicability of the platform in the domain of digital cameras, since the integrated ethernet interface can also be used to connect the platform to digital cameras sharing an ethernet port. Furthermore, the audio interface can be used to augment the visual perception of any camera by audio information.

The computational resources of the platform may be scaled using the expansion module and additional DSPs, allowing for the application of more demanding algorithms from the field of computer vision. Any information, either generated or extracted, can be transmitted to a given destination in various ways, like sharing a conventional computer network or any other technology (modems or mobile phones may be interfaced using the integrated serial ports). Thereby, the amount of information to be transmitted may directly depend on the infrastructural resources available at site, thus, ranging from a

few bytes for a simple text message to a fully encoded MPEG4 video stream [235] at 1.5 MBit/s. In summary, the platform can be used in a lot of different ways, for example, as a simple compression device producing high-quality video for general surveillance tasks, or as a complete video telephone box using Voice over IP (VoIP).

2.5.2 TRICam Specific Software Development

Irrespective of all aspects of integrability and commercialization, we use the platform to test and develop different high-level computer vision algorithms for object detection and recognition. Though the TMS320C6414 is a fixed-point processor lacking a FPU, efficient execution of programs is only possible if they are using fixed-point arithmetic. No additional devices or components but the basic setup as described above are used. We neither make use of the expansion module or additional DSP power. This places several limitations, concerning real-time capability of methods and memory consumption of programs, but also states clear goals for the development and integration of approaches on this fixed hardware setup.

We follow a principal way of development, which is divided into multiple steps and allows rapid prototyping. The first stage of algorithm development is conducted on a usual PC. First, algorithms are designed and implemented using a high-level programming and design language, which is Mathworks MATLAB [233]. This allows for fast development using already available code sections, also neglecting possibly needed library support. After verifying the functionality and suitability of an algorithm, we recode critical code sections bit by bit into MATLAB executables, so-called MEX files. In doing so, we can make the first step to transfer algorithm behavior straight into plain ANSI-C code. Thereby, we are still able to use MATLAB functions and libraries, but are also capable of estimating algorithm performance on the C programming language level. Note that we do not make use of Simulink [63]. Finally, a plain C code executable is built, either under Windows or Linux, where also all library dependencies have been removed. Using special, open-source profiling tools like `valgrind` [239], we proof the final program to be free of memory leaks or other bugs.

The second step of algorithm development is the deployment on our hardware platform. The plain, library-free C code version of our algorithm is compiled using TIs Code Composer Studio 3.2. Naturally, the main goal, to get the algorithm to run on the hardware platform for the first time, is achieved through reorganization of memory usage. In this context, we mainly speak of reducing the overall memory requirements of algorithms, and refer to a well-structured procedure for exchanging data blocks between internal and external memory banks, if necessary. After meeting that requirement, the implementation is refined using all optimization and design rules explained in the previous section 2.4. The refinement and optimization is iterated until the algorithm performance is satisfying, or the amount of expected, additional performance gain does not justify further development work. During optimization work, debugging on the hardware platform is done using a

console on the onboard serial port interface, and the JTAG boundary scan interface. To transmit input and output data for the final program, we mainly use the onboard network interface or the analog video grabber. Video data is transmitted in raw, MPEG4 or JPEG encoded format. Parametrization of algorithms is usually done at startup using XML files and including a lightweight XML parser into the DSP program.

Our entire software setup forms a nice framework for fast development and rapid prototyping of image processing algorithms on our hardware platform. As described above, we follow rather general design rules for software development, which should also, up to a certain extend, be applicable on other DSP-based hardware platforms.

2.6 Conclusion

Smart cameras are a very interesting field of research, bringing together developers from the area of computer vision and embedded systems. In principle, the reasonability of smart cameras is given by the necessity to perform dedicated tasks at camera site, to take the burden from the human observer who is monitoring a large number of video streams at the same time. It is clear, that for processing embedded hardware is preferred over usual PCs, as there are special requirements for robustness against environmental stress, power consumption and communication issues. In this context, the introduction of smart cameras pushes the design space in many dimensions, such that smart camera related development of embedded hardware forms its own field of research nowadays. In terms of performance, DSP-based setups are good choices, as they are very powerful, yet flexible and because they are strongly related to general purpose processors in terms of programmability.

It is clear that the advantages of embedded systems, especially the low power consumption, the small form factor and the robustness against environmental adversities, come at the cost of complex memory management and more complicated design processes. Also the lack of powerful design tools which can fully exploit the hardware properties play an important role here, which makes the realization of computer vision algorithms on smart cameras a tricky task. All the more, developers of computer vision algorithms have to take care of these special issues at design time, to allow for high performance of approaches, even with restricted computational and memory resources. Moreover, it is necessary to be aware of the properties of the underlying hardware architecture, which allows for performing different computations in a highly optimized manner. Summarizingly, software development for smart cameras is a difficult task, which involves both, knowledge about computer vision and experience in programming embedded computers.

The TRICam platform, that was presented in Section 2.5, can be described as a prototypical smart camera platform. Although it does not feature a video sensor, the computational and memory resources, together with the different interfaces for communication and the possibility to connect to digital or analog cameras, form a suitable setup for performing state-of-the-art computer vision tasks. As a big advantage, the platform is very

flexible due to the powerful DSP, which can be programmed easily. Because we are mainly interested in the aspects of algorithm development for visual surveillance on smart cameras, the setup proposed can be stated as an ideal playground for developing, investigating and evaluating powerful computer vision approaches for surveillance purposes.

Philosophers say a great deal about what is absolutely necessary for science, and it is always, so far as one can see, rather naive, and probably wrong.

Richard Feynman
US educator & physicist, 1918 - 1988

Chapter 3

Related Work

A big amount of research work in the computer vision community is focused on object detection and object recognition. Mostly, direct applications can be found in public surveillance, where security is one of the most often heard keywords since the beginning of the new millennium. The amount of literature about detection and recognition becomes more and more unmanageable. Algorithms have been proposed, starting from raw object detection from aerial images, up to iris recognition for access control systems of buildings. The range of applications for computer vision is nearly unlimited.

However, even if good solutions already exist, algorithms steadily become better and more accurate than the existing approaches. This often comes at the cost of practicability. Many algorithms are highly efficient in terms of accuracy and performance. Unfortunately, mostly they do not take environmental constraints into account which limit their deployment in the real-world. Another aspect is the need for robust algorithms, that still offer a certain amount of reconfigurability and adaptability. This effectively means, that a given set of algorithms is wanted, which is applicable and tunable to successfully perform a big variety of tasks, not just a small number of applications in a small niche. The search for an algorithm, being the universal remedy, is utopistic. However, it is clear that the development must be towards a set of algorithms, keeping the most important keywords in mind to be successively applicable: *robustness*, *reconfigurability*, *adaptability*, *performance* and *speed*.

In the following we refer to a large number of approaches which can be denoted as state-of-the-art methods for object detection and object recognition. We will mainly focus on surveillance, since we believe it to be the largest area of related development. Note, that this does not foreclose the application of specific methods in other situations and applications. Starting with some general categorization of algorithms in computer vision, we discuss the methods proposed for building an object representation in section 3.1. Related work in the context of object detection is discussed in section 3.2, and the literature available on object recognition is repeated in section 3.3, respectively. In this regard, we

also list the available approaches proposed for application on embedded systems.

Note, that it is not possible to draw a clear border between approaches for detection and recognition, due to the ambiguous use of both terms in the literature. Likewise, it is very hard to limit the focus on approaches, which are dedicated especially to surveillance. This is a consequence of the nomenclature *surveillance* describing a wide range of development, rather than a small corner. However, we make a set of assumptions, which allows us to structure existing literature in a well arranged way. First, we assume *Learning* to be a general tool, rather than a special method itself. This interpretation is ambiguous and cannot be pushed through entirely without lack of clarity, but already captures a large range of methods.

We do not draw upon performing the best structuring over the entire set of approaches. However, we still try our best to perform a grouping based on the conceptual and basic methodologies of approaches. In the final section 3.5, we summarize the most important aspects and justify our own selection of algorithms for closer investigation on our embedded platform.

3.1 Computer Vision for Surveillance Applications

As already stated before, one of the driving forces in computer vision development is the increasing need for security in public places. Another reason is, that due to the deployment of more and more cameras in our environment, the amount of data (*i.e.* video material) to be analyzed has become by far too large to be processable by humans. Thus it is necessary to develop and deploy devices and algorithms which are doing the main processing task automatically. In the context of surveillance, this mainly implies robust detection of objects, recognizing their class membership or identity, and the interpretation of behavior, events and threads of various kinds. This information is passed to a human supervisor for further verification and inducement.

While the overall objective can be formulated in a few sentences, in practice it requires knowledge from a multitude of technological areas to be fused to achieve this goal. Looking at the algorithms and methods, the most important features are robustness, accuracy and real-time capabilities. The first two properties are mainly determined by the ingredients of the algorithms themselves; they are subject to optimization by using machine learning techniques or expert knowledge. However, real-time capabilities finally make the difference between prevention of an assault on public safety and reproducing the incident afterwards.

3.1.1 Algorithm Categorization

In principle it is necessary for object detection and object recognition to form some type of representation including the main characteristics of the object class. In principle one can use any type of visual feature, or also combinations of them, and a lot of different approaches have evolved in the past. From the large amount of algorithms proposed in

the literature for object representation, generally two groups of methods can be defined, namely (1) **global/patch-based methods** and (2) **local feature based methods**.

1. **Global/patch-based methods** are algorithms where a single descriptor of an entire object is created. In other words, an image patch showing an instance of an object is taken and a transformation (or learning algorithm) is applied to the whole patch to calculate an overall object representation.

2. **Local feature based methods** are approaches where local features are used to describe parts of an object. This means that, given an image of an object instance, single parts of the object are described locally and the final object representation is a collection of these local descriptors. Optionally, also a collection of descriptors, which reflect the constellation and relationships between the local features, are used.

For the direct application of these methods for the individual tasks of object detection and object recognition, a large number of solutions have been proposed likewise using algorithms from both groups. In principle, both types of methods have their advantages and disadvantages in their application for the specific task, be it object detection or object recognition. Although no general statement about the special suitability of one group for a given task can be made, recently, *Global Methods* were used more often for object detection, while *Local Methods* were more often used for object recognition. The reasons will be elucidated in the following as we will shortly outline the principles of both global and local approaches in the following sections 3.1.2 and 3.1.3. For now note that in Figure 3.1 the ideas behind both approaches are illustrated.

One important thing to mention is, that the rigidity of objects plays a vital role in the development of algorithms for both tasks, object detection and recognition. Rigid objects form the largest group of objects in our environment, and the task of building a suitable object or class representation is already hard enough, due to varying illumination, intra-class variation and other adversities. In contrast, algorithms for non-rigid objects have to incorporate additional knowledge about geometry or deformability of the object or class. In fact, this information is sometimes not available or hard to acquire. But even if it is available, incorporating it definitely comes at considerable algorithm complexity, which is often not worth the additional effort. Summarizingly, the development of approaches becomes harder, and algorithms become more complex with increasing deformability of objects. Therefore, one motto, often followed in algorithm design, is to develop methods for rigid objects, which are also able to cope with a limited amount of deformability. This variability can be treated as additional intra-class variation, such that the principle algorithm design must not be changed at all. In this context, a good example is the representation of faces for detection or recognition, which treat the facial expression as supplementary intra-class variation. Another example is the representation of humans in describing the appearance of pedestrians, where mainly the human torso and the head is described, omitting details about the arms and the legs, which are the "degrees of freedom" in this case.

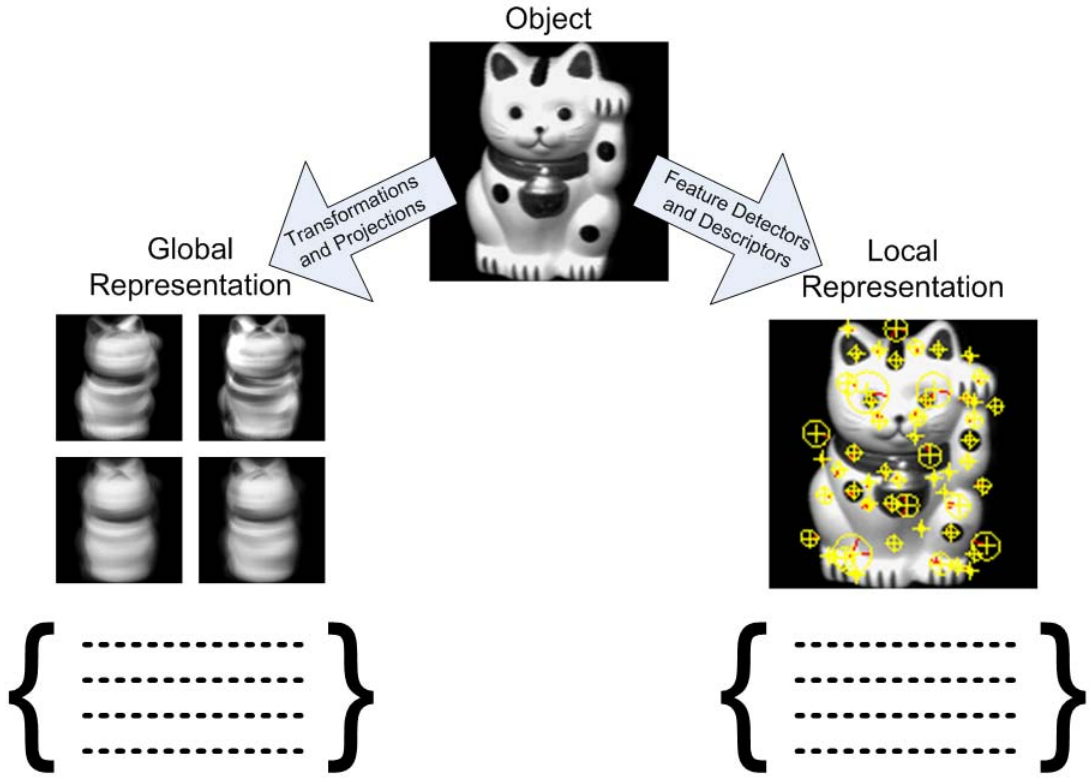


Figure 3.1: Global versus local description of an object. Most general methods are based on calculating an appropriate transformation and on projections, thus building a global representation is sloppy denoted as *transformations and projections*. Local representations are mostly based on interest point detection and description, which is depicted on the right and denoted as *Feature Detectors and Descriptors* here.

3.1.2 Global Methods

A lot of different approaches have been proposed in recent years for creating a global object representation. Giving an overview of all classes of approaches would go far beyond the scope of this thesis. However we will still mention the biggest groups of algorithms here and focus on describing the ones related to our work in more detail later.

3.1.2.1 Template Matching

A rather primitive method of representing an object class is to use a patch showing a prototypical instance of an object, a so-called *template*. This template is used, together with some mathematical function, to calculate a similarity measurement between a given patch and this template. The most common measurement in this respect is the *Normalized Cross Correlation* (NCC). For searching object instances in larger images, an exhaustive matching is performed on subwindows of images, and for multi-scale search, exhaustive

matching in an appropriate image pyramid is necessary. Note that this is more or less equivalent to filtering the larger image with the template as a filter, and that this is a very computationally expensive task.

Template matching has been proposed by various authors, especially in the context of object models consisting of dominant edges. So-called wireframe models were proposed by Koller *et al.* [124], or by Ferryman *et al.* [72]. Jain and Zongker proposed the use of deformable templates for recognition of handwritten digits [107]. More recently, Cole *et al.* used template matching for recognizing different types of Lego bricks in a database of 100,000 images [50]. Template Matching has also been used in larger detection and recognition frameworks for matching shape templates of object parts, for example by Opelt *et al.* [172]. However, the large computational costs of template matching is a big drawback, especially foreclosing the real-time capability of this type of approaches.

3.1.2.2 Subspace Methods

From all methods working on a complete image or an entire image patch, the biggest group of approaches are the *subspace methods*. This group is based on projections and transformations from high dimensional image space into lower dimensional spaces where the resulting dimensions of space can be adjusted.

Subspace methods mainly include linear approaches like *Principal Component Analysis* (PCA) [101, 109, 113, 183], *Linear Discriminant Analysis* (LDA) [20, 73, 277], *Nonnegative Matrix Factorization* (NMF) [130, 177, 215], *Independent Component Analysis* (ICA) [8, 17, 105] and *Canonical Correlation Analysis* (CCA) [96, 150]. As a subgroup it contains Kernel extended methods to make all approaches mentioned applicable in the nonlinear case, Kernel PCA [211], Kernel LDA [152], Kernel NMF [276], Kernel ICA [14] and Kernel CCA [150]. Other methods for the nonlinear case are *Local Linear Embedding* (LLE) [200], or *Isometric Feature Mapping* (ISOMAP) [230, 231]. For a short introduction and outline of global methods the reader is referred to the work of Roth [196].

The major problem of subspace methods is their sensitivity to background noise and occlusion. Moreover finding the right transformations is highly computationally and memory intensive in case of large number of images. Coevally the calculations sometimes suffer from numerical inaccuracies which are additional sources of error.

3.1.2.3 Shape Based Methods and Moments

Shape based methods are another big group of global methods complementary to *subspace methods*. For an overview of different shape matching approaches the reader is referred to the survey of Veltkamp and Hagedoorn [249].

Popular approaches are based on pointwise sampling of object outlines and registration of point sets. Chui and Rangarajan proposed the use of a *thin-plate-spline* model for registering point sets by searching for a global optimal transformation and image warp [48]. The modeling of point sets using statistical descriptions of their location was proposed

by Cootes *et al.* and is known as *Active Shape Models* (ASMs) [54]. Later, Cootes *et al.* extended the approach to also capture some degree of appearance within the models, which is known as *Active Appearance Models* (AAMs) [53]. Another popular approach was proposed by Belongie *et al.* [22] called the *Shape-Context* descriptor to form an object representation as a whole by a set of circular edge histograms. A given number of t sample points from the edges of an object are sampled randomly and for each sample point all $t-1$ remaining points are accumulated in a histogram in log-polar space. The sum of all histograms finally builds the representation of an object and matching is done by solving a bipartite graph matching problem.

Thuresson and Carlsson [241] propose the use of a combined shape and appearance histogram. By using an ordering function over the relative orientation between triples of edge points, the shape and the appearance is coevally encoded in a histogram, while the gradient strength at the selected points serves as a weighting factor.

Closely related to shape-based methods, the usage of different types of moments has been proposed by various authors. *Geometric* moments were first introduced by Hu who applied moments to the task of character recognition in 1962 [102]. The usage of the Fourier transform of object boundaries to build a translation and rotation invariant description was proposed by Fu and Persoon [80]. *Zernike* moments were introduced by Teague [228], *Rotational* moments were investigated by Boyce and Hossack [28] and *Pseudo-Zernike* moments were proposed by Teh and Chin [229]. For a general survey on moment-based methods for object detection and recognition in image analysis the interested reader is referred to the survey of Prokop and Reeves [186]. Closely related to moments, the theory of *Wavelets* in image analysis was summarized by Mallat [143], which forms the base for image compression, but also for coding and, thus, efficient describing images or image patches. Another approach was proposed by Ballard and is known as the *Generalized Hough Transform* (GHT). This approach is an extension to the original Hough transform to allow for representing objects with more complex shapes [16].

Usually, shape-based methods use segmented input data, which is the boundary of an object respectively. These methods have been shown to be suitable for special tasks, like optical character recognition, recognition of logos on the MPEG7 database, or for retrieval of binary images from a database based on a similarity criterion. One disadvantage of shape-based methods using point sets is their high computational complexity, as the matching of object representations, consisting of sets of points, is a difficult task. Moreover, the boundary of an object can be highly disturbed through occlusions or noise, which makes the correspondence finding problem even harder. However, the major drawback of shape-based methods is the large deformability of shape, which makes it really hard to define and build a compact, yet robust and efficient representation for a single object or object classes.

3.1.3 Local Methods

The main idea behind local methods is maintaining object representations from collections of locally sampled descriptions. In other words, the appearance of local parts of a single object is encoded in descriptors, and a set of these descriptors forms the final object representation. For finding the distinguishable regions so-called interest region detectors are used, which find regions or points of special visual distinctiveness. The neighborhood of such regions is subsequently encoded using a special transform to build a description, and the multitude of descriptors forms an object representation.

Two aspects are important when working with local features. First, local features are usually computed without spatial information between them, so they are usually treated separately. However, different approaches have been proposed to take advantage of spatial information, either to achieve a higher performance during the first detection or recognition step, or to verify a hypothesis in a verification stage. Second, efficient descriptor matching is inevitable to allow for real-time performance. Determining matches between descriptors is based on comparing them one by one, which is an exhaustive search technique. Due to the computational complexity of the task and the resulting amount of time needed to solve it, different strategies have been investigated to achieve a considerable speedup [203]. However, no methods are known to solve the matching problem in high dimensional spaces in a satisfactory way concerning runtime. Thus, tree-like data structures have been proven to be favorable for approximating the optimal solution and establishing correspondences.

The vast amount of approaches for detectors and descriptors makes it impossible to give an extensive overview about all methods here. Thus we will only describe the general ideas behind local methods and mention the most popular algorithms.

3.1.3.1 Interest Region Detectors

The basic principle of interest points and regions is the search for spots and areas in an image which exhibit a predefined property making them special in relation to their local neighborhood. This property should make the region distinguishable from its neighborhood and detectable repeatedly. Furthermore the detection of these features should be - to the best possible - illumination and viewpoint invariant.

The first important interest point detector, the so-called Harris Corner detector, was proposed in 1988 by Harris and Stephens [91]. It exhibits excellent repeatability and was subsequently used for object recognition purposes by Schmid and Mohr [209]. An extension to the Harris detector to include scale information was later reported by Mikolajczyk and Schmid as Harris-Laplace detector [156] and was used by Schaffalitzky and Zisserman [204] for multi-view matching of unordered image sets. Another approach to detect blob-like image structure is to search points where the determinant of the Hessian matrix assumes a local extremum, which is called the Hessian detector. Further developments to include affine covariance resulted in the Harris-Affine and Hessian-Affine detectors proposed by Mikolajczyk [153, 155].

The currently most popular two-part approach known as SIFT (*Scale Invariant Feature Transform*) was proposed by Lowe [142], where the first part is an interest point detector. The DoG (*Difference of Gaussian*) detector takes the differences of Gaussian blurred images as an approximation of the scale normalized Laplacian and uses the local maxima of the responses in scale space as an indicator for a keypoint. A complementary feature detector, the MSER (*Maximally Stable Extremal Regions*) detector, was proposed by Matas *et al.* [145]. In short, the MSER detector searches for regions which are brighter or darker than their surroundings, *i.e.* are surrounded by darker, vice-versa brighter pixels. First, pixels are sorted in ascending or descending order of their intensity value, depending on the region type to be detected. The pixel array is sequentially fed into a union-find algorithm and a tree-like shaped data structure is maintained, whereas the nodes contain information about pixel neighborhoods, as well as information about intensity value relationships. Finally, nodes which satisfy a set of predefined criteria are sought by a tree traversing algorithm[1].

Two affine covariant region detectors were proposed by Tuytelaars and van Gool [245], IBR (*Intensity Based Regions*) and EBR (*Edge Based Regions*). IBRs are based on extrema in intensity. Given a local intensity extremum, the brightness function along rays emanating from the extremum is studied. This function itself exhibits an extremum at locations where the image intensity suddenly changes. Linking all points of the emanating rays corresponding to this extremum forms and IBR. EBRs are determined from corner points and edges nearby. Given a single corner point and walking along the edges in opposite directions with two more control points, a one-dimensional class of parallelograms is introduced using the corner itself and the vectors pointing from the corner to the control points. Studying a function of texture and using additional constraints a single parallelogram is selected to be an EBR.

Another algorithm termed Salient Region detector was proposed by Kadir and Brady [110] and is based on the *Probability Density Function* (PDF) of intensity values computed over an elliptical region. For each pixel the entropy extrema for an ellipse centered at this pixel is recorded over the ellipse parameters orientation θ, scale s and the ratio of major to minor axis λ. From a sorted list of all region candidates the n most *salient* ones are chosen.

For an extensive evaluation of a large number of affine region detectors refer to the work of Mikolajczyk *et al.* [159].

3.1.3.2 Descriptors

Generally speaking, a *descriptor* is an abstract characterization of an image patch. Usually the image patch is chosen to be the local environment of an interest region. Based on various algorithms, methods or transformations, the resulting characterization can be made rotation invariant or, at least partially, insensitive to affine transformations.

[1]The algorithm described has complexity $\mathcal{O}(Nlog(log(N)))$. A more efficient algorithm based on the use of component tree analysis is to be found in the work of Donoser and Bischof [65].

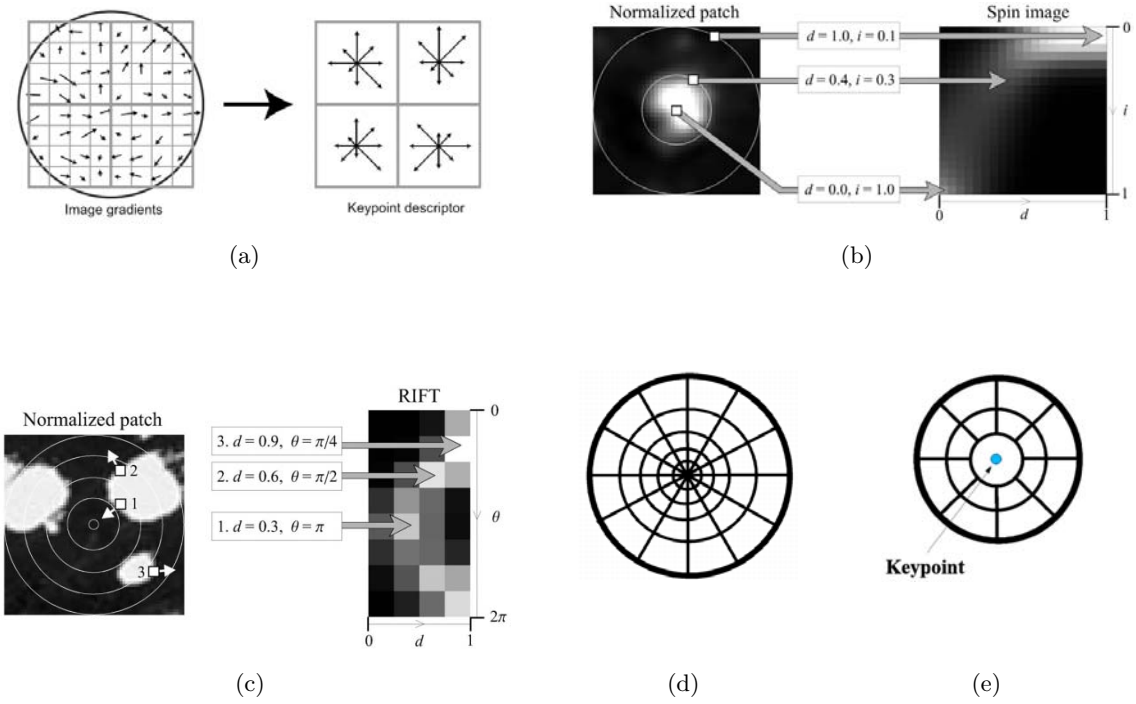


Figure 3.2: Overview about different descriptors and their calculation. (a) SIFT descriptor calculation scheme, taken from [142]. (b) SPIN descriptor and (c) RIFT descriptor, taken from [129]. (d) Shape Context descriptor mask, as described in [22]. (e) GLOH descriptor calculation mask, as described in [157].

Most approaches are based on gradient calculations or image brightness values. As a second part of the SIFT approach, Lowe [142] proposed the use of descriptors based on stacked gradient histograms. The single histograms are calculated in a subdivided patch discretizing the gradient orientation in order to cover spatial information. Finally they are concatenated to form a 128-dimensional descriptor. Recently Ke and Sukthankar [115] proposed the so-called PCASIFT descriptor based on eigenspace analysis. They calculated a PCA (*Principal Component Analysis*) eigenspace on the gradient images of a representative number of over 20000 image patches. The descriptor of a new image tile is generated by projecting the gradients of the tile onto the precalculated eigenspace keeping only the d most significant eigenvectors. Thus an efficient compression in descriptor dimensionality ($d \leq 36$) is achieved, coevally keeping the performance at a rate comparable to the original SIFT descriptor. Closely related to the SIFT approach the GLOH (*Gradient location and orientation histogram*) descriptor was proposed by Mikolajczyk and Schmid [157]. Opposed to SIFT gradient histograms are calculated on a finer circular rather than on a coarser rectangular grid, which results in a 272 dimensional histogram. PCA is subsequently used to reduce the descriptor dimensionality to 128 again. Two rotation invariant descriptors were proposed by Lazebnik *et al.* [129], the RIFT (*Rotation-Invariant Fea-*

ture Transform) and the SPIN-Image descriptors. The RIFT descriptor is calculated on a circular normalized patch which is divided into concentric rings of equal width. Within each ring the gradient orientation histogram is computed while the gradient direction is calculated relative to the direction of the vector pointing outward from the center. The SPIN-Image is a two dimensional histogram encoding the distribution of image brightness values in the neighborhood of a particular center point. The histogram has two dimensions, namely the distance from the center point and the intensity value. Quantizing the distance, the value of a bin corresponds to the histogram of the intensity values of pixels located at a fixed distance from the center point.

A descriptor based on edge pixels was proposed by Belongie *et al.* [22]. Given n sample points from the shape of an object, the Shape Context describes the coarse distribution of the $n-1$ points with respect to a given point p_i. For this point, a coarse histogram of the relative coordinates of the remaining $n-1$ points in log-polar space is computed and called the Shape Context of p_i. Note that a limit on the distance of the neighboring points can be used to build a local or a global object description.

Filter based approaches to build local descriptions, also known as differential-based approaches, include Steerable Filters or Complex Filters amongst others. To approximate a point neighborhood a set of image derivatives computed up to a given order is used. Using local derivatives, also known as *local jet*, Freeman and Adelson [74] developed Steerable Filters, which steer derivatives in a particular direction given the components of the local jet. To obtain a stable estimate of the derivatives, Gaussian filters are used for convolution. Based on steerable filters, Carneiro and Jepson proposed the use of features, which include phase-based information [42]. Schaffalitzky and Zisserman [204] proposed to use Complex Filters, which are derived from the family $K(x, y, \theta) = f(x, y) \cdot e^{(j\theta)}$. The orientation is denoted as θ and a polynomial is used for the function $f(x, y)$.

The multitude of most prominent descriptors were evaluated by Mikolajczyk and Schmid [157]. For an explanation and more detailed listing please refer to their work.

Detectors and descriptors are very powerful tools to describe objects as a set of parts on a very abstract level. This type of representation has several advantages, such as illumination insensitivity, or at least partial insensitivity to occlusion and background noise. However, these advantages come at considerable computational cost, and the amount of memory needed is also substantial. Nevertheless, the approaches are especially suited if the high distinctiveness of the features is used in the right way, thus the approaches are very popular.

3.1.4 Learning Algorithms

Methods, involving special learning algorithms, have attracted a lot of attention recently. The main reason is their ability to be applicable to various different object categories, without changing the basic principles of algorithms. Furthermore, their usage allows for scaling and adapting approaches to given situations. Especially in the area of object

detection, learning algorithms have been successfully used as classification techniques to solve a number of important problems. In contrast, the use of learning algorithms in the area of object recognition is rather evident and needs no further explanation. For a basic introduction to learning in the context of pattern recognition, the interested reader is referred to the book of Duda, Hart and Stork [194] and to the book of Bishop [27]. Hereafter, only approaches are mentioned, which introduced a given learning algorithm into the domain of object detection and significantly influenced algorithm development in the past.

The work of Hopfield inspired the use of Neural Networks for pattern recognition in the beginning of the 1980s [99, 100]. For example, Fukushima proposed the use of neural networks for digit recognition [81]. Rumelhart *et al.* introduced the *Back-Propagation* algorithm for training neural networks [202]. Neural Networks (NN) [26] for building descriptions of faces were proposed by Vaillant *et al.* [246] and by Rowley *et al.* [201]. The neural network is trained to implement a filter which is separating objects, respectively faces in this case, from background. The general object representation included in the neural network is trained from a set of objects. In the approach of Sung and Poggio, clustering and distance metrics are used in combination with a neural network [227]. First the object and non-object manifolds are modeled by clusters. The neural network is then used to classify patches by their distances to and from clusters as objects or non-objects.

The use of *Support Vector Machines* (SVMs) [248] for representing faces was proposed by Osuna *et al.* [174] and by Romdhani *et al.* [195]. In the original approach of Osuna *et al.* a polynomial-2 SVM is used to separate face images from non-face images, reaching detection rates similar to those of Sung and Poggio. Later Romdhani *et al.* modified the SVM algorithm to reduce the number of support vectors needed and to speed up the approach considerably by using a sequential evaluation and early rejection of patches.

Colmenarez and Huang proposed the use of Markov Chains for face detection [51] and Dass and Jain extended the approach using *Markov Random Fields* (MRFs) [59]. MRFs for building models for object recognition have been proposed by Caputo *et al.* [41]. They presented a special version of MRFs and demonstrated the plausibility of their approach on a 100 object database.

The currently most popular algorithm for generating a description of objects from a predefined category was proposed by Viola and Jones [250]. The algorithm was invented for rapid object detection and is based on combining classifier responses using Boosting [206, 207]. The appearance of objects within a given class is modeled using a set of weak classifiers which are arranged locally and are based on simple Haar wavelets or more complex features like gradient histograms. Each single weak classifier delivers a binary decision, or discrete valued confidence rate, which just has to be better than random. Multiple of these classifiers are combined as a weighted linear combination into a stage by sequentially selecting those which deliver the smallest classification error on a given training set. If a predefined performance threshold is reached, the stage is terminated and a new stage is begun, after the training set is refilled to include enough positive

and negative examples. Multiple stages are combined to form a single strong classifier capturing the representation of the object class. Details about Boosting will be given in chapter 4.

As already mentioned before, one important aspect of learning based approaches is their ability to be adaptable to different situations, objects and environmental conditions. Recent advances in artificial intelligence allow for efficient training and building of highly accurate and extensive object representations. Most of recent approaches include some type of learning, which also fortifies the special relevance of learning for computer vision.

3.1.5 Other Algorithms

Other types of algorithms proposed in the context of forming object representations include *Nearest Neighbor* (NN) classifiers, *Perceptron* classifiers or the *Winnow* algorithm, amongst others. For a more in-depth description and review, again, the interested reader is referred to the book of Duda, Hart and Stork [194] and to the book of Bishop [27].

3.2 Related Studies in Object Detection

For the task of object detection a few popular approaches exist whereas the majority of them is based on global object representations. After building an object representation an exhaustive scan over entire images in multiple scales is performed to locate instances of an object. In the literature almost all algorithms using a global representation were introduced given the problem of face detection. However, the basics of the methods are not limited to the face detection problem but were formulated more generally later and can be applied to any other object category under certain circumstances. Beside global methods only a few approaches exist which are based on local features. We will treat both types of approaches in the following. Note that we have used a rather rough classification in a set of subgroups, and that some approaches, only mentioned once, might belong to an other group as well. A table summarizing all approaches mentioned here is given at the end of this section in Table 3.1.

3.2.1 Subspace based Detection

Kirby and Sirovich introduced the *Principal Component Analysis* (PCA) to computer vision for characterization of human faces [120]. The so-called *Eigenface* approach was then proposed and investigated more exhaustively by Turk and Pentland for face recognition [244], applying PCA to a set of training images. Interpreting each image in the training set as a long linear array, they all reside in a small low-dimensional portion of the high dimensional image space, the so called *face space*. By employing PCA, this face space is described by a given number of eigenvectors which best describe the variation of the data. Given these eigenvectors, any image patch can be transformed into the *face space* and its distance can be calculated. Applying this transformation on all subpatches of larger

images, a distance map can be generated and detection can be done by putting a threshold on the distance. Later, Moghaddam and Pentland enhanced the detection approach using a probabilistic measure instead of the Euclidean distance to improve detection accuracy [160, 161].

Yang *et al.* proposed the combination of several linear subspaces to form a powerful face detector [272]. First, a *Self-Organizing-Map* (SOM) is used to separate face and non-face examples into different clusters. Labels are assigned to the training examples by assigning labels to the clusters. Then, FLD is used to maximize the ratio of between-class scatter and within-class scatter. Finally, the training sets are projected into a lower dimensional feature space, and a Gaussian distribution is used to model the class-conditional density function for the individual classes. For detecting faces in a test image, a detection is determined upon the maximum-likelihood criterion calculated from the Gaussian density functions.

3.2.2 Support Vector Machine and Neural Network based Detection

The approach invented by Osuna *et al.* [174] and later enhanced by Romdhani *et al.* [195] uses an object representation based on a Support Vector classifier, which is applied to subwindows of an image exhaustively to deliver the number and locations of detected faces. Similarly, Papageorgiou *et al.* used an overcomplete set of Haar wavelets and a SVM classifier for the face detection task [178]. For the task of person detection in natural street imagery, they also trained a person detector. The detection results of the exhaustive scan are enhanced by incorporating motion information between consecutive frames and adding a special interest region to hypotheses of previously detected persons.

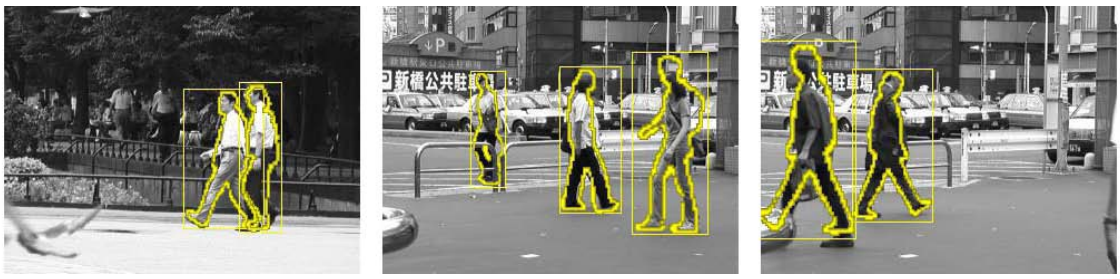
Similar approaches were proposed by Vaillant *et al.* [246] and by Rowley *et al.* [201]. In these methods, a single subpatch is the input for a neural network classifier. Input images are consecutively subsampled by a factor of 1.2 to build an image pyramid. 20x20 pixel subpatches are exhaustively extracted from each layer of the pyramid and forwarded to the classifier to achieve multi-scale detection. The neural network contains three different types of receptive fields as input and a variable number of hidden units. Due to the backpropagation based training of the network, in each training cycle additional false positives are added as new negatives for the next training step, which is also denoted as *bootstrapping* procedure. After an entire detection process overlapping detections are discarded by selecting the detection with the highest confidence rate and suppressing all others. The approach leads to acceptable performance rates in terms of detection accuracy and Rowley *et al.* also demonstrated the real-time capability of the approach [201].

Another neural network based approach was proposed by Sung and Poggio [227]. From a set of 19x19 pixel face images, a canonical face model is generated by fitting six multi-dimensional Gaussian distributions into the image space by using a modified k-means clustering algorithm. Another six multi-dimensional Gaussian distributions are fitted to a set of non-face, but face-like image samples. The models are simplified by a PCA-like

projection to a lower dimensional "face-space", and two distance metrics are introduced to measure face or non-face similarity. A multi-layer neural network is finally trained to classify single image patches.

A shape based approach for object detection was proposed by Gavrila and Philomin [83]. Shape templates of pedestrians are hierarchically ordered and a distance transform is used to generate initial object hypotheses. Finally a *Radial-Basis-Function* (RBF) neural network, whose application and structure is not described in detail, is used to verify the initial detection result. This work is mentioned, because the authors try to optimize the algorithm for a Pentium based hardware platform, addressing aspects of hardware-related algorithm optimization similar to our own considerations treated earlier. Later, Leibe *et al.* proposed a combined approach of local and global features for pedestrian detection [133]. Similar to the work of Gavrila and Philomin, for the global detection cue a silhouette template and Champfer matching is used.

(a)



(b)

Figure 3.3: Pedestrian detection. (a) Some results of the detector developed in [251]. (b) Some results presented in [133]

More recently, Dalal and Triggs proposed the use of linear SVMs and *Histograms of Oriented Gradients* (HOGs) for pedestrian detection in cluttered scenes [57]. The method is mainly based on the accumulation of several gradient histograms on small blocks of a given sample image. This collection of histograms forms a relatively simple, yet powerful set of spatial descriptors. By training a SVM on a positive and negative training set, the weighting function for the individual blocks is determined. Finally the presence or absence of a pedestrian in a test image is, again, determined by an exhaustive scan and non-maximum suppression.

3.2.3 Boosting based Detection

Schneiderman and Kanade proposed to use a statistical combination of visual features, which are represented as histograms of quantized wavelet coefficients [210]. For training a detector containing the most suitable features, the AdaBoost algorithm is used. The approach allows for 3D object detection, using separate detectors for different object viewpoints, but is far from real-time capability due to the detector size and complexity.

The ground-breaking work of Viola and Jones established the use of Boosting as a very powerful method for object detection in the computer vision community [250]. The first important property of their approach is the use of an intermediate image representation, which mainly allows the calculation of simple Haar wavelet features in constant time. The second contribution is the combination of weak classifiers, trained on those features, in a cascade like structure to allow for early rejection of non-object images during detection. After the approach of Rowley *et al.* [201], this approach also allowed for real-time detection of objects based on appearance, and was shown to perform well on the task of non-rigid object detection in single views.

Since the initial publication of the basic algorithm, a number of extensions have been proposed, mainly focusing on the calculation accuracy, the cascade building approach and the type of features used. Lienhart and Maydt proposed an enhanced set of Haar wavelets that are calculated using a second type of image representation [139]. Implementations of the original approach and the enhanced set of features is nowadays available as open-source software in the OpenCV library [236]. For the task of face detection, Chen *et al.* proposed the use of Gabor wavelets [45] as weak classifiers. Gabor filter based features are computationally more expensive, however, also being more powerful than simple Haar wavelets. As the evaluation of their approach can be interpreted, the number of features can be reduced through using more powerful features. However, the overall computational complexity is not reduced significantly, which coevally means that the use of advanced features only has limited benefits.

For detecting pedestrians, Viola *et al.* proposed features combined from simple Haar wavelets and motion information [251]. The main advantage of the approach is that the important queue of motion information, if motion is present at all, is directly used to build a more powerful detector with less false detections. However, if pedestrians are not moving, the purpose of the special motion patterns is lost and the approach reduces to the original approach proposed. Another approach for human detection was based by Mikolajczyk *et al.*, using a probabilistic model of body parts [158]. Seven different body parts are defined, which are learned using gradient and filter response features and AdaBoost. The final strong classifiers for individual body parts are combined to vote for the final hypothesis of a human body.

A general framework for object detection and recognition, which is based on a variety of different local features, was proposed by Opelt *et al.* [170]. Using two different interest point detectors and four different descriptors, a rich description of the content of an

image is created. Weak classifiers are built from single features, whereas their presence or absence, together with a threshold, is taken as parameters. Final strong classifiers are built using the AdaBoost algorithm and a set of training images from a given category. The approach was shown to achieve good performance for generic object detection in natural images. Later, Opelt *et al.* proposed a different set of features for object detection, so-called *boundary fragments* [171]. These features are edge templates, from which a vocabulary is learned using the AdaBoost algorithm. The fragments are mainly used to model the outline of objects and to vote for an object hypothesis. The approach is shown to perform well for the detection of complex categories, such as cows or bottles.

(a) (b)

Figure 3.4: Images taken from the work of Agarwal and Roth [2]. (a) Image patches extracted using the Förstner operator on a set of training images. (b) Examples for clusters of representative, visually similar patches.

For generating detectors for multiple object classes, or classes with high intra-class variation, different approaches have been proposed recently. The main idea is to use a tree-like structure, because it allows for compact and efficient detection. Amongst others, approaches were proposed by Tu [243] or Huang *et al.* [103]. Wu and Nevatia proposed the *Cluster Boosting Tree* method, which combines an unsupervised clustering approach with the idea of Boosting for forming a classifier tree [266]. Torralba *et al.* proposed the combination of "stumps" of boosted classifiers for forming a classifier for multi-class object detection [242].

3.2.4 Local Feature based Detection

Agarwal and Roth introduced an approach for object detection based on a representative vocabulary of visually similar parts of an object class [2]. First, interest points are detected from a training set of images using the Förstner operator and fixed-size patches are extracted around the detected keypoints. Based on a similarity measure, visually similar patches are clustered (see Figure 3.4). All training images are subsequently represented as binary feature vectors, which capture both the presence of a visual feature and its spatial relation to other features. The *Sparse Network of Winnows* (SNoW) algorithm, a Winnow based learning algorithm, is used to learn a powerful classifier over the training vectors. For detection, the notion of a *Classifier Activation Map* is introduced, which allows to

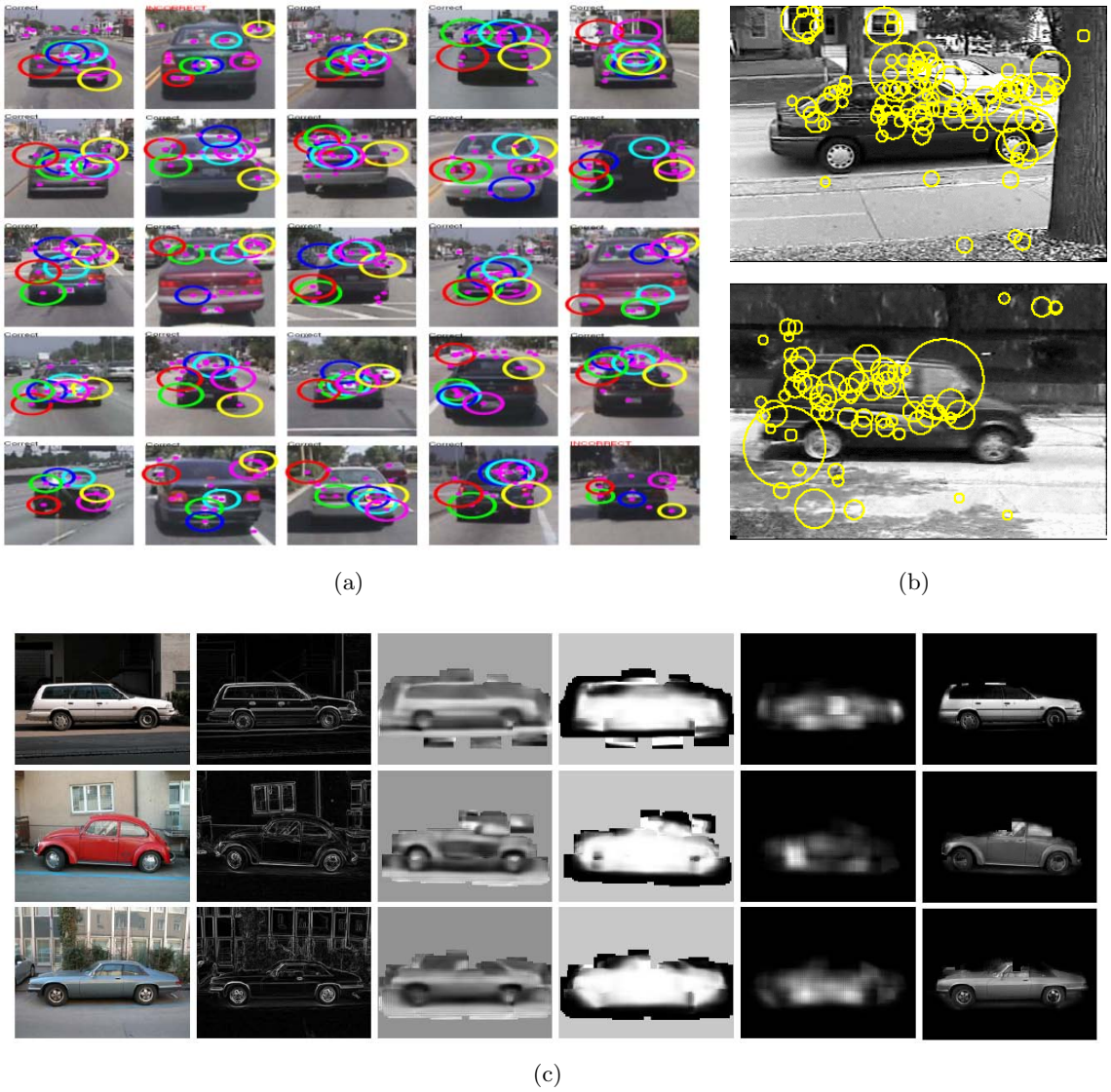


Figure 3.5: Vehicle detection results. (a) Results for rear-view vehicle detection in [71]. (b) Results for side-view vehicle detection, taken from [66]. (c) Results for side-view vehicle detection and vehicle segmentation, taken from [132].

assign each location in an image a confidence value for the presence or absence of an object.

Dorko and Schmid proposed a framework for constructing object part descriptors from scale-invariant local features [66]. Two learning approaches, namely SVM and *Gaussian Mixture Models* (GMMs) are used to learn a classifier for a single part. By combining multiple part detectors, they achieve a robust detection under scale changes and variations in viewing conditions.

Leibe and Schiele proposed a system for interleaved object detection and segmentation [132]. Based on the idea of Agarwal and Roth, a vocabulary of representative image patches is built, using the Harris detector to acquire seeds and an agglomerative clustering algorithm to cluster visually similar patches. In the detection stage, the parts detected in objects are voting individually for the center of a potential object hypothesis. For each pixel, a probability function is calculated, which predicates the likelihood of the pixel being part of an object hypothesis. The approach is shown to perform well and delivers good results in segmenting objects from background.

Later, Leibe *et al.* proposed a combined approach of local and global features for pedestrian detection [133]. As local features DoG points are used, which serve as seeds for the extraction of plain image patches. These patches are resized to a common size of 25x25 pixels and are clustered using an agglomerative clustering approach. Only the centers of the resulting clusters are stored as representatives of appearance in a compact codebook. During detection, the keypoint detection is performed, and matches are recorded to vote for object hypotheses, in which the normalized grayscale correlation is used for determining correspondences together with a Hough voting procedure.

Mikolajczyk *et al.* evaluated a representative set of detector and descriptor combinations for building an object representation, and showed the plausibility of their approach on the task of pedestrian detection [154]. The evaluation of the approach is similar to the one of Leibe *et al.*, but uses the best performing combination of detector and descriptor as local feature cue.

A real-time capable object detection approach based on local features was proposed by Lepetit *et al.* [137]. In an offline stage, keypoints are detected on a specific object, whose views are artificially translated, rotated and affinely distorted. The keypoints are organized in a set of multiple randomized trees. The approach has two nice properties. First, since the query and matching during runtime can be accomplished efficiently, it is possible to detect an object in realtime. Second, also partially occluded, deformable and 3D objects can be detected due to the use of local features and the artificial distortion mechanism during training. As an extension to randomized trees, so-called *ferns* were proposed by Ozuysal *et al.* to further speed up the matching step [176]. By using a Naive Bayesian framework, faster and more robust classification results are achieved.

3.2.5 Notes

There exists a large number of different approaches for object detection proposed in the literature without reference to any hardware platform. A listing of all approaches mentioned in this Section is given in Table 3.1. It is on the developer to choose the most suitable algorithm from this set, to meet performance limits and predefined requirements. This is of special importance, for example, in the automotive domain, as algorithms have to be highly robust and have to deliver high performance. Any failure of a system may be fatal to somebodys health and, possibly, life.

Subspace Based Detection	**SVM and NN Based Detection**
Kirby and Sirovich [120] Turk and Pentland [244] Moghaddam and Pentland [160, 161] Yang *et al.* [272]	Osuna *et al.* [174], Romdhani *et al.* [195] Papageorgiou *et al.* [178], Vaillant *et al.* [246] Rowley *et al.* [201], Sung and Poggio [227] Gavrila and Philomin [83], Leibe *et al.* [133] Dalal and Triggs [57]

Boosting Based Detection	**Local Feature Based Detection**
Schneiderman and Kanade [210], Chen *et al.* [45] Viola and Jones [250], Lienhart and Maydt [139] Viola *et al.* [251], Mikolajczyk *et al.* [158] Opelt *et al.* [170, 171], Tu [243], Huang *et al.* [103] Wu and Nevatia [266], Torralba *et al.* [242]	Agarwal and Roth [2], Dorko and Schmid [66] Leibe and Schiele [132], Leibe *et al.* [133] Mikolajczyk *et al.* [154], Lepetit *et al.* [137] Ozuysal *et al.* [176]

Table 3.1: Summary of the object detection approaches mentioned in Section 3.2.

Recently, a lot of attention is paid to the Viola-Jones algorithm, due to its robustness, high performance and, even more important, because of the large range of possible applications, adaptations and modifications for optimization. It is evident that some form of this approach can definitely serve as one module within a larger system, which delivers object hypotheses with satisfying accuracy and high reliability. Thus we also feel confident, that exploiting the possibilities of adaptation of this method onto embedded hardware is a very important step towards development of a reliable smart camera setup for surveillance.

3.3 Related Studies in Object Recognition

Object recognition is the task of identifying a specific object, or the membership of a given object to a predefined class. Depending on the object appearance, rigidity and object shape, different approaches have been proposed, using likewise global and local object descriptions and representations. However, recently, local feature based representations have attracted much interest, due to their flexibility in usage, the advances in rapid computing, and their other advantages over global methods.

For an overview of a lot of different methods in the area of face recognition, the interested reader is referred to the *Handbook of Face Recognition* by Li and Jain [138]. A prominent, yet incomplete set of approaches follows, which is on object recognition from a more general point of view and is, however, especially focusing on local feature based methods. Again note that we roughly structure the methods into subgroups, and that affinities with other groups are evident.

3.3.1 Subspace based Recognition

The *Eigenface* approach of Turk and Pentland was also used to recognize faces from several different individuals [244]. In this context, a probabilistic similarity measurement was proposed by Moghaddam and Pentland and applied to different recognition tasks, such as hand gesture recognition or face recognition [160, 161]. Using the method proposed, it is possible to also determine the pose of a face looking at the face space of a given individual.

Belhumeur *et al.* proposed the use of Fishers Linear Discriminant (FLD) to build an object representation, the so-called *Fisherface* approach [20]. While PCA is used to retain the subspace with the highest amount of variation in the data, this becomes definitely a problem, if this variation is mainly caused by changing illumination or change of (facial) expression. For discriminating objects from each other, the FLD can be successfully applied to find the best linear separation between individual classes, regardless of this type of variation. Consequently this approach achieves better performance than the original Eigenface approach. For a more elaborate introduction and overview to these and other subspace methods in the context of face recognition, the reader is referred to the work of Shakhnarovich and Moghaddam [213].

For more general object recognition tasks, the use of a robust method and multiple Eigenspaces was proposed by Leonardis and Bischof [135]. By combining a random sampling strategy with the selection of the right hypothesis according to the *Minimum Description Length* (MDL) principle, a framework for robust object recognition in small size databases is presented. Later, Leonardis *et al.* extended the approach to incrementally select and grow the number and shape of Eigenspaces to cope with a larger number of objects [136].

3.3.2 Local Feature based Recognition

Object representations are collections of local, usually unordered, features. The recognition task is usually accomplished by a voting based scheme, where correspondences between features of a query object and several object representations are accumulated. The hypotheses cumulating the highest number of votes is assumed to be the correct match. Spatial information is used optionally to achieve higher performance rates or to verify hypotheses.

A set of successful approaches was proposed combining probability theory and spatial information of features. A framework for object category detection and recognition, which combines both appearance and spatial locality of features in a probabilistic model, was proposed by Fergus *et al.* [71]. In their approach salient regions are detected during a training stage, and generative models for different object categories are learned. The correct object category is determined by detecting features and evaluating the models according to a maximum likelihood criterion.

A graphical overview about the multitude of other constellation models for incorporat-

ing spatial information is given in the work of Carneiro and Lowe [43], together with their own approach. They propose a model in which each feature is geometrically dependent on its k nearest neighbors. This is based on the assumption that the local geometry of features is more constrained than the global one and less likely to change severely.

Concerning the problem of efficient descriptor matching, Lowe used a *kd*-tree with best-bin-first modification to organize SIFT descriptors from training images and to find approximate nearest neighbors to the descriptors in query [142]. A post-verification step is subsequently used for consistency checks and to confirm or reject matching correspondences. A binary decision tree to index keypoints and to minimize the average time to decision was used by Obdrzalek and Matas [254]. A few local image areas are represented by the leafs of the tree and each inner tree node is associated with a *weak classifier*. Robustness of their approach against partial occlusion, background clutter and large viewpoint changes was shown on a database of multiple hundred objects.

Sivic and Zisserman first introduced the ideas from text retrieval into the area of image matching [219]. As a visual analogy to a *word* they defined the vector quantization of descriptors and introduced the *inverted file* mechanism to facilitate efficient retrieval. By quantizing multiple descriptors into clusters using the k-means algorithm a *visual vocabulary* is built and used for keyframe retrieval in videos. Consequently in the approach of Nistér and Stewénius [167] hundred thousands of descriptor vectors are quantized using k-means clustering in a hierarchical vocabulary tree for image retrieval, being capable of organizing a database of 1 million images. The results of an initial scoring scheme are verified by using the geometry of keypoints matched to further improve image retrieval quality. Combining the ideas of vector quantization and hierarchical clustering results in real-time behaviour of matching.

3.3.3 Notes

Local feature based approaches have attracted a lot of attention recently, as remarkable results on large databases with a huge number of different objects were achieved. Especially the introduction of tree structured databases for efficient matching has inspired a lot of work in the direction of vocabulary trees and large databases.

Not only the robustness to occlusion and background noise, but also the possibility to apply local features in large scale object recognition systems, using efficient storage and management methods, has made local feature based approaches quite popular. Vocabulary trees, together with a suitable combination of feature detectors and descriptors can be seen as state-of-the-art in todays object recognition. Especially the modular design of current object recognition systems, and the architecture of different local feature algorithms, make them attractive for realization on embedded devices. We conclude that in the foreseeable future, the clever combination of an interest point detector and a descriptor will definitely be part of any successful object recognition system, in order to deploy certain object recognition capabilities in the area of household robotics, autonomous navigation

and mobile computing. We are certain, that local feature based object recognition technology can serve as a key component in embedded system technology for a large variety of applications, like public surveillance amongst others.

3.4 Special Approaches for Embedded Devices

In the last two Sections, we listed a large number of different algorithms from the field of object detection and object recognition. However, it is rather obvious that only a small set of algorithms is really suitable for implementation on embedded hardware. This fact is also reflected by the, relatively small, amount of literature dedicated to the development of high-level algorithms for embedded hardware.

In the following, we will list a representative number of approaches dedicated especially to embedded systems. Note, that due to the ambiguous use of the terms *detection* and *recognition* in the literature a clear boundary between both areas cannot be drawn.

3.4.1 Object Detection

For a long time object detection on embedded hardware was performed by primitive image processing techniques, like background subtraction and morphological operations. However, due to the increase in computational and memory resources more evolved methods have been proposed recently. This also implies that the trend in development is from the detection of specific objects, eg. human faces by colour-based algorithms, to more generic methods for detection of various objects through reconfiguring the basic algorithm. A good example in this case is the Viola-Jones algorithm for detection of vehicles, faces or other object types. However, there are only a few dominant domains in which detection algorithms are benchmarked on embedded systems. These mainly include 1) human face detection and 2) person and vehicle detection for surveillance purposes and in automotive engineering.

In the first case the main reason is simply the (relatively) small variability in appearance of faces and the large variety of applications in customer market. The motivation behind pedestrian and vehicle detection in surveillance is simply the necessity to monitor and actively control the traffic flow on todays transportation routes to decrease delays and avoid incidents. Furthermore in the individual topic of automotive engineering the motivation is simply the need for "helping hands" as driver assistance systems due to the considerable increase in traffic, vehicles, accidents and injuries respectively. Clearly there is also a commercial motivation from the automotive industry driving the development in this domain.

In the following, we give a short and fragmentary overview about the current state-of-the-art on object detection on embedded hardware. We will also treat other object detection methods that have been reported, mainly in the context of vehicle detection from stationary cameras. To justify this, indisputably face and vehicle detection both

have special relevance to surveillance and aspects of public security. For an extensive overview about computer vision in the automotive domain the interested reader is referred to the review of Sun *et al.* and the included references to other surveys and summaries [226]. In the context of pedestrian detection in the autonomous domain, the interested reader is referred to the work of Gavrila and Munder [82]. We will mainly summarize the approaches here, that help to cover the variety of methods proposed yet.

We divide the methods proposed in the literature into two groups. The first group contains the algorithms implemented in hardware, and the second group is the set of approaches developed entirely in software. We believe, that the separation of algorithms into these two groups helps to understand the basic differences in algorithm behaviour and development.

3.4.1.1 Dedicated Hardware Solutions

McCready proposed a detector based on the *Quantized Magnitude/Phase* (QMF) transform [148]. Using a total of 9 FPGAs the system is able to detect faces in 320x240 pixel images at 30 fps. However, detailed evaluations on the detection rate are omitted. Paschalakis and Bober proposed the FPGA implementation of a colour based face detection and tracking system for mobile video conferencing [180]. The detection algorithm relies on a statistical histogram-based skin colour model which is created offline and can be adapted in a setup step. After assigning each pixel a probability for containing a facial colour a simple segmentation algorithm is used to find connected components. Due to the focus on mobile communication, the system is designed to run on 176x144 pixel QCIF images and can theoretically perform at frame rates of over 430 fps. Another advantage is the invariance to viewpoint, however, erroneously also detecting other regions of skin colour, like human hands.

The implementation of a Radial-Basis Function (RBF) neural network in hardware on a FPGA and a *Zero Instruction Set Computer* (ZISC) was proposed by Yang and Paindavoine [269]. The network was built to detect and coevally recognize the faces of persons. The performance of the FPGA and the ZISC implementation on the ORL face database was 92% and 85% respectively. The implementations allow for processing 352x288 pixel images at 14 fps and 25 fps. Theocharides *et al.* reported the implementation of the neural network based method proposed earlier by Rowley [201] in hardware on an FPGA [240]. The solution is able to perform at 424 fps on 300x300 pixel images, achieving about 75% of detection rate (compared to a time consumption of about 2 seconds per image and 84 % detection rate using the software approach). Later Murali *et al.* proposed an enhanced version being able to process 320x240 pixel images at 40 frames per second while maintaining 94% detection accuracy [163].

Wei *et al.* [259] propose the implementation of the Viola-Jones detector for face detection on an FPGA architecture. The system is able to process 120x120 pixel images at 15 frames per second, however the performance of the system in terms of detection

rate is only moderate. Yang *et al.* proposed the implementation of the Viola-Jones detector on an Altera Cyclone II FPGA for inclusion in handheld cameras or mobile phones equipped with a camera module [271]. They propose to use a special method to early discard subwindows which are unlikely to contain a face, in order to meet hard real-time constraints and to achieve real-time performance. Although the method performs well, the detection performance is still significantly lower than in the original approach proposed by Viola and Jones [250]. The implementation of a full detection processor based on a FPGA architecture was proposed by Hiromoto *et al.* [97]. In their approach an entire classifier is divided into two parts where the separation threshold is determined by looking on the amount of time spent in each stage during a usual sequential detection process. By calculating weak classifiers of stages in the first part in parallel a significant speedup over sequential execution can be achieved, allowing for real-time execution at 30fps on images with 640x480 pixel resolution.

Nair *et al.* proposed a Viola-Jones person detector for indoor corridor scenes based on JPEG compressed images using an FPGA [166]. The algorithm is working on 216x288 pixel images corresponding to 3079 subimages and can process 2.5 frames per second. The main contribution of this work is the method for approximation of the integral image, which can be done from the DCT coefficients of the compressed image directly. This renders the computation of the inverse transform to restore the uncompressed image unnecessary.

Cucchiara *et al.* reported the implementation of a vehicle detection and tracking system on a GigaOps G800 Spectrum board [56]. The system is able to detect and track vehicles at day- and nighttime using a total of 4 Xilinx FPGAs and 16 MB of RAM. The approach is reported to work on full PAL resolution in real-time where the algorithmic core of the implementation is a background differencing method. Kaszubiak *et al.* proposed the use of disparity map calculation using a stereo camera setup in a car for vehicle detection on highways [114]. By clustering fragments in the depth map the presence and distance of a vehicle is determined, while the relative velocity is estimated by a Kalman tracking step. Their system is a hardware and software co-design approach, consisting of an FPGA which contains the entire depth map calculation as a hardware solution, and an ARM9 RISC processor running at 166MHz for clustering and Kalman filtering.

3.4.1.2 Algorithmic Software Solutions

Yang *et al.* proposed to approximate the oval shape of faces with an ellipse [270]. By using a variant of the Hough transform and parallelizing the calculation process on two TI TMS320C40 DSPs the detection of faces in 143x123 pixel images at 3.2 frames per second was accomplished. Later, the implementation of a RBF neural network for usage on a TI TMS320C62xx DSP was proposed by Yang and Paindavoine [269]. An average throughput of 4.8 CIF frames per second is reported.

Fatemi *et al.* reported the implementation of a color based face detection algorithm on the predecessor of todays WiCa platform [70, 122]. Skin colored blobs are detected based

on a color space transformation and thresholding. False detections of skin colored regions such as the human hands are discarded looking at the aspect ratio of the blobs. Various aspects about Viola-Jones detector implementation on SIMD architectures were first treated by Reuvers [193]. While the use of separable filters for efficient feature computation is proposed, more complex filters like Gabor filters are also evaluated. The approach is dedicated to run on a Xetal architecture and is reported to run at 4-5 frames per second on 640x480 pixel images on a TriMedia 1300 DSP based smart camera running at 133 MHz. Jeanne *et al.* proposed the detection of faces on the WiCa platform by using a so-called smooth edges technique and a stereo camera setup [108, 123]. In principle the approach is based on edge detection in the horizontal and vertical direction and searching for locations in the binary image where a predefined number of edges is present. Finally a validation of the results is performed by comparing the results from both camera streams and forming a final result. Wu *et al.* presented the implementation of a gesture recognition algorithm on the WiCa platform [268]. After background subtraction, the k-means algorithm is used to find dominant color modes. Ellipses are fitted to segmented color regions which are the extremities of the actor. By combining the ellipse information from multiple cameras an articulated 3D skeleton model is animated.

Bramberger *et al.* proposed a smart camera platform based on a TI TMS320C6416 DSP [33, 30]. The plausibility of the solution was demonstrated using a background modeling algorithm in a tunnel environment for the task of stationary vehicle detection. Chiu *et al.* proposed an embedded vision system based on a CMOS imaging chip and a RISC processor in order to perform real-time car counting [47]. The entire system consists of a network of 13 embedded vision platforms that are applied to monitoring and controlling the access to a parking facility. The system is mainly based on detection of markers and their occlusion through passing vehicles. It is able to perform at day and nighttime and can operate at 30 fps on 320x240 pixel images. Mathew *et al.* performed an evaluation of various nameable algorithms like the Rowley [201] and the Viola Jones [250] face detectors to gain insight into the suitability of these algorithms for implementation in the embedded domain [146, 147]. Although the study contains not an explicit evaluation of any algorithm in terms of detection performance, important aspects of the caching strategy inside a system and potential ways of parallelism are discussed. Rahimi *et al.* deployed an object detection algorithm on their Cyclops smart camera platform [190]. The algorithm is based on background modeling using a moving average technique and is able to perform at about 4 frames per second on 128x128 pixel frames. Yeh *et al.* [275] proposed a background subtraction based approach as preprocessing step on a mobile phone. Only the segmentation step is performed, the object information is sent to a central server for further processing.

3.4.2 Object Recognition

Object recognition is a much more inexperienced field of development in the context of embedded systems. In contrast to the field of object detection, recognition methods have been limited to very specific object categories and dedicated applications. Only a few areas of research exist, like character recognition for industrial environments or mobile devices [15], gesture recognition [190], but also for license plate recognition in the area of surveillance [21, 112, 140]. Another example is the automotive domain where algorithms have been proposed for sign or obstacle detection [69], or the recognition of human faces in biometric systems [18, 19, 269], which often perform both detection and recognition coevally. However, in the case of vehicle recognition or reacquisition the approaches are merely limited to license plate reading, which itself is based on the use of SVMs and neural networks. There simply exists no comparative work in the domain of generic object recognition on embedded systems, thus the following collection of related publications is fragmentary and diverse.

In the following we will especially treat related work in the context of face recognition on embedded systems to show that the main types of algorithms, that have been deployed yet, are based on learning methods and/or subspaces. We also refer to recognition systems based on other algorithms that have been proposed for diverse applications and describe what is the current state-of-the-art in recognition.

3.4.2.1 Dedicated Hardware Solutions

Regarding the area of local features, Benedetti and Perona proposed the implementation of the Harris corner detector [91] on a GigaOps G800 Spectrum board using six Xilinx FPGAs [23]. The system is able to run in real-time on 320 x 240 pixel images. Another implementation on a Spartan-3 FPGA was proposed by Giacon *et al.*, whose system is able to run at 512x512 pixel images at 46 frames per second [85].

Se *et al.* denote the implementation of Lowes SIFT for autonomous robot navigation on an Virtex II Xilinx FPGA [212]. However, details about the implementation and design of the system are not public, although they claim to extract features from a 640x480 pixel image in about 60 ms, compared to about 600 ms for a Pentium III 700MHz processor. Recently Cabani and MacLean proposed a pipeline architecture for implementing a multi-scale Harris corner detector on an Altera Stratix S80 FPGA [39]. Candidate points are fed into an iterative procedure to determine the local affine shape of the features neighborhood and to achieve affine invariance. Although the design and occupancy of diverse parametrized versions of the algorithm on FPGAs is outlined in detail, experimental results in terms of detection rate and throughput are omitted.

3.4.2.2 Algorithmic Software Solutions

Following the approach of Turk and Pentland [244] Batur and Flinchbaugh built a system based on a TI TMS320C6416 DSP running at 500Mhz [18, 19]. The system needs about 3.7 seconds to detect and recognize a face in a 640x480 pixel image. However, two different approaches in the face classification stage were evaluated, achieving 88% and 93% success rate respectively. Yang and Paindavoine built a detection and recognition system based on a RBF neural network for usage on a TI TMS320C62xx DSP [269]. The system is able to recognize and track 10 different persons at 4.8 fps on 352x288 pixel images and a recognition rate of 98.2 % on the ORL database is reported. Fatemi *et al.* presented an RBF neural network implementation for face recognition on the predecessor of the WiCa platform, at that time using a Trimedia DSP running at 166 MHz [70, 122]. Based on a database of 5 known faces the algorithm is able perform at recognition rates between 90% and 97% depending on the parametrization and needs 4.2ms to recognize a single 64x72 pixel face image. Wei and Bigdeli reported the implementation of a face detection and recognition system based on an Analog Devices ADSP-BF535 EZ-KIT Lite^{TM}platform featuring a Blackfin DSP running at 300 MHz [258]. A neural network is used for detection and recognition and the authors emphasize the special computational burden during detection and the considerable memory consumption during the recognition step, omitting detailed evaluations concerning the recognition performance. Günlü proposed the the use of a TI TMS320C6713 floating point DSP and a neural network classification of DCT coefficients [88]. Though the goal of the work is to find the optimal cache parameters for DSP configuration detailed evaluation results are omitted. The nice contribution of the approach is that training was performed on the development board and not on a usual computer, however, with only a small database.

Estevez and Kehtarnavaz [69] built a system to detect and recognize three different road signs using a Texas Instruments TMS320C40 DSP for a driver assistance system. Their approach is mainly based on color edge detection and using a special filter mask to scan over an image coevally accumulating edge pixel information. Ortmann and Eckmiller [173] built a real-time capable system for detecting and recognizing objects on a Texas Instruments TMS320C50 DSP running at 80 MHz. However, a big part of their work was focused towards integrated camera control, thus a detailed description of the methods involved and evaluations of recognition performance on their 58 object database were omitted.

Using a modified *kd*-tree for efficient feature vector organization was proposed by Bishnu *et al.* [24]. An object database is built from so-called Euler vectors which are calculated from binary forms of object images. While an implementation and evaluation on hardware is not performed explicitly, the design of a hardware system and a pipelining structure is motivated in the work of Bishnu *et al.* [25] and Dey *et al.* [62].

Recently Munich *et al.* built a library around Lowes SIFT which has already been applied in several commercial products [162]. However, although they have implemented

a full-scale object recognition system on DSPs and other platforms, details about implementation issues and performance evaluations are omitted.

Rahimi *et al.* reported the implementation of a gesture recognition algorithm on their Cyclops hardware system [190]. The system can differentiate 5 different gestures from the American Sign Language (ASL) at about 2 frames per second and 92 % of recognition rate.

3.4.3 Notes

The literature available on object detection and recognition algorithms for embedded systems is very limited to special applications. High level algorithms for object detection, such as the Viola-Jones algorithm, have been investigated rarely in the past. However, the large amount of research done in the area of Boosting and the development of powerful approaches strengthens our assumption, that this type of algorithm is a suitable choice for closer investigation in the context of embedded systems.

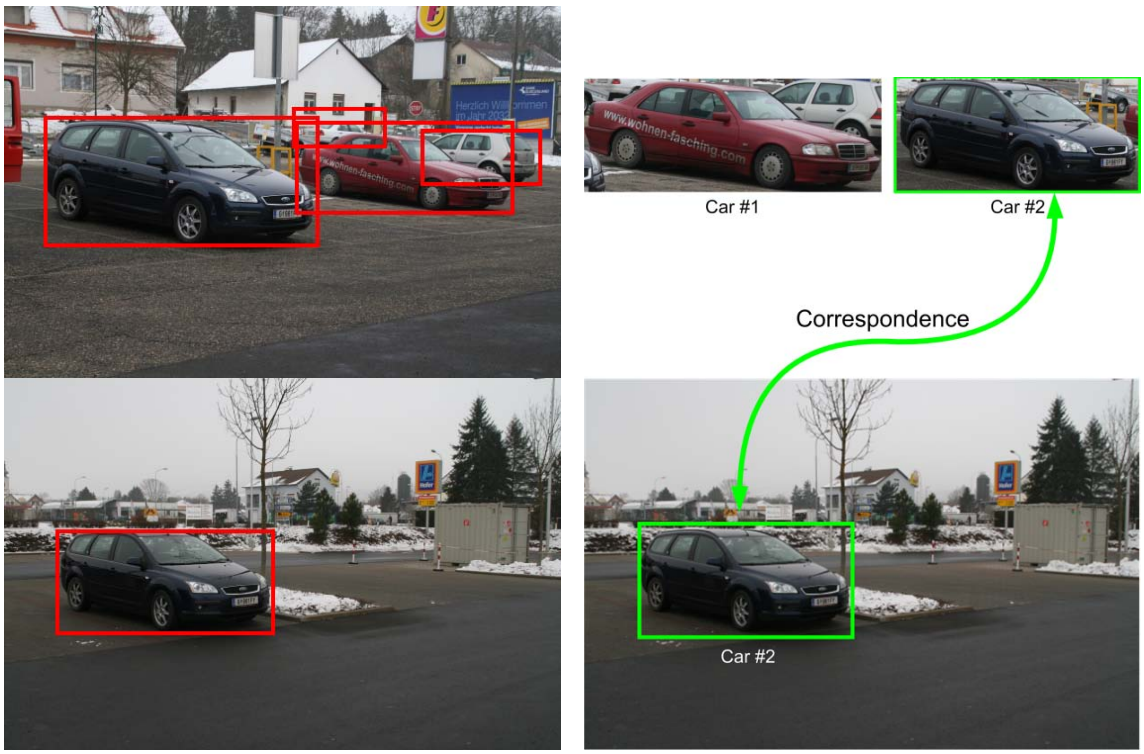
In the context of object recognition, since the rise of local feature based methods, there is a clear trend towards the implementation of filter-based algorithms in hardware, such as the Harris detector [23] or the SIFT approach [212]. Clearly, algorithms, which are data-independent in large parts, are very suitable for this type of implementation. The need for certain object recognition capabilities in smart cameras requires the implementation of a robust and high performant module for object recognition. Consequently, we assume that this methodology is also the right choice for building a generic object recognition system on an embedded device.

3.5 Algorithm Selection Criteria

To summarize the most prominent and popular algorithms for object detection and recognition, there is a large set of algorithms which have been proposed in the literature. Only a few algorithms have turned out to be suitable in the context of surveillance. Moreover, the selection of algorithms for smart cameras is rather difficult due to the restrictive nature of embedded hardware. Hereafter, general aspects concerning algorithm development and embedded systems are discussed. Subsequently, we revisit the main task to be solved in our thesis, ans give a justification and explanation of our choice of algorithms.

3.5.1 Embedded System Aspects

Since the development of smart cameras is a relatively young field of activity, literature about computer vision on smart cameras in a general context is only rarely available. Due to the restrictive nature of embedded hardware most of the existing approaches are based on rather primitive image processing techniques. Especially triggered by dedicated conferences on embedded computer vision [67, 68, 106], also more evolved methods are introduced into the area of smart camera development recently.



(a) (b)

Figure 3.6: (a) Detecting cars in images. (b) Recognizing a specific one as seen previously.

One important aspect of the survey of methods is, that pure hardware implementations of algorithms feature much higher performance in throughput over software based solutions. However, one crucial property of implementations in hardware is that this improvement in speed is mostly achieved by sacrificing a significant amount of performance in terms of detection or recognition accuracy. Although this effect has diminished recently due to the improvement in development tools for hardware implementations and the increasing reconfigurability of typical hardware solutions like FPGAs, this trend is still prevalent. This also means, that DSP or SoC based solutions have the additional striking advantage, that they can deliver high accuracy and performance. Even if real-time capabilities are not present first, however, an approximated prediction of the time of meeting real-time constraints can be given following Moore's law. Nevertheless, innovation in hardware design and development tools that get constantly better justify the usage of flexible solutions for rapid prototyping of vision algorithms in the domain of embedded systems.

3.5.2 Detection and Recognition on our Smart Camera - Thesis Goals Revisited

To recapitulate and redefine the main tasks of this thesis, we want to investigate the suitability of the currently top-performing algorithms in object detection and object recognition in the context of DSP-based embedded platforms. Without loss of generality, we do this given some specific tasks from the area of traffic surveillance. As an example, consider the task of object detection and recognition in the following scenario.

In Figure 3.6 (a), two time-separated images from two cameras on parking lots are depicted. Both images are taken from smart cameras, so there is no centralized processing engine available, but the data is processed at site using the DSP-based platform described previously.

The first task is to localize all cars in the upper and lower image in (a). Needless to say, that this task is done on each smart camera independently and there is no flow of information between individual cameras. After detection of the individual vehicles, the recognition task is to find out, if the object in the lower image in (b) was already seen previously in the upper image of (a), which coevally means, that one car has been sighted at both parking lots at different times of day. To state in an more general context, the recognition task can also be defined as finding out, where this individual vehicle has been sighted previously, given a set of vehicle observations at different locations and different times of day. This implies that a certain amount of communication between individual smart cameras is necessary.

We emphasize, that we do not explicitly discuss issues of entire camera networks in this thesis. Our main focus is on the investigation of state-of-the-art algorithms on standalone DSP-based smart cameras. However, it is clear that our work is also influenced by several aspects of smart camera and visual sensor networks. Thus we also use dedicated applications from these areas as examples to justify several of our own design choices and evaluate our algorithms. In the context of object recognition, note that we do not take license plate information into account, but, as a prerequisite, rely on appearance information of objects, or parts of objects, only. Neither do we take information about the geometry of objects into account. Moreover, as a special aspect of smart cameras, we emphasize, that a special goal of our investigation is the possibility to compress data as much as possible, without loosing too much information. It is rather obvious, that this is necessary due to the restricted amount of memory resources available, and the high cost of data transmission between individual smart cameras. This is an important aspect, because efficient data transmission is also a major point of investigation in recent work on smart camera networks [179, 217].

The best car safety device is a rear-view mirror with a cop in it.

Dudley Moore
English movie actor & musician, 1935 - 2002

Chapter 4

Object Detection

Object Detection denotes the process of determining the presence of a single or multiple objects and locating them. Needless to say, that object detection is a task which is of major importance, because object detection forms the first step of a larger setup for surveillance. A lot of different object detection approaches were proposed in the literature in the past, which are based on different assumptions and were proposed for different scenarios. Many approaches are very interesting because of their high robustness and high performance in terms of accuracy. However, often the advantages of these approaches are based on computations with high complexity and demanding requirements on the infrastructure in terms of computational and memory resources. Hence, it is rather obvious, that for applications with high demands on reliability, strongly limited resources and real-time constraints, only those approaches are suitable, that offer a certain amount of robustness, are applicable in various different scenarios, and are applicable for a reasonable computational effort. In this respect, mainly methods are considered, which allow trading computational effort against accuracy and performance in terms of speed. Under this perspective, the large number of methods proposed in the literature shrinks to a rather small number of interesting approaches.

As a reformulation of the term "object detection" in terms of a pattern recognition task, the goal is to find a classifier, which separates the group of objects to be detected from the huge set, formed by the overwhelming majority of objects that are not subject of interest. This set can also be described as *"everything else but the object"*, respectively. An important question is, how such a classifier is applied in a given algorithm. One method is the use as a verification step, where a preliminary result is tested to be confirmed or revoked. The main requirement for such a classifier is high accuracy. Compactness in representation and being computable in a fast way is less important, because the classifier is mostly applied on a small group of already available results. An other method of using a classifier is the application in an exhaustive manner. This is mainly performed in sliding window and scanning approaches, where a very large number of input samples has to be classified in a very short time interval. In this case other demands are made on the nature of

a classifier. The main requirements are the compactness in representation and, moreover, the possibility to calculate a single result extremely fast. This is necessary, because in scanning approaches efficiency can only be reached, if classification results of individual input samples can be delivered rapidly. In this respect, object detectors, which are based on exhaustive search, have to be based on classifiers, which can be computed quickly. In the context of object detection on embedded systems, compactness in representation is also of major importance, as the issue of limited computational and memory resources states a strict limitation on the properties of the method used. This is also a reason, why learning based object detectors have only been rarely deployed on smart camera platforms yet.

In the following, we will discuss the task of object detection in the context of DSP-based embedded systems, where we focus on the latter method of using a classifier, the exhaustive way. After some introductory notes, we focus on the realization of a particular, representative and popular approach from the available set of methods proposed recently. First, some theoretical background of the approach is given in Section 4.2, followed by a description of the methods for the rapid calculation of features, the training and the detection step of the approach in Sections 4.3 and 4.4, respectively. A conceptual evaluation of the entire algorithm is given in 4.5. In Section 4.6, we investigate several aspects of algorithm realization in the context of our prototypical DSP-based embedded platform. Finally, the Chapter closes with some concluding and summarizing notes in Section 4.7.

4.1 Introduction and Motivation

There is a long list of features, which are desirable for a general object detection algorithm. First of all, it is important that the approach delivers results with high accuracy and a low number of false alarms. Furthermore, the effort to set up a detector for a given object class and scenario must be reasonable. As already stated before, a detector should be able to discriminate between the object class and everything else but this group. Such a *generic* detector would be applicable in any scenario without any additional effort. Indeed, describing the space of all possible negative examples given only a small group of objects of interest is impossible. Thus, it is also impossible to build a fully generic detector. Though, the natural way is to build a detector, which can discriminate the object class from all possible samples, which are likely to occur during application of the detector. To allow for large scale application of such an algorithm, the building of detectors should be possible in an adequate amount of time using a moderate amount of resources. Finally, real-time capabilities under predefined environmental conditions are inevitable, to make the approach a reasonable choice for inclusion into a larger framework.

Because of the prerequisite of the method to work under different environmental conditions, approaches which have some setup or *training* stage are quite suitable choices. This mainly refers to the list of approaches, summarized previously in Sections 3.2.2 and 3.2.3 of Chapter 3. Appearance based methods have been proven to be advantageous in this respect. First, encoding algorithms for capturing the appearance of an object

class are very efficient. This means that object specific visual features can be described very well, very fast and in a very compact form. Even more important, compared to approaches based on other features, appearance based object class representations can be built for a large number of different object categories, which also allows for applications in many different domains.

However, it is clear, that training a classifier is a complex task, subject to both, high computational effort and problematic selection of examples and counter-examples. Moreover, the complexity of building a good detector increases with the variability of the objects, and becomes harder and harder with the increase of intra-class variation. Thus it is necessary to choose algorithms, which can cope with a certain amount of noise, offer robustness and adequate performance, and are still able to perform at real-time. One of the most promising approaches in this respect is the Boosting based detection framework of Viola and Jones, dating back to 2001 [250]. The detector is based on the efficient calculation of features and the cascaded structure of the classifier, which is applied to classify subwindows of images exhaustively. The approach forms a nice framework for object detection in real-time and has been shown to perform well in a large variety of applications and scenarios. Therefore, it is treated as "the" state-of-the-art algorithm for object detection and discussed closer in the following.

Since the initial proposal of the Viola-Jones algorithm, a large amount of referred work was published, dealing with different aspects and applications of this method. The approach was successfully deployed in a big number of tasks, such as face, vehicle or pedestrian detection, just to mention a few areas with special relevance to surveillance. Moreover, a few main problems of the original approach were targeted and, at least partially, solved since. As an important example, a large variability in object appearance enforces detectors to be based on a large set of classifiers and complex features. In turn, this results in increased computational complexity and leads to a significant slowdown of the detectors. Recently, approaches have been proposed, especially targeting this problem, such as the *feature sharing* approach of Torralba *et al.*, which is based on combining stumps of boosted classifiers [242]. Such ideas can be used to speed up the detectors again and to keep them applicable under real-time constraints. One can summarize, that the Viola-Jones detector is steadily growing in its capabilities to detect various different object classes. Furthermore, its performance is continuously improved both, in terms of detection accuracy and in terms of execution speed.

The Viola-Jones detector can be denoted as state-of-the-art in object detection. Because we feel confident, that this algorithm will also play a vital role in the development of object detection systems in the foreseeable future, a closer investigation of the algorithm in the context of smart cameras is motivated. Especially the study of several aspects, such as detection performance in terms of accuracy and temporal behaviour, given restricted amounts of computational and memory resources is of major importance. In the following, we introduce our approach and investigate it in detail to gain knowledge about its capabilities and the problems in realizing it on DSP-based embedded systems.

4.2 Algorithm Principles

The Viola-Jones detector is based on two major parts. The first contribution is the use of a Boosting algorithm for combining weak classifiers. The organization of the classifiers in multiple closed cascades and the use of bootstrapping for generating a suitable set of training images is proposed. The second part deals with the efficient computation of weak features and classifiers, which can also be calculated rapidly during detection.

In the following the Boosting concept, together with the formation of cascades is introduced. Thereafter the basics for rapid calculation of features is described. Lastly, the training process for generating the final detector is outlined, together with aspects of scale-invariant detection and scanning.

4.2.1 The Boosting Concept

The nomenclature *Boosting* was first mentioned by Kearns [116] and was later introduced officially by Kearns and Valiant [117]. Basically, Boosting denotes a framework to improve general learning algorithms. Given a set of learning hypotheses, which perform only slightly better than random, the basic principle of Boosting is to combine several of these *weak* classifiers into one single *strong* classifier. While the first algorithm in closed form, proposed by Schapire, had polynomial running time [205], a more efficient version was developed by Freund [75], however, still suffering from some problems in practical use.

The first practical approaches were proposed as "***A**daptive **r**eweighting and **c**ombin**ing***" (Arcing) by Breiman [35], and the "***Ada**ptive **Boost**ing*" (AdaBoost) algorithm by Freund and Schapire [77]. Breiman also introduced a related method called "***B**ootstrap **agg**regat**ing***" (Bagging) [34], which is often used nowadays to improve the performance of complex classifiers, such as neural networks or decision trees.

A lot of other boosting approaches have been proposed since then. A few examples are the *Gentle AdaBoost* by Schapire and Freund [206], the *LogitBoost* by Friedman *et al.* [78], and the *BrownBoost*, presented by Freund [76]. Of special relevance in this thesis is the *RealBoost* algorithm, proposed by Schapire and Singer, which is a modification of the classical AdaBoost algorithm to use confidence rated predictions [208]. A introductory description of Boosting can be found in the work of Meir and Rätsch [149].

4.2.2 AdaBoost

The first practically usable approach was introduced by Freund and Schapire [77] and denoted AdaBoost. We mainly follow their notation in the following outline of the algorithm, which is also given in the thesis of Leistner [134].

To capture the main idea of AdaBoost, consider a set of training samples $S = \langle (x_1, y_1), \dots (x_m, y_m) \rangle$, where each x_i belongs to some domain or instance space $\mathcal{X}$, and each label y_i stems from a set of discrete labels $\mathcal{Y} = \{-1, +1\}$, denoting a positive and a negative set respectively. A distribution of weights w_i is maintained, assigning

Algorithm 1 AdaBoost

Given: M training samples $\langle (x_1, y_1), \ldots, (x_M, y_M) \rangle$ where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y} = \{-1, +1\}$

Initialize $D_1(i) = \frac{1}{M} \qquad \forall i \in [1, M]$

for $t = 1, \ldots, T$ **do**

- Normalize weights w_i so that D_t is a probability distribution

$$w_{t,i} = \frac{w_{t,i}}{\sum_{k=1}^{M} w_{t,k}} \qquad \forall i \in [1, M]$$

- Train weak learner using the distribution D_t
- Get weak hypothesis $h_t^{weak} : \mathcal{X} \rightarrow \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_i}[h_t^{weak}(x_i) \neq y_i]$$

- Break if $\epsilon_t = 0$ or $\epsilon_t > \frac{1}{2}$
- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
- Update:

$$D_{t+1}(i) \;=\; D_t(i) \times \begin{cases} e^{-\alpha_t} & \text{if } h_t^{weak}(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t^{weak}(x_i) \neq y_i \end{cases} \;=\; D_t(i)\, \exp(-\alpha_t\, y_i\, h_t^{weak}(x_i))$$

end for

The final hypothesis results as:

$$h^{strong}(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \cdot h_t^{weak}(x)\right)$$

each sample i a specific weighting factor. This distribution is altered in each round t of Boosting and denoted as $D_t(i)$, where, again, i is the training example.

As a first step, *weak* classifiers $h_t^{weak}(x)$ have to be found with an accuracy greater than 50%, whereas a weak classifier has to draw a decision on the examples x_i as

$$h_t^{weak}(x_i) :\rightarrow \{-1, +1\}. \tag{4.1}$$

The resulting classification error ϵ_t can be calculated according to equation 4.2 as a sum of the individual weights of all misclassified samples:

$$\epsilon_t = \Pr_{i \sim D_i}[h_t^{weak}(x_i) \neq y_i] = \sum_{i : h_t^{weak}(x_i) \neq y_i} D_t(i) \tag{4.2}$$

Note, that there is no need to use a special algorithm for the weak learner. Alternatively, a subset of the training data can be sampled according to D_t, which is then used for training the weak learner, without special weighting of individual samples. After learning a weak hypothesis and calculating its error ϵ_t, the classifier h_t^{weak} is convexly combined into the strong classifier h^{strong} using a parameter α_t, which is computed from the error ϵ_t. This inclusion into the final strong classifier h^{strong} is denoted in equation 4.3:

$$h^{strong}(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \cdot h_t^{weak}(x)\right) \tag{4.3}$$

In Algorithm 1, which is taken from [134], the AdaBoost algorithm is summarized. First, each training example (x_i, y_i) gets an initial weight w_i. In each round t, the weights have to be normalized to form a probability distribution D_t. Using this distribution, a weak classifier $h_t^{weak}(x)$ is computed and is evaluated with respect to its classification error ϵ_t. The parameter α_t is calculated reciprocally from the error, which means that the weighting of the weak classifier $h_t^{weak}(x)$ increases with a decreasing error $epsilon_t$. The algorithm is stopped, if the error is larger than 50%, or, alternatively, if the error is zero. This equivalently means that all training examples are classified correctly. Finally, the distribution D_{t+1} is updated by reweighting the training examples. The weights of correctly classified examples are decreased, and the weighting of misclassified samples is increased. These can be examples, which are close to the decision boundary and are, thus, hard to classify. By increasing their weights, special importance is focused on classifying them correctly in the next round.

Training and Generalization Error Freund and Schapire have shown, that the training error of h^{strong} decreases exponentially fast with the number of weak classifiers [77]. First, assume, that for all weak classifiers $h_{t=1\ldots T}^{weak}$, the error $\epsilon_t \leq \frac{1}{2} - \gamma$ for some $\gamma > 0$. Furthermore, note that $\gamma_t = 1 - \epsilon_t$ can be denoted as the superiority of a given weak classifier h_t^{weak} over deciding randomly (50%). Then, the training error of the strong classifier h^{strong} can be bounded to

$$\Pr_{(x,y)\sim S}[h^{strong}(x) \neq y] \leq \prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] = \prod_t \sqrt{1-4\gamma_t^2} \leq \exp\left(-1\sum_t \gamma_t^2\right). \tag{4.4}$$

Note that equation 4.4 states the fundamental proof, that AdaBoost is able to convert several weak learners into a strong classifier. Moreover, if γ is known in advance, an exponential decrease in the probability can be achieved by slightly modifying the algorithm. However, gathering a priori knowledge of γ is difficult in practice, hence this is of theoretical interest mainly.

Not only the training error, but also the generalization error is important to be investigated. With high probability, the generalization error of h^{strong} is bounded to the

training error, as indicated in equation 4.5 given in [77]:

$$\Pr_{(x,y)\sim D}[h^{strong}(x) \neq y] \leq \Pr_{(x,y)\sim S}[h^{strong}(x) \neq y] + O\left(\sqrt{\frac{T \cdot d}{m}}\right) \tag{4.5}$$

In this formula, m denotes the sample size, T denotes the number of weak classifiers (or boosting rounds, respectively), and d denotes the *Vapnik-Chervonenkis dimension* (VC-dimension). The VC-dimension was introduced by Vapnik and Chervonenkis as a standard measure of the "complexity" of a space of hypotheses [247].

Early experiments have shown empirically, that boosting most times does not overfit, although this bound suggests that. Moreover, in some cases the generalization error tends to be further decreased, although the training error has already reached zero, which is also in opposition to the upper bound defined in 4.5. A closer explanation of this phenomenon by using the *margin theory* is given by Schapire *et al.* [207]. The margin of an example (x, y) is a number in $[-1, +1]$, which is some kind of "distance" from the decision boundary calculated as

$$\text{margin}(x, y) = \frac{y \ \sum_{t=1}^{T} \alpha_t \ h_t^{weak}(x)}{\sum_{t=1}^{T} \alpha_t}. \tag{4.6}$$

The magnitude of the margin can equivalently be interpreted as confidence value of the prediction. As the margins on the training set increase, the upper bound on the generalization error is improved. This is described in equation 4.7:

$$\Pr_{(x,y)\sim D}[h^{strong}(x) \neq y] \leq \Pr_{(x,y)\ S}[\text{margin}(x, y) \leq \theta] + O\left(\sqrt{\frac{d}{m \cdot \theta^2}}\right) \tag{4.7}$$

Note, that the number of iterations T has no influence for an arbitrary threshold $\theta > 0$.

4.2.3 RealBoost

RealBoost is an improved boosting algorithm based on the work of Schapire and Singer [208]. In the following, we mainly follow their remarks.

The main modifications are, that the outputs of weak classifiers are real values, compared to discrete values in AdaBoost, and that α_t is treated independently from ϵ_t. Moreover, the idea of confidence-rated predictions is introduced, which makes the algorithm converge faster during training.

To describe the *RealBoost* algorithm in detail, remember, that the goal of the weak learning algorithm in AdaBoost is to learn a weak hypothesis h_t^{weak} over a given distribution D_t with a small error ϵ_t. As indicated above, an approach to minimize the upper bound on the training error is to minimize the so-called *loss-function* Z_t in each round. This basic idea is useful as a general criterion for the choice of a weak hypothesis h_t^{weak}, where the formulation is given in equation 4.8, assuming that α_t might be chosen arbitrary:

$$Z_t = \sum_i D_t(i) \ \exp(-\alpha_t \ y_i \ h_t^{weak}(x_i)) \tag{4.8}$$

α_t can be folded into h_t^{weak} and can, basically, scale a weak hypothesis by a constant factor in $\mathbb{R}$. Under this assumption, Schapire and Singer have shown, that Z_t has direct impact on the training error, and that the upper bound of the training error of a strong hypothesis h^{strong} can be given by,

$$\frac{1}{m}|\{i : h^{strong}(x_i) \neq y_i\}| \leq \prod_{t=1}^{T} Z_t. \tag{4.9}$$

Domain partitioning weak hypotheses and prediction smoothing To make the approach more powerful, the weak learner can be adapted to better suit the general mathematical concept of Boosting by folding α_t into the weak learning algorithm. Though we are dealing with a 2-class problem, *i.e.* $j = 1$ denoting the object class and $j = 2$ denoting the non-object class, each weak hypothesis can be associated with a partition of $\mathcal{X}$ into disjoint blocks X_j. The collectivity of X_j with $j \in \{1, 2\}$ covers all of the instance space $\mathcal{X}$, for which $h^{weak}(x) = h^{weak}(x')$ for $x, x' \in X_j$. The prediction of h depends only on which block X_j a given instance falls into.

To recapitulate the main modification of RealBoost over discrete AdaBoost, the initial prediction form $h_t^{weak} : \mathcal{X} \to \{-1, +1\}$ is changed to the prediction space $h_t^{weak} : \mathcal{X} \to \mathbb{R}$, where the sign of h_t^{weak} still indicates the class membership, and the magnitude $|h_t^{weak}(x)|$ now denotes the confidence of this prediction. Supposing $c_j = h^{weak}(x)$ for $x \in X_j$, the goal is now to find proper choices for c_j. Again, given a one-dimensional problem, in principle one threshold ϑ, splitting the two domains X_1 and X_2, has to be determined. Moreover, just two values c_1 and c_2, have to be determined. These values are later denoted as *alpha* (α) and *beta* (β).

Let $b \in \{-1, +1\}$ denote the labels of classes. For $j \in \{1, 2\}$, the weighted fraction of examples which fall in block j with label b can be written as

$$W_b^j = \sum_{i:x_j \in X_j \wedge y_i = b} D(i) = \Pr_{i\ D}[x_i \in X_j \wedge y_i = b] \tag{4.10}$$

The original loss function of the discrete AdaBoost algorithm, given in equation 4.8, can be rewritten as

$$Z = \sum_{j \in \{1,2\}} \sum_{i:x_i \in X_j} D(i) \exp(-y_i\ c_j) = \sum_{j \in \{1,2\}} \left(W_+^j e^{-c_j} + W_-^j e^{-c_j} \right). \tag{4.11}$$

Z reaches a minimum at y

$$c_j = \frac{1}{2} \ln \left(\frac{W_+^j}{W_-^j} \right), \tag{4.12}$$

which leads to a further simplification of 4.11:

$$Z = 2 \cdot \sum_{j \in \{1,2\}} \sqrt{W_+^j \cdot W_-^j} = 2 \cdot \left[\sqrt{W_+^1 \cdot W_-^1} + \sqrt{W_+^2 \cdot W_-^2} \right] \tag{4.13}$$

For numerical stability, Schapire and Singer [208] suggest to smooth the values c_j by including the error ϵ into the calculation, most commonly as $\epsilon = \frac{1}{M}$ with M being the number of training examples. Including *epsilon* may be necessary, because accidentally W_+^j and W_-^j can become very small or even zero. This results in very large or infinite c_j, which, in turn, causes numerical problems. This limiting of the predictions magnitudes is formulated in equation 4.14:

$$c_j = \frac{1}{2} \ln \left(\frac{W_+^j + \epsilon}{W_-^j + \epsilon} \right) \tag{4.14}$$

Since W_+^j and W_-^j are bounded between 0 and 1, $|c_j|$ can be bounded by the following equation 4.15:

$$\frac{1}{2} \ln \left(\frac{1+\epsilon}{\epsilon} \right) \approx \frac{1}{2} \ln \left(\frac{1}{\epsilon} \right) \tag{4.15}$$

The main advantage of RealBoost over the discrete AdaBoost algorithm is its faster convergence during training. This makes it more applicable in practice, because the training phase can be stopped after fewer iterations. The algorithm is summarized in Algorithm 2, and a similar description can be found in the work of Wu *et al.* [265] or in the master thesis of Kálal [111].

4.2.4 Cascaded Classifiers as Object Detectors

Without going too much into detail, we will formulate the principal ideas of the object detection concept here. This is the first main contribution of the work of Viola and Jones, the organization of multiple strong classifiers in a cascaded structure in order to perform fast classification of samples [250].

A strong classifier is a linear combination of T weak classifiers, which contain a threshold θ and a parity value p to form a classification boundary on a feature $f_t(x)$. A weak classifier using RealBoost can thus be formulated as

$$h_t^{weak}(x) = \begin{cases} \alpha & \text{if } p_t \cdot f_t(x) < p_t \cdot \theta_t \\ \beta & \text{otherwise} \end{cases} \qquad \alpha, \beta \in \mathbb{R}. \tag{4.16}$$

Likewise, the structure of a single strong classifier $h_k^{strong}(x)$ is defined as

$$h_k^{strong}(x) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t \cdot h_t^{weak}(x) - \Theta_k \right), \tag{4.17}$$

where the threshold Θ is used to steer the classification performance of the single stage.

A classifier cascade is composed of K strong classifiers with increasing classification power, which are linked in sequential stages. During classification, samples have to pass the cascade stagewise, while an intermediate classification result is available after each stage $k = 1 \ldots K$, or single strong classifier h_k^{strong} respectively. Samples, which obtain a positive classification result in stage k, are passed to the next stage $k+1$, while those,

Algorithm 2 RealBoost for Two-Class Problems

Given: M training samples $\langle (x_1, y_1), \ldots, (x_M, y_M) \rangle$ where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y} = \{-1, +1\}$, L weak learners

Initialize $D_1(i) = \frac{1}{M} \qquad \forall i \in [1, M]$

for $t = 1, \ldots, T$ **do**

- Normalize weights w_i so that D_t is a probability distribution

$$w_{t,i} = \frac{w_{t,i}}{\sum_{k=1}^{M} w_{t,k}} \qquad \forall i \in [1, M]$$

for $l = 1, \ldots, L$ **do**

- Train weak learner with feature f using the distribution D_t
- For $b \in \{-1, +1\}$ and $j \in \{1, 2\}$, calculate

$$W_b^j = \sum_{i: x_j \in X_j \wedge y_i = b} D(i) = \Pr_{i\ D}[x_i \in X_j \wedge y_i = b]$$

- Determine threshold θ with minimal Z

$$Z = 2 \cdot \sum_{j=1}^{2} \sqrt{W_+^j \cdot W_-^j}$$

- $\forall i \in [1, M]$, set the output of $h^{weak}(x_i)$ to

$$h^{weak}(x_i) = \begin{cases} \alpha = \frac{1}{2} \ln\left(\frac{W_+^1 + \epsilon}{W_-^1 + \epsilon}\right) & \text{if} \quad f(x_i) < \theta \\ \beta = \frac{1}{2} \ln\left(\frac{W_+^2 + \epsilon}{W_-^2 + \epsilon}\right) & else \end{cases}$$

with $\epsilon = \frac{1}{M}$ being a small positive number.

end for

- Select weak learner h_t^{weak} with smallest Z
- Update:

$$D_{t+1}(i) = D_t(i) e^{-y_i h_t^{weak}(x_i)}, \quad \forall i \in [1, M]$$

end for

The final hypothesis and the confidence value result as:

$$h^{strong}(x) = \text{sign}\left(\sum_{t=1}^{T} h_t^{weak}(x)\right) \qquad \text{Conf}(h^{strong}(x)) = \left|\sum_{t=1}^{T} h_t^{weak}(x)\right|$$

obtaining a negative result, are discarded as negatives immediately after the stage. For object detection this means, that the majority of samples, which correspond to the non-object or background class, can be discarded very fast. More complex samples like highly textured patches, which are hard to distinguish from the object class, survive the leading cascade levels and are classified by more complex classifier stages. The basic detector structure is also depicted in Figure 4.1.

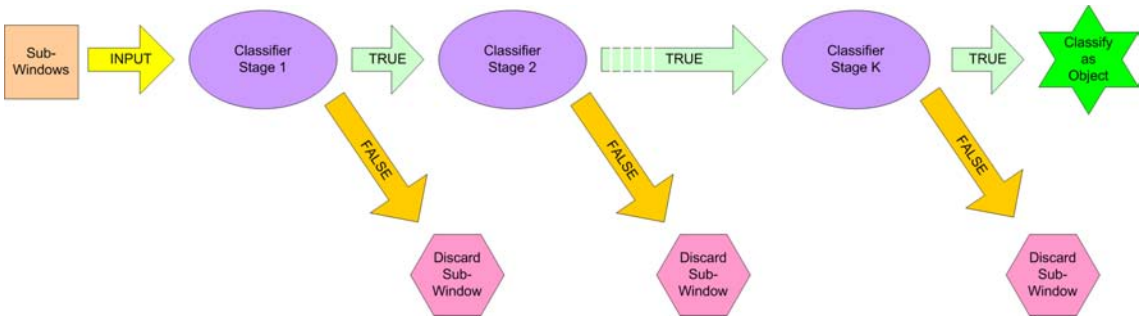


Figure 4.1: Cascaded structure of the object detector. After each stage, the negatively classified samples are discarded and the positively classified samples are forwarded to the next stage. Samples, reaching the end of the cascade are classified as objects.

In principle, the strength of a strong classifier related to a given training set is fixed by putting a threshold on the classification performance to achieve. If the threshold is reached, the strong classifier is finalized and treated as a single stage. The training set is refilled to contain only examples, which have passed the former stages and have been assigned a wrong label erroneously. This results in a more challenging training set for the next stage, and, consequently, in an increase in classifier complexity from stage to stage. The procedure of refilling a training set with negative examples on the basis of former classification results is also known as *bootstrapping* and was first mentioned by Sung and Poggio [227].

For object detection using the classifier cascade, it is of special importance that positive and negative examples are treated differently in terms of their classification accuracy. This means, that it is more important to classify positive examples as positives, as to classify negative examples as negatives. The reason for this is rather obvious. A false classification in the first case leads to an irreversible loss of the sample and to a bad detection performance consequently. In the latter case, the sample is simply passed to the next stage, where it will possibly be correctly discard. In other words, the cascade building procedure has to be conservative about treating examples, focusing on keeping positives even if unsure.

To formulate this, each stage k has a *minimum detection rate* d_k and a *maximum false positive rate* fp_k. Both rates are determined and set as training parameters. The overall detector performance for a K-stage classifier cascade is measured as *overall detection rate* D and the *overall false positive rate* FP, and calculated as products of the real rates in

the individual stages to

$$D = \prod_{k=1}^{K} d_k \qquad \text{and} \qquad FP = \prod_{k=1}^{K} fp_k. \tag{4.18}$$

To achieve a detector with a high detection rate, the minimum detection rate d of the stages has to set to a very high level, in order not to miss correct detections. In contrast, the false positive rate fp has to be set to a significantly smaller value, to allow FP to decrease much faster than D.

4.3 Rapid Feature Calculation

The second important contribution to allow for real-time performance of the detector is the proposal of an intermediate image representation. The basic idea of the so-called *Integral Image* originally stems from a work of Crow on texture mapping [55]. The mathematical foundation behind was described later in the work of Heckbert [92], and can also be found in the work of Simard *et al.* [216]. In the following, the basics for calculating the integral image are repeated first. Then we describe the set of features for fast calculation of features, and weak classifiers respectively.

4.3.1 Integral Image

The *Integral Image* was first introduced by Crow in the context of texture mapping and called *Summed Area Table* [55]. It is defined to be the sum of all pixels to the left and above the current position. The value of the integral image ii at location (x, y) is defined as

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{4.19}$$

with $i(x, y)$ being the original value of the input image at the same location. The integral image can be calculated in one pass over the entire image, using the preliminary settings

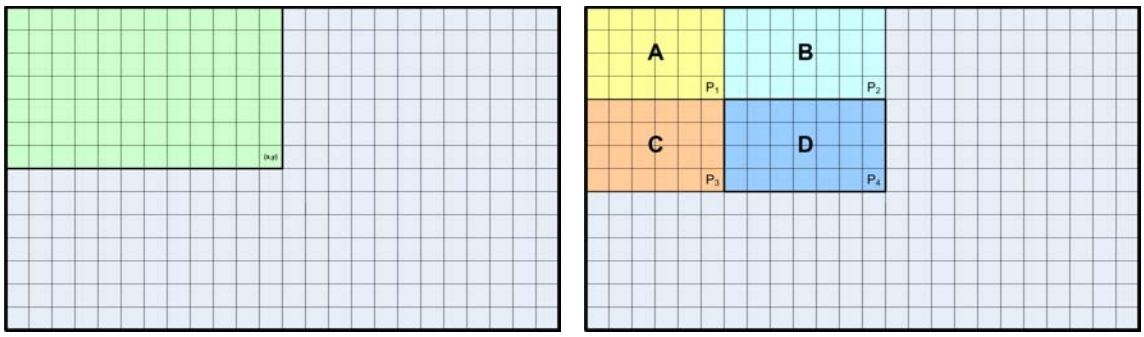
$$s(x, -1) = 0 \qquad \text{and} \qquad ii(-1, y) = 0 \tag{4.20}$$

and the following two recursions:

$$s(x, y) = s(x, y - 1) + i(x, y) \tag{4.21}$$

$$ii(x, y) = ii(x - 1, y) + s(x, y) \tag{4.22}$$

The basic idea is also depicted in Figure 4.2. The integral image is used to calculate any rectangular sum over image pixels in the original image by using four array references only. This allows for very fast calculation of very simple features in constant time with a few array references only. It also allows the scaling of features. Only the indices to the correct array entries have to be scaled, while the calculation procedure of the sums remains the same plus a correction factor for possible interpolation errors.



(a) (b)

Figure 4.2: (a) The integral image value at location (x, y) is defined to be the sum of all values above and left of (x, y). (b) The calculation of any rectangular sum in the original image simplifies to four array references in the integral image. Four array references are needed to calculate the sum of the pixels in rectangle D: At point P_1 the integral image has the value of rectangle A, at P_2, it is the sum of A and B. At location P_3, the sum is A plus C and at P_4, it is A + B + C + D. The sum of all pixels in rectangle D can then be calculated as the references at the points $P_4 + P_1 - (P_2 + P_3)$.

4.3.2 Features and Weak Classifiers

Using the integral image, simple features, which are made of several rectangles, can be calculated efficiently. The simplest features of this type are commonly known as *Haar* wavelets. For an introduction to wavelets, the reader is referred to the work of Stollnitz *et al.* [222, 223].

The set of features used for our experiments is depicted in figure 4.3. The first two features a) and b) can characterize edges, while features c)-f) describe thin and thick line like structures. Features g) and h) are special features to characterize corners and point-shaped visual structures. Features are composed of at least two rectangles. Each single rectangle contains not less than one pixel, which essentially means, that a feature can be calculated from at least two pixel values. However, such features are highly noisy, because the visual singularity to capture has be located directly beneath it. Thus, features with a larger spatial extend are preferable, as they exhibit higher description power and, therefore, higher performance.

Although we do not use more advanced features in our investigations, we want to mention several extensions and adaptations proposed in the literature to create more advanced features. Lienhart and Maydt proposed an extended set of Haar features, which are rotated by 45° to better describe diagonal features [139]. A modified version of the integral images is necessary for a rapid calculation of the new feature values. Porikli proposed the integral image for extracting histogram based information from images [185]. On this account, the use of *histograms of oriented gradients* has been proposed in various ways. Grabner *et al.* have shown this concept to work well for extracting features similar

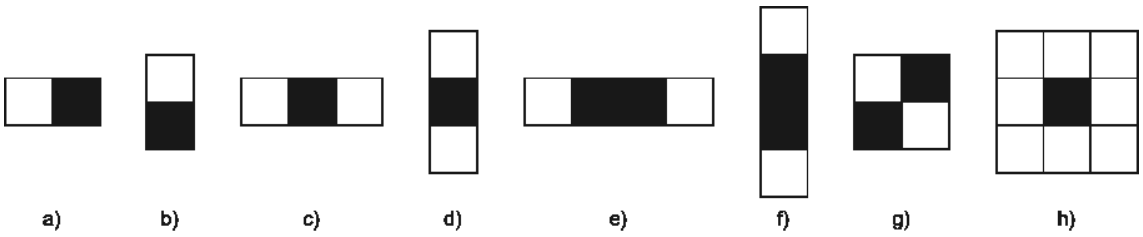


Figure 4.3: Simple Haar features. Each feature compromises two or more smaller rectangles. The white rectangles have a positive, the black rectangles a negative weight.

to the original SIFT approach [87]. Sun and Si have proposed the usage of the gradient histograms for detection of symmetrical structures in images [225].

Note, that the integral image is a fundamental tool to trade computational expenses towards memory resources. Since integral images are accumulating images values, large datatypes have to be used even for small-sized images. This places a clear contradiction to the principles of embedded systems, as memory is a way larger concern there than on usual computers. This is also a reason, why we use simple features and focus on more principal aspects of the algorithm, rather than investigate features of higher complexity.

4.4 Training and Detection

Here we shortly outline the training process. Furthermore, we describe a method to increase the accuracy of a detector, coevally decreasing its computational complexity. Finally, we shortly outline the fundamental application of a detector for scale-invariant detection and post-processing of the intermediate results.

4.4.1 Detector Training

The training of a cascade shaped detector proceeds as follows. First, the desired detection rate D and false positive rate FP for the training process is chosen. Then, an initial set of positive and negative examples is loaded, whereas the size of the dataset is in the range of several hundred examples. The set is divided into a training set and a validation set. All examples are scaled to a predefined, uniform rectangular size. For computational efficiency, from the huge amount of possible features, given the various types of features described above, a rather small set of features (usually 2-10%) is randomly chosen.

Each feature is trained on the training set, whereas the feature value is calculated on all training images, and for the resulting probability distribution, a threshold is sought which minimizes the loss-function Z. From all resulting weak classifiers, the one with the smallest value Z is chosen to be included into the strong classifier stage, and the training samples are reweighted for the next iteration. The training and selection process is repeated and weak classifiers are chosen according to the minimum-Z criterion, until

the stage meets the desired criteria in respect of D and FP [1]. Note, that these rates are evaluated on the validation set.

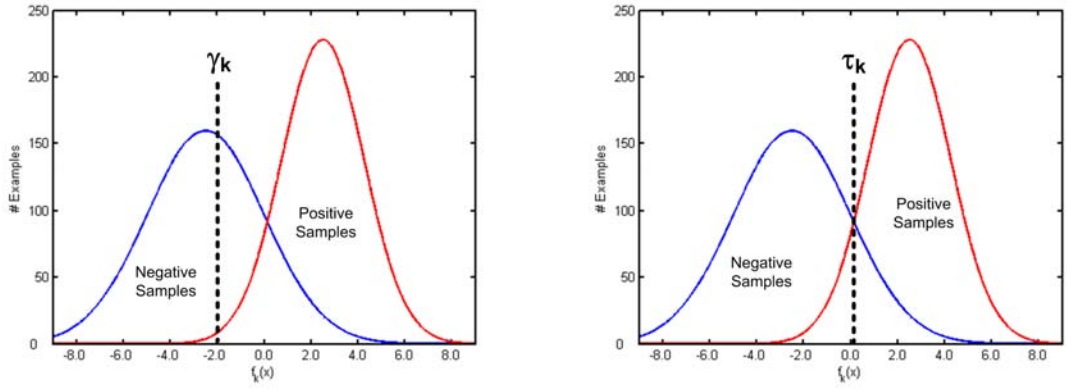
After finalizing a single stage, it is appended as a new cascade to the final classifier. The training set and the validation set are cleaned up, which means that all misclassified examples are removed from the sets. Both datasets are refilled to the original size, whereas the data to refill the sets, in contrast, has to be misclassified by the preliminary cascade classifier. As mentioned earlier, this *bootstrapping* process causes the training procedure to focus on harder examples in later stages.

The cascade building process is finally stopped, if a predefined overall performance criterion is met. This criterion is usually defined to be an *overall maximum false positive rate*, which can also be understood as a percentage value of misclassified samples, given the (not infinite, but huge) set of possible images in the universe.

4.4.2 Inter-Stage Feature Propagation

It is important to mention, that, for a good performance in terms of speed, each classifier stage has to be kept as short as possible. This is important, because false patches should be discarded as soon as possible, with only a minimum number of weak classifiers to evaluate.

Several different approaches have been proposed, aiming at this goal. Šochman and Matas proposed the use of *Inter-Stage Feature Propagation* (ISPF) [252]. In principle, the idea is to let stage k participate from the classification power of the weak classifiers of stage $k-1$.



(a) (b)

Figure 4.4: Inter-Stage Feature Propagation principle. (a) The threshold γ_k is selected to be the threshold of stage k. (b) The first weak classifier of stage $k+1$ is using the same distributions, but takes an other threshold τ_k.

[1] Note that each feature is usually limited to be used only once, as there is no additional gain in discriminative power, if it would be chosen multiple times

More precisely, in the original approach of Viola and Jones, each stage is trained without information of the previous stages. As a consequence, the only connection between the individual stages during training is created through the bootstrapping process. By definition, each stage has a stage threshold, which discards a significant amount of negative examples, coevally keeping a predefined amount of positives. However, as the primary focus is on meeting the criteria regarding the minimum rate of positive examples discarded, a lot of discriminatory power of this *strong* classifier is lost.

The idea of ISPF is to search for a threshold for the *strong* classifier (or stage classifier), which performs a classification with minimal loss function. In other words, given a finalized stage, the output of the stage for the new set of bootstrapped training samples is treated as *feature values*, and same loss function as for the *weak* classifiers is applied to build a first *weak* classifier. In fact, this means, that the previous stage k is treated as the first *weak* classifier of stage $k+1$, which can be interpreted as the propagation of a *prior* from stage k into stage $k+1$.

To formalize this, remember, that for the RealBoost the binary decision rule for the strong classifier of stage k is

$$h_k^{strong}(x) \;=\; \text{sign}(f_k(x) - \gamma_k) \tag{4.23}$$

with

$$f_k(x) \;=\; \sum_{t=1}^{T_k} h_t^{weak}(x), \tag{4.24}$$

being the sum over a all T_k weak classifiers of the stage k, and γ_k being the optimal threshold with respect to the predefined minimum detection rate. Applying the idea described above, in stage $k+1$, equation 4.24 is modified to

$$f_{k+1}(x) \;=\; h_0^{weak}(x) + \sum_{t=1}^{T_{k+1}} h_t^{weak}(x). \tag{4.25}$$

with

$$h_0^{weak}(x) \;=\; \text{sign}(h_k^{strong}(x) - \tau_k). \tag{4.26}$$

being the modified strong classifier of stage k with a threshold τ_k, which is optimal in the minimum-loss-function sense. ISFP was originally proposed for the discrete AdaBoost algorithm, but can be applied directly to RealBoost, as the loss function stays the same as already described in section 4.2.3, and also given in equation 4.27.

$$Z_0 = \sum_i D_0(i)\, \exp(-y_i\; h_0^{weak}(x_i)) \tag{4.27}$$

A graphical example of the ISPF idea is given in figure 4.4. Inter-Stage Feature Propagation is an efficient method to reduce the number of features. This results from the exhaustive use of hard-earned discriminatory power from stage to stage. Even more important, while the algorithm is forced towards harder examples earlier in the training stage,

the approach also leads to a significant decrease in computational complexity. This means that not only the detector performance in terms of accuracy, but also the performance in terms of speed is increased, which are both desirable effects.

4.4.3 WaldBoost

WaldBoost was originally proposed by Šochman and Matas [253] and is based on the *Sequential Probability Ratio Test* of Wald [255]. A in-depth investigation of WaldBoost for face detection is given in the thesis of Kálal [111]. Although we do not use WaldBoost in this thesis, we strongly emphasize a closer investigation of this algorithm in the context of embedded systems because of its striking benefits, *i.e.* a reduced number of features and, thus, a faster way of decision making. Hereafter, we shortly outline the principal idea.

Let x be characterized by an unobservable hidden state $y \in \{-1, +1\}$, which has to be determined by successive measurements $x_1, x_2, \ldots$. Further assume, that for $c \in \{-1, +1\}$ the joint conditional density $p(x_1, \ldots, x_T|y = c)$ to be known for all T. Wald defined the likelihood ratio R_T as

$$R_T = \frac{p(x_1, x_2, \ldots, x_T|y = -1)}{p(x_1, x_2, \ldots, x_T|y = +1)} \tag{4.28}$$

and the SPRT as a strategy S^*

$$S^* = \begin{cases} +1, & R_T \leq B \\ -1, & R_T \geq A \\ \#, & B < R_T < A \end{cases} \tag{4.29}$$

with # denoting "take one more measurement". Because optimal values for A and B are hard to compute in practice, Wald suggests to set the thresholds to

$$A' = \frac{1-\beta}{\alpha} \qquad B' == \frac{\beta}{1-\alpha} \tag{4.30}$$

For building a detector using Boosting, α and β denote the desired false negative rate and the desired false positive rate, respectively. Furthermore, Šochman and Matas propose to approximate the likelihood ratio by projecting the T-dimensional space into a one-dimensional space using the current classifier $h_T^{strong}(x)$, which is based on T measurements, *i.e.* T weak learners.

$$\hat{R}_T = \frac{p(h_T^{strong}(x)|y = -1)}{p(h_T^{strong}(x)|y = +1)} \tag{4.31}$$

The classifier is the fastest possible, because it requires only the minimum number of measurements that is necessary to make a decision with a given classification error.

4.4.4 Scale-Invariant Scanning and Postprocessing

As already mentioned earlier, the detection process is done by exhaustively scanning and classifying overlapping windows in a larger image. Needless to say, that the image is first transformed into the integral image representation to allow for fast feature calculation.

The original detector, which was trained on a given patch size, is directly used to find objects of the predefined scale. For scanning at different scales, the individual weak classifiers are scaled by the scale factor, which equals the resizing of rectangles respectively. This results in computational inaccuracies, as a raw scaling of the rectangles might lead to calculations on "fractions" of pixel values. This problem is mostly solved by reformating the rectangle dimensions to integer numbers. A more critical problem in this matter is the possible elimination of weak classifiers, if the scale factor is smaller than one and features have a very small spatial extent. As a consequence, for training a detector, it is meaningful to choose a sample size, which is close to the minimum size of objects to be detected. In this way, the detector only has to scale up, which avoids this type of errors.

Due to the small translation invariance of the detector and the overlapping scan of windows, multiple detections of single objects are unavoidable. Many different approaches have been proposed to solve this problem, however with varying rate of success, but mostly coming at a considerable amount of computational costs. One method for combining multiple detections, given confidence rated detections, is the *Mean-Shift* algorithm, which is a nonparametric estimator of the density gradient and was originally proposed by Comaniciu and Meer [52]. The Mean-Shift algorithm is used to cluster correct results, where all acquired detection responses are first represented as a two-dimensional probability distribution. Each detection is registered and weighted using a 2-dimensional Gaussian kernel, whose size is equal to the detection size. Given a search window with a fixed size, the Mean-Shift algorithm iteratively climbs the gradients of this distribution to find dominant modes. Dominant modes are found on algorithm convergence, which is also known as *mode seeking*. A modified version, the *Continuously Adaptive Mean-Shift* (CAMShift) algorithm, was proposed by Bradski [29]. The main difference is the adaptively adjusted window size in order to find proper modes. In contrast to the approaches mentioned before, we employ a rather simple idea, which is called *Maximum Suppression*. In this approach, all detections with significant overlap are simply replaced by the detection with the highest confidence rate. This approach is relatively simple, yet achieves satisfactory performance.

4.5 Concept Evaluation on a Desktop Computer

Before we start with an investigation of the detection algorithm properties on our embedded platform, we compare the original Viola-Jones detection approach with the enhanced method using ISFP on a usual PC. To anticipate the justification for using the ISFP method, any enhancement of the detection approach on the algorithmic level results in a

more remarkable improvement than any other type of hardware-related adaptation. To say in the context of the explanations given in chapter 2, any type of advance in terms of detector size or accuracy finally leads to more significant benefits, than exhaustively exploiting issues of parallelism on any platform. Needless to say, that the detector size is equal to the number of weak classifiers in this case.

Before going into detail, we define some evaluation criteria, which allow us to investigate the detector performance and to compare both approaches objectively. This is mainly a repetition of the evaluation criteria commonly used in the literature. For evaluating both approaches, we have chosen three different datasets, which come from the area of traffic surveillance. The first dataset is the publicly available *UIUC Database* [238], which contains side views of cars with different scales. Furthermore, we built two additional datasets to further gain insight into the behaviour of the detection algorithm. The first dataset, the "A10" dataset, contains images extracted from a video stream, recorded on a gantry over a highway in Austria. The major task is to detect vehicles, which are passing and are moving away from the camera location. The major difficulty of this dataset is the drastic change in aspect ratio and the relatively low resolution of images and objects, respectively. The last testset, the "License Plate" dataset, was originally acquired for license plate recognition tests and contains a set of vehicle images taken from a pedestrian bridge over an urban street. Each image contains a single license plate, which should be located by the detector.

Note, that we are not interested in building a detector which is close to perfection in terms of detection accuracy, because this would result in large classifiers with several hundred features. We trade detection accuracy against computational complexity and aim at reduced numbers of weak classifiers and cascades to keep the detectors fast. Needless to say, that this comes at the cost of additional false positives. Our main intention is to simply train several detectors from scratch, and examine the principal behaviour of the algorithms. This is also important, because for product development or system setup, it is of major relevance, which performance rates are to be expected, if a detector is trained without special tuning to a given scenario. Moreover, we will show, that only a small modification of the algorithm, *i.e.* using ISFP, can have a large impact on both, detector size and accuracy.

4.5.1 Evaluation Criteria

In the following experiments, all detectors are evaluated on full images, rather than on single patches. Agarwal and Roth defined some criteria, which allow us to compare different types of detectors and different approaches in a fair way [2]. Also the performance of multi-scale detectors can be measured, where a correct detection of an object is defined as

$$\frac{|i-i^*|^2}{\alpha^2_{height}} + \frac{|w-w^*|^2}{\alpha^2_{scale}} + \frac{|j-j^*|^2}{\alpha^2_{width}} \leq 1, \tag{4.32}$$

$$\alpha_{height} = 0.25 \cdot h^* \tag{4.33}$$
$$\alpha_{width} = 0.25 \cdot w^* \tag{4.34}$$
$$\alpha_{scale} = 0.25 \cdot w^* \tag{4.35}$$
$$\tag{4.36}$$

In equation 4.32, (i^*, j^*, w^*) denote the center coordinates and the width of the true location of an object, while (i, j, w) denote the center coordinates and width of the detection delivered by the detector. h^* denotes the true height of the object. All parameters α_{height}, α_{width}, α_{scale} are chosen, that for a correct detection the ellipsoid, defined in equation 4.32, is allowed to have 25% of the size of the true object at max.

In cascaded detectors with K stages, the threshold Θ_K of the last stage is used to indicate, whether an evaluation is considered as a detection, or discarded as non-object. During the evaluation on a test set, this threshold can be varied to trade the number of false positives against the number of correct detections. Usually, one is interested in the behaviour of the detector in between the two possible extremas, where both detection and false positive rate are either zero or one. Due to the cascaded structure of the detectors, the extremas can only be reached under special circumstances. On the one hand, setting the threshold $\Theta_K = +\infty$ causes a false-positive and a detection rate of zero. On the other hand, a false-positive and a detection rate of 1 can only be reached, if the thresholds of all stages $1 \dots K$ are set to $-\infty$, which is equivalent with treating all evaluations as objects. However, during the following evaluations, only the threshold of the last stage Θ_K is varied to investigate the behaviour of the detectors, which causes the sacrifice of some curve smoothness.

To allow for comparison of individual detectors, in the literature most often the *receiver operator characteristic* (ROC) curve is used. It allows for illustration of the trade-off between true-positive (TP) and false-positive (FP) detections. The detection rate and false-positive rate used in the ROC curve are calculated according to equations 4.37 and 4.38:

$$\text{detection rate} = \frac{\#\text{correct true positives } (TP)}{\#\text{all positives in the test set } (nP)} \tag{4.37}$$

$$\text{false-positives rate} = \frac{\#\text{false positives } (FP)}{\#\text{all negatives in the test set } (nN)} \tag{4.38}$$

The false-positive rate and the detection rate are plotted on the x-axis and y-axis of the diagram, respectively. Comparing different detectors is still difficult due to the problematic definition of the total number of negatives (nN). Treating the detector as a classifier, (nN) can be defined to be the number of all negative evaluations, which is a very large number compared to (nP). Furthermore, for a detector, the number of evaluations is more dependent on the implementation, than on the input or output data.

For evaluating a detector, a more convenient visualization method, called the *Recall-Precision Curve* (RPC), can be used to avoid the need for setting (nN). The RPC allows

for a more practical graphical description of the detector behaviour, as it only plots the real number of false detections versus the number of correct detections. The individual rates are defined as

$$\text{Recall} = \frac{TP}{nP} \qquad \text{Precision} = \frac{TP}{TP+FP} \qquad 1-\text{Precision} = \frac{FP}{TP+FP} \tag{4.39}$$

More precisely, Recall is identical to the true-positive rate in the ROC curve, and 1−Precision defines the percentage of false detections from all detections delivered by the detector. The trade-off between Recall and Precision can also be expressed using the so-called *F-measure*, as defined in equation 4.40. Summarizingly, the RPC is a more practical description of the detector behaviour during real application.

$$F\text{-measure} = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}} \tag{4.40}$$

4.5.2 UIUC Vehicle Database

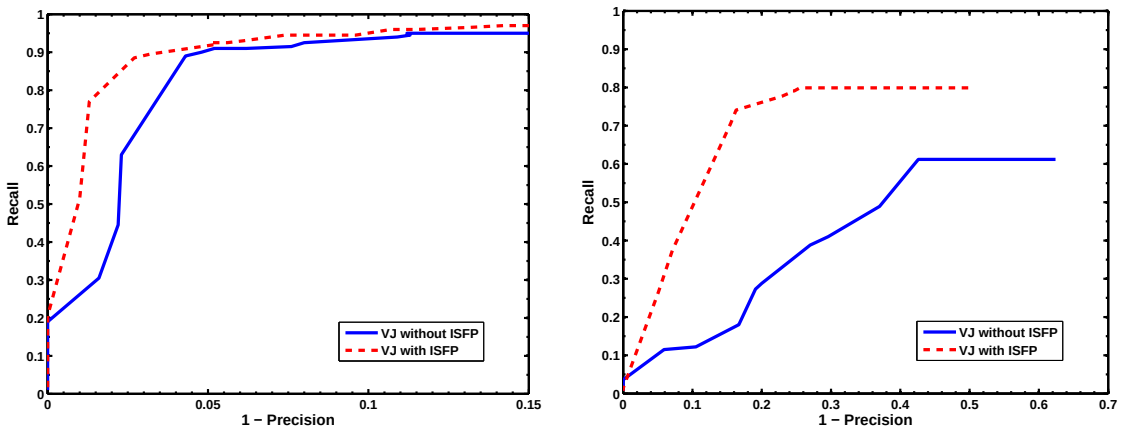
The algorithms were first evaluated on the free *UIUC Car Database*, which was published by Agarwal and Roth [2]. The database contains 550 positive images with a size of 100×40 pixel values, which show cars from the side. Some examples are depicted in figure 4.5. We used the parameter settings listed in Table 4.1 as configuration for training the detector. As can be seen, we split the set into 350 training samples and 200 validation samples. The negative images, as well as the images used for bootstrapping, were randomly chosen from a set of pictures which were downloaded from the Internet. Only a small percentage, about 10%, of all possible features were randomly chosen to form the feature pool. The pool of all possible features contains 633,334 features. The detectors were trained, until the overall maximum false positive rate of 0.0005% was reached. Note, that this implies, that the detectors may have different numbers of stages.

Figure 4.5: Some samples of the UIUC car database.

For evaluating the detector performance, the UIUC database includes a set of 170 images, which contain a total of 200 cars of the same scale (100×40 pixels). Furthermore, a test set is included, which contains 108 images and a total of 139 cars in scales from 0.8 to 2.0 of the original scale. For our detectors, for exhaustive scanning, we chose the step

UIUC Training Parameters	
object width	50 pixels
object height	20 pixels
positive Training examples	350
negative Training examples	300
positive Validation examples	250
negative Validation examples	200
bootstrap images	500
detection rate per stage	0.99
false-positive rate per stage	0.7
overall maximum false positive rate	0.0005
features used	63.334

Table 4.1: Settings used for the detectors trained on the UIUC dataset.



(a) (b)

Figure 4.6: Recall-Precision curves for both classifiers trained on the UIUC dataset. a) Detector performance for the testset with cars of the same scale. b) Detector performance for the testset with varying object scale.

size for the detection window to $\Delta_{xy} = 2$. This means, that we only scan about 1/4 of all possible subwindows. However, due to the small translation invariance of the detector, this does not have a bad effect on detection performance, but additionally reduces the number of false detections.

A comparison of the original Viola-Jones approach and the detector with ISFP is given in Table 4.2. The detectors contain 13 and 11 stages, with a total of 101 and 59 features, respectively. The first weak classifier for each stage of the ISFP detector is not counted here, because the evaluation costs are negligible. Theoretically, the number of features in each stage is expected to increase monotonically. However, this is not the case with our detectors, which happens, because the influence of "noisy" samples is too strong. The

Stage	# features		D		FP	
	wo. ISFP	w. ISFP	wo. ISFP	w. ISFP	wo. ISFP	w. ISFP
1	4	4	0.995	0.99	0.51	0.445
2	5	3	0.995	1.0	0.395	0.585
3	5	2	0.995	1.0	0.395	0.17
4	5	3	0.995	0.995	0.375	0.38
5	6	6	0.995	1.0	0.575	0.575
6	7	6	0.995	1.0	0.505	0.535
7	3	6	0.995	0.995	0.645	0.595
8	10	11	0.995	0.995	0.68	0.625
9	8	2	0.995	0.995	0.695	0.66
10	9	9	0.995	0.995	0.535	0.585
11	14	7	0.995	0.995	0.615	0.39
12	14	—	0.995	—	0.645	—
13	11	—	0.995	—	0.58	—
Total	**101**	**59**	**0.9369**	**0.9607**	**0.0003**	**0.0003**

Table 4.2: Comparison of the cascaded detector with inter-stage feature propagation (w. ISFP) and the original Viola-Jones (wo. ISFP) algorithm on the UIUC test dataset.

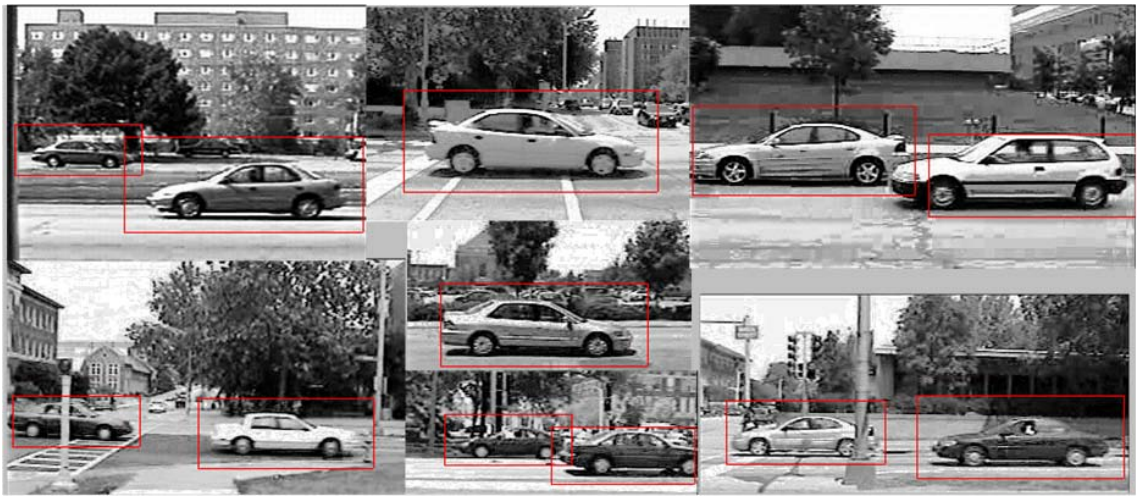


Figure 4.7: Detection results for the ISFP detector on the UIUC testset with varying object scale.

number of features needed in each stage increases with the stage number, as the decision process gets harder. However, the number of features used in the ISFP approach increases slower, which leads to a reduced total number of features needed.

The RPC on both testsets are depicted in Figure 4.6. As can be seen, both detectors perform quite well on the testset with a fixed scale. Because there is no prior scale information available for the dataset with varying object scale, the overall false positive rate of the detector has a higher influence on the final detection result. We scanned the test images exhaustively using 9 different scales in the range of $[0.9, 2.0]$. As can be seen clearly,

	Single Scale	**Multiple Scales**
Agarwal *et al.* [1]	76.5 %	39.6 %
Fergus *et al.* [71]	88.5 %	–
Leibe *et al.* [131]	97.5 %	–
Fritz *et al.* [79]	–	87.8 %
Mutch and Lowe [165]	99.94 %	90.6 %
VJ wo. ISFP	91.0 %	61.2 %
VJ w. ISFP	92.5 %	74.1 %

Table 4.3: Comparison of the cascaded detector with inter-stage feature propagation (w. ISFP) and the original Viola-Jones (wo. ISFP) algorithm to other results found in the literature for the UIUC databases.

the approach with ISFP achieves a way better performance on this testset, although it needs less features. The highest F-measures of the original approach are 92.861% and 59.239%, while the highest F-measures of the ISFP approach are 93.685% and 78.608%, respectively. A comparison to other approaches proposed in the literature is given in Tables 4.3. As can be seen, the performance of our detectors is comparable to others, but remember that we limited the number of weak classifiers for computational efficiency. Some sample detection results of the approach with ISFP are depicted in Figure 4.7.

As can be seen clearly, the approach using ISFP achieves a higher accuracy and better detection rate than the original approach. Coevally, it needs about 40% less features, which improves detector performance in terms of speed.

4.5.3 Vehicle Detection on Highways

The second testset for the Viola-Jones detector is a set of frames taken from a surveillance camera on an Austrian highway A10. The images have a resolution of 352×288 pixels. For simplicity, detection is only performed on the back views of vehicles. A set of training samples for the detectors is depicted in Figure 4.8. The training size was chosen to be very small, because object size decreases fast with increasing object distance. The number of 630 positive training samples was split into 400 examples for training and 230 for validation. The feature pool contains 50% of all possible features, which is a number of 50,665 features. A summary of the parameter settings for training the detectors with and without using ISFP is given in Table 4.4.

Because the scenario is recorded by a mounted camera at a fixed location, we have chosen only 30 images for bootstrapping, which also contain some pictures of the given scenario at different times of day and under different weather conditions, but always without vehicles. We strongly point out, that this is only a weak help for the training process to build a suitable detector for this scenario. Consider just one false positive, maybe somewhere nearby the supervised road section. This single false positive has a major impact on any benchmarking test on any database, because its detection is simply

multiplied by the number of images tested. Thus, it is simply necessary to tune the detector for a given setup to avoid such types of rather primitive, but impacting errors. For a closer discussion of these issues and the introduction of a method to autonomously learn a scene optimized detector, the interested reader is referred to the thesis of Roth [196].

Detailed information about both detectors is given in Table 4.5. The detectors consist of 15 and 13 stages, respectively. The testset contains 1057 images with a total number of 1656 cars in various scales. Each image contains between 1 and 5 vehicles. In this case, we did not use prior scale information to fit the detection and scanning process to the underlying scene geometry. For both detectors, the RPCs are depicted in Figure 4.9, which clearly reflect the unoptimized way of using the detector in an exhaustive way. The highest F-measures for the original approach is 53.10%, and 56.47% for the ISFP approach, respectively. In Figure 4.10, some detection results on the testset are depicted.

On the A10 dataset, the approach using ISFP is, again, performing better than the original approach, although both approaches deliver only moderate performance given the exhaustive scanning method. The main problem responsible for this is the missing ability of the detector to deliver detections with an accurate scale factor. Some erroneous detection results are depicted in Figure 4.11. Due to the restrictive criteria of Agarwal and Roth, only detections with an accurate scale factor are registered as correct detections. Many detections are counted as false detections in our case, because the size estimation of the detected object does not fit the real object dimensions well. However, from an algorithmic point of view, again, the major benefit of the ISFP approach is the reduced number of features needed, which, leads to increased detector performance in terms of speed. The approach with ISFP needs only half of the features, the original approach is using.

4.5.4 License Plate Detection

The third testset was originally acquired for license plate recognition tests. A video stream with 352×288 pixel resolution was recorded from a pedestrian bridge over an urban street.

Figure 4.8: Some samples of the A10 car database.

Figure 4.10: Detection results for the ISFP detector on the A10 highway testset. Note, that the detector was configured to only detect vehicles on the right half of the images.

Figure 4.11: Erroneous results for the ISFP detector on the A10 highway testset. The detector is not able to detect the correct scale of the objects, finally delivering a lot of inaccurate detections.

Highway A10 Training Parameters	
object width	20 pixels
object height	20 pixels
positive Training examples	400
negative Training examples	280
positive Validation examples	230
negative Validation examples	250
bootstrap images	30
overall maximum false positive rate	0.0001
features used	50665

Table 4.4: Settings used for the detectors trained for vehicle detection on the A10 dataset.

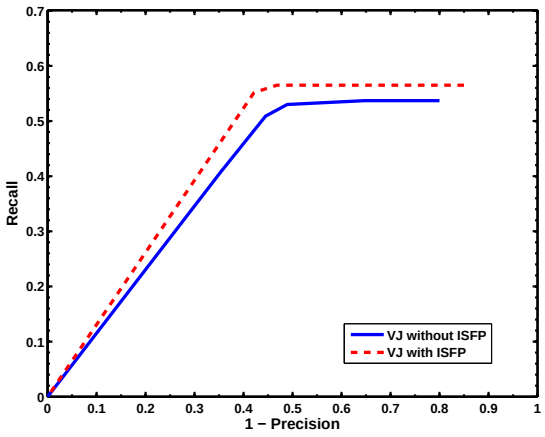


Figure 4.9: Recall-Precision curves for both vehicle detectors trained on the A10 dataset.

Stage	# features		D		FP	
	wo. ISFP	w. ISFP	wo. ISFP	w. ISFP	wo. ISFP	w. ISFP
1	4	3	0.995	0.991	0.404	0.332
2	3	3	0.995	1.0	0.564	0.56
3	9	2	0.991	1.0	0.592	0.576
4	5	3	0.991	0.991	0.592	0.552
5	6	6	0.991	0.991	0.588	0.208
6	7	5	0.991	0.991	0.588	0.584
7	8	6	0.991	1.0	0.46	0.444
8	9	2	0.991	0.991	0.432	0.528
9	6	6	0.991	0.991	0.432	0.516
10	5	6	0.991	0.991	0.58	0.584
11	5	2	0.991	0.991	0.584	0.384
12	4	4	0.99	0.995	0.564	0.576
13	13	4	0.99	0.991	0.568	0.58
14	8	—	0.99	—	0.58	—
15	9	—	0.995	—	0.532	—
Total	**101**	**52**	**0.881**	**0.917**	**0.000079**	**0.000065**

Table 4.5: Comparison of the cascaded detector with inter-stage feature propagation (w. ISFP) and the original Viola-Jones (wo. ISFP) algorithm on the A10 test dataset.

Vehicles were passing at residential speeds, so the amount of motion blur is negligible. Again, two detectors were trained, given the parameters summarized in Table 4.6. A training set of Austrian license plates was collected with a handheld photo camera, taking pictures of cars on a public parking lot. Some sample images are depicted in Figure 4.12. The total number of 266 positive samples was divided into 130 samples for training and 136 sample images for validation. Note, that we, again, used only 30 bootstrap examples for training the detector in the same way and for the same reason as described above.

Figure 4.12: Some sample images of Austrian license plates.

License Plate Training Parameters	
object width	75 pixels
object height	25 pixels
positive Training examples	150
negative Training examples	200
positive Validation examples	130
negative Validation examples	136
bootstrap images	30
detection rate per stage	0.992
false-positive rate per stage	0.6
overall maximum false positive rate	0.00005
features used	111326

Table 4.6: Settings used for the detectors trained for license plate detection.

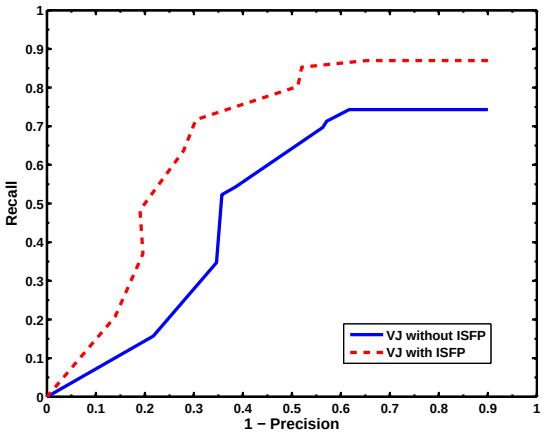


Figure 4.13: Recall-Precision curves for both license plate detectors trained on the LP dataset.

Detailed information about both detectors, which consist of 15 stages each, is given in Table 4.7. The testset contains 300 images with one car and one license plate in each image. The scale of the plates is between 1.0 and 1.2 of the original training size. The RPCs for both detectors are depicted in Figure 4.13. The highest F-measures of the original approach is 57.683 %, and 70.634 % for the ISFP approach, respectively. In Figure 4.14, some detection results on the testset are depicted.

As can be seen from the RPCs in Figure 4.13, the original Viola-Jones approach performs worse than the approach with ISFP. The number of features used by the original approach is only a little larger in this case, however, the approach using ISFP clearly outperforms the original approach in terms of detection accuracy. Both detectors perform rather poor, which is a result of the fact, that not all false positives on the background were removed completely, This happens, although we have tuned the detector by selecting special bootstrapping images. To increase detector performance, it would be necessary to exclude all false positives, which might occur in the scenario. However, as stated earlier, our main goal was to show the superiority of the ISFP approach over the original approach, thus we can leave this issue open for further development work.

Stage	# features		D		FP	
	wo. ISFP	w. ISFP	wo. ISFP	w. ISFP	wo. ISFP	w. ISFP
1	3	3	1.0	0.993	0.295	0.4
2	3	1	1.0	1.0	0.345	0.41
3	3	2	0.993	0.993	0.51	0.465
4	3	2	0.993	1.0	0.34	0.455
5	3	2	0.993	1.0	0.595	0.325
6	5	6	0.992	0.993	0.575	0.53
7	2	5	0.992	0.992	0.565	0.35
8	3	4	0.992	0.992	0.455	0.49
9	3	4	0.992	0.992	0.385	0.135
10	3	2	0.992	0.992	0.32	0.355
11	3	—	0.992	—	0.32	—
11	3	—	0.992	—	0.51	—
Total	**37**	**31**	**0.926**	**0.948**	**0.000049**	**0.000031**

Table 4.7: Comparison of the cascaded detector with inter-stage feature propagation (w. ISFP) and the original Viola-Jones (wo. ISFP) algorithm on the license plate test dataset.

Figure 4.14: Detection results for the ISFP detector on the license plate testset. Note, that the detector also detects license plates of foreign countries (as seen in the upper right image).

4.5.5 Summarizing Notes

To summarize the conceptual evaluation of both approaches, the ISFP approach achieves better performance in terms of accuracy on all testsets used. Moreover, the number of weak classifiers used is significantly lower, which leads to improved performance in terms of detection speed and smaller detectors.

On this account, we conclude, that the extension of the original Viola-Jones approach using Inter-Stage Feature Propagation is a more reasonable choice for building a detector than using the original approach. As will become clear in the following, a compact detector is necessary to allow for satisfying performance on an embedded system.

4.6 Experimental Evaluation on the DSP

In the following, we investigate several aspects and procedures of the Viola Jones algorithm on the prototypical embedded platform. We mainly focus on the real-time aspect of a system, rather than on the question of energy and exact memory consumption. The main focus of this section is to investigate and experimentally evaluate different aspects of algorithm development, which were described previously in chapter 2.

4.6.1 Prerequisites

As a first test database, we again choose the UIUC database. We reuse the detectors trained previously on a standard computer, which are stored in XML files. We use a lightweight XML parser on the embedded platform to load the detector structure during boot time. The binary program is compiled using the Code Composer Studio 3.2 from TI and uploaded using a JTAG emulator device. It is rather obvious, that training of detectors can not be done easily on an embedded platform, due to the lack of memory and the problematic need of exhaustive data transfer between a standard computer and the platform. Hence, training is always done on a standard PC, and the final classifier is simply uploaded as some type of "configuration" file. For uploading the images during run-time, the onboard Ethernet interface is used. Because we are transferring image data directly to the memory of the platform, the data is not compressed or modified in any way. Thus, we can directly compare the impact of fixed point calculations or other hardware related issues, as well as detection accuracy and throughput of our detector, without care of additional boundary effects.

The following evaluations are based on an 352×288 pixel input image. There are three good reasons for choosing this image size. Firstly, the integral image can be calculated in one pass and fits into the internal DSP memory entirely, which makes memory swapping between internal and external memory unnecessary. This allows us to focus on the algorithm directly, rather than on problems of memory transfer timing. Secondly, this image size enables us to still detect rather small objects, as shown in the previous section 4.5.3, where we have used a rather small object size for our highway vehicle detector. Moreover, weak classifiers are mostly based on sums of image areas rather than on single pixel values, thus a reduction in image resolution is not critical in this respect. Lastly, choosing a fixed image size allows us to define a performance limit for our individual functional parts, which makes it possible to define overall *real-time* operation criteria in the context of a detector.

Given a detector for the UIUC dataset with a fixed scale of 100×40 pixels and a step size of two pixels in horizontal and vertical direction, respectively, a total number of 15,875 subwindows has to be classified. Given this single scale scan, the average time of classifying a single subwindow must not exceed $2.5\mu s$, to allow for operation of the detector at 25 frames per second. This really states a hard limit, because it only allows for object detection at this single scale. Scanning for object instances at multiple scales clearly

Scale Factor	hor.	vert.	sum
0.875	133	127	16,891
1.0	127	125	15,875
1.125	121	122	14,762
1.25	114	120	13,680
1.375	108	117	12,636
1.5	102	115	11,730
1.625	96	112	10,752
1.75	89	110	9,790
1.875	83	107	8,881
2.0	77	105	8,085
Total:			**123,082**

Table 4.8: UIUC scenario exhaustive scanning. Listing of total number of subwindows to be scanned, given an image size of 352×288 pixels and a base size of 100×40 pixels for the detector. Note, that the calculations are based on a step size of two pixels in horizontal and vertical direction, respectively.

Scale Factor	hor.	vert.	sum
0.75	169	137	23,153
1.00	167	135	22,545
1.25	164	132	21,648
1.50	162	130	21,060
1.75	159	127	20,193
2.00	157	125	19,625
2.25	154	122	18,788
2.50	152	120	18,240
2.75	149	117	17,433
3.00	147	115	16,905
3.25	144	112	16,128
3.50	142	110	15,620
3.75	139	107	14,873
Total:			**246,211**

Table 4.9: A10 highway scenario exhaustive scanning. Listing of the total number of subwindows to be classified, given an image size of 352×288 pixels and a base size of 20×20 pixels for the detector. Again, we assume a step size of two pixels in horizontal and vertical direction.

implies a much faster operation of the detector, as the sum of subwindows to be scanned rises considerably. This is also indicated in Table 4.8. Note, that the costs for classifying subwindows of different sizes are the same, no matter of the scale the detector is currently operating at. However, for exhaustively scanning images at 10 different scales, the average classification time for each subwindow must not exceed $0.3\mu s$. As another example, consider the more practical A10 highway scenario. A listing of the number of subwindows to be scanned for object search at multiple scales is given in Table 4.9. For scanning at 13 different scales, the average classification time might not exceed $0.15\mu s$. Although we are aware of the fact, that achieving this goal is rather impossible exploiting the possibilities of current hardware and algorithms, this example impressively illustrates the hopelessness of a straight-forward realization of the approach on an underlying DSP platform of this kind.

In the following we will first investigate several aspects of a hardware related, functional realization of the approach. The UIUC database constitutes our playground for investigating this type of hardware related modifications. Thereafter, we will discuss issues on the algorithmic level to improve the detector performance. In a more practical scenario, the need for exhaustively scanning images at multiple scales is only hardly the case, as most often the underlying scene geometry defines clear rules for searching at multiple scales. On this account, we will examine the impact of a tuned object scan instead of exhaustively scanning. We use the "A10 highway" testset, which we find to be a suitable setup in this context.

4.6.2 Core Functions

We define three core functions, which contain the main functionality of the detector. These functions are the *Integral Image Calculation* function, the *Classify Function*, applied to each single subwindow during exhaustive scan, and the *Non-Maxima Suppression* function, which is post-processing all detections and formulates the final detector output.

4.6.2.1 Integral Image Calculation

The integral image is calculated according to the equations given previously in section 4.3.1. We assume, that the original grayscale image and the integral image are both aligned to an 8-byte boundary, so we can test several different aspects of memory access and instruction parallelism. All timings were measured directly on the DSP. The counting of cycles was done using the *Cycle Accurate* simulator included in the Code Composer Studio 3.2.

Table 4.10 lists the results of two different evaluations and different adaptations to the integral image calculation function. A realization of the plain functionality achieves a low performance and takes almost 1.8 ms for a 352×288 pixel image. By informing the compiler about the independence of pointers to separate memory buffers, a small improvement can be achieved. The main reason is, that now the compiler is able to introduce a higher amount of parallelism into the object code. As can be seen from the rather larger gap, using an more suitable memory access scheme, the highest benefits can be achieved. Using word-wise memory access, the total time needed can be cut by almost 3/4. Using double-word memory access leads to a small additional enhancement. A graphical comparison of the different modifications in favour of the DSP is shown in figure 4.11.

A realization of the function in *Linear Assembly* is expected to achieve an additional improvement. We conclude, that by exhausting the entire set of possibilities of parallelism on the DSP, the performance can be increased by a factor of about 10.

4.6.2.2 Classify Function

The classify function is the main procedure, which is deciding upon the presence or absence of an object. It is applied to each single subwindow during exhaustive search, and performs the cascaded classification. Basically, our simple weak classifiers consist of several rectangular areas, whose integral sums are added or subtracted. The feature value is compared to a classifier threshold and a classifier response is accumulated. The response of an entire stage is a sum of *alphas* and *betas*, respectively, as already indicated in section 4.2.3. This sum is again compared to a stage threshold, which lets the subwindow pass to the next stage, or discards it as false positive.

The high amount of additions, subtractions and comparisons gives rise to an investigation of the functional realization in a more hardware promoted fashion. On usual computers, the use of large datatypes, *i.e.* `float` and `double`, is common due to their

Evaluation Type	DSP	Simulation
plain C	1.814 ms	11,575,984 cyc.
anti-aliasing of pointers	1.669 ms	9,356,537 cyc.
word memory access	0.422 ms	3,062,143 cyc.
double-word memory access	0.320 ms	1,762,656 cyc.

Table 4.10: Integral image calculation on a 352×288 pixel image. As can be seen, by exploiting the possibilities of efficient memory access and parallelism, a drastic improvement in performance can be achieved.

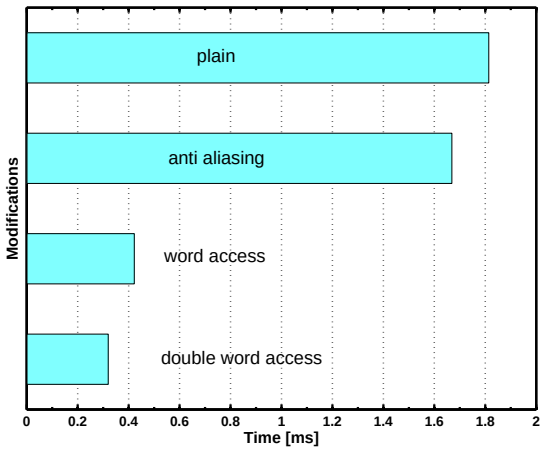


Table 4.11: Graphical illustration of the advantage of the realization of the integral image calculation function in a hardware favoured way.

comfortable use and the lack of accuracy problems in handling. However, it is clear that the use of datatypes, which are not directly supported by an embedded platform, leads to a significant loss in performance. On this account, we want to investigate the need for high accuracy in this context, and evaluate the benefits of a more hardware oriented realization, using suitable datatypes for α, β, and the stage thresholds. Moreover, remember that for calculating the feature value a scaling is necessary, which is dependent on the basic feature size. Also this type of scaling is converted into a fixed-point version to better fit the hardware conditions.

In Table 4.12, a listing of results for different datatypes is given. We have used the detector with ISFP from above, containing 11 stages and 59 weak classifiers in total (see Table 4.2 for details). As an input image, we used a single positive example, which passes all stages, respectively. This also defines an upper bound for the amount of time, a patch needs to pass all stages. As can be seen, using the the `fastrts` library, the computational effort can be reduced for both, single- and double-precision datatypes. However, the advantage of `short` or `char` datatypes is apparent, as they can be used to achieve a remarkable improvement.

We also evaluated several different images from the UIUC testimage dataset with a fixed scale, and averaged the time needed for each patch to be classified. The total number of patches evaluated is 11,179, while the results for the individual datatypes are listed in Table 4.13. In Table 4.14, we have listed the average time and averaged number of cycles for a single feature to be calculated. Note, that these values are independent of the size of the detector and denotes the average time for a feature value to be calculated as a normalized sum over several rectangular areas in the integral image. This Table pinpoints the dilemma of real-time operation of a detector rather clearly. Remember, that, at the

Datatype	DSP	Simulation
double (wo. fastrts)	0.183 ms	292,878 cycles
float (wo. fastrts)	0.143 ms	223,763 cycles
double (w fastrts)	0.145 ms	226,553 cycles
float (w fastrts)	0.076 ms	157,856 cycles
short	0.045 ms	107,524 cycles
char	0.045 ms	107,652 cycles

Table 4.12: Comparison of different datatypes for the weak classifier responses and the stage thresholds. Note, that we have used the detector with ISFP from above, containing 11 stages and 59 weak classifiers in total.

Datatype	DSP	Simulation
double (wo. fastrts)	0.0331 ms	35,668 cycles
float (wo. fastrts)	0.0257 ms	26,864 cycles
double (w. fastrts)	0.0253 ms	26,906 cycles
float (w. fastrts)	0.0128 ms	18,191 cycles
short	0.0077 ms	12,833 cycles
char	0.0072 ms	11,977 cycles

Table 4.13: Comparison of different datatypes in respect of the average time of classification given 11179 patches from several different testimages of the UIUC. Again, we used the detector mentioned before.

beginning of this section 4.6.1, we calculated an upper bound for the time available for classifying each subwindow during an exhaustive object scan to be 2μs. In this regard, we can retain, that this implies the rejection of most subwindows after one or two features, which is really hard to achieve given the current Boosting approach.

To summarize the results, it can be seen, that the advantage of the datatypes, which are especially suitable for the DSP, is striking. The total computational effort can be cut by more than half. Due to the high number of subwindows scanned during detection, even a small improvement has a remarkable effect on the performance of the detector on a full-sized image. This is also graphically illustrated in Figure 4.15.

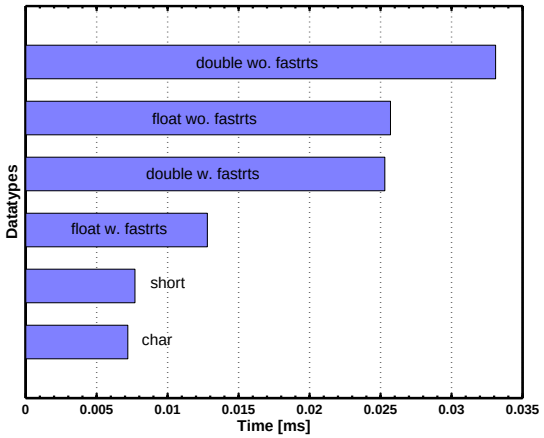


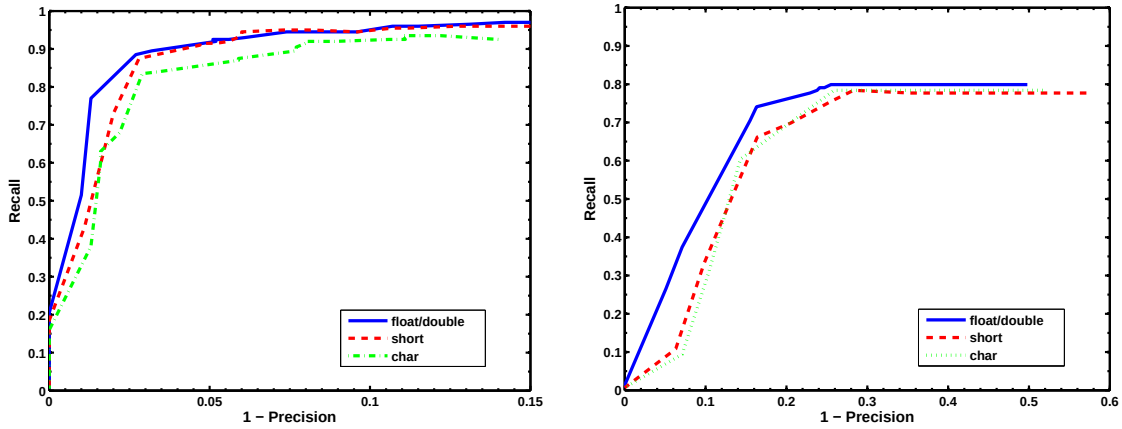
Figure 4.15: Graphical illustration of the striking benefits of using datatypes, which are suitable for the DSP, given the average classification time for a single subwindow.

Datatype	DSP	Simulation
double (wo. fastrts)	4.113 μs	4,934 cycles
float (wo. fastrts)	3.192 μs	3,702 cycles
double (w. fastrts)	3.157 μs	3,680 cycles
float (w. fastrts)	1.606 μs	2,560 cycles
short	0.950 μs	1,725 cycles
char	0.904 μs	1,723 cycles

Table 4.14: Average computational complexity for a single feature to be calculated, given the different datatypes used. Note, that in this case, only the normalization of the rectangle areas cause the difference.

One major question when using fixed-point arithmetic is, if there is a significant loss in detection performance due to the more inaccurate calculation of feature values and thresholds, respectively. We evaluated the detector with different datatypes, using both

datasets of the UIUC database. As can be seen in Figure 4.16 a), the use of approximations does not lead to a remarkable degradation in detection performance. The detection accuracy is not significantly lower than using high-precision calculations. Note, that the performance of the detector is almost the same when using `char` and `short` datatypes, and is exactly the same for `double` and `float` values. Also on the second testset, where exhaustive scanning is used, the use of approximated *alpha*s and *beta*s does not lead to a significant decrease in accuracy. This can be seen in Figure 4.16 b). Therefore, we can conclude, that it is sufficient, to use approximated values instead of exact values. Needless to say, that this is a big step towards real-time capabilities of a detector on an embedded platform without explicit hardware support for high precision datatypes.



(a) (b)

Figure 4.16: a) RPC curve of the detector on the fixed-scale testset, using exact values or approximations of the thresholds, the *alpha*s and the *beta*s. b) RPC curve of the detector on the testset with multiple scales, again, using exact values or approximations of the thresholds, the *alpha*s and the *beta*s.

4.6.2.3 Non-Maxima Suppression

The Non-Maxima Suppression function forms the last phase of our detector. Multiple detections of single objects are combined into final results, which express the final detector output. In principle, the total functionality can be described by sorting all detections in decreasing order by their confidence rate, and by two nested loops to calculate rectangle overlaps. Given a detector with satisfying performance as used here, the number of detections to be processed by the method is typically in the range of 10 to 250 detections. The calculation of overlaps can be done efficiently using integer values, and the total time increases almost linearly. This is also illustrated in Figure 4.17 and in Table 4.15.

Because of the low computational effort for calculating the result, the postprocessing step can be ignored for global considerations on real-time performance of a detector.

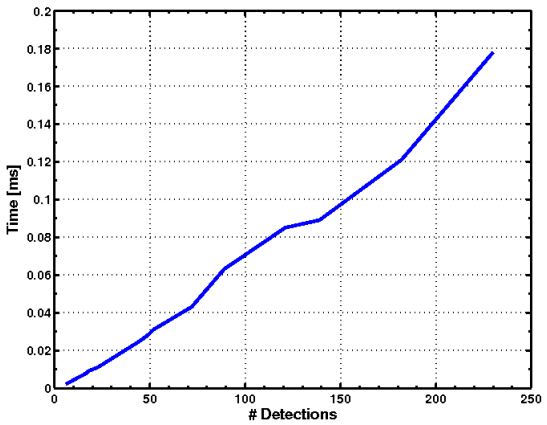


Figure 4.17: Linear dependency of the computational effort for the number of detections to be processed.

# of detections	DSP	Simulation
6	0.002 ms	5,652 cycles
18	0.009 ms	23,224 cycles
26	0.013 ms	34,842 cycles
45	0.025 ms	66,804 cycles
52	0.030 ms	95,052 cycles
72	0.042 ms	120,294 cycles
89	0.063 ms	178,794 cycles
121	0.085 ms	233,328 cycles
139	0.089 ms	293,772 cycles
182	0.121 ms	410,610 cycles
230	0.178 ms	473,388 cycles

Table 4.15: Listing of the amount of time and number of cycles, needed for post-processing. A graphical illustration of the almost linear dependency is depicted in Figure 4.17.

4.6.3 Distance-dependent Scaling

In practical situations, often objects at multiple scales might occur frequently. However, in most cases, the underlying scene geometry allows for the assumption of a particular object scale at a given position in the image. Thus, it is not necessary to scan images exhaustively with fixed scales at every position. This is an important hint, which can be applied on the algorithm level to speed up the approach considerably.

For a closer investigation of this modification, consider the highway scenario, that we have introduced in section 4.5.3. The underlying scene geometry and the physical real-world conditions enforce, that vehicles have their largest appearance close to the camera location. Needless to say, that the object size smoothly decreases with the distance of the object. We can scale the detection window accordingly to avoid searching for objects at impossible scales.

In Figure 4.18, the idea of distance dependent scaling is illustrated. As can be seen clearly, the size of the detection window smoothly decreases with the distance to the camera location. In Table 4.16, a listing of the subwindows to be scanned is given, assuming the smoothing decrease in scale and a factor of 90% overlap between adjacent subwindows. Note that, compared to our original calculations in Table 4.9, only about 2.6% of all possible patches has to be classified, which leads to a tremendous increase in detector performance. The total number of different scales is 62 in this case, starting with a factor of about 3.75 in the front of the scenario, down to about 0.75 in the distance.

For evaluating this idea in practice, we have chosen the detector with ISFP, which was trained and described before in Section 4.5.3. To recapitulate, the detector has 52 weak classifiers in 13 stages. In Figure 4.19, some results of the detector are depicted for detecting cars by using a smoothly decreasing scale factor. Because we apply the detector on the whole image for demonstration purposes, we can also detect cars moving into the

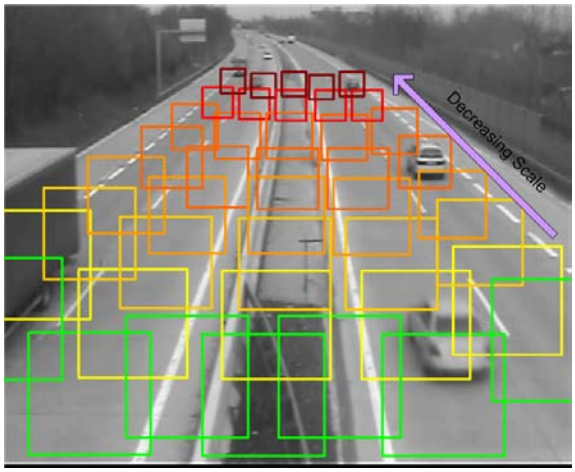


Figure 4.18: Illustration of the distance-dependent scaling of the detection window. The decreasing size of the subwindows is also denoted as a color change of the rectangles from *green* (large scale) to *dark red* (small scale).

Scale Range		# of Subwindows
0.75	1.00	1,680
1.00	1.25	1,656
1.25	1.50	652
1.50	1.75	750
1.75	2.00	395
2.00	2.25	388
2.25	2.50	245
2.50	2.75	240
2.75	3.00	149
3.00	3.25	145
3.25	3.50	123
3.50	3.75	81
Total:		**6,504**

Table 4.16: Listing of subwindows to be scanned, using a smoothly decreasing scale. Note, that we have listed the numbers by scale range, rather than individual scales, because the total number of different scales is 62 in this case.

opposite direction reasonably well, although we have not explicitly trained the detector to do so. Some results are depicted in Figure 4.20. Another very nice effect of the distance-dependent scaling approach is, that the performance of the algorithm in terms of the recall rate rises considerably as shown in Figure 4.21. Because of the restricted search, a lot of false positives are removed and also the scale of the detections is much more accurate, which reduces the number of bad detections by a considerable factor. As a consequence, a much higher F-measure is achieved given the new approach. From the original level of 56.47%, it now rises to 86.15%, which is an excellent advancement.

Although the distance-dependent scaling approach delivers satisfying results in most cases, some problems cannot be overcome, as depicted in Figure 4.22. As can be seen clearly, large vehicles might erroneously be detected twice. Other problems occur at image boundaries, where objects are not fully visible and the detector still works. Clearly, the restrictive Agarwal and Roth criteria register such a detection as an error, although the detector is not able to deliver a better result.

A very important aspect is the drastically reduced number of subwindows to be classified, as listed in Table 4.16. Classifying all patches on the embedded platform now takes only $17,29ms$ on average, which is a very nice result, compared to about $708,22ms$ on average for a full range scan at 13 scales, as previously discussed and listed in Table 4.9. This result is also a strong argument, such that the application of a detector should be carefully tuned to a given scenario. This aspect is also important, because it impressively shows, that the saving of a big amount of computational costs to achieve a speedup, must not necessarily come with a bad influence on algorithm accuracy. Quite the contrary, in

Figure 4.19: Some sample results of the detector, given the smoothly decreasing scale search. The detector is able to scale nicely to fit the vehicle size at a given position.

this case a dramatic increase in detection performance can be achieved. Thus, we can conclude, that the real-time property of an overall system can be achieved by tuning the underlying algorithm to the given application carefully and under consideration of all aspects of embedded systems, but with the main focus on modifications on the algorithmic level.

4.7 Concluding Notes

In this chapter, we have investigated several aspects of object detection on embedded platforms. Following a description of the theoretical foundations, we have shown the plausibility of our approach on various different scenarios. While we were focusing on the principal aspects of the realization of our approach on embedded platforms, significant advantages of hardware related development have been revealed. An important result

Figure 4.20: Some detection results for detecting vehicles, moving into the opposite direction. As can be seen clearly, the approach also works for detecting vehicles driving into the opposite direction, although we did not train the detector to do so.

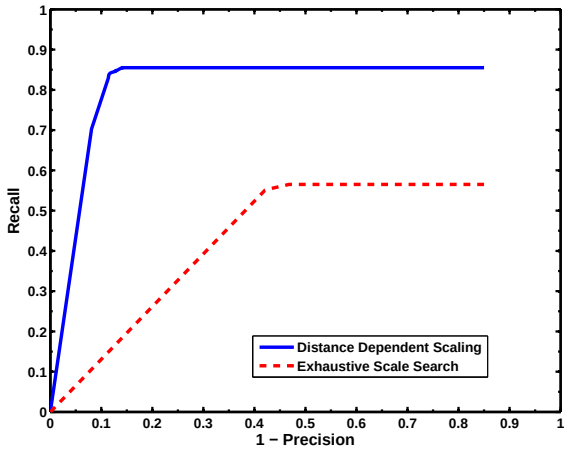


Figure 4.21: Recall-Precision curve for both, the exhaustive scan and the tailored scanning approach. As can be seen, the performance is considerably higher, if scanning is restricted to reasonable scales for each individual location.

is, that real-time behaviour of a detector can only be achieved by incorporating multiple different aspects of algorithm realization. Hardware related considerations, like parallelism and suitable datatypes, can lead to a significant benefit on the smart camera, compared to a straight-forward implementation. However, only modifications on the algorithmic level enable an approach to deliver acceptable performance in both, accuracy and speed. This was demonstrated by achieving tremendous speedups by adapting the approach using reasonable assumptions and limitations.

In his thesis, Reuvers discussed the suitability of the original Viola-Jones approach for implementation on SIMD architectures [193], and also discussed the suitability of different

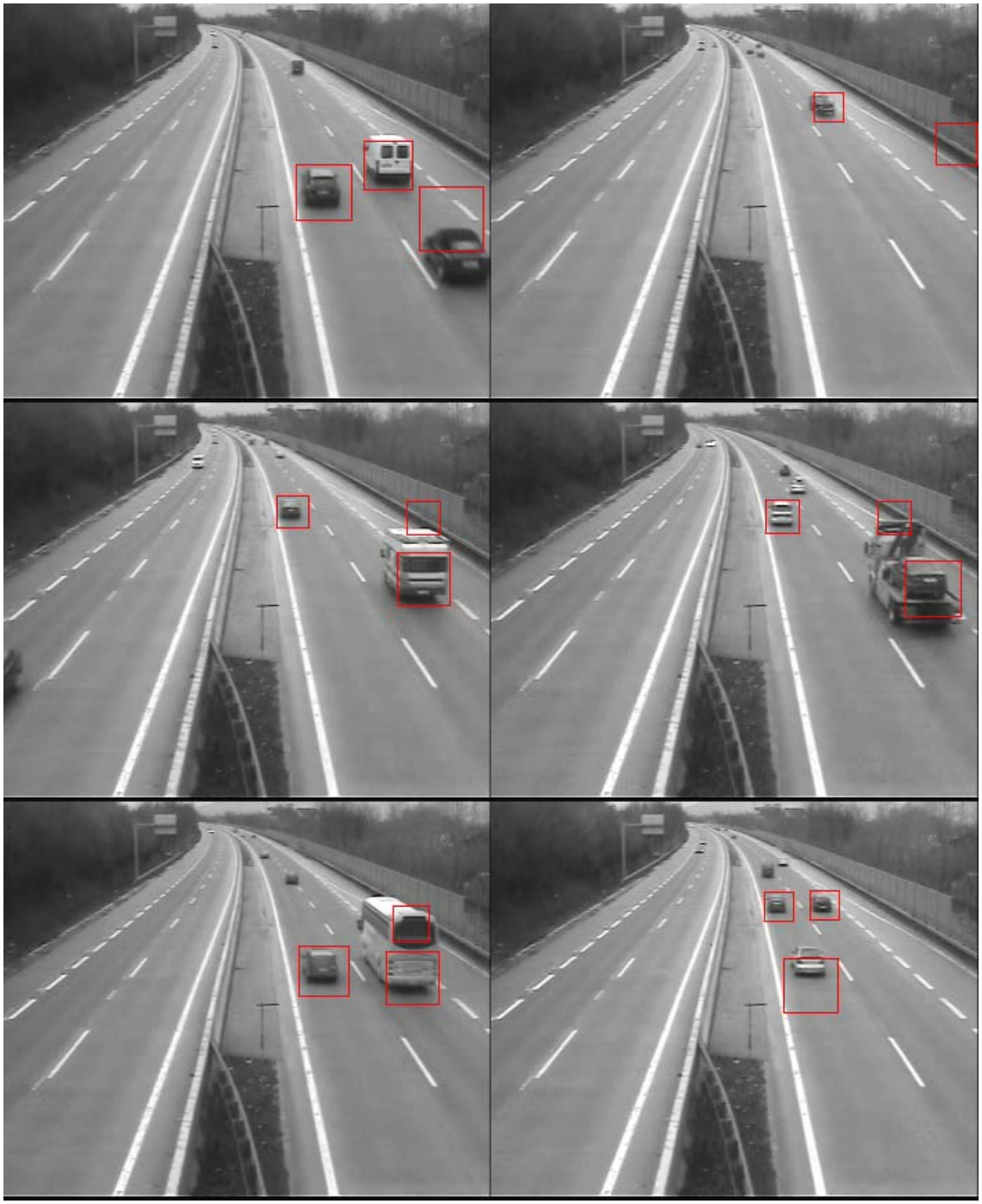


Figure 4.22: Some erroneous results of the distance-dependent scaling approach. Large vehicles might be detected twice, and problems occur at the image boundaries, where objects are not fully visible, but are still detected by the detector.

types of features. We agree with one of his main findings, that the Haar filters, and separable filters in general, are very suitable for DSP-based platforms, because they can be calculated efficiently. However, one of the main unsolved problems in this respect is, that the building of an image pyramid is necessary to calculate filters with a larger spatial extent. Needless to say, that this comes at considerable costs in terms of memory. Even more important, the methodology of detecting is still restricted to a sliding window approach, in which a classifier has to be applied at each image location exhaustively. Out of the scope of this thesis, we ran some premature tests on an algorithm based on separable filters and image pyramids to get an idea about the resulting detector behaviour. Although we found out, that the idea works in general, we could not build a detector, which was competitive to the original approach, neither in terms of accuracy, nor in terms of speed. Thus we came to the conclusion, that for successfully deploying the Viola-Jones algorithm on embedded devices, modifying the classifier structure, *i.e.* the number of weak classifiers, is preferable over changing the way of feature calculation. However, both starting points do not exclude each other, thus we also look forward to a closer investigation of the ideas of Reuvers for building an improved approach.

We demonstrated, that our approach is suitable to perform in real-time. It is still necessary to keep several important issues in mind, regarding the size of the detector and the number of weak features. Moreover, limited resources in terms of memory only allow for competitive performance, if the image size allows for operating without need for swapping memory blocks between external and internal banks. It was shown, that the detector size has a major impact on the overall performance, say, the less weak classifiers are used, the higher the performance in terms of speed. Hence, an important result in respect of our hardware related investigations is, that algorithms, aiming at minimizing the number of weak classifiers, are especially suitable for the development on embedded systems. From an other perspective, this outcome can also be interpreted as one more indication, that modifications on an algorithmic level has a much higher influence on the overall performance of an approach than adaptations on the implementation level. In this regard, we especially refer to the work of Šochman and Matas on *WaldBoost* [253], which was already described in Section 4.4.3, respectively. *WaldBoost* seems to be the most suitable approach, aiming at a minimization of the number of weak classifiers and, thus a maximization of detector performance in terms of speed. Although we have not evaluated it in the context of our work on embedded systems, a closer investigation is strongly recommended.

The creation of something new is not accomplished by the intellect but by the play instinct acting from inner necessity. The creative mind plays with the objects it loves.

Carl Jung
Swiss psychologist (1875 - 1961)

Chapter 5

Object Recognition

Object recognition[1] is one of the most popular tasks in the field of computer vision. In the past decade big efforts were made to build robust object recognition systems based on local appearance features [44, 167, 219, 221]. For such a framework to be applicable in the real world, several attributes are very important: insensitivity against rotation, illumination or viewpoint changes, as well as real-time behavior and large-scale operation. Current systems, which run on usual computers, already have a lot of these properties and, though not all problems have been solved yet, nowadays they become more and more attractive to the industry for inclusion in products for the customer market. However, yet, object recognition has not been investigated widely in the context of embedded systems. On this account, this chapter deals with the realization of a state-of-the-art object recognition approach on DSP-based smart cameras.

The remainder of this chapter is structured as follows. First, we shortly define the importance of object recognition in the context of embedded systems. We also outline our work in section 5.1. A detailed description of the methods involved in building our object recognition algorithm is given in the following section 5.2. We also outline our framework and give details about training and implementation of our system. We closely describe all steps in designing our approach and give side notes on alternative methods. In section 5.3 we experimentally evaluate our system on a challenging object database and discuss real-time and real-world issues. Furthermore we investigate some special features of our approach and elucidate the dependencies of several parameters on the overall system performance. In section 5.4, an application of the approach to vehicle reacquisition on public streets is described. In this context, also special issues concerning larger smart camera networks are discussed. The chapter concludes with a discussion of the results and some final notes in section 5.5.

[1] This chapter is an adaption and extension of the work of [9] and [11].

5.1 Introduction and Motivation

On the one hand, there is an indisputable need for certain object recognition capabilities in any system for visual surveillance. Needless to say, that for smart cameras this places a clear justification for development in that area. On the other hand, also a big range of commercial products deals with mobile, handheld devices, like mobile phones, PDAs or portable music players. Almost everyone's mobile phone is equipped with a camera and, thus, can also be treated as a small embedded vision system. Hence, embedded vision platforms are already present in our everydays life, however, only offering a limited amount of computational and memory resources [264]. Clearly this gives rise to new applications, like navigation tools for visually impaired persons, or collaborative public monitoring using millions of artificial eyes. It is obvious, that certain object recognition capabilities for surveillance, household robotics, entertainment, or military and industrial robotics are indispensable features of embedded systems, and, thus object recognition on embedded devices is an important field of research and development.

Some attributes of embedded platforms strictly limit the practicability of current state-of-the-art object recognition approaches. For example, the amount of memory available on a device strictly limits the number of objects in the database. Therefore for building an embedded object recognition system, one goal is to make the amount of data to represent a single object as small as possible in order to maximize the number of recognizable objects. Another important aspect is the real-time capability of these systems. Algorithms have to be fast enough to be operational in the real world; they have to be robust and user-friendly, otherwise a product equipped with such functionality is simply unattractive to a potential customer. For example, in an interactive tour through a museum object recognition on a mobile device has to be fast enough to allow for continuity in guidance. To summarize, building a full-featured recognition system on an embedded platform turns out to be a challenging problem given all the different aspects and environmental restrictions to consider.

In the following, we describe a method to deploy an object recognition system on our prototypical DSP-based embedded platform. To the best of our knowledge, we are the first to extensively investigate issues related to object recognition in the context of embedded systems; by now this is the only work studying the influence of various parameters on recognition performance and runtime behaviour. The major goal is to deploy a medium-sized database, which contains 250 objects, on a DSP-based platform, and to understand the tradeoffs between the database size, recognition performance, computation accuracy and other issues specific to embedded devices. We pick a set of high-level algorithms to describe objects by a set of appearance features. As a prototypical local feature based recognition system we use DoG (*Difference of Gaussian*) keypoints [142] and PCASIFT (*Principal Component Analysis Scale Invariant Feature Transform*) descriptors [115] to build compact object representations. By arranging this information in a clever tree-like data structure based on k-means clustering, a so-called *vocabulary tree*, real-time

behaviour is achieved. By applying a dedicated compression mechanism, the size of the data structure can be traded off against the recognition performance. Thereby accurate tuning the properties of a recognition system to an arbitrary DSP-based hardware platform can be performed. As it is shown in extensive evaluations by considering both, special properties of the algorithms and dedicated advantages of special hardware, considerable gains in recognition performance and throughput can be achieved. To further prove the reasonability of our approach, we demonstrate its suitability in a more practical application in traffic surveillance, namely vehicle reacquisition.

5.2 Object Recognition Framework

We start with a description of our keypoint detector and the region descriptor used. Thereafter the vocabulary tree building approach and its usage is explained in detail. Finally, our methods to successfully compress the amount of data in our database is elucidated at length. A schematic illustration of our approach is depicted in Figure 5.1. We also shortly explain the object database, that is used to evaluate the approach, and shortly outline the preparations for experimental evaluations of the approach.

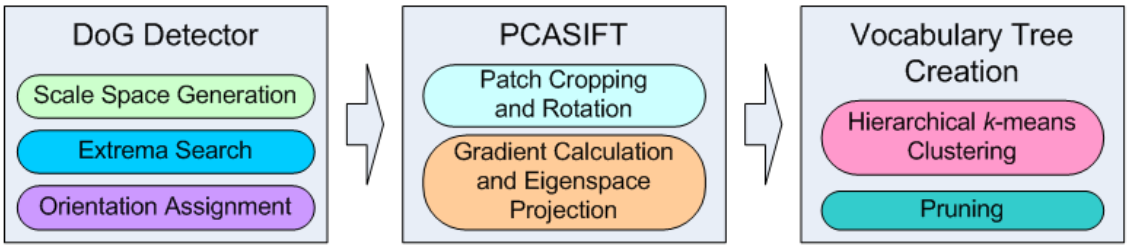


Figure 5.1: Schematic illustration of our approach.

5.2.1 Keypoint Detection and Descriptor Calculation

Based on grayscale images, we use the DoG detector to extract interest regions. The DoG detector is mainly based on Gaussian filtering and differencing the resulting filtered images (see Figure 5.2) [142]. The differences can be interpreted as an approximation of the scale normalized Laplacian. By doing so a scale space is built in multiple octaves, and maxima and minima in the scale space are determined. These extremas are keypoints, which indicate the presence of blob-like (more or less circular) structures in images. For each keypoint a circular region around the keypoint is cropped whose size is dependent on the scale factor delivered during detection. By summing up the gradients in the image patch, the main gradient direction is determined and assigned as orientation to the keypoint. The image size is downsampled by a factor of 2 with each doubling of the sigma of the Gaussian filter kernel (after each octave) to form the initial image for the next octave.

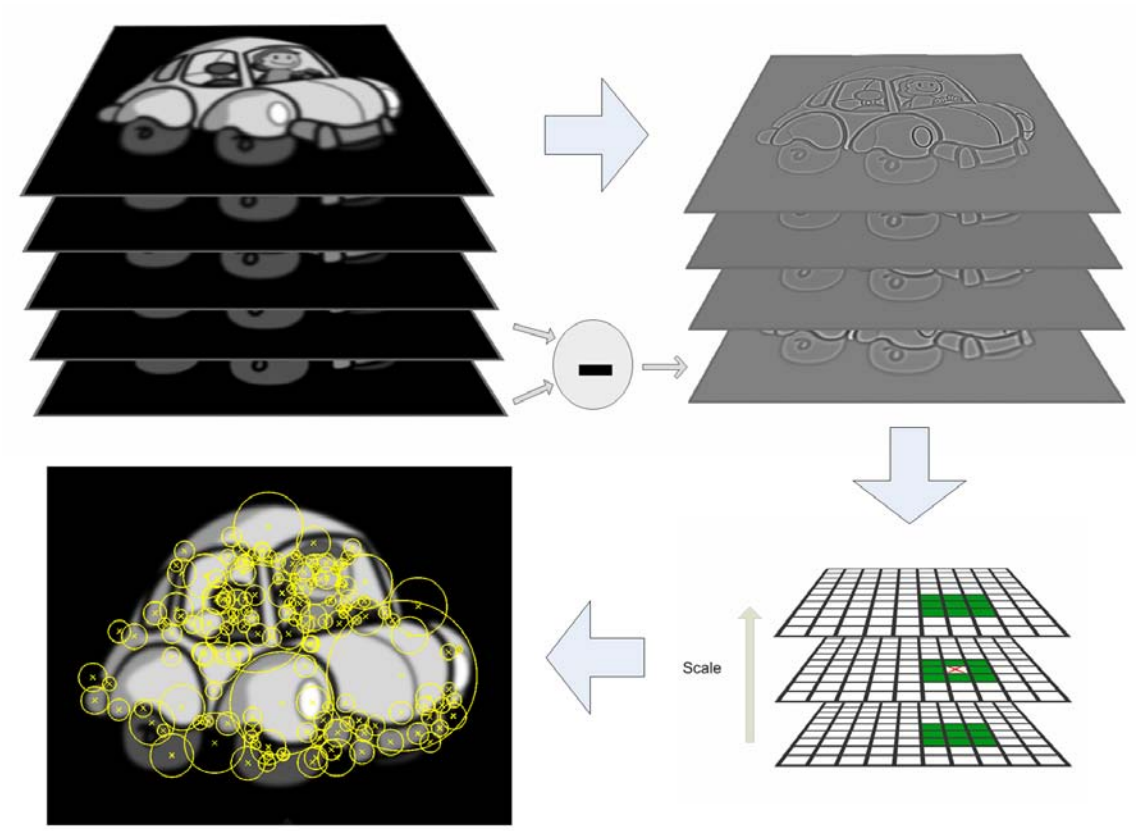


Figure 5.2: Illustration of DoG keypoint detection. First the scale-space is built by generating a difference image stack from multiple Gaussian filtered images. The maximas found in scale space are the keypoints sought.

A nice feature of the DoG detector is that it is almost purely based on image filtering and addition/subtraction operations. While a clever arrangement of filtering and search operations makes the algorithm also efficient in terms of memory usage, the algorithm is very well suited for DSP platforms, as they are mainly designed for fast filter operations. We implemented the Gaussian filtering in fixed-point as the hardware platform has no floating point unit and floating point operations have to be emulated in software. Due to the small amount of internal memory the filtered images and the difference images are consecutively swapped between the external memory and the internal cache of the DSP. To reduce the number of difference images to be stored in the stack for extrema search, the search is performed on each difference image stack immediately after creation. By doing so the difference image can be discarded immediately and only the valuable information about maxima and minima has to be kept. For determining the main orientation of a keypoint, a scale-normalized patch is cropped from the original image around the keypoint and resized to a fixed size to fix the runtime of this orientation assignment step. After calculating the gradients on this patch, the main gradient orientation is determined by finding the maxima

in the accumulated orientation histogram. For all local peaks that are within 80 % of the highest peak another new keypoint with the same scale and location but with different orientation is created. This significantly increases stability in matching for keypoints in highly textured areas with multiple dominant orientations [142].

For calculating the description of the appearance around a detected interest region, we use the PCASIFT approach from Ke and Sukthankar [115]. This descriptor has several advantages, especially for our utilization. First, the algorithm mainly consists of *multiply-accumulate* (MAC) operations, which fits the properties of embedded platforms very well. Secondly, the descriptor is much more compact, because they have proven the $d = 36$ dimensional descriptor to exhibit the same discriminatory power as the 128-dimensional SIFT descriptor. A third big advantage is, that a further decrement of d results in only a slight loss in discriminatory power, thereby making the descriptor calculation itself scalable. Finally, the amount of storage for the a large set of descriptors is also reduced by a factor of ≥ 4, because of the smaller amount of memory needed to store the individual descriptors.

A scale-normalized patch which exhibits the same dimensions as proposed in their original work is extracted from the original image and rotated to compensate the specific orientation. The dimensions are chosen such that we can use the same coefficients as Ke and Sukthankar [115]. Furthermore we converted their Eigenspace projection matrices to a fixed point version. By doing so we can take advantage of the benefits of fixed point calculations on our platform. The dimensionality of the resulting descriptor d can be adjusted, which allows for a tradeoff between discriminability and final recognition performance, but also between more and less computationally expensive calculation. The final descriptor is a d-dimensional vector of 1-byte elements.

5.2.2 Vocabulary Tree Generation

Although linear exhaustive search can be implemented very efficiently given embedded hardware properties, it is very computationally expensive. The exhaustive matching of descriptors in databases is impracticable for databases with more than several hundred descriptors. Though a tree-like data structure combined with an approximated nearest neighbor search is preferred for the management of our object representations, even if data-dependent control flow and control code containing random conditional statements cannot be executed very efficiently on DSPs. The vocabulary tree allows for a approximated nearest neighbor search in moderate dimensional spaces with a huge numbers of candidates. Because this method has been shown to be highly efficient in terms of speed and accuracy [167, 219], we also use this technique in our own approach.

For building our vocabulary tree we largely follow the approach of Nistér and Stewénius [167]. Given a database of descriptors for training, hierarchical k-means clustering is used to divide the feature space into separate partitions without interruption. As the final result of this approach, a structure is generated where each descriptor from the training

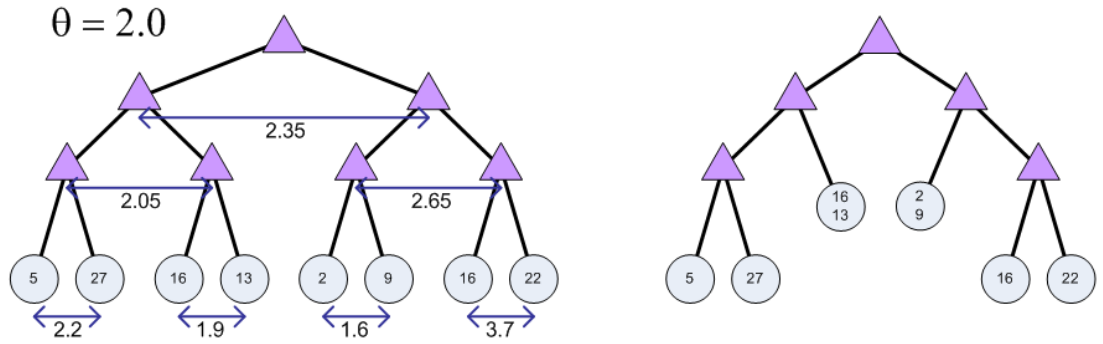
set forms a single leaf of the tree. The inner nodes of the tree are the cluster centers identified during training. Note, that we train and build the vocabulary tree on a usual PC, because of the high computational effort and the high amount of memory resources needed. The vocabulary tree is stored as a raw data buffer, such that it can be uploaded onto a hardware platform easily.

Given a new descriptor determining the approximate nearest neighbor is performed by traversing the tree from the root downwards to the leafs. In each stage the actual descriptor is compared to the inner nodes of the tree and the one with the smallest distance is chosen to be the next node along the path. This procedure is repeated until a leaf - and thus the nearest neighboring descriptor - is reached. The parameter k chosen in the clustering step can thus be used to trade the depth of the tree structure against computational complexity in early stages of the tree in the final recognition step. The step of approximate nearest neighbor search is also performed directly after building the tree to build the so called *inverted file structure*. This structure holds the information which leaf nodes are hit by the descriptors of an object (or which object inversely obtains a vote if a descriptor hits the leaf). Given a fixed tree the inverted file structure can be altered during runtime, which means that object representations can be added and removed from the file structure arbitrarily. This is also subject to investigations in section 5.3.

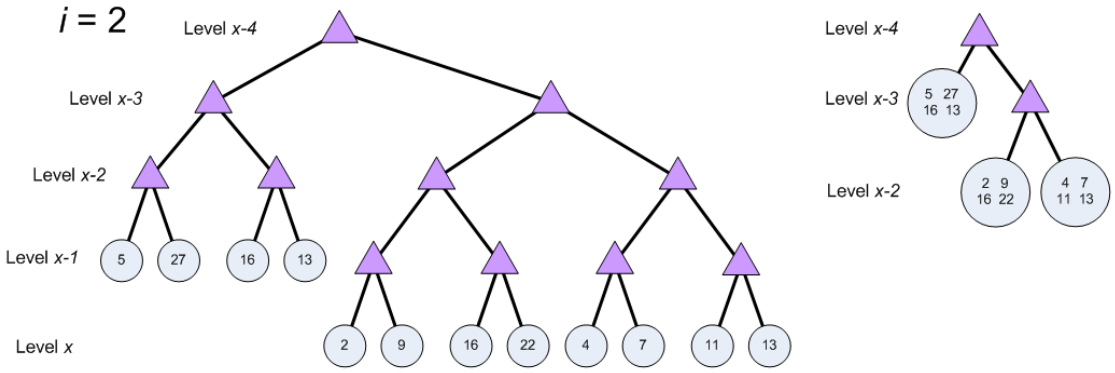
One point to mention is the fact that the k-means clustering based tree structure is just a method to approximate the optimal nearest neighbor search. Experiments show that determining the absolute nearest neighbor in descriptor space fails in about 2-5 % of all cases. The reason for this is that the clustering algorithm enforces linear borders between clusters and partitions the feature space locally using global cluster information. Thus, for points close to cluster borders the procedure of nearest neighbor search can fail[2]. However, for the overwhelming majority of points in space the partitioning is accurate enough and the algorithm is robust enough to overcome this additional amount of noise. For the descriptors in the training set, the indexing procedure is still performed once again after building to minimize inaccuracies, even if the build procedure delivers an initial indexing result implicitly.

One major drawback of the algorithm is that it is based on a data-dependent control flow. It is impossible to foresee the path of descriptors through the individual branches of a tree. Thus it is nearly impossible to optimize the query any further to better fit the environmental conditions. Though a maximum execution time is implicitly given by the depth of the tree, thus a worst case runtime of the query can be calculated. However, the majority of resources is spent anywhere in other steps of the recognition algorithm, as it is shown in section 5.3, thus optimizations here are not critical to the overall system success. Nevertheless, modifications are possible to limit the number of descriptors and to set a tight upper bound for the algorithm runtime. For example, the *Adaptive Non-Maximal Suppression* (ANMS) strategy can be used to select a fixed number of interest points from

[2]Note, that in contrast to the approach of Nistér and Stewénius [167], we assume a binary decision tree here, thus the number of failures is much lower.



(a) distance based pruning



(b) level based pruning

Figure 5.3: (a) Distance and (b) Level based leaf pruning given a predefined threshold θ or a number i of levels, respectively.

images, spaced equally over the entire image, as proposed by Brown *et al.* [37]. For reasons of computational efficiency, we did not employ such a strategy in our current approach and left this issue as an open point for optimization in our framework.

5.2.3 Tree Compression and Pruning

The vocabulary tree obtained by the procedure described above contains the full amount of information. Consequently each final leaf of the tree votes for a single object only. While this guarantees for the finest possible partitioning in feature space given the training data, the amount of information to store is large such that it cannot be easily deployed on a platform with limit memory resources. Due to the robustness of the local feature based approach a lot of redundancy can be removed. The most efficient way to achieve this is to prune the tree and replace the single votes by a set of votes from the pruned leafs. In other words, if leafs of the tree meet a given criterion they are collapsed into a single one

which now votes for several objects. Note that there is a strong relation to Decision Trees in Machine Learning, where pruning is used to obtain a better generalization [93, 144]. We have evaluated two possibilities for tree pruning.

- Leafs are collapsed if their distance in feature space is smaller than a predefined threshold θ. In this case the partitioning in densely populated areas of the feature space is made coarser first while the partitioning in sparely populated regions of the feature space is not affected. In the following, this strategy is denoted as *distance-based* pruning.
- All subtrees with a predetermined number i of inner nodes are pruned. Thus the partitioning is made coarser equally well in all regions of the feature space by simply merging leafs (and partitions respectively). We also refer to this method as *level-based* pruning.

Both ideas are illustrated in Figure 5.3. By doing so the amount of data needed to store the tree is reduced considerably trading against recognition performance.

5.3 Experiments

In this section we investigate several aspects and properties of our approach in detail. While we want to evaluate the recognition performance in detail regarding different parameters, we also want to evaluate our algorithm in terms of memory usage, timing profile and calculation accuracy on our hardware platform.

First we note some informative details about building our object database, the system setup and our evaluation strategy in Section 5.3.1. In 5.3.2, we discuss the influence of the descriptor dimensionality on the recognition performance. Then we show how our pruning strategies described in Section 5.2.3 influence the recognition performance and the size of the database in 5.3.3. The robustness against occlusion and background noise is subject to experimental evaluations in 5.3.4. We also describe a nice feature of our approach, namely the addition and removal of object descriptions at runtime, in Section 5.3.5. In Section 5.3.6 we investigate the throughput of our system; in detail, we list the average timing results for each individual step of the algorithm on our hardware platform and specify the amount of memory needed to store the individual data buffers. Detailed information about the calculation accuracy in the most important steps of our approach is given in Section 5.3.7. In Section 5.3.8 we shortly demonstrate the speed advantage by choosing fixed point calculations over floating point calculations on our platform. Furthermore, in Section 5.3.9 we discuss optimization issues for further speeding up the algorithm.

5.3.1 System Settings

Our image database is a subset of the publicly available ALOI (*Amsterdam Library of Object Images*) database from Geusebroek *et al.* [84] (Figure 5.4 shows some sample

Figure 5.4: Sample images of objects selected from the ALOI database for our experiments.

Figure 5.5: The images of two objects recorded over the complete viewpoint range. For ease of illustration only the images of 15° steps are depicted here.

images). We preselected those 250 objects out of 1000 which deliver the highest number of DoG points (see Appendix B for a listing of the object IDs). The main reasons for doing so is that deploying this medium-sized object database on our system is already challenging, but to a greater extent because the database contains a lot of objects which can not be sufficiently represented using DoG points alone, as the number of them is too small. To overcome this problem multiple different detectors can be used, but for now we left that as an open issue.

We evaluate our recognition system for a viewpoint range of ±90°. In Figure 5.5 the images for two objects over the complete viewpoint range are depicted. For building our

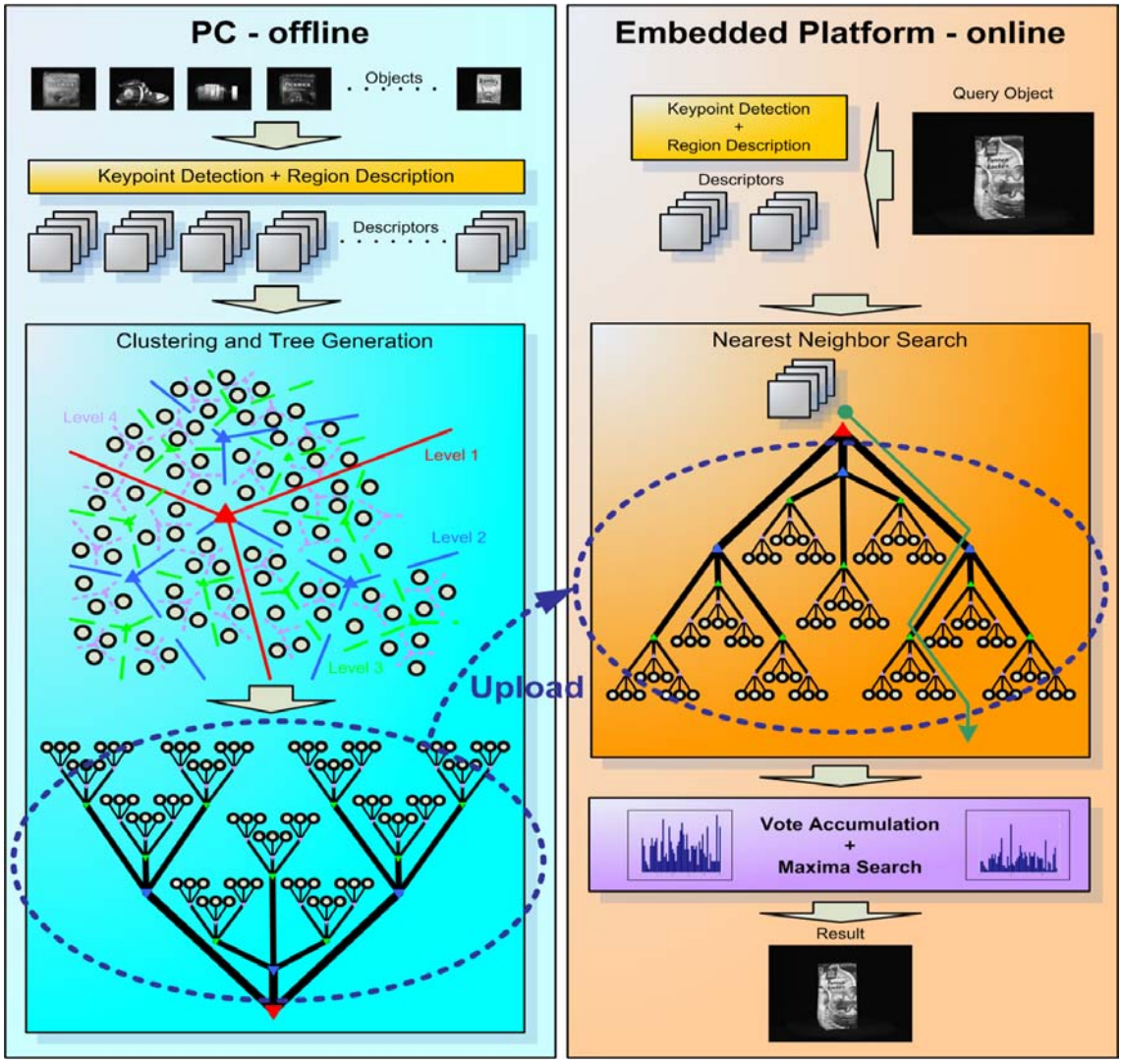


Figure 5.6: Schematic map of our object recognition system. On the left side, the database generation algorithm performed on a PC is shown. Descriptors are calculated on interest regions detected, and the features are iteratively clustered to form the final vocabulary tree (with $k = 3$ in this case). This data structure is subsequently uploaded to the embedded device. On the right side, the process of object detection on the embedded device is depicted. While the extraction of features works the same as on the desktop computer, the final object match is determined following the nearest neighbor search for each descriptor in the tree and finding the maximum vote in the overall voting histogram.

database we use all descriptors at steps of 15°, while we calculate the recognition results at intermediate steps of 5° and omitted testing the system at the angles learned. Because of the memory restrictions of the hardware platform, we limit the image size to CIF format, 352x288 pixels respectively. As the original ALOI images exhibit a size of 768x576 pixels, all images are converted to grayscale and resized to CIF format prior to database building and evaluation. Note that we do not change the aspect ratio of the images but scale the images to 384x288 pixels and crop 16 columns from both the left and right image border to arrive at the desired image size.

For the vocabulary tree, for all experiments k was chosen to be 2, which means that each vocabulary tree is a binary decision tree. Without loss of generality k can be chosen arbitrary to trade vocabulary tree depth against calculation costs and accuracy in practice. Although Nistér and Stewénius [167] have shown a large k to result in better recognition performance, our choice of $k = 2$ is inspired by the idea of simplified control flow for handling a vocabulary tree and the approximate-nearest-neighbor query on an embedded device. If $k = 2$, the query of a descriptor along the path in a vocabulary tree can be implemented by simple $if - then - else$ statements, largely avoiding costly branching instructions. Choosing k to be larger would result in a more complex control structure.

In our approach to calculate distances in feature space use the *sum of squared distances* (SSD) metric rather than the *Euclidean* metric. In doing so the partial ordering of elements is not changed (which essentially means that the voting result is not affected). However, we can omit calculating the square root which is a computationally expensive task on our embedded platform. Due to the memory restrictions we further assume that the critical limit for our database residing in the external memory of the platform is 12.5MB, as we also need a small piece of memory to store other data buffers.

The training of the object recognition system is done on a standard desktop computer using MATLAB^{TM}. After building the vocabulary tree we upload it onto our smart camera platform, where all further evaluations are performed. The vocabulary and the tree structure is represented as raw blocks of data in memory which are accessible interpreting pointers from a binary executable on the platform. The binary program for recognizing objects is built using the Code Composer Studio 3.2 from TI and uploaded together with all necessary data buffers using a JTAG emulator device. During evaluation, images are presented to the algorithm, which extracts local features, calculates descriptors, searches correspondences in the vocabulary tree and finally returns the ID of the best object match. A schematic illustration is shown in figure 5.6. On the left side, the database generation algorithm on the PC is depicted, while on the right side the object recognition procedure on the embedded platform is described.

5.3.2 Feature Dimensionality

As shown by Ke and Sukthankar [115], the PCASIFT descriptor exhibits almost the same discriminability as the original SIFT descriptor proposed by Lowe [142]. However, the

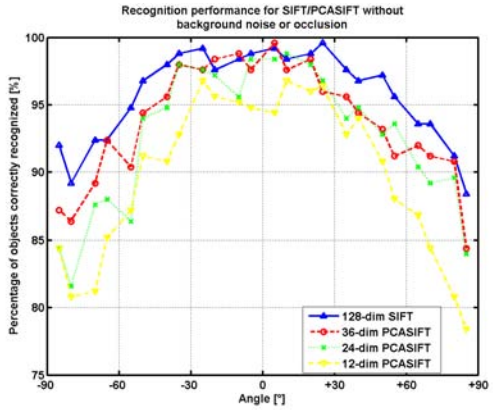


Figure 5.7: Average recognition performance for different feature types and descriptor dimensions over the whole viewpoint range. For ease of illustration only 4 plots are drawn here.

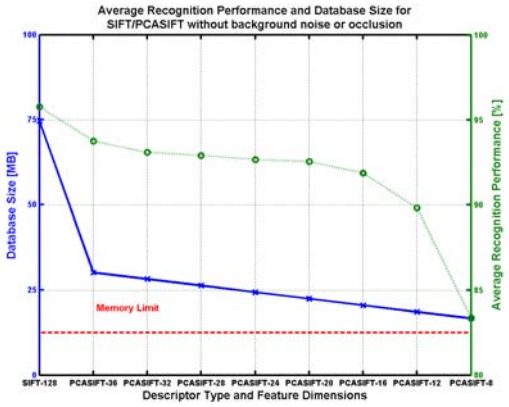


Figure 5.8: Average recognition performance and database size for different feature types and descriptor dimensions are marked as dotted and full line. The critical limit of 12.5MB for our database is marked as red dashed line.

amount of data to be stored is much less, at least a factor of about 1/4, which is desirable in our case. In Figure 5.7, the recognition performance for the 128-dimensional SIFT descriptor and different dimensional PCASIFT descriptors over the complete viewpoint range of $\pm 90°$ is shown. For a fair comparison the evaluations were performed without background noise or occlusions.

As can be seen the PCASIFT descriptor performs only slightly worse than the SIFT descriptor. In general recognition performance decreases with the distance in viewpoint from an angle learned during training (we omitted the learned poses). At 0° objects are recorded in frontal view. The objects often exhibit a lower number of DoG points when they are viewed from the side. Thus the recognition performance generally decreases when the viewpoint changes from 0° towards $\pm 90°$.

In Figure 5.8, the database size and the average recognition performance for different feature types and descriptor dimensions are shown. The critical limit of 12.5MB for our database is marked as a red dashed line. Choosing the PCASIFT descriptor over the SIFT descriptor results in a major decrease in database size from about 78MB to about 32MB for the 36 dimensional descriptor. Note that the database size is only reduced to a factor of about 0.41, because the internal tree structure needs a fixed amount of space which is not influenced by the choice of the descriptor type. As can be seen, the database size only slightly decreases with the number of descriptor dimensions and never reaches the critical limit. However, the recognition performance drops below 90% when the descriptor dimensions are reduced to 12. Note that the average recognition performance was calculated without background noise or occlusions.

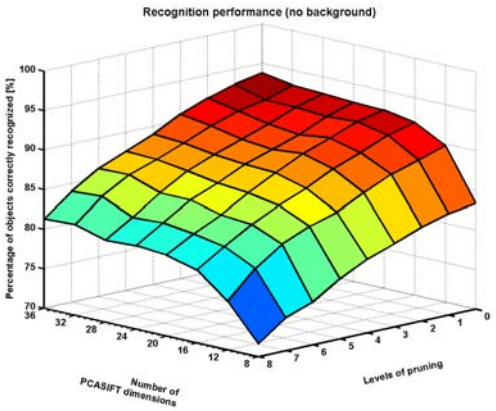


Figure 5.9: Average recognition performance for the level based pruning method and different parameter settings.

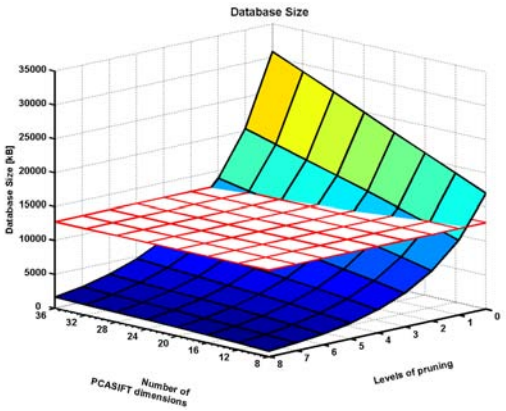


Figure 5.10: Database size for the level based pruning strategy and different parameter settings. The size limit of $12.5MB$ is depicted as plane here.

5.3.3 Tree Pruning

As shown in the previous section, a much more compact database can be generated choosing the PCASIFT descriptor over the SIFT descriptor. However, the size of the database is still much too large to fit onto our platform (see full line in Figure 5.8). Thus we employ different strategies to further compress the database, still trying to preserve the best recognition performance possible. In Figure 5.9 the influence of the level based pruning strategy and the resulting performance levels are visualized. In Figure 5.10 the size of the resulting databases is shown. For the distance based pruning strategy, in Figure 5.11, the resulting recognition performance is visualized. In Figure 5.12, the amount of memory to store the resulting databases is shown. Figures 5.13 and 5.14 show the number of leafs in the vocabulary tree for both pruning strategies.

As can easily be seen, using the level based method the pruning only slightly influences the recognition performance, but has a major impact on the database size. Note that level based pruning is much more predictable in terms of database size and recognition performance. Moreover, as a comparison of the Figures 5.9 and 5.11 reveals, by choosing the level based pruning method over the distance based a much higher average recognition performance can be achieved. Another interpretation of the superior performance of the level based pruning method is that the tree is pruning uniformly in all branches. This means that the partitioning in feature space is made coarser while the amount of additional noise can be controlled very well. Distance based pruning in contrast merges individual branches without respect to information loss. The partitioning in densely populated regions of the feature space is made coarser at the loss of discriminability. A way to solve this problem would be the use of an entropy based measure to decide on pruning the tree. By doing so the loss in information could be better traded off against database compression.

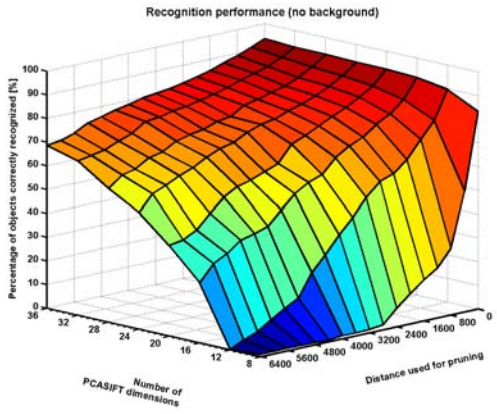


Figure 5.11: Average recognition performance for the distance based pruning method and different parameter settings.

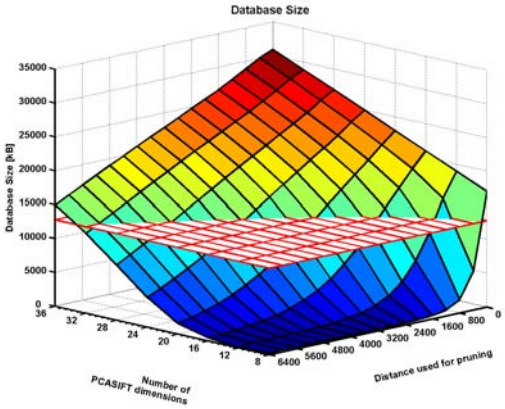


Figure 5.12: Database size for the distance based pruning strategy and different parameter settings. Again, the 12.5MB boundary is depicted as plane here.

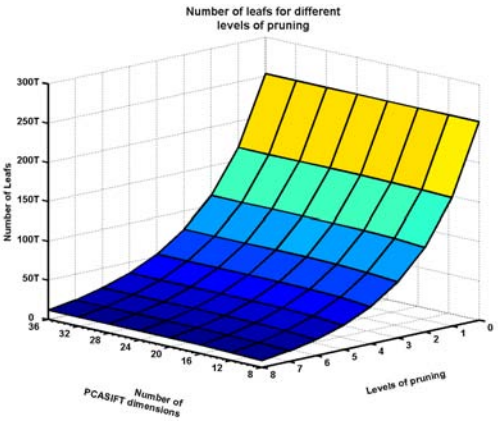


Figure 5.13: Number of leafs in the tree for the level based pruning method and different parameter settings.

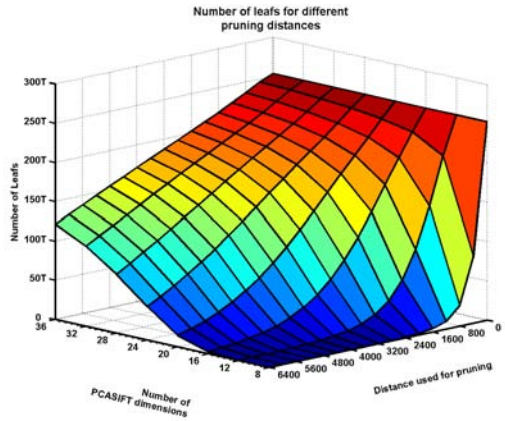


Figure 5.14: Number of leafs in the tree for the distance based pruning method and different parameter settings.

For the following experiments we choose the dimensionality of the descriptors to be 28 and the level based pruning method with a level of 2. By doing so we generate a database with about 12.1MB, still keeping an average recognition performance of about 90.6%. This setting is used to generate all following results.

5.3.4 Background Noise and Occlusion

To illustrate the robustness of the approach against background noise, we projected the object images onto different background images, which are shown in Figure 5.15. Some sample results of these projections are shown in Figure 5.16. As can easily be seen, some

Figure 5.15: The four background images onto which we have projected the objects to further challenge our recognition system.

Figure 5.16: Some sample projection results. The amount of background noise is severe, some of the objects itself occupy less than 20% of the total image area (352x288 pixels).

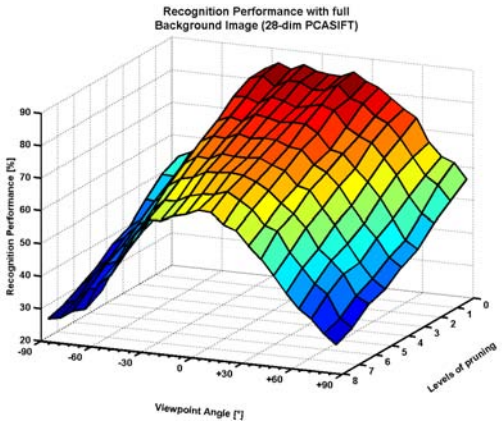


Figure 5.17: Average recognition performance on images with background noise for the 28-dim. PCASIFT descriptor and the level based pruning method.

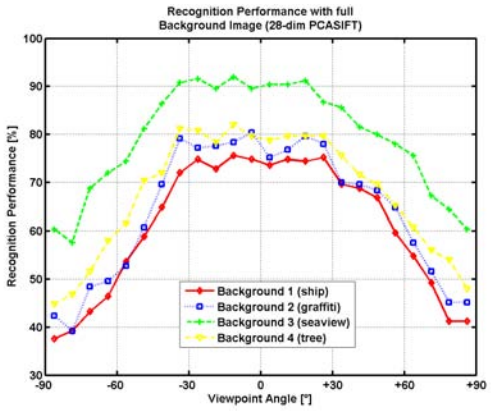


Figure 5.18: Average recognition performance for projections onto the four different background images for the settings chosen (28-dim. PCASIFT, pruning level 2).

of the objects are very small, thus they occupy less than 20% of the total image area. In Figure 5.17 the recognition performance on our background images for different levels of pruning for the complete viewpoint range are shown. In Figure 5.18, the recognition performance of our chosen setting for the four different background images is shown. It is easy to see that the approach performs best on the *seaview* image as most parts of the image are low textured. On all other images, our approach performs almost equally well.

The occlusion of an object was simulated by replacing parts of the object by additional background noise (as a rectangular bar from the middle of the object towards outside with

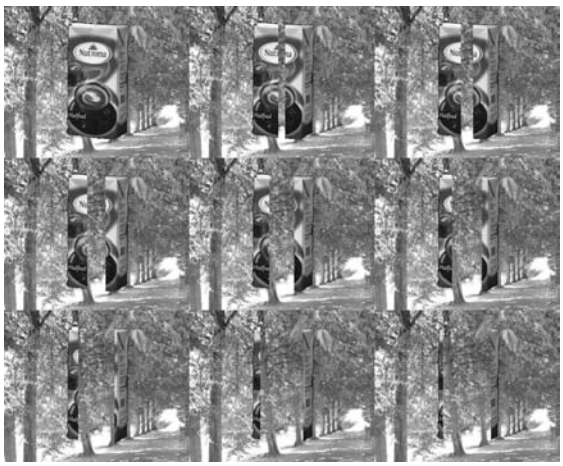


Figure 5.19: Different amounts of occlusion of object 847, starting with no occlusion (left upper image) to 80% (right lower image) in steps of 10%.

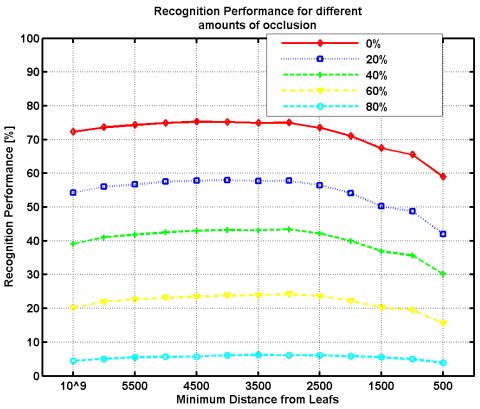


Figure 5.20: Average recognition performance vs. the distance threshold for different amounts of occlusion and our predefined settings (28-dim. PCASIFT, pruning level 2).

an increasing width as depicted in Figure 5.19). With the increasing amount of occlusion the recognition performance decreases almost linearly, as can be seen in Figure 5.20. Because features detected on the background do not accurately correspond to any leaf of the vocabulary tree the nearest neighbors of these descriptors found in the tree are only weak representatives. By thresholding the distance between the leaf descriptor and the descriptor in query we can discard these weakly represented features and thus reduce the noise level. In Figure 5.20 the recognition performance for different distance thresholds is depicted. As can be seen, a dedicated threshold can be used to increase the average performance by about 4% compared to using all descriptors without thresholding (denoted by a minimum distance value of 10^9 here).

5.3.5 Adding and Removing Object Representations

By now we have only evaluated our approach based on a static vocabulary tree where the full object database was available from the beginning. However, as such a system is very inflexible we test how well our methods can perform if only parts of the data are available during training and others are added at runtime.

First, we built a vocabulary tree given keypoints and descriptors from the first 100 objects. After building the tree, the indexing of the descriptors is performed on these 100 objects. Then we added up to 150 additional objects where the descriptors were not used for tree creation but were indexed only. In Figures 5.21 and 5.22 the results for different numbers of objects over the full viewpoint range are shown. Note that using 28-dimensional descriptors and a pruning of level one the database containing 250 objects has a size of about 10.0*MB*. The difference in size compared to section 5.3.4 results from

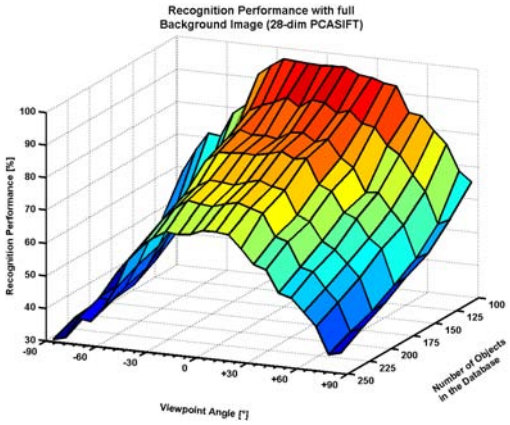


Figure 5.21: Average recognition performance on images with background noise for the 28-dim. PCASIFT descriptor with level 1 pruning and different numbers of objects in the database.

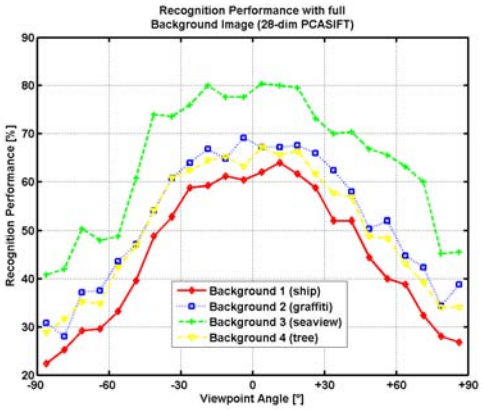


Figure 5.22: Recognition performance for projections onto the four different background images for 250 objects in total over the complete viewpoint range.

the different vocabulary trees trained on 250 objects and on 100 objects respectively. As can be seen, the recognition performance is considerably lower due to the much coarser partitioning in feature space and the inaccurate nearest neighbor assignment during indexing. Nevertheless, a very important advantage of this system property is that the database content can be altered at runtime, meaning that objects exhibiting a moderate amount of texture (and thus an adequate number of keypoints and descriptors) can be added to a fixed recognition system at runtime. Thus it is possible to tune a system to a new set of objects up to a given degree, even if the build process has long time passed and the vocabulary tree structure has been fixed once and forever. For further evaluations and extensions to this system features, the reader is referred to the work of Ober *et al.* [168].

5.3.6 Timing Results and Memory Profile

To test the final performance of our algorithm on our platform we have measured the average time consumption of each individual step and evaluated the amount of memory spent on each task. We have divided the approach into several subsections which are listed in Table 5.1. The *scale space generation* step, consisting of *image filtering* and *image subtraction*, takes a constant amount of computation time as there is no dependency on the data being processed. All other steps of the approach are dependent on the number of DoG points found in the *minima/maxima search* and updated in the *orientation assignment* step. The timing results for the *descriptor calculation* and the *vocabulary tree query* step are computed, given an average detection rate of 100 DoG points. Note that a detection rate of 50-200 points is reasonable (referring to Appendix B where Figure B.1 shows the average number of DoG points detected per angle over the complete viewpoint range.) The high standard deviation in the *orientation assignment* is due to the possibility that multiple

Algorithm	Avg.Time [ms]	Std.Dev.
Scale Space Generation	35.78	0.014
Minima/Maxima Search	35.07	17.18
Orientation Assignment	107.75	98.56
Descriptor Calculation	75.59	11.40
Vocabulary Tree Query	3.62	1.14
Total:	257.82	127.73

Table 5.1: Timing results for the individual algorithmic parts of our approach. The *scale space generation* step can also described as combination of *image filtering* and *image subtraction*.

Algorithm	Memory Consumption [kB]
Scale Space	1,386
PCA Transformation Matrices	219
Final Descriptors	2.7
Vocabulary Tree	12,471

Table 5.2: Memory consumption of the individual algorithmic steps. The size of the data buffer holding the final descriptors is based on the 28-dimensional descriptor used in our setup and a detection rate of 100 descriptors.

keypoints might be created or discarded and thus the time for assigning the orientation varies drastically. Based on the detection of about 100 DoG points, the algorithm can process 4 frames per second. As most of the parts of the algorithm have no fixed execution time, it is hard to estimate the system timing performance under real conditions. One way of predicting the worst case execution time is to limit the number of keypoints allowed to be detected. By placing an upper limit, say 250 keypoints, we can guarantee a worst case execution time be calculated to $500ms$, which is 2 frames per second. Limiting the number of keypoints can be performed by putting a threshold on the DoG response and selecting the 250 keypoints having the highest DoG response.

A severe drawback of simply thresholding is that one cannot guarantee the resulting set of keypoints and descriptors to capture the appearance of objects well, given a complex background. In other words, a strategy should be used, such that keypoints are selected from all image regions with equal probability, to avoid local clustering of keypoints in highly textured regions. For example, the *Adaptive Non-Maximal Suppression* (ANMS) strategy proposed by Brown *et al.*, which was already mentioned previously, can be used to select a fixed number of interest points from images, spaced equally over the entire image [37]. For reasons of computational efficiency, in our current approach we left this as an open issue for optimization.

In Table 5.2 the size of the individual memory buffers is listed. Due to the fixed spacing in the scale space and the fixed number of octaves, the *scale space* takes a fixed amount of $1386kB$. The size of the data buffers holding the transformation matrices for the PCASIFT descriptor takes about $219kB$. The amount of memory needed to store the

descriptors increases linearly with their number. The size of the memory buffer holding the vocabulary tree is determined by the parameters chosen during the tree construction. The size of the data buffers for our tree is about 12.1MB.

We have implemented almost all parts of the approach in fixed point as this is basically necessary for algorithms to perform in acceptable time on our platform. Note that we did not write any parts of the algorithms in assembler code or made any extensive use of other optimization techniques like intrinsics.

5.3.7 Calculation Accuracy

As almost all parts of our algorithm are fixed-point versions, we want to investigate how much performance is lost due to this approximations of all calculations compared to floating-point version. For these tests we used the first and second image of the *Graffiti* scene from Mikolajczyk [159], starting with an initial image size of 352x288 pixels. For a more extensive evaluation, we chose 800 random natural images to get more exact statistics.

Using our 16-bit fixed point filters we calculated the root mean square Signal-To-Noise ratio (SNR_{rms}) to floating point filtered images. The evaluation on 800 natural images revealed that on average the SNR starts from 13.77 dB and decreases about 0.27 dB per filtering operation, down to about 9.57 dB in our case for the last image in the last octave.

The filtered images are not used directly to detect keypoints, but their differences are used. Now the question arises how much this approximation influences the extrema search process and if there is any loss in keypoint localization accuracy. In Figure 5.23 the results for both versions of the algorithm on both images are shown. The number of points found using the floating point is 294 and 276, while the number is only 290 and 273 for the fixed point version. This means that due to the reduced accuracy some points are simply lost (especially keypoints which are extremas of marginally low hills or shallow sinks in the scale space). Again, based on our evaluation on 800 natural images we detected 59,062 and 58,769 points, thus we can estimate that a negligible number of 0.45 % of all points get lost due to the fixed point approximations.

Overlap Error	10.0	20.0	30.0	40.0	50.0	60.0
Repeatability (float)	14.7	50.0	67.6	73.5	82.4	82.4
Repeatability (fixed)	14.7	52.9	67.6	73.5	82.4	82.4

Table 5.3: Repeatability for both versions of the keypoint detector.

Treating the keypoints found by the floating point unit as the *true* keypoints, a minimal localization error of $0.16 \cdot 10^{-3}$ and $0.66 \cdot 10^{-3}$ pixels is introduced in x and y direction in the fixed point version. To allow for a fair comparison between different interest region detectors, Mikolajczyk [159] defined some general criteria. The main goal is to measure, to what extent detected regions overlap exactly the same scene area, given the results of a

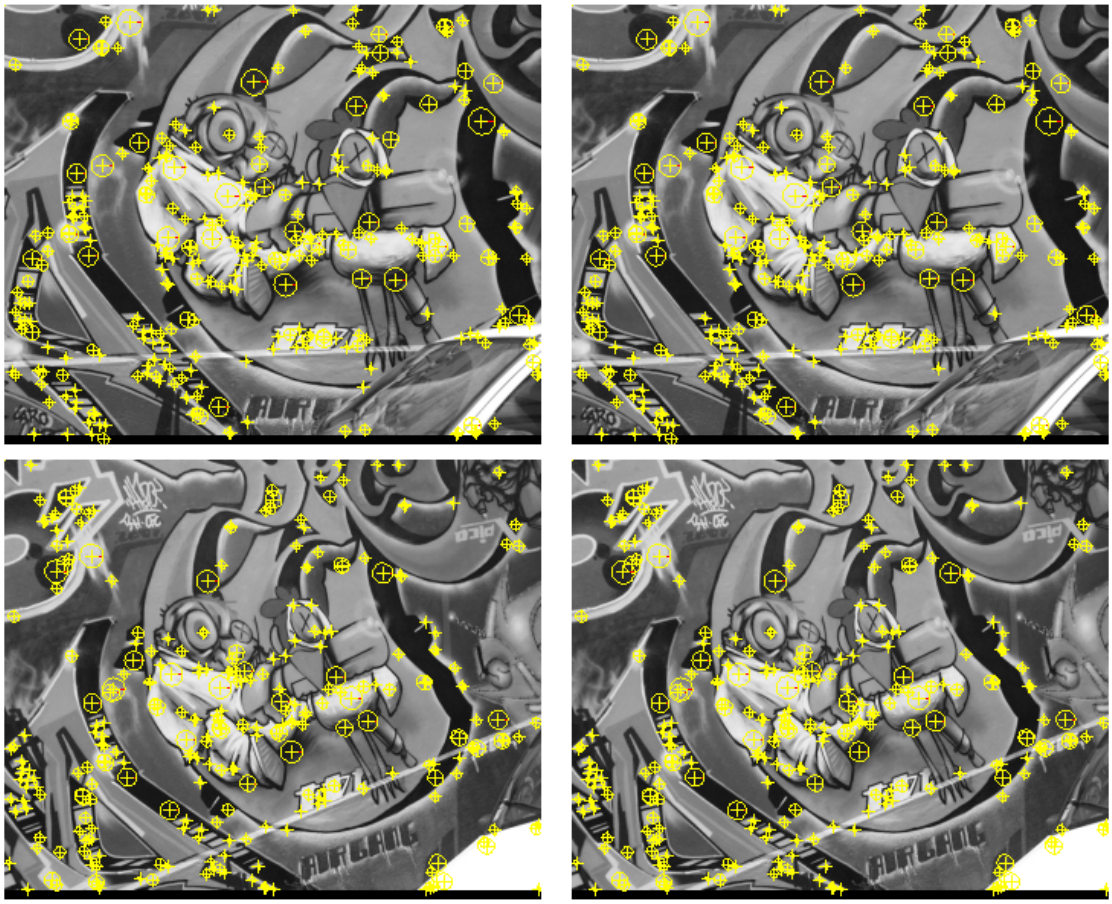


Figure 5.23: Detection results for the first two images of the (resized) *Graffiti* sequence.

detector on two images of a planar scene taken from different viewpoints. In this respect, the *overlap percentage* is defined to be the amount of overlap between detected interest regions, which are projected back and forth from one image into the other by a known Homography. The *overlap error* defines the discrepancy, respectively. The *repeatability* is defined to be the average number (or percentage) of corresponding regions detected in both images, *i.e.* the number of regions with an overlap error smaller than a given threshold. In our experiment, we calculated the repeatability values for both versions of algorithms. As can be seen in Table 5.3, there is no loss in repeatability.

As the descriptor calculation is also performed in fixed point, the question arises how much the calculation accuracy influences the descriptor performance. In our 800 image database, for all points that are detected coevally in both versions we calculate 36-dimensional PCASIFT descriptors, again in floating point and in fixed point, and normalize them. On average the mean squared error (MSE) is about $8.8134 \cdot 10^{-4}$ which is negligible.

Algorithm	float (wo. fastrts.lib)	float (w. fastrts.lib)	fixed
Filtering (5x5)	130,038,113	54,586,734	1,091,749
Descriptor Calculation	16,491,804	5,991,365	166,414

Table 5.4: Cycle count for two algorithmic parts of our approach.

5.3.8 Performance Comparison

Here we shortly compare several individual parts of our algorithms calculated in fixed point and in floating point on our hardware regarding calculation cycles. Due to the complexity of the approach it is not possible to compare the approach as a whole. However, just comparing the calculation times for a simple filtering operation and a single descriptor calculation impressively shows the superiority of the fixed point approach on the hardware platform used (see Table 5.4).

We have evaluated the floating point approach with the `fastrts` library, which can be used for enfastened floating point calculations, enabled and disabled. As can be seen the fixed point approach is about 119x/50x faster in the filtering and about 99x/36x faster in the case of the descriptor calculation. Although the use of the library cuts floating point calculation time by a considerable factor, the advantage is only tiny compared to the striking speedup using fixed point calculations. It also clearly illustrates the suitability of fixed point arithmetic for DSPs without hardware floating point support.

5.3.9 Optimization Issues

Concerning different optimization strategies, all parts of the approach can definitely be optimized to make better use of the hardware resources available. This can mainly be achieved by using intrinsics and rewriting critical parts in assembler code. However, the biggest amount of optimization potential lies in the algorithm itself.

First the approach is not optimized to memory swapping issues. A lot of external memory accesses decreases the overall performance. Moreover most parts of the algorithms have no fixed execution time. One workaround for this could be an upper threshold on the number of keypoints to be detected. Another possibility is to perform the descriptor calculation and nearest neighbor search interchangingly. By doing so, and by putting a threshold on the ever-changing voting histogram, one can calculate a likelihood ratio for each object in the database and can interrupt the calculations if a desired confidence rate is reached.

Concerning the vocabulary tree structure, the indexing strategy for finding the nearest neighbors is suboptimal in terms of memory accesses. Though the matching path of a single descriptor through the tree is not predictable in advance, a well-engineered data swapping strategy has to be implemented. Reorganizing the nearest-neighbor search to minimize external cache accesses can definitely speed up the search by a considerable factor.

5.4 Vehicle Reacquisition

As has been shown in the previous section, state-of-the-art object recognition approaches can be successfully deployed on current smart camera platforms, as long as some general aspects of embedded systems are considered during development. Now we want to prove the suitability of the approach for object recognition in one task of surveillance.

Vehicle reacquisition is an interesting topic in the area of traffic monitoring. In the following, we present an application of our algorithm for object reacquisition in smart camera networks. First, we apply the algorithms described earlier to form a very efficient object representation, the so called *signature*. Then we demonstrate, how this representation can be efficiently communicated throughout a network of smart cameras. At other camera locations, objects are again described and a specific signature for the object is created. The matching algorithm compares object signatures and allows for efficient reacquisition and tracking of the objects through entire camera networks.

One nice feature of this approach is that our algorithms minimize the amount of information per object to be transmitted between adjacent camera nodes. Another benefit of this setup is, that it has not to be re-trained for every single camera view. Additionally, one major goal of our work was to use uncalibrated setups since multi-camera calibration is still a tedious task. Finally, an important property of our system is that it runs fully autonomous coevally fulfilling real-time demands. The power of our approach is shown on the simulation of several traffic scenarios; first, a two-camera setup is considered and then a medium-scale camera network is simulated. The results presented proof our concept useful and motivate further research in the area of local features for object fingerprinting on embedded systems.

5.4.1 Related Studies

Before we start describing our approach in detail, we give a short review of existing methods in the field of vehicle reacquisition. We emphasize that most of the algorithms in this field come from the area of computer vision and do not take any embedded system related issues into account. Thus, in contrast to the work presented here, communication constraints or computational issues are usually hardly considered, since all necessary computations are performed on a centralized engine. Additionally, most algorithms are designed for aerial imagery, thus capturing larger areas but, on the other hand, due to their lower resolutions are not able to benefit from the usage of local features, which we propose in this context.

In literature, many work concentrates on object instance recognition and fingerprinting, mostly for vehicles. For example, Guo *et al.* [89, 90] address the problem of matching vehicles across multiple sightings. They extract multiple features, *e.g.* line segments, of poor quality aerial images and hold these features in an integrated matching framework. In one of most recent works Ali *et al.* [6] use both motion and appearance contexts for

tracking of vehicles in aerial images. In contrast to Guo *et al.* they use a clustering scheme based on the Lyapunov Characteristic Exponent (LCE) to learn the motion context of multiple trajectories. Additionally, they use the motion of a car to interpret the behavior of neighbored cars. By using appearance information they are, furthermore, able to handle occlusions.

Coifman *et al.* [49] tackle the problem of single loop vehicle reidentification in order to reliably deviate the traffic flow as well as travel time data. Oh *et al.* [169] use a Bayesian approach to identify vehicles in freeway traffic. The probability of identity is derived from physical observations and events, *i.e.* trajectories of vehicles, and can be improved online. Additionally, they are able to derive appearance probabilities for each object. Huang and Russell [104] use a probabilistic approach to reidentify vehicles on a traffic highway. Color appearance and transition times are modeled using Gaussian distributions. Kettnaker and Zabith [119] use a Bayesian formalization to track persons over multiple non-overlapping cameras. Yet, their system has to be calibrated and the number of possible objects has to be known.

Shan *et al.* in [214] present a system capable of reidentifying cars between two non-overlapping cameras formulated as same-different classification problem without direct feature matching. Their system, however, builds on a SVM classifier which has to be trained and uses edge-based object matching which does not achieve comparable discrimination rates compared to, e.g, DoG or MSER based approaches. In the work of Ram [191], DoG keypoints and MSERs were detected, and SIFT features were used to match vehicle images across different camera locations. Similarly, Weber used the *Multi-Modal Neighborhood Signature* (MNS) to find representative color features for matching vehicles in low-resolution images [257]. Sun *et al.* [224] perform vehicle reidentification using a multidetector fusion approach. While detection is performed using a nearest neighbor classifier and a linear fusion strategy, the features are based on object color and inductive loop information.

Our proposed system differs from the former mentioned in (i) that the entire system is designed and implemented in order to run on resource constrained embedded systems, (ii) we successfully and solely employ local features, as described earlier, for object reidentification, (iii) we concentrate our work on minimizing communication costs rather than local processing, (iv) we avoid the necessity of tedious learning and (v) our approach works on spatially separated, uncalibrated, non-overlapping cameras, and finally (vi), no global optimization has to be performed.

5.4.2 Object Fingerprinting and Reacquisition

As previously mentioned, the entire object fingerprinting and re-identification approach used here is solely based on local appearance features, as described in section 5.2. Hereafter, the application of our approach to build a compact object representation is introduced.

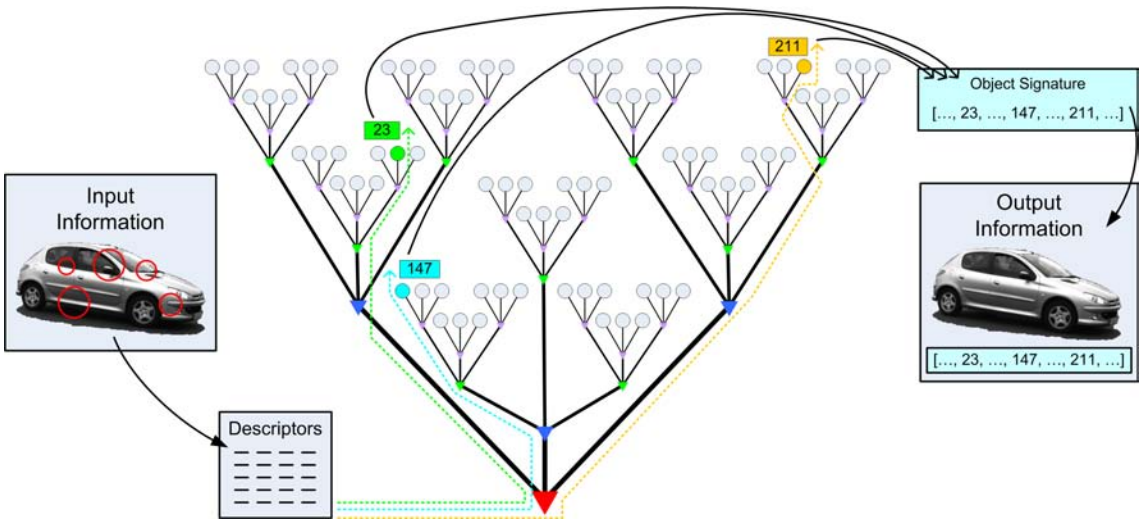


Figure 5.24: Illustration of the object signature generation mechanism. After detection of keypoints and descriptor calculation, the leaves representing the nearest neighbors are determined. The unique indices of the leaves are sorted and stored as object signature.

5.4.2.1 Object Signature and Signature Matching

First, we build a vocabulary tree from a set of descriptors. It is important, that this set stems from images containing vehicles, or from some sample backgrounds of possible street scenes. This is relevant, since we want to tune the size of the vocabulary tree to fit our embedded resources. Descriptors, hence, should be sampled from the space of potential descriptors, which also can be understood as a try to build a more scene specific visual alphabet.

The vocabulary tree contains a fixed number l leaves which are numbered in increasing order and (hopefully) populate the feature space as desired. For generating an object signature, the keypoints are detected on the object image and corresponding descriptors are calculated. For each of the t descriptors extracted, the nearest neighbor in feature space is determined by traversing the vocabulary tree from the root downwards to the leaves. The unique IDs of the corresponding leaves representing the nearest neighbors are increasingly sorted to form the final object signature (which is in fact a vector of length t numbers). This vector is a maximally compact representation given a fixed vocabulary tree (see Figure 5.24 for illustration).

For matching of individual object signatures, the number of identical elements has to be determined. Though the object signatures are relatively short vectors (typically in the order of a few hundred elements) and can be matched very fast, sorting of the elements in increasing order still simplifies the algorithm complexity and makes more efficient matching possible.

5.4.2.2 Removal of Insufficiently Represented Features

Although sample images from the object domain are used for the construction of the vocabulary tree, populating the whole feature space equally well is almost impossible. Especially features from background noise or objects not belonging to the object category often can not be represented well by any leaf of the tree since they populate different areas of the feature space. We can leverage this to early discard features from being included into an object signature. If the Euclidean distance, or the Sum-of-Squared distances (SSD), between a feature and its nearest leaf is above a given threshold θ_{dist}, we simply treat it as noise and eliminate it. By doing so, we can guarantee, that only these features are used to build a signature, which are not likely to change their affiliation to a specific leaf when being extracted from different object views.

5.4.3 Camera Network Setup and Framework Overview

It is important to note, that we focus on a network of camera nodes in this application of our approach. All design considerations are driven by the fact, that the entire system has to be used in typical outdoor traffic scenarios, as well as under harsh environmental conditions.

5.4.3.1 Camera Network Setup

In order to ensure an easy and scalable setup, our proposed network architecture is hierarchical organized in camera groups. A group simply consists of one or more single cameras and is defined as a set of neighboring video sensors. Each camera itself is uniquely identified by its group and camera ID, respectively. A complete camera network is temporally synchronized using NTP, constituting the only dependency of our system to some kind of local coordinator. Two types of communication paths are possible, as illustrated in Figure 5.30 showing our multi-camera simulation setup, namely *intra-group* and *inter-group* communication. For each camera, the neighboring cameras are defined as internal neighbors if they are in the same group, or external neighbors if they belong to a different group.

For each new object passing a camera an object signature is created and multicasted together with a timestamp, a preliminary empty history, and the camera and group ID to all neighbors defined. The signature is also stored together with a predefined timeout in a temporary output buffer for later acknowledgment. All receiving cameras store the incoming message in a dedicated buffer. If objects are passing the individual node, a signature is created and compared to all available signature messages in the buffer. The matching score is evaluated against a special threshold. If it is lower than the threshold it is considered as "new" (object having entered the scenario in between two neighboring camera nodes), and processing proceeds as described above. If it is higher than the threshold, the car is reidentified, the old signature is exchanged by the new one, the history is updated and the message is multicasted to this cameras neighbors again. Furthermore,

the camera which had delivered the signature before is notified that the object has been reidentified.

For overall surveillance purposes two types of notifications to a supervisor might occur. Either a message is created if the object in the temporary output buffer was not acknowledged within the given timeout. This means that an object might has left the scene in between two neighboring cameras. Another message is created if an object is passing a boundary camera node having no more neighbors. Anyway, both types of messages passed to the supervisor contain the complete tracking history of the object making statistical flow analysis on a global level possible.

5.4.4 Evaluation

In this section we evaluate our approach in a real-world traffic scenario. First we describe the way how we collected images for our algorithm evaluations. Then we examine our methods on two different setups and give detailed results. A discussion of the results and further notes conclude the section. Note, that all calculations have been simulated under MATLAB^{TM}and finally been examined in the context of our hardware platform. We have not built a real network of smart cameras due to the high administrative effort. However, the temporal behaviour and the memory profile of most parts of our approach have already been evaluated in the previous section 5.3 and remain valid in this context.

Figure 5.25: Projection of various cars onto different backgrounds. We simulate the various levels of background noise by cropping the vehicle out of the images again with a varying border added around the object.

5.4.4.1 Data Acquisition and Database Creation

A well-known problem with the evaluation of algorithms for sensor networks is the lack of training and test data, respectively images in our case. Clearly, collecting data from a multi-node sensor network is difficult due to time synchronization problems. Moreover, the

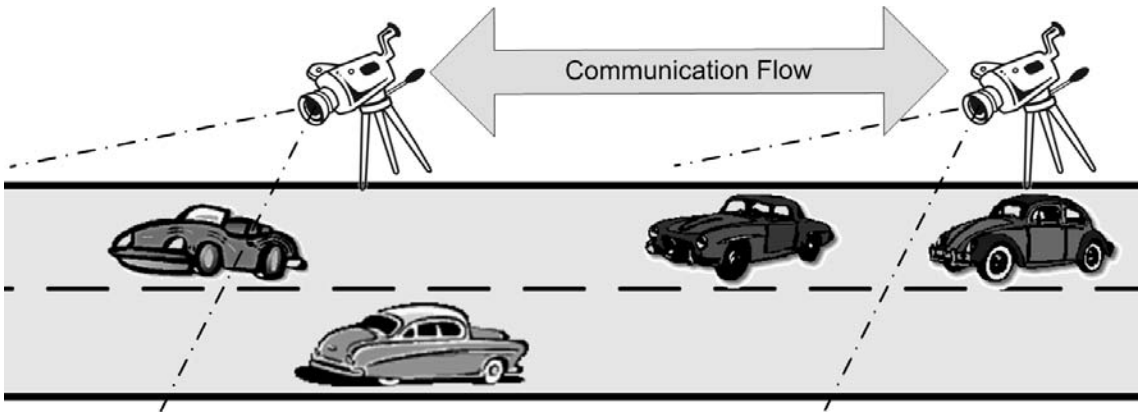


Figure 5.26: 2-Camera setup. Vehicle signatures are communicated between the cameras.

amount of data to be stored and the lack of automatic video annotation tools complicate the creation of a useful database.

We cope with the problem by applying several tricks from the area of computer vision to collect a set of pictures with two cameras only, which we can still use to test our algorithms under maximally realistic conditions for an entire camera network.

Precisely, we recorded 171 vehicles with two different cameras in a roundabout, in order to get maximal viewpoint changes, and then segmented the objects using a simple background modeling. Note that the total viewpoint change between images of the individual vehicles is up to $\pm 30°$. This is a reasonable scenario for the application of the DoG keypoint detector, but for larger viewpoint changes other methods have to be used [159, 157]. Additionally, we took 29 pictures of many different urban traffic scenarios. Having this basic data setup, we projected the segmented objects to the different backgrounds and varied the ratio between background and foreground object, thus, allowing for arbitrary reidentification simulation under various amounts of background noise and various viewpoints (see Figure 5.25 for illustration). For simplicity we assume, that in a practical setup a preliminary detection result is available, such as provided by the detection approach described in the previous chapter. In other words, the re-identification algorithm is only applied, if an object is present in the image and the major part of an image is covered by only this single object in query. Partial occlusions or parts of other objects are sufficiently well simulated by background noise.

The vocabulary tree used for our evaluation contains about 43.000 leaves, which is equivalent to about 5.1 MB of memory needed for storage. The tree was purely built from images of street crossings and vehicle images from an other database. We did not prepare the tree in any additional way, say, we did not employ any leaf pruning strategy. The high performance achieved, as shown in the following evaluations, is a clear indication, that

Method	Data to transmit	Recognition Performance
SIFT keys	19.060,8 kB	90,64 %
PCA-SIFT keys	5.360,8 kB	82,11 %
Object Signatures (no removal)	148,9 kB	87,60 %
Object Signatures (removal)	**49,5 kB**	**88,18 %**

Table 5.5: The amounts of data to be transmitted between individual nodes and the recognition performances for our two-camera setup. We assume that all information is encoded in *integer* or *float* units with 4 bytes each.

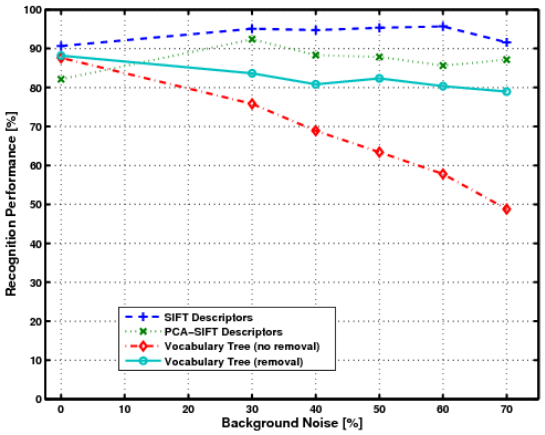


Figure 5.27: Recognition performance for different levels of background noise. For example, 50% background noise indicates that the object only covers half of the total image area.

even a more restrictive memory usage policy still leads to adequate recognition accuracy and satisfying system performance.

5.4.4.2 Two-Camera setup

In our first experiment we simulate a camera network consisting of two single cameras, as depicted in Figure 5.26. For simplicity we assume that vehicles are simply passing the scenario in random order without leaving or entering the scenario in between the cameras. Furthermore, we simulate traffic in one direction of the road only, so cars are not allowed to turn around. Our simulation runs in a 10000 *ticks* long time loop and the vehicles pass the first camera in random order and random intervals. Each vehicle has 500 *ticks* at max to pass the second camera. Thus, the buffer timeout of the cameras is set to 500 *ticks*. To highlight the benefits of our approach we compare it with a reacquisition system based on pure matching of SIFT or PCA-SIFT descriptors.

In Figure 5.27, the dependency of the recognition performance on the additional amount of background noise is depicted. To illustrate the importance of removing inadequately represented features, we have evaluated our approach for both strategies, with and without removal. In the first case the recognition performance significantly drops as the influence of noise on the signature generation increases. In the latter case the recognition performance only slightly decreases. This indicates that a rough object segmentation, together with our removal strategy, is sufficient to allow for satisfying performance. Note that we have not evaluated our approach for noise levels below 30% because using a simple bounding box around the segmented car already includes at least 25% of background clutter. Note, that the results for a significant amount of background noise agree with our previous results from section 5.3.4.

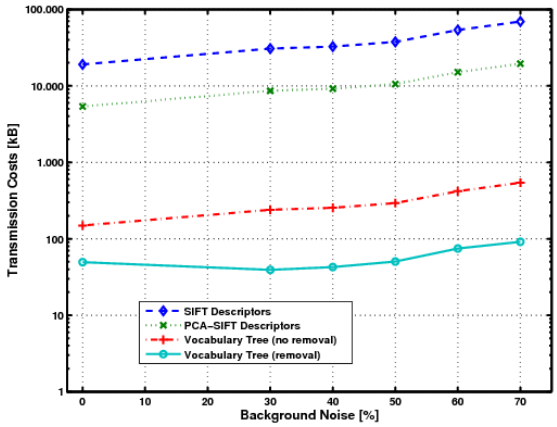


Figure 5.28: Total Transmission Costs for different levels of background noise. As the difference in the amount of data is that severe, logarithmic scaling is chosen.

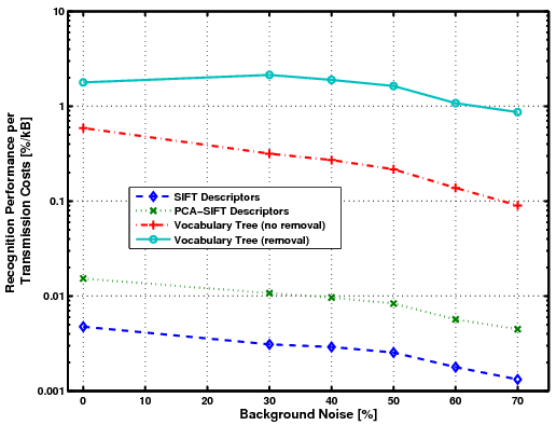


Figure 5.29: Information Content for different levels of background noise. While the amount of information contained in a single transmission unit ([kB]) is low in matching-based approaches, it is high in our tree-based approach.

In Table 5.5 the amount of data, together with the recognition performance achieved for different types of strategies for fully segmented vehicles (no background noise), is summarized. While our strategy already reduces the amount of data to be sent by a dramatic factor, removing noisy features once again cuts the transmission costs by 2/3. As can easily be seen, our setup achieves satisfying results, coevally minimizing the amount of data to be transmitted between individual camera nodes. In Figure 5.28, the amount of data to be transmitted for various levels of background noise is depicted. While the transmission costs for our tree-based approach only slightly increase, the transmission costs for matching-based approaches explodes, as there is no mechanism to early discard bad features. Figure 5.29 shows the amount of information contained in a single transmission unit ([kB]). It is easy to see that our approach nicely compresses the information necessary, while a high amount of data with low information content is transmitted in matching-based approaches. Note that our vocabulary tree based method performs more than two powers of ten better than the original approach based on PCA-SIFT or SIFT feature matching.

5.4.4.3 Multi-Camera Setup

For this experiment the camera network setup is as depicted in Figure 5.30. The conditions for vehicle movements are the same as described in the previous section, but additionally the paths of the cars through the scenario are determined randomly. For ease of illustration we allowed all cars in either case solely to enter the scenario from one group. However, note that we achieved similar performance rates with arbitrary entry groups.

In Figure 5.31 we have depicted the traffic flows for our setup based on a random

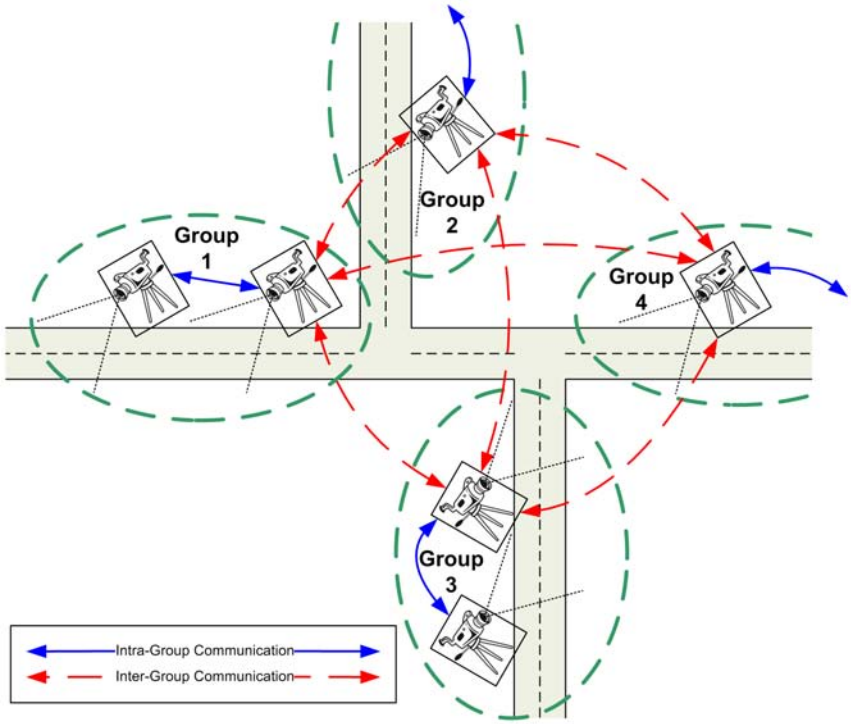


Figure 5.30: Multi-Camera setup. Intra-group communication of signatures and inter-group communication occurs between separate camera groups.

traffic setup and with four different entry groups. Though recognition rates in each case depend on the entering group, for all four starting conditions our framework achieved at least *87%* of successful vehicle tracking through the entire scenario. Note that, although compared to the two-camera setup one might expect a significant worse recognition rate, the relative high performance comes also from the fact that due to the higher amount of cameras, each camera has to handle a smaller signature message buffer.

5.5 Concluding Notes

In the previous experiments, we showed, that state-of-the-art object recognition methods can be successfully deployed on DSP-based embedded systems. It is essential, that the development of algorithms is strongly hardware oriented, such that a recognition system is real-time capable and fulfills the predefined criteria in terms of accuracy. As we were interested in a principal evaluation of current methods on embedded hardware, only a premature implementation of the methods exist. However, in the context of our vehicle reacquisition applications, the experimental evaluation has shown, that already this system would be able to work at about 3 frames per second on an image

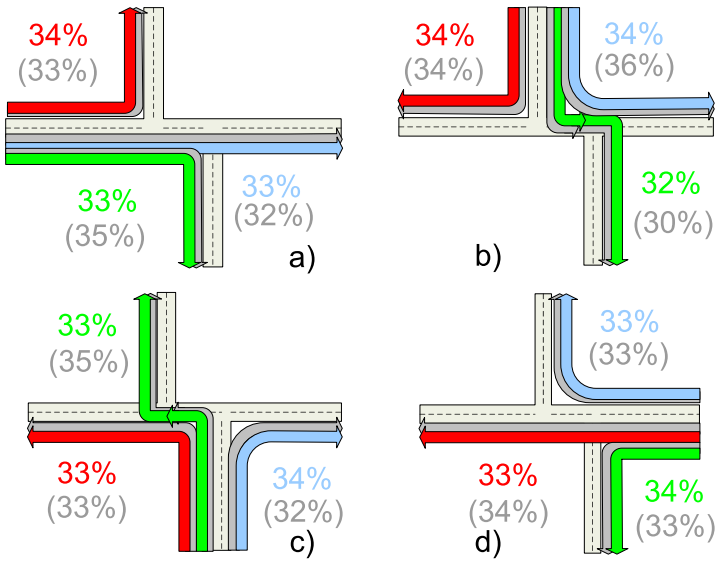


Figure 5.31: Four different charts indicating the measured traffic flow in percent. In each illustration a different entry group is chosen. The groundtruth is illustrated in gray.

stream of 352x288 pixel resolution. This is equivalent to an average traffic flow of about 10000 cars per hour. Needless to say, that a more in-depth optimization of the individual algorithm parts would result an a more performant system with higher throughput.

An important property of our approach is that it nicely scales with the number of objects. Because our approach is based on highly distinctive local features, a fixed vocabulary can be used to build up representations for several hundred objects. Furthermore, applying the algorithm in the application proposed, the communication effort between individual camera nodes can be reduced drastically, without a remarkable loss in reacquisition performance. As communication costs are an important issue in the development of software for entire sensor networks, we strongly emphasize, that our algorithm is a valuable option for setting up a recognition - or reacquisition - system in a smart camera network of larger scale.

One of our main findings is, that it is essential to select suitable algorithms for building a recognition framework, given the properties of embedded hardware. Algorithms should consist of a set of operations, which can be executed efficiently on DSPs. In this respect, filter-based algorithms and methods mainly consisting of MAC instructions are very suitable. By choosing algorithms of that type, drawbacks of embedded systems in terms of computational power can be turned into advantages, given an architecture especially optimized for parallelism and fast and recurring computations. Another important conclusion is, that it is inevitable to find ways to cope with limited memory resources. We showed, that even a moderate amount of storage

capacity is sufficient to perform high-level vision algorithms on smart cameras. To achieve this goal, we reduced the amount of memory needed at many different points of the approach, finally arriving at a finely tunable recognition framework. Hence, we can summarize, that, beside the selection of suitable algorithms, also the reduction in memory requirements plays a vital role in algorithm development for embedded systems.

The need for object recognition capabilities, especially for mobile and portable devices, is evident. Features, like the incremental adding and removing of objects at runtime, bring visual object recognition closer to visionary applications in real world scenarios. For example, in supermarkets autonomous shopping carts can be used, which collect items autonomously given a shopping list. In household robotics autonomously cooking robots can be used to prepare dinner, while everybody is at work. The area of recognition is large, and we will see a lot of applications being introduced in the next few years. We are certain, that local feature based recognition technology is a powerful tool, which will also proof valuable on embedded systems in the near future.

People do not like to think. If one thinks, one must reach conclusions. Conclusions are not always pleasant.

Helen Keller
US blind & deaf educator, 1880 - 1968

Chapter 6

Conclusion

In this thesis, we were concerned with the current state-of-the-art in object detection and object recognition for visual surveillance on embedded systems. We discussed several aspects and typical problems in detail, and proposed new approaches especially addressing and tackling the limitations appointed by current hardware platforms. We described the state-of-the-art in smart camera development, and discussed several issues of hardware related algorithm realization. We referred to a large number of related publications, which deal with the topic of object detection and object recognition in general, and also focus on the issues of approaches involved with the research in the area of embedded systems. On this account, we will summarize our results in object detection and object recognition, and discuss several modifications in the following.

6.1 Discussion

Following the introduction and motivation of our work on visual surveillance for smart cameras in Chapter 1, we described current state-of-the-art smart camera platforms in Chapter 2. We summarized the most important properties of embedded systems, and also discussed the aspects of software development for smart sensors. We pinpointed issues like limited memory and computational resources, or the need for autonomous processing and minimized communication effort to server nodes. In Chapter 3, we gave a rather comprehensive overview about current research in the area of object detection and recognition. We listed the most important approaches from computer vision research, and also listed numerous methods proposed especially for embedded platforms.

For object detection, we have presented a flexible framework in Chapter 4, which is able to perform in real time, without the need for relying on any presegmentation of the object. We vote for several adaptations and modifications, such that object detection can be realized in a very compact and efficient fashion, achieving highly accurate results in real time. We showed, that it is essential to reduce the computational complexity of a detector to allow for acceptable performance in terms of speed. In other words, on the algorithmic

level, the number of weak classifiers, and coevally the size of the detector, should be reduced to the best possible, which allows for a considerable speedup without a nameable loss in detection accuracy. Moreover, we demonstrated, that considerable improvements are achieved by considering the suitability of DSP-based systems for particular types of parallelism, like instruction level and operation level parallelism. We evaluated our approach on three different datasets and benchmarked several of the modifications and adaptations. It is worth to note, that the results achieved suggest the deployment of our approach in a larger framework for visual surveillance, for example in the area of traffic monitoring.

Similarly, in Chapter 5 we proposed the deployment of a state-of-the-art object recognition setup on our prototypical DSP-based platform. We have used the DoG and SIFT algorithms for finding distinctive local features on objects and for building descriptors. A visual vocabulary tree is used to organize object representations by collections of local features in an efficient way. By using dedicated compression mechanisms, we demonstrated the plausibility of our approach on a moderate size database, allowing for operation at real time speeds and achieving remarkable recognition performance. As in object detection, our experiments showed, that fixed-point arithmetic leads to a significant speedup of our approach, without sacrifice of accuracy. We evaluated our algorithm in the context of a real world application, namely vehicle reacquisition on residential streets. We simulated a small camera network, and demonstrated, that the communication effort between adjacent camera nodes can be minimized, using our algorithm for building a compact representation of passing vehicles. The results achieved encourage a further investigation of the algorithm in a large-scale camera network and further motivate work on local features for object recognition in robotics or navigation on embedded smart cameras.

6.2 Outlook

When developing the approaches proposed in this thesis, one goal was to include the newest accomplishments in research on embedded systems and computer vision. In this respect, the most recent algorithms were considered as base for our own development. However, a few further improvements and investigations in both fields, object detection and object recognition, are recommended, to achieve higher robustness and higher performance in terms of speed.

In object detection, we have used the standard Viola-Jones algorithm as an initial step for real-time object detection on our prototypical DSP-based platform [250]. It was shown, that a massive reduction in the number of weak classifiers is necessary, to allow for real-time performance in a practical application of the approach. On this account, we have proposed *Inter-Stage Feature Propagation*, originally developed by Šochman and Matas [252], which has turned out to achieve a considerable improvement in detector compactness and speedup over the original approach. However, for further improving the detector we propose the investigation of *WaldBoost*, which was introduced by Šochman

and Matas [253]. *Waldboost* is the most suitable algorithm, generating the most compact, and thus fastest, detector currently available by reducing the number of features to one in each cascade. Related to product development, an important issue is the reuse of detectors across different scenarios. Adaptability of an existing detector to different scenarios is a key feature to allow for deployment of detectors in large scale. A closer investigation of the performance loss or the degree of reusability, given a fixed set of training examples, would be of interest. In the current approach, we have not considered multiple detectors for different types of objects at the same time, *e.g.* detectors for cars and trucks. In fact, the major problem is the need for hypothesising, given multiple observations of different classes. This problem is strongly related to multiple-sensor fusion, which is also a complex topic of research. A general investigation or approach, maybe combining multiple rudimentary detector stumps, such as in the work of Torralba [242], would be of great help to build a flexible setup for detecting objects from multiple different classes.

Like other promising approaches for object recognition, we have also used vocabulary trees as base for our embedded object recognition setup. We have used DoG keypoints and SIFT keys, as originally proposed by Lowe [142]. However, the approach is not limited to any special type of detector or descriptor, so other possible choices include the set of detectors and descriptors investigated by Mikolajczyk *et al.* [159, 157]. Furthermore, we have not used any post-verification step in our recognition setup. The use of additional geometric information about neighboring interest points could definitely improve the recognition accuracy, however, being still problematic due to the limited amount of memory available on embedded systems. One possible solution would be the use of Co-occurrences, as proposed by Winter *et al.* [261]. Given a concrete application of the recognition approach, we have just simulated a rather small smart camera network, due to the lack of hardware available and the high administrative effort needed for a larger evaluation of the approach. However, the initial results are promising and encourage further research in that direction. Possible extensions to increase robustness of the approach at reasonable costs would be the use of additional features based on color combinations, such as the *Multimodal Neighborhood Signature* (MNS) method, proposed by Koubaroulis *et al.* [125] and already used by Weber [257]. Especially in the context of vehicles, color information might help to increase recognition performance and reduce mismatches.

For future research and development, a lot of high-level goals can be listed, particularly the deployment of adaptive approaches on smart cameras. This includes autonomous calibration of entire camera networks, and the capability to find communication networks with minimal energy consumption [46, 60, 61]. Furthermore, in the area of object detection, the unsupervised learning of object appearance by using online Boosting is desirable. In the area of object recognition, the investigation of categorization issues is an important issue, which would provide many possibilities for application of object recognition in categorization tasks. The number of desirable

improvements is large, and lot of work has to be done in the future. However, as the development of algorithms advances, we will see visionary ideas become reality, and, for sure, mobile and embedded devices will also play a vital role in this context.

Appendix A

Abbreviations and Terms

The following abbreviations and terms are used frequently in this thesis. However, we only write out the single abbreviations here. A more elaborate discussion of individual terms is given throughout the thesis.

- ALU **A**rithmetic **L**ogic **U**nit
- ANSI **A**merican **N**ational **S**tandards **I**nstitute
- ASIC **A**pplication **S**pecific **I**ntegrated **C**ircuit
- CISC **C**omplex **I**nstruction **S**et **C**omputer
- CMOS **C**omplementary **M**etal **O**xide **S**emiconductor
- DMA **D**irect **M**emory **A**ccess
- DoG **D**ifference **o**f **G**aussians
- DRAM **D**ynamic **R**andom **A**ccess **M**emory
- DSP **D**igital **S**ignal **P**rocessor
- FPGA **F**ield **P**rogrammable **G**ate **A**rray
- FPU **F**loating **P**oint **U**nit
- GPU **G**raphics **P**rocessing **U**nit
- IC **I**ntegrated **C**ircuit
- IEEE **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers, Inc.
- JTAG **J**oint **T**est **A**ction **G**roup
- MAC **M**ultiply-**AC**cumulate

- MATLAB High level computing language developed by Mathworks
- MEX **M**ATLAB **Ex**ecutable
- MPEG **M**oving **P**icture **E**xperts **G**roup
- PCA **P**rincipal **C**omponent **A**nalysis
- PDA **P**ersonal **D**igital **A**ssistant
- RISC **R**educed **I**nstruction **S**et **C**omputer
- SIFT **S**cale **I**nvariant **F**eature **T**ransform
- SIMD **S**ingle **I**nstruction **M**ultiple **D**ata
- SoC **S**ystem **o**n **C**hip
- SDRAM **S**ynchronous **D**ynamic **R**andom **A**ccess **M**emory
- SRAM **S**tatic **R**andom **A**ccess **M**emory
- TI **T**exas **I**nstruments
- VLIW **V**ery **L**ong **I**nstruction **W**ord
- VLSI **V**ery **L**arge **S**cale **I**ntegration

Appendix B

ALOI object selection

In Table B.1 the IDs of the 250 objects selected from the ALOI database for our object recognition experiments are listed. These objects have been selected because they deliver the highest number of DoG points. To illustrate this in Figure B.1 the number of DoG points for the top 500 ALOI images is depicted.

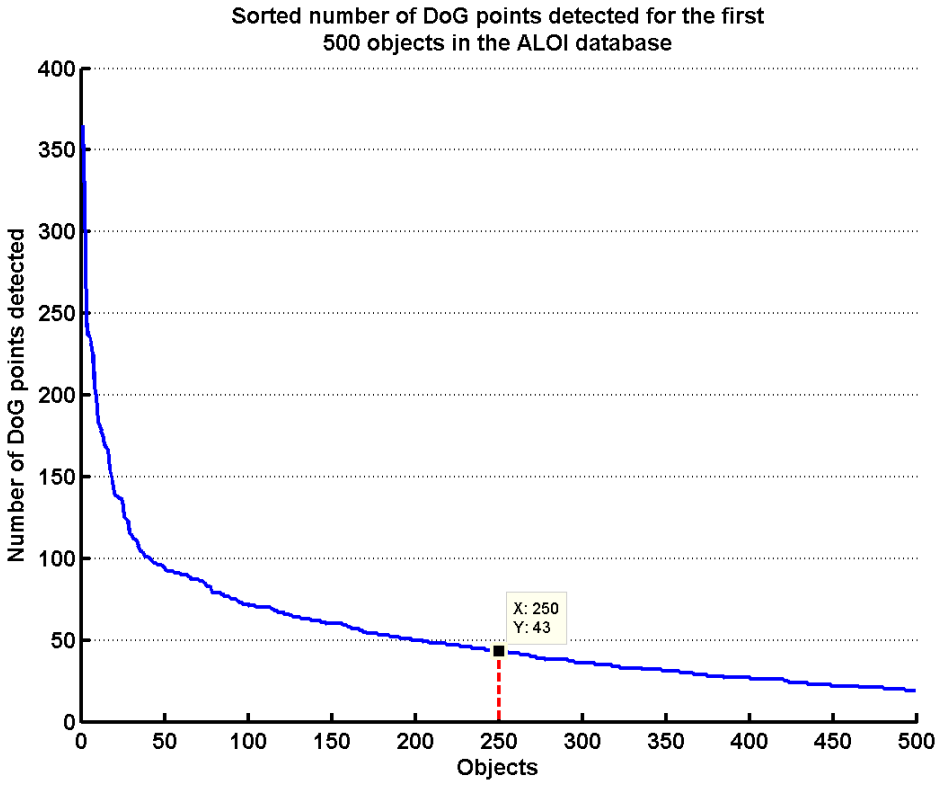


Figure B.1: Number of DoG points for the first 500 objects in the ALOI database decreasingly sorted.

Object IDs									
2	111	234	329	466	558	629	761	835	927
9	119	239	330	467	575	637	766	838	928
18	126	243	335	469	576	640	769	840	930
20	133	246	345	478	577	648	770	842	938
37	135	247	355	488	578	673	771	846	940
39	138	253	357	489	580	676	772	847	942
41	151	263	362	498	581	678	774	850	945
46	154	269	368	499	582	684	782	853	946
48	155	282	369	511	583	687	783	854	958
49	156	284	374	517	584	688	795	858	959
71	157	285	377	519	588	696	796	861	960
72	160	293	381	525	595	698	797	864	963
74	162	297	405	530	599	703	798	868	964
75	185	298	406	531	602	704	805	872	967
76	195	300	407	534	606	729	807	874	969
77	199	301	445	538	610	731	809	876	972
86	210	307	447	539	612	736	811	877	973
93	212	310	448	541	615	740	814	879	974
95	216	313	451	542	616	744	815	890	975
99	219	315	453	543	618	746	821	893	977
101	222	317	455	544	619	748	828	907	978
103	225	319	456	546	621	749	829	909	985
104	226	322	458	547	622	753	830	910	987
108	227	323	463	554	625	755	832	913	990
109	229	325	464	556	627	756	834	917	994

Table B.1: Object IDs selected for experiments.

Appendix C

Key publications

Publications developed during the research work are reported in chronological order in the following report. The aims and the key findings of each paper are shortly described by the abstract of the individual paper.

- Arth, C. and Leistner, C. and Bischof, H. (2006). TRICam - An Embedded Platform for Remote Traffic Surveillance. *Proceedings of the 2nd Workshop on Embedded Computer Vision, IEEE International Conference on Computer Vision and Pattern Recognition (ECV'06)*, pp 125–125. [10]

  Abstract: In this paper we present a novel embedded platform, dedicated especially to the surveillance of remote locations under harsh environmental conditions, featuring various video and audio compression algorithms as well as support for local systems and devices. The presented solution follows a radically decentralized approach and is able to act as an autonomous video server. Using up to three Texas Instruments^{TM}TMS320C6414 DSPs, it is possible to use high-level computer vision algorithms in real-time in order to extract the information from the video stream which is relevant to the surveillance task. The focus of this paper is on the task of vehicle detection and tracking in images. In particular, we discuss the issues specific for embedded systems, and we describe how they influenced our work. We give a detailed description of several algorithms and justify their use in our implementation. The power of our approach is shown on two real-world applications, namely vehicle detection on highways and license plate detection on urban traffic videos

- Martin Winter and Sandra Ober and Clemens Arth and Horst Bischof. Vocabulary Tree Hypotheses and Co-Occurrences. Proceedings of the 12th Computer Vision Winter Workshop (CVWW'07) - BEST PAPER AWARD, [261]

  Abstract: This paper introduces an efficient method to substantially increase the recognition performance of a vocabulary tree based recognition system. We propose to combine the hypothesis obtained by a standard inverse object voting algorithm with reliable descriptor co-occurrences. The algorithm operates on different depths

of a standard k-means tree, coevally benefiting from the advantages of different levels of information abstraction. The visual vocabulary tree shows good results when a large number of distinctive descriptors form a large visual vocabulary. Co-occurrences perform well even on a coarse object representation with a few number of visual words. We demonstrate the achieved performance increase, robustness to occlusions and background clutter in a challenging object recognition task on a subset of the Amsterdam Library of Object Images (ALOI).

- Arth, C. and Leistner, C. and Bischof, H. (2007). Robust Local Features and their Application in Self-Calibration and Object Recognition on Embedded Systems *Proceedings of the 3nd Workshop on Embedded Computer Vision, IEEE International Conference on Computer Vision and Pattern Recognition (ECV'07) - BEST PAPER AWARD*, pp 1–8. [12]

  Abstract: In recent years many powerful Computer Vision algorithms have been invented, making automatic or semiautomatic solutions to many popular vision tasks, such as visual object recognition or camera calibration, possible. On the other hand embedded vision platforms and solutions such as smart cameras have successfully emerged, however, only offering limited computational and memory resources. The first contribution of this paper is the investigation of a set of robust local feature detectors and descriptors for application on embedded systems. We briefly describe the methods involved, i.e. the DoG (Difference of Gaussian) and MSER (Maximally Stable Extremal Regions) detector as well as the PCA-SIFT descriptor, and discuss their suitability for smart systems and their qualification for given tasks. The second contribution of this work is the experimental evaluation of these methods on two challenging tasks, namely fully embedded object recognition on a moderate size database and on the task of robust camera calibration. Our approach is fortified by encouraging results we present at length.

- Arth, C. and Limberger, F. and Bischof, H. (2007). Real-Time License Plate Recognition on an Embedded DSP-Platform *Proceedings of the 3nd Workshop on Embedded Computer Vision, IEEE International Conference on Computer Vision and Pattern Recognition (ECV'07)*, pp 1–8. [13]

  Abstract: In this paper we present a full-featured license plate detection and recognition system. The system is implemented on an embedded DSP platform and processes a video stream in real-time. It consists of a detection and a character recognition module. The detector is based on the AdaBoost approach presented by Viola and Jones. Detected license plates are segmented into individual characters by using a region-based approach. Character classification is performed with support vector classification. In order to speed up the detection process on the embedded device, a Kalman tracker is integrated into the system. The search area of the detector is limited to locations where the next location of a license plate is predicted.

Furthermore, classification results of subsequent frames are combined to improve the class accuracy. The major advantages of our system are its real-time capability and that it does not require any additional sensor input (e.g. from infrared sensors) except a video stream. We evaluate our system on a large number of vehicles and license plates using bad quality video and show that the low resolution can be partly compensated by combining classification results of subsequent frames.

- Sandra Ober and Martin Winter and Clemens Arth and Horst Bischof (2007). Dual-Layer Visual Vocabulary Tree Hypotheses For Object Recognition Proceedings of the IEEE International Conference on Image Processing (ICIP'07), vol VI, pp 345–348. [168]

  Abstract: This paper introduces an efficient method to substantially increase the recognition performance of a vocabulary tree bas-ed recognition system. We propose to enhance the hypothesis obtained by a standard inverse object voting algorithm with reliable descriptor co-occurrences. The algorithm operates on different layers of a standard k-means tree benefiting from the advantages of different levels of information abstraction. The visual vocabulary tree shows good results when a large number of distinctive descriptors form a large visual vocabulary. Co-occurrences perform well even on a coarse object representation with a small number of visual words. An arbitration strategy with minimal computational effort combines the specific strengths of the particular representations. We demonstrate the achieved performance boost and robustness to occlusions in a challenging object recognition task.

- Arth, C. and Leistner, C. and Bischof, H. (2007). Object Reacquisition and Tracking in Large-Scale Smart Camera Networks *Proceedings of the 1st IEEE International Conference on Distributed Smart Cameras (ICDSC'07)*, pp 156–163. [11]

  Abstract: Object reacquisition or reidentification is the process of matching objects between images taken from separate cameras. In this paper, we present our work on feature based object reidentification performed on autonomous embedded smart cameras and applied to traffic scenarios. We present a novel approach based on PCA-SIFT features and a vocabulary tree. By building unique object signatures from visual features, reidentification can be done efficiently coevally minimizing the communication overhead between separate camera nodes. Applied to large-scale traffic scenarios, important parameters including travel time, travel time variability, section density, and partial dynamic origin/destination demands can be obtained. The proposed approach works on spatially separated, uncalibrated, non-overlapping cameras, is highly scalable and solely based on appearance-based optical features. The entire system is implemented and evaluated with regard to a typical embedded smart camera platform featuring one single Texas Instruments^{TM}fixed-point DSP.

- Donoser, M. and Arth, C. and Bischof, H. (2007). Detecting, Tracking and Recognizing License Plates *Proceedings of the Asian Conference On Computer Vision (ACCV'07)*, pp 447-456. [64]

  Abstract: This paper introduces a novel real-time framework which enables detection, tracking and recognition of license plates from video sequences. An efficient algorithm based on analysis of Maximally Stable Extremal Region (MSER) detection results allows localization of international license plates in single images without the need of any learning scheme. After a one-time detection of a plate it is robustly tracked through the sequence by applying a modified version of the MSER tracking framework which provides accurate localization results and additionally segmentations of the individual characters. Therefore, tracking and character segmentation is handled simultaneously. Finally, support vector machines are used to recognize the characters on the plate. An experimental evaluation shows the high accuracy and efficiency of the detection and tracking algorithm. Furthermore, promising results on a challenging data set are presented and the significant improvement of the recognition rate due to the robust tracking scheme is proved.

- Arth, C. and Bischof, H. (2008) Real-Time Object Recognition using Local Features on a DSP-based Embedded Systems *Journal of Real-Time Image Processing*, to appear. [9]

  Abstract: In the last few years object recognition has become one of the most popular tasks in computer vision. In particular, this was driven by the development of new powerful algorithms for local appearance based object recognition. So-called "smart cameras" with enough power for decentralized image processing became more and more popular for all kinds of tasks, especially in the

  eld of surveillance. Recognition is a very important tool as the robust recognition of suspicious vehicles, persons or objects is a matter of public safety. This simply makes the deployment of recognition capabilities on embedded platforms necessary. In our work we investigate the task of object recognition tion based on state-of-the-art algorithms in the context of a DSP-based embedded system.We implement several powerful algorithms for object recognition, namely an interest point detector together with an region descriptor, and build a medium-sized object database based on a vocabulary tree, which is suitable for our dedicated hardware setup. We carefully investigate the parameters of the algorithm with respect to the performance on the embedded platform. We show that state-of-the-art object recognition algorithms can be successfully deployed on nowadays smart cameras, even with strictly limited computational and memory resources.

In science one tries to tell people, in such a way as to be understood by everyone, something that no one ever knew before. But in poetry, it's the exact opposite.

Paul Dirac
English physicist in the US, 1902 - 1984

Bibliography

[1] Shivani Agarwal, Aatif Awan, and Dan Roth. Learning to Detect Objects in Images via a Sparse, Part-Based Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(11):1475–1490, 2004.

[2] Shivani Agarwal and Dan Roth. Learning a Sparse Representation for Object Detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 113–130, London, UK, 2002. Springer-Verlag.

[3] Hamid Aghajan, Franois Berry, Horst Bischof, Richard Kleihorst, Bernhard Rinner, and Wayne Wolf. CVPR Short Course: Distributed Vision Processing in Smart Camera Networks, 2007.

[4] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

[5] L. Albani, Pietro Chiesa, Daniele Covi, Pedegani G., Alvise Sartori, and Monica Vatteroni. VISoc: A Smart Camera SoC. In *Proceedings of the 28th European Conference on Solid-State Circuits (ESSCIRC)*, pages 367–370, September 2002.

[6] Saad Ali, Vladimir Reilly, and Mubarak Shah. Motion and Appearance for Tracking and Re-Acquiring Targets in Aerial Videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[7] Vicki H. Allan, Reese B. Jones, Randall M. Lee, and Stephen J. Allan. Software Pipelining, September 1995.

[8] Bernard Ans, Jeanny Hérault, and Christian Jutten. Adaptive Neural Architectures: Detection of Primitives. In *Proceedings of the of COGNITIVA*, pages 593–597, 1985.

[9] Clemens Arth and Horst Bischof. Real-Time Object Recognition using Local Features on a DSP-based Embedded System. *Journal of Real-Time Image Processing*, page to appear, 2008.

[10] Clemens Arth, Christian Leistner, and Horst Bischof. TRICam - An Embedded Platform for Remote Traffic Surveillance. In *Proceedings of the 2nd Workshop on Embedded Computer Vision , IEEE International Conference on Computer Vision and Pattern Recognition*, page 125, Washington, DC, USA, 2006. IEEE Computer Society.

[11] Clemens Arth, Christian Leistner, and Horst Bischof. Object Reacquisition and Tracking in Large-Scale Smart Camera Networks. In *Proceedings of the 1st IEEE International Conference on Distributed Smart Cameras*, pages 156–163, September 2007.

[12] Clemens Arth, Christian Leistner, and Horst Bischof. Robust Local Features and their Application in Self-Calibration and Object Recognition on Embedded Systems. In *Proceedings of the 3rd Workshop on Embedded Computer Vision , IEEE International Conference on Computer Vision and Pattern Recognition - BEST PAPER AWARD*, June 2007.

[13] Clemens Arth, Florian Limberger, and Horst Bischof. Real-Time License Plate Recognition on an Embedded DSP-Platform. In *Proceedings of the 3rd Workshop on Embedded Computer Vision , IEEE International Conference on Computer Vision and Pattern Recognition - BEST PAPER AWARD*, June 2007.

[14] Francis R. Bach and Michael I. Jordan. Kernel Independent Component Analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

[15] K. S. Bae, K. K. Kim, Y. G. Chung, and W. P. Yu. Character recognition system for cellular phone with camera. In *COMPSAC '05: Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05) Volume 1*, pages 539–544, Washington, DC, USA, 2005. IEEE Computer Society.

[16] Dana H. Ballard. Generalizing the Hough Transform to Detect Arbitrary Shapes. *PR*, 13 (2):111–122, 1981.

[17] Marian Stewart Bartlett, Javier R. Movellan, and Terrence J. Sejnowski. Face Recognition by Independent Component Analysis. *IEEE Transactions on Neural Networks*, 13(6):1450–64, 2002.

[18] Aziz Umit Batur and Bruce E. Flinchbaugh. Performance Analysis of Face Recognition Algorithms on TMS320C64x, 2002.

[19] Aziz Umit Batur, Bruce E. Flinchbaugh, and Monson H. III. Hayes. A DSP-based Approach for the Implementation of Face Recognition Algorithms. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 253–256, 2003.

[20] Peter N. Belhumeur, Joao Hespanha, and David J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(7):711–720, 1997.

[21] Nikolaos Bellas, Sek M. Chai, Malcolm Dwyer, and Dan Linzmeier. FPGA Implementation of a License Plate Recognition SoC using Automatically Generated Streaming Accelerators. *20th International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–8, 2006.

[22] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape Matching and Object Recognition Using Shape Contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(4):509–522, 2002.

[23] Arrigo Benedetti and Pietro Perona. Real-time 2-D Feature Detection on a Reconfigurable Computer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 586, Washington, DC, USA, 1998. IEEE Computer Society.

[24] Ariji Bishnu, Piyush K. Bhunre, Bhargab B. Bhattacharya, Malay K. Kundu, C.A. Murthy, and Tinku Acharya. Content Based Image Retrieval: Related Issues using Euler Vector. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, volume 2, pages 585–588, 2002.

[25] Arijit Bishnu, Bhargab B. Bhattacharya, Malay K. Kundu, C.A. Murthy, and Tinku Acharya. A Pipeline Architecture for Computing the Euler Number of a Binary Image. *Journal of Systems Architecture*, 51(8):470–487, August 2005.

[26] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

[27] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[28] James F. Boyce and W.J. Hossack. Moment Invariants for Pattern Recognition. *Pattern Recognition Letters*, 1:451–456, 1983.

[29] Gary R. Bradski. Computer Vision Face Tracking For Use In Perceptual User Interface. In *Intel Technology Journal*, pages 123–140, 1998.

[30] Michael Bramberger, Josef Brunner, Bernhard Rinner, and Helmut Schwabach. Real-Time Video Analysis on an Embedded Smart Camera for Traffic Surveillance. *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 174–181, 2004.

[31] Michael Bramberger, Andreas Doblander, Milan Jovanovic, Arnold Maier, Bernhard Rinner, and Allan Tengg. Embedded Smart Cameras as Key Components in Reactive Sensor Systems. In *COGnitive systems with Interactive Sensors (COGIS)*, 2006.

[32] Michael Bramberger, Andreas Doblander, Arnold Maier, Bernhard Rinner, and Helmut Schwabach. Distributed Embedded Smart Cameras for Surveillance Applications. *IEEE Computer*, 39(2):68–75, February 2006.

[33] Michael Bramberger, Roman P. Pflugfelder, Arnold Maier, Bernhard Rinner, Bernhard Strobl, and Helmut Schwabach. A Smart Traffic Camera for Stationary Vehicle Detection. In *Proceedings of the Workshop on Intelligent Solutions in Embedded Systems (WISES)*. Vienna University of Technology, June 2003.

[34] Leo Breiman. Bagging Predictors. *Machine Learning*, 26 (2):123–140, 1996.

[35] Leo Breiman. Bias, Variance, and Arcing Classifiers. Technical report, Statistics Department, University of California, 1996.

[36] Ilja N. Bronstein, Konstantin A. Semendjajew, Gerhard Musiol, and Heiner Mühlig. *Taschenbuch der Mathematik*. Deutsch (Harri), 5 edition, 2001.

[37] Matthew Brown, Richard Szeliski, and Simon Winder. Multi-Image Matching using Multi-Scale Oriented Patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 510–517, 20-25 June 2005.

[38] Wouter Caarls, Pieter Jonker, and Henk Corporaal. SmartCam: Devices for Embedded Intelligent Cameras. In *Proc. 3rd PROGRESS Workshop on Embedded Systems*, pages 1–4, October 2002.

[39] Cristina Cabani and W. James MacLean. A Proposed Pipelined-Architecture for FPGA-based Affine-Invariant Feature Detectors. In *Embedded Computer Vision Workshop (held in conjunction with CVPR)*, pages 121–121, 17-22 June 2006.

[40] Branislav Kisačanin. Examples of Low-Level Computer Vision on Media Processors. In *Embedded Computer Vision Workshop (held in conjunction with CVPR)*, page 135, Washington, DC, USA, 2005. IEEE Computer Society.

[41] Barbara Caputo, Sahla Bouattour, and Heinrich Niemann. Robust Appearance-Based Object Recognition Using a Fully Connected Markov Random Field. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages III: 565–568, 2002.

[42] Gustavo Carneiro and Allan Jepson. Phase-Based Local Features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 282–296, 2002.

[43] Gustavo Carneiro and David Lowe. Sparse Flexible Models of Local Features. In *ECCV*, pages 29–43, 2006.

[44] Chad Carson, Serge Belongie, Hayit Greenspan, and Jitendra Malik. Region-based Image Querying. In *Workshop on Content-Based Access of Image and Video Libraries (held in conjunction with CVPR)*, 1997.

[45] Jie Chen, Shiguang Shan, Peng Yang, Shengye Yan, Xilin Chen, and Wen Gao. Novel Face Detection Method Based On Gabor Features. *Advances in Biometric Person Authentication, SINOBIOMETRICS*, 12:90–99, 2004.

[46] Zhaolin Cheng, Dhanya Devarajan, and Richard J. Radke. Determining Vision Graphs for Distributed Camera Networks using Feature Digests. *EURASIP Journal on Applied Signal Processing*, 2007(1):220–220, 2007.

[47] Ming-Yee Chiu, Remi Depommier, and Thomas Spindler. An Embedded Real-Time Vision System For 24-Hour Indoor/Outdoor Car Counting Applications. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 338–341, 2004.

[48] Haili Chui and Anand Rangarajan. A New Algorithm for Non-Rigid Point Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2044–2051, 2000.

[49] Benjamin Coifman. Vehicle Reidentification and Travel Time Measurement in Real-Time on Freeways Using the Existing Loop Detector Infrastructure. In *Transportation Research Board*, 1998.

[50] Luke Cole, David Austin, and Lance Cole. Visual object recognition using template matching. In *Proceedings of the Australian Conference on Robotics and Automation (ACRA)*, 2004.

[51] Antonio J. Colmenarez and Thomas S. Huang. Face Detection with Information-Based Maximum Discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 782–787, 1997.

[52] Dorin Comaniciu and Peter Meer. Mean Shift Analysis and Applications. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1197–1203, 1999.

[53] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active Appearance Models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 2, pages 484–498, 1998.

[54] Timothy F. Cootes, Christopher J. Taylor, David H. Cooper, and Jim Graham. Active Shape Models - Their Training and Application. *Computer Vision and Image Understanding (CVIU)*, 61:38–59, 1995.

[55] Franklin C. Crow. Summed-Area Tables for Texture Mapping. In *SIGGRAPH '84: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, pages 207–212, New York, NY, USA, 1984. ACM.

[56] Rita Cucchiara, Andrea Prati, Massimo Piccardi, and Nello Scarabottolo. Real-time Detection of Moving Vehicles. In *Proceedings of the International Conference on Image Analysis and Processing (ICIAP)*, page 618, Washington, DC, USA, 1999. IEEE Computer Society.

[57] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[58] Mark Daniels, Kate Muldawer, Jason Schlessman, Burak Ozer, and Wayne Wolf. Real-Time Human Motion Detection with Distributed Smart Cameras. In *Proceedings of the IEEE International Conference on Distributed Smart Cameras (ICDSC)*, pages 187–194, 25-28 Sept. 2007.

[59] Sarat C. Dass and Anil K. Jain. Markov Face Models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 680–687, 2001.

[60] Dhanya Devarajan and Richard J. Radke. Calibrating Distributed Camera Networks using Belief Propagation. *EURASIP Journal on Applied Signal Processing*, 2007(1):221–221, 2007.

[61] Dhanya Devarajan, Richard J. Radke, and Haeyong Chung. Distributed Metric Calibration of Ad-hoc Camera Networks. *ACM Transactions on Sensor Networks*, 2(3):380–403, 2006.

[62] Sabyasachi Dey, Bhargab B. Bhattacharya, Malay K. Kundu, and Tinku Acharya. A Fast Algorithm for Computing the Euler Number of an Image and its VLSI Implementation. In *Proceedings of the International Conference on VLSI Design*, pages 330–335, 2000.

[63] Andreas Doblander, Dietmar Gösseringer, Bernhard Rinner, and Helmut Schwabach. An Evaluation of Model-Based Software Synthesis from Simulink Models for Embedded Video Applications. *International Journal of Software Engineering and Knowledge Engineering*, 15(2):343–348, 2005.

[64] Michael Donoser, Clemens Arth, and Horst Bischof. Detecting, Tracking and Recognizing License Plates. In *Proceedings of the 8th Asian Conference on Computer Vision (ACCV)*, volume II, pages 447–456, November 2007.

[65] Michael Donoser and Horst Bischof. Efficient Maximally Stable Extremal Region (MSER) Tracking. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:553–560, 2006.

[66] Gyuri Dorkó and Cordelia Schmid. Selection of Scale-Invariant Parts for Object Class Recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 634–639 vol.1, 13-16 Oct. 2003.

[67] Workshop on Distributed Smart Cameras (DSC).

[68] IEEE Workshop on Embedded Computer Vision (ECVW) (in conjunction with IEEE CVPR).

[69] Leonardo Estevez and Nasser Kehtarnavaz. A Real-time Histographic Approach to Road Sign Recognition. In *Southwest Symposium on Image Analysis and Interpretation*, pages 95–100, 1996.

[70] Hamed Fatemi, Richard Kleihorst, Henk Corporaal, and Pieter Jonker. Real Time Face Recognition on a Smart Camera. In *Proceedings of the Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS)*, 2003.

[71] Rob Fergus, Pietro Perona, and Andrew Zisserman. Object Class Recognition by Unsupervised Scale-Invariant Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 264–271, June 2003.

[72] James M. Ferryman, Anthony D. Worrall, Geoffrey D. Sullivan, and Keith D. Baker. A Generic Deformable Model for Vehicle Recognition. In *Proceedings of the British Machine Vision Conference (BMVC)*, volume 1, pages 127–136, September 1995.

[73] R. A. Fisher. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7:179–188, 1936.

[74] William T. Freeman and Edward H. Adelson. The Design and Use of Steerable Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13(9):891–906, 1991.

[75] Yoav Freund. Boosting a Weak Learning Algorithm by Majority. *Information and Computation*, 121 (2):256–285, 1995.

[76] Yoav Freund. An Adaptive Version of the Boost by Majority Algorithm. *Proceedings of the Conference on Computational Learning Theory*, pages 102–113, 1999.

[77] Yoav Freund and Robert E. Schapire. A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55 (1):119–139, 1995.

[78] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. Technical report, Department of Statistics, Stanford University, 1998.

[79] Mario Fritz, Bastian Leibe, Barbara Caputo, and Bernt Schiele. Integrating Representative and Discriminative Models for Object Category Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1363–1370, Washington, DC, USA, 2005. IEEE Computer Society.

[80] King-Sun Fu and Eric Persoon. Shape Discrimination Using Fourier Descriptors. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 126–130, 1974.

[81] Kunihiko Fukushima. A Neural Network for Visual Pattern Recognition. *IEEE Computer*, 21(3):65–75, March 1988.

[82] Dariu M. Gavrila and Stefan Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *International Journal of Computer Vision (IJCV)*, 73 (1):41–59, 2007.

[83] Dariu M. Gavrila and Vasanth Philomin. Real-time Object Detection for "Smart" Vehicles. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 87–93 vol.1, 1999.

[84] Jan-Mark Geusebroek, Gertjan J. Burghouts, and Arnold W. M. Smeulders. The Amsterdam Library of Object Images. *International Journal of Computer Vision (IJCV)*, 61(1):103–112, 2005.

[85] Paolo Giacon, Saul Saggin, Gionanni Tomasi, and Matteo Busti. Implementing DSP Algorithms using Spartan-3 FPGAs. *Xcell Journal*, 53:2225, 2005.

[86] Michael Grabner. *Visual Tracking and Online Learning Discriminative Features*. PhD thesis, Institute for Computer Graphics and Vision, Technical University Graz, 2008.

[87] Michael Grabner, Helmut Grabner, and Horst Bischof. Fast Approximated SIFT. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pages 918–927. Springer-Verlag, 2006.

[88] Göksel Günlü. Improving DSP Performance for Artificial Neural Networks Based Face Recognition. In *Signal Processing and Communications Applications Conference, 2005. Proceedings of the IEEE 13th*, pages 551–556, 2005.

[89] Yanlin Guo, Steven C. Hsu, Ying Shan, Harpreet S. Sawhney, and Rakesh Kumar. Vehicle Fingerprinting for Reacquisition and Tracking in Videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 761–768, 2005.

[90] Yanlin Guo, Steven C. Hsu, Ying Shan, Harpreet S. Sawhney, and Rakesh Kumar. Robust Object Matching for Persistent Tracking with Heterogeneous Features. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(5):824–839, 2007.

[91] Chris Harris and Mike J. Stephens. A Combined Corner and Edge Detector. In *Alvey Vision Conference*, pages 147–152, 1988.

[92] Paul S. Heckbert. Filtering by Repeated Integration. *ACM SIGGRAPH Computer Graphics*, 20(4):315–321, 1986.

[93] David P. Helmbold and Robert E. Schapire. Predicting Nearly as Well as the Best Pruning of a Decision Tree. In *Computational Learing Theory*, pages 61–68, 1995.

[94] Stephan Hengstler, Daniel Prashanth, Sufen Fong, and Hamid Aghajan. Mesheye: a Hybrid-Resolution Smart Camera Mote for Applications in Distributed Intelligent Surveillance. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, pages 360–369, New York, NY, USA, 2007. ACM.

[95] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., 4 edition, 2006.

[96] Jeanny Hérault, Bernard Ans, and Christian Jutten. Circuits neuronaux á synapses modifiables: Décodage de messages composites par apprentissage non supervisé. *Comptes Rendus de lAcadémie des Sciences*, 299 (III-13):525–528, 1984.

[97] Masayuki Hiromoto, Kentaro Nakahara, Hiroki Sugano, Yukihiro Nakamura, and Ryusuke Miyamoto. A Specialized Processor Suitable for AdaBoost-Based Detection with Haar-like Features. In *Embedded Computer Vision Workshop (held in conjunction with CVPR)*, pages 1–8, 2007.

[98] Andrew Hollingworth, Gary Schrock, and John M. Henderson. Change Detection in the Flicker Paradigm: The Role of Fixation Position within the Scene, 2001.

[99] John J. Hopfield. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences USA (PNAS)*, 79:2554–2558, 1982.

[100] John J. Hopfield. Neurons with Graded Response have Collective Computational Properties like those of Two-State Neurons. *Proceedings of the National Academy of Sciences USA (PNAS)*, 81:3088–3092, May 1984.

[101] Harold Hotelling. Analysis of a Complex of Statistical Variables with Principal Components. *Journal of Educational Psychology*, 24:417–441, 1933.

[102] Ming-Kuei Hu. Visual Pattern Recognition by Moment Invariants. *IEEE Transactions on Information Theory*, 8(2):179–187, February 1962.

[103] Chang Huang, Haizhou Ai, Yuan Li, and Shihong Lao. Vector Boosting for Rotation Invariant Multi-View Face Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 446–453, Washington, DC, USA, 2005. IEEE Computer Society.

[104] Timothy Huang and Stuart J. Russell. Object Identification in a Bayesian Context. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1276–1283, 1997.

[105] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.

[106] ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC).

[107] Anil K. Jain and Douglas Zongker. Representation and Recognition of Handwritten Digits using Deformable Templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(12):1386–1390, Dec 1997.

[108] Vincent Jeanne, Francois-Xavier Jegaden, Richard Kleihorst, Alexander Danilin, and Ben Schueler. Real-time Face Detection on a Dual-Sensor Smart Camera using Smooth Edges Technique. In *DSC 2006 (Workshop on Distributed Smart Cameras)*, 2006.

[109] Ian T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[110] Timor Kadir, Andrew Zisserman, and Michael Brady. An Affine Invariant Salient Region Detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 1, pages 228–241, 2004.

[111] Zdeněk Kálal. Face Detection with WaldBoost Algorithm. Master's thesis, Czech Technical University, Prague, 2007.

[112] Varsha Kamat and Subramaniam Ganesan. An Efficient Implementation of the Hough Transform for Detecting Vehicle License Plates using DSPs. In *Proceedings of the Real-Time Technology and Applications Symposium (RTAS)*, page 58. IEEE Computer Society, 1995.

[113] Kari Karhunen. Über lineare Methoden in der Wahrscheinlichkeitsrechnung. *Annales Academiae Scientiarum Fennicae, Series A1: Mathematica-Physica*, 37:3–79, 1947.

[114] Jens Kaszubiak, Michael Tornow, Robert Kuhn, Bernd Michaelis, and Carsten Knöppel. Real-time Vehicle and Lane Detection with Embedded Hardware. In

Proceedings of the IEEE Intelligent Vehicles Symposium, pages 619–624, 6-8 June 2005.

[115] Yan Ke and Rahul Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 506–513, 2004.

[116] Michael Kearns. Thoughts on Hypothesis Boosting. Project for Ron Rivest's machine learning course at MIT, 1988.

[117] Michael J. Kearns and Leslie G. Valiant. Cryptographic Limitations on Learning Boolean Formulae and Finite Automata. *Journal of the Association of Computer Machinery*, 41 (1):67–95, 1994.

[118] Arezou Keshavarz, Ali Maleki Tabar, and Hamid Aghajan. Distributed Vision-Based Reasoning for Smart Home Care. In *DSC 2006 (Workshop on Distributed Smart Cameras)*, 2006.

[119] Vera Kettnaker and Ramin Zabih. Bayesian Multi-Camera Surveillance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 253–259, 1999.

[120] Michael Kirby and Lawrence Sirovich. Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 12(1):103–108, 1990.

[121] Richard Kleihorst, Anteneh Abbo, Andre van der Avoird, Marc Op de Beeck, Leo Sevat, Paul Wielage, Rutger van Veen, and Hany van Herten. Xetal: A Low-Power High-Performance Smart Camera Processor. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 5, pages 215–218, 2001.

[122] Richard Kleihorst, Martijn Reuvers Reuvers, Ben Krose, and Harry Broers. A Smart Camera for Face Recognition. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, volume 5, pages 2849–2852 Vol. 5, 2004.

[123] Richard Kleihorst, Ben Schueler, Alexander Danilin, and Marc Heijligers. Smart Camera Mote with High Performance Vision System. In *DSC 2006 (Workshop on Distributed Smart Cameras)*, 2006.

[124] Dieter Koller, Kostas Daniilidis, and Hans-Hellmuth Nagel. Model-Based Object Tracking in Monocular Image Sequences of Road Traffic Scenes. *International Journal of Computer Vision (IJCV)*, Vol. 10:257–281, 1993.

[125] Dimitri Koubaroulis, Jiří Matas, and Josef Kittler. Colour-based Object Recognition for Video Annotation. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, volume 2, pages 1069–1072, 2002.

[126] Sen M. Kuo, Bob H. Lee, and Wenshun Tian. *Real-Time Digital Signal Processing: Implementations and Applications*. John Wiley & Sons, 2006.

[127] Monica Lam. Software Pipelining: an Effective Scheduling Technique for VLIW Machines. *ACM SIGPLAN Notices*, 23(7):318–328, 1988.

[128] Phil Lapsley, Jeff Bier, Edward A. Lee, and Amit Shoham. *DSP Processor Fundamentals: Architectures and Features*. Wiley-IEEE Press, 1996.

[129] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A Sparse Texture Representation using Local Affine Regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(8):1265–1278, 2005.

[130] Daniel D. Lee and H. Sebastian Seung. Learning the Parts of Objects by Non-negative Matrix Factorization. *Nature*, 401:788–791, 1999.

[131] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Combined Object Categorization and Segmentation with an Implicit Shape Model. In *Workshop on Statistical Learning in Computer Vision (held in conjunction with ECCV)*, 2004.

[132] Bastian Leibe and Bernt Schiele. Interleaved Object Categorization and Segmentation. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 759–768, Norwich, UK, Sept. 2003.

[133] Bastian Leibe, Edgar Seemann, and Bernt Schiele. Pedestrian Detection in Crowded Scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 878–885, Washington, DC, USA, 2005. IEEE Computer Society.

[134] Christian Leistner. Robust Real-time Vehicle Detection on an Embedded DSP Platform. Master's thesis, Institute for Computer Graphics and Vision, Technical University Graz, 2006.

[135] Aleš Leonardis and Horst Bischof. Robust Recognition Using Eigenimages. *Computer Vision and Image Understanding: CVIU*, 78(1):99–118, 2000.

[136] Aleš Leonardis, Horst Bischof, and Jasna Maver. Multiple Eigenspaces. *Pattern Recognition*, 35(11):2613–2627, November 2002.

[137] Vincent Lepetit, Vincent Lagger, and Pascal Fua. Randomized Trees for Real-time Keypoint Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 775–781, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[138] Stan Z. Li and Anil K. Jain, editors. *Handbook of Face Recognition*. Springer, 2005.

[139] Rainer Lienhart and Jochen Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, volume 1, pages I–900–I–903 vol.1, 2002.

[140] Florian Limberger. Robust Real-Time License Plate Recognition on an Embedded DSP Platform. Master's thesis, Institute for Computer Graphics and Vision, Technical University Graz, 2007.

[141] Chung L. Liu and James W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 20(1):46–61, 1973.

[142] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[143] Stephane G. Mallat. A Theory for Multiresolution Signal Decomposition: The Wavelet Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 11(7):674–693, 1989.

[144] Yishay Mansour. Pessimistic Decision Tree Pruning based on Tree Size. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 195–201. Morgan Kaufmann Publishers Inc., 1997.

[145] Jiří Matas, Ondřej Chum, Martin Urban, and Tomáš Pajdla. Robust Wide baseline Stereo from Maximally Stable Extremal Regions. In Paul L. Rosin and David Marshall, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, volume 1, pages 384–393, London, UK, September 2002. BMVA.

[146] Binu Mathew, Al Davis, and Robert Evans. A Characterization of Visual Feature Recognition. In *IEEE International Workshop on Workload Characterization (WWC)*, pages 3–11, 27 Oct. 2003.

[147] Binu Mathew, Al Davis, and Mike Parker. A Low Power Architecture for Embedded Perception. In *International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, pages 46–56, 2004.

[148] Rob McCready. Real-time Face Detection on a Configurable Hardware System. In *Proceedings of the International Workshop on Field-Programmable Logic and Applications (FPA)*, pages 157–162, London, UK, 2000. Springer-Verlag.

[149] Ron Meir and Gunnar Rätsch. An Introduction to Boosting and Leveraging. *Advanced Lectures on Machine Learning*, 41 (1):119–184, 2003. `http://www.boosting.org`.

[150] Thomas Melzer, Michael Reiter, and Horst Bischof. Appearance Mmodels Based on Kernel Canonical Correlation Analysis. *Pattern Recognition*, 36(9):1961–1971, 2003.

[151] Giovanni De Micheli and Rajesh K. Gupta. Hardware/Software Co-Design. *Proceedings of the IEEE*, 85(3):349–365, Mar 1997.

[152] Sebastian Mika, Gunnar Rätsch, Jason Weston, Bernhard Schölkopf, and Klaus-Robert Müller. Fisher Discriminant Analysis with Kernels. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 41–48, 1999.

[153] Krystian Mikolajczyk. *Interest Point Detection Invariant to Affine Transformations*. PhD thesis, Institut National Polytechnique de Grenoble, 2002.

[154] Krystian Mikolajczyk, Bastian Leibe, and Bernt Schiele. Local Features for Object Class Recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1792–1799 Vol. 2, 17-21 Oct. 2005.

[155] Krystian Mikolajczyk and Cordelia Schmid. Indexing Based on Scale Invariant Interest Points. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 525–531, 2001.

[156] Krystian Mikolajczyk and Cordelia Schmid. An Affine Invariant Interest Point Detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 1, pages 128–142, 2002.

[157] Krystian Mikolajczyk and Cordelia Schmid. A Pperformance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(10):1615–1630, 2005.

[158] Krystian Mikolajczyk, Cordelia Schmid, and Andrew Zisserman. Human Detection based on a Probabilistic Assembly of Robust Part Detectors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume I, pages 69–81, 2004.

[159] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiří Matas, Frederick Schaffalitzky, Timor Kadir, and Luc Van Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision (IJCV)*, 65(1-2):43–72, 2005.

[160] Baback Moghaddam and Alex Pentland. Probabilistic Visual Learning for Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 00, page 786, Los Alamitos, CA, USA, 1995. IEEE Computer Society.

[161] Baback Moghaddam and Alex Pentland. Probabilistic Visual Learning for Object Representation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, volume 19 (7), pages 696–710, Washington, DC, USA, 1997. IEEE Computer Society.

[162] Mario Enrique Munich, Paolo Pirjanian, Enrico DiBernardo, Luis Goncalves, Niklas Karlsson, and David Lowe. Break-through Visual Pattern Recognition for Robotics and Automation. In *IEEE International Conference on Robotics and Automation*, 2005.

[163] Srinivasan Murali, Theocharis Theocharides, Narayanan Vijaykrishnan, Mary Jane Irwin, Luca Benini, and Giovanni De Micheli. Analysis of Eerror Recovery Schemes for Networks on Chips. In *Design & Test of Computers, IEEE*, volume 22 (5), pages 434–442, Sept.-Oct. 2005.

[164] Praveen K. Murthy and Shuvra S. Bhattacharyya. *Memory Management for Synthesis of DSP Software*. CRC Press, Inc., Boca Raton, FL, USA, 2006.

[165] Jim Mutch and David Lowe. Multiclass object recognition with sparse, localized features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[166] Vinod Nair, Pierre-Olivier Laprise, and James J. Clark. An FPGA-based People Detection System. *EURASIP Journal on Applied Signal Processing*, 2005(1):1047–1061, 2005.

[167] David Nistér and Henrik Stewénius. Scalable Recognition with a Vocabulary Tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2161–2168, June 2006.

[168] Sandra Ober, Martin Winter, Clemens Arth, and Horst Bischof. Dual-Layer Visual Vocabulary Tree Hypotheses For Object Recognition. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'07)*, volume VI, pages 345–348, September 2007.

[169] Songhwai Oh, Stuart Russell, and Shankar Sastry. Markov Chain Monte Carlo Data Association for General Multiple-Target Tracking Problems. In *Proceedings of the 43rd IEEE Conference on Decision and Control, Paradise Island, Bahamas*, 2004.

[170] Andreas Opelt, Michael Fussenegger, Axel Pinz, and Peter Auer. Weak Hypotheses and Boosting for Generic Object Detection and Recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 2, pages 71–84, 2004.

[171] Andreas Opelt, Axel Pinz, and Andrew Zisserman. Incremental Learning of Object Detectors using a Visual Shape Alphabet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 3–10, 17-22 June 2006.

[172] Andreas Opelt, Axel Pinz, and Andrew Zisserman. A Boundary-Fragment-Model for Object Detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 2, pages 575–588, 2006.

[173] Valerij Ortmann and Rolf Eckmiller. Real-time Object Recognition Based on Active Vision and Sequential Analysis. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3325–3328, Washington, DC, USA, 1999. IEEE Comp. Society.

[174] Edgar Osuna, Robert Freund, and Federico Girosi. Training Support Vector Machines: An Application to Face Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1997.

[175] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, and Tim Purcell. A Survey of General-Purpose Computation on Graphics Hardware. In *Eurographics, State of the Art Reports*, pages 21–51, August 2005.

[176] Mustafa Ozuysal, Pascal Fua, and Vincent Lepetit. Fast keypoint recognition in ten lines of code. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2007.

[177] Pentti Paatero and Unto Tapper. Positive Matrix Factorization: A Non-negative Factor Model with Optimal Utilization of Error Estimates of Data Values. *Environmetrics*, 5(2):111–126, 1994.

[178] Constantine P. Papageorgiou, Michael Oren, and Tomaso Poggio. A General Framework for Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 00, page 555, Los Alamitos, CA, USA, 1998. IEEE Computer Society.

[179] Johnny Park, Priya C. Bhat, and Avinash C. Kak. A Look-up Table Based Approach for Solving the Camera Selection Problem in Large Camera Networks. In *DSC 2006 (Workshop on Distributed Smart Cameras)*, 2006.

[180] Stavros Paschalakis and Miroslaw Bober. A Low-cost FPGA System for High Speed Face Detection and Tracking. In *Proceedings of the IEEE International Conference on Field-Programmable Technology (FPT)*, pages 214–221, 15-17 Dec. 2003.

[181] David A. Patterson. Reduced Instruction Set Computers. *Communications of the ACM*, 28(1):8–21, 1985.

[182] David A. Patterson and John L. Hennessy. *Computer Organization and Design: the Hardware/Software Iinterface*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3 edition, 2005.

[183] Karl Pearson. On Lines and Planes of Closest Fit to Systems of Points in Space. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Sciences*, 6(2):559–572, 1901.

[184] Thang V. Pham, Marcel Worring, and Arnold W. M. Smeulders. A Multi-Camera Visual Surveillance System for Tracking of Reoccurrences of People. In *Proceedings of the IEEE International Conference on Distributed Smart Cameras (ICDSC)*, pages 164–169, 25-28 Sept. 2007.

[185] Fatih Porikli. Integral Histogram: A Fast Way to Extract Histograms in Cartesian Spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 829–836 vol. 1, 20-25 June 2005.

[186] Richard J. Prokop and Anthony P. Reeves. A Survey of Moment-Based Techniques for Unoccluded Object Representation and Recognition. *CVGIP: Graph. Models Image Process.*, 54(5):438–460, 1992.

[187] Markus Quaritsch, Markus Kreuzthaler, Bernhard Rinner, Horst Bischof, and Bernhard Strobl. Autonomous Multicamera Tracking on Embedded Smart Cameras. *EURASIP Journal on Embedded Systems (Special Issue on Embedded Vision System)*, 2007:48–57, 2007.

[188] Markus Quaritsch, Markus Kreuzthaler, Bernhard Rinner, and Bernhard Strobl. Decentralized Object Tracking in a Network of Embedded Smart Cameras. In *DSC 2006 (Workshop on Distributed Smart Cameras)*, 2006.

[189] Shehrzad Qureshi. *Embedded Image Processing on the TMS320C6000 DSP: Examples in Code Composer Studio and MATLAB*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[190] Mohammad Rahimi, Rick Baer, Obimdinachi I. Iroezi, Juan C. Garcia, Jay Warrior, Deborah Estrin, and Mani Srivastava. Cyclops: In-situ Image Sensing and Interpretation in Wireless Sensor Networks. In *SenSys '05: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, pages 192–204, New York, NY, USA, 2005. ACM.

[191] Surinder Ram. Object Recognition from Local Features. Master's thesis, Institute for Computer Graphics and Vision, Technical University Graz, 2006.

[192] Ronald A. Rensink, J. Kevin O'Regan, and James J. Clark. To See or Not to See: The Need for Attention to Perceive Changes in Scenes. *Psychological Science*, 8:368–373, 1997.

[193] Martijn Reuvers. Face Detection on the INCA+ System. Master's thesis, University of Amsterdam, 2004.

[194] Peter E. Hart Richard O. Duda and David G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[195] Sami Romdhani, Philip Torr, Bernhard Schölkopf, and Andrew Blake. Computationally Efficient Face Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 695–700 vol.2, 2001.

[196] Peter Martin Roth. *Autonomous Learning an Object Representation by On-line Conservative Learning*. PhD thesis, Graz University of Technology, Institute for Computer Graphics and Vision, 2008.

[197] Anthony Rowe, Adam G. Goode, Dhiraj Goel, and Illah Nourbakhsh. CMUcam3: An Open Programmable Embedded Vision Sensor. Technical Report CMU-RI-TR-07-13, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2007.

[198] Anthony Rowe, Charles Rosenberg, and Illah Nourbakhsh. A Low Cost Embedded Color Vision System. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[199] Anthony Rowe, Charles Rosenberg, and Illah Nourbakhsh. A Second Generation Low Cost Embedded Color Vision System. In *Embedded Computer Vision Workshop (held in conjunction with CVPR)*, 2005.

[200] Sam T. Roweis and Lawrence K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290:2323–2326, 2000.

[201] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural Network-Based Face Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 20(1):23–38, 1998.

[202] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-Propagating Errors. *Neurocomputing: Foundations of Research*, pages 696–699, 1988.

[203] Hanan Samet. The Quadtree and Related Hierarchical Data Structures. *ACM Computing Surveys*, 16(2):187–260, 1984.

[204] Frederik Schaffalitzky and Andrew Zisserman. Multi-view Matching for Unordered Image Sets, or "How Do I Organize My Holiday Snaps?". In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 1, pages 414–431, 2002.

[205] Robert E. Schapire. The Strength of Weak Learnability. *Machine Learning*, 5 (2):197–227, 1990. `http://www.boosting.org`.

[206] Robert E. Schapire and Yoav Freund. Experiments with a New Boosting Algorithm. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 148–156. Morgan Kaufmann Publishers Inc., 1996.

[207] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the Margin: a new Explanation for the Effectiveness of Voting Methods. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 322–330. Morgan Kaufmann Publishers Inc., 1997.

[208] Robert E. Schapire and Yoram Singer. Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, 37 (3):297–336, 1999.

[209] Cordelia Schmid and Roger Mohr. Local Grayvalue Invariants for Image Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(5):530–535, 1997.

[210] Henry Schneiderman and Takeo Kanade. A Statistical Method for 3D Object Detection applied to Faces and Cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 746–751 vol.1, 2000.

[211] Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319, 1998.

[212] Stephen Se, Timothy Barfoot, and Piotr Jasiobedzki. Visual Motion Estimation and Terrain Modeling for Planetary Rovers. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2005.

[213] Gregory Shakhnarovich and Baback Moghaddam. *Handbook of Face Recognition*, chapter Face Recognition in Subspaces, page 35. Springer-Verlag, 2004.

[214] Ying Shan, Harpreet S. Sawhney, and Rakesh Kumar. Vehicle Identification between Non-Overlapping Cameras without Direct Feature Matching. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 378–385, 2005.

[215] J. Shen and G. W. Israël. A Receptor Model using a Specific Non-negative Transformation Technique for Ambient Aerosol. *Atmospheric Environment*, 23(10):2289–2298, 1989.

[216] Patrice Simard, Lon Bottou, Patrick Haffner, and Yann Le Cun. Boxlets: a Fast Convolution Algorithm for Signal Processing and Neural Networks. In *Advances in Neural Information Processing Systems*, volume 11, pages 571–577, 1999.

[217] Erik Simmons, Erik Ljung, and Richard Kleihorst. Distributed Vision with Multiple Uncalibrated Smart Cameras. In *DSC 2006 (Workshop on Distributed Smart Cameras)*, 2006.

[218] Daniel J. Simons and Daniel T. Levin. Change Blindness. *Trends in Cognitive Sciences*, 1(7):261–267, October 1997.

[219] Josef Sivic and Andrew Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 02, page 1470, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[220] Andrew Sloss, Dominic Symes, and Chris Wright. *ARM System Developer's Guide: Designing and Optimizing System Software*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[221] David Squire, Wolfgang Müller, Henning Müller, and Jilali Raki. Content-based Query of Image Databases, Inspirations from Text Retrieval: Inverted Files, Frequency-based Weights and Relevance Feedback. In *Proceedings of the Scandinavian Conference on Image Analysis (SCIA)*, 1999.

[222] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. Wavelets for Computer Graphics: A Primer, Part 1. *IEEE Computer Graphics and Applications*, 15 (3):76–84, 1995.

[223] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. Wavelets for Computer Graphics: A Primer, Part 2. *IEEE Computer Graphics and Applications*, 15 (4):75–85, 1995.

[224] Carlos C. Sun, Glenn S. Arr, Ravi P. Ramachandran, and Stephen G. Ritchie. Vehicle Reidentification Using Multidetector Fusion. In *IEEE Transactions on Intelligent Transportation Systems*, volume 5 (3), pages 155–165, 2004.

[225] Changming Sun and Deyi Si. Fast Reflectional Symmetry Detection Using Orientation Histograms. *Real-Time Imaging*, 5(1):63–74, 1999.

[226] Zehang Sun, George Bebis, and Ronald Miller. On-Road Vehicle Detection: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(5):694–711, 2006.

[227] Kah Kay Sung and Tomaso Poggio. Example-Based Learning for View-Based Human Face Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 20(1):39–51, 1998.

[228] Michael Reed Teague. Image Analysis via the General Theory of Moments. *Journal of the Optical Society of America*, 70(8):920–930, August 1980.

[229] Cho-Huak Teh and Roland T. Chin. On Image Analysis by the Methods of Moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 10(4):496–513, July 1988.

[230] Joshua B. Tenenbaum. Mapping a manifold of perceptual observations. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

[231] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[232] `DSPnano RTOS and Unison DSP RTOS`. http://www.rowebots.com/embedded_system_software/dsp_rtos.

[233] `MATLAB - The Language of Technical Computing`. http://www.mathworks.com.

[234] `MIPS Technologies`. http://www.mips.com/.

[235] `Moving Picture Experts Group`. http://www.chiariglione.org/mpeg/index.htm.

[236] `OpenCV - Open Source Computer Vision Library`. http://www.intel.com/technology/computing/opencv/index.htm, http://opencvlibrary.sourceforge.net.

[237] `TMS320C64x Technical Overview (Rev. B)`. http://focus.ti.com/lit/ug/spru395b/spru395b.pdf.

[238] `UIUC Image Database for Car Detection`. http://l2r.cs.uiuc.edu/ cogcomp/data/car/. (January 9, 2008).

[239] `Valgrind`. http://valgrind.org/.

[240] Theocharis Theocharides, Greg Link, Narayanan Vijaykrishnan, Mary Jane Irwin, and Wayne Wolf. Embedded Hardware Face Detection. In *Proceedings of the 17th International Conference on VLSI Design*, pages 133–138, 2004.

[241] Johan Thuresson and Stefan Carlsson. Appearance Based Qualitative Image Description for Object Class Recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages Vol II: 518–529, 2004.

[242] Antonio Torralba, K.P. Murphy, and W.T. Freeman. Sharing Visual Features for Multiclass and Multiview Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(5):854–869, May 2007.

[243] Zhuowen Tu. Probabilistic Boosting-Tree: Learning Discriminative Models for Classification, Recognition, and Clustering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1589–1596, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[244] Michael Turk and Alex Pentland. Face Recognition Using Eigenfaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–591, 1991.

[245] Tinne Tuytelaars and Luc Van Gool. Matching Widely Separated Views Based on Affine Invariant Regions. *International Journal of Computer Vision (IJCV)*, 59(1):61–85, 2004.

[246] Régis Vaillant, Christophe Monrocq, and Y. LeCun. Original Approach for the Localization of Objects in Images. *IEE Proceedings of the on Vision, Image, and Signal Processing*, 141 (4):–, 1994.

[247] Vladimir Vapnik and Aleksei Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[248] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[249] Remco Veltkamp and Michiel Hagedoorn. State-of-the-art in Sshape Matching. Technical Report UU-CS-1999-27, Utrecht University, the Netherlands, 1999.

[250] Paul Viola and Michael J. Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume II, pages 511–518, 2001.

[251] Paul Viola, Michael J. Jones, and Daniel Snow. Detecting Pedestrians Using Patterns of Motion and Appearance. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 734–741, 2003.

[252] Jan Šochman and Jiří Matas. Inter-stage Feature Propagation In Cascade Building With AdaBoost. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 236–239, 2004.

[253] Jan Šochman and Jiří Matas. WaldBoost - Learning For Time Constrained Sequential Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 150–156, Washington, DC, USA, 2005. IEEE Computer Society.

[254] Štěpán Obdržálek and Jiří Matas. Sub-linear Indexing for Large Scale Object Recognition. In *Proceedings of the British Machine Vision Conference (BMVC)*, volume 2, 2005.

[255] Abraham Wald. *Sequential analysis*. Dover Publications, 1947.

[256] F. Walzer and B. Krell. Object Oriented Programming for Digital Signal Processing Systems. *GSPx Conference*, 2002.

[257] Thomas Weber. Evaluation of the Mean Link Travel Time using Digital Image Processing. Master's thesis, Institute for Computer Graphics and Vision, Technical University Graz, 2007.

[258] Mao Wei and A. Bigdeli. Implementation of a Real-time Automated Face Recognition System for Portable Devices. In *IEEE International Symposium on Communications and Information Technology (ISCIT)*, volume 1, pages 89–92 vol.1, 26-29 Oct. 2004.

[259] Yu Wei, Xiong Bing, and C. Chareonsak. FPGA Implementation of AdaBoost Algorithm for Detection of Face Biometrics. In *IEEE International Workshop on Biomedical Circuits and Systems*, pages S1/6–17–20, 1-3 Dec. 2004.

[260] Adam Williams, Dan Xie, Shichao Ou, Roderic Grupen, Allen Hanson, and Edward Riseman. Distributed Smart Cameras for Aging in Place. In *DSC 2006 (Workshop on Distributed Smart Cameras)*, 2006.

[261] Martin Winter, Sandra Ober, Clemens Arth, and Horst Bischof. Vocabulary Tree Hypotheses and Co-Occurrences. In *Proceedings of the 12th Computer Vision Winter Workshop (CVWW'07) - BEST PAPER AWARD*, February 2007.

[262] Wayne Wolf. *High-Performance Embedded Computing: Architectures, Applications, and Methodologies*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[263] Wayne Wolf. A Decade of Hardware/Software Co-Design. *Computer*, 36(4):38–43, April 2003.

[264] Wayne Wolf, Burak Ozer, and Tiehan Lv. Smart Cameras as Embedded Systems. *IEEE Computer*, 35(9):48–53, 2002.

[265] Bo Wu, Haizhou Ai, Chang Huang, and Shihong Lao. Fast Rotation Invariant Multi-View Face Detection Based on Real Adaboost. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition (FGR)*, 2004.

[266] Bo Wu and Ram Nevatia. Cluster Boosted Tree Classifier for Multi-View, Multi-Pose Object Detection. In *ICCV*, pages 1–8, 2007.

[267] Chen Wu and Hamid Aghajan. Collaborative Gesture Analysis in Multi-Camera Networks. In *DSC 2006 (Workshop on Distributed Smart Cameras)*, 2006.

[268] Chen Wu, Hamid Aghajan, and Richard Kleihorst. Mapping Vision Algorithms on SIMD Architecture Smart Cameras. In *Proceedings of the IEEE International Conference on Distributed Smart Cameras (ICDSC)*, pages 27–34, 25-28 Sept. 2007.

[269] Fan Yang and Michel Paindavoine. Implementation of an RBF Neural Network on Embedded Systems: Real-time Face Tracking and Identity Verification. *IEEE Transactions on Neural Networks*, 14(5):1162–1175, Sept. 2003.

[270] Fan Yang, Michel Paindavoine, and Hervé Abdi. Parallel Implementation on DSPs of a Face Detection Algorithm. In *Proceedings of the IEEE International Conference on Signal Processing (ICSP)*, pages 69–72 vol.1, 1998.

[271] Ming Yang, Ying Wu, James Crenshaw, Bruce Augustine, and Russell Mareachen. Face Detection for Automatic Exposure Control in Handheld Camera. In *Proceedings of the International Conference on Computer Vision Systems*, page 17, Washington, DC, USA, 2006. IEEE Computer Society.

[272] Ming-Hsuan Yang, Narendra Ahuja, and David Kriegman. Face Detection Using Mixtures of Linear Subspaces. In *FG '00: Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, page 70, Washington, DC, USA, 2000. IEEE Computer Society.

[273] Charles R. Yates. Fixed-Point Arithmetic: An Introduction. http://www.digitalsignallabs.com/, August 2007.

[274] Charles R. Yates. Practical Considerations in Fixed-Point FIR Filter Implementations. http://www.digitalsignallabs.com/, March 2007.

[275] Tom Yeh, Kristen Grauman, Konrad Tollmar, and Trevor Darrell. A Picture is Worth a Thousand Keywords: Image-Based Object Search on a Mobile Platform. In *CHI Extended Abstracts*, pages 2025–2028, 2005.

[276] Daoqiang Zhang, Zhi-Hua Zhou, and Songcan Chen. Non-negative Matrix Factorization on Kernels. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, pages 404–412, 2006.

[277] Xiao-Sheng Zhuang and Dao-Qing Dai. Inverse Fisher Discriminate Criteria for Small Samples Size Problem and its Application to Face Recognition. *Pattern Recognition*, 38(11):2192–2194, 2005.