System Management Algorithms for Distributed Vision Networks

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades
Doktor-Ingenieur
genehmigte Dissertation

von Dipl.-Ing. Martin Hoffmann

geboren am 1. Januar 1981 in Bielefeld

2010

1. Referent:	Prof. Dr. rer. nat. Jörg Hähner
2. Referent:	Prof. Dr.-Ing. Bernardo Wagner
Tag der Promotion:	6. Juli 2010

Acknowledgements

First of all I would like to thank my advisor Jörg Hähner for his guidance and many fruitful discussions about my work. I would also like to thank Christian Müller-Schloer for giving me the chance to work in his research group.

I owe my gratitude to my exceptional colleagues in the System and Computer Architecture group, who helped me at countless occasions with worthy suggestions and lots of motivation. In particular I would like to thank Yvonne Bernard, Jürgen Brehm, Emre Çakar, Uwe Jänen, Monika Steinberg, Sven Tomforde, and Michael Wittke for many valuable discussions as well as Monika Lorenz and Lars Maasjost for their sedulous support.

I would also like to seize the opportunity to express my respect to numerous students, who contributed with their bachelor and master theses to the Smart Camera project this dissertation evolved from. Namely, these are: Helge Hoffmann, Elena Mogilevska, Jochen Witt, Ramin Soleymani, Liang Han, Anna Averbakh, Tobias Kavemann, Lars Friedrichs, Philipp Kleybolte, Christian Schulz, Florian Blatt, Ahmed Fares, Henning Perl, Markus Thielecke, and Andrea Kittner.

I would like to thank Franziska Linsmeier, my family, and all dear friends for their constant encouragement and understanding.

Finally, I appreciate the financial support from the *Bundesministerium für Wirtschaft und Technologie* that funded parts of the research discussed in this dissertation.

Zusammenfassung

Schlagworte: Smart Camera Netzwerke, verteilte Algorithmen zur Kameraausrichtung, Interaktion zwischen Nutzern und Kameras

Diese Dissertation behandelt Verwaltungsalgorithmen für große Kamerasysteme. In zukünftigen Sicherheits- und Überwachungssystemen werden intelligente Kameras zum Einsatz kommen. Diese Smart Cameras verfügen über integrierte Recheneinheiten, die Bilddaten noch am Bildsensor analysieren. Das Ergebnis dieser Analyse kann über ein Netzwerk übertragen werden. Somit reagieren Kameras weitestgehend autonom, werten kooperativ Bilddaten aus und fordern nur im Falle des Auftretens vordefinierter Ereignisse Unterstützung durch menschliches Personal an.

Eine bisher ungelöste Aufgabe bestand darin, das Aufnahmeverhalten dieser Kameras mit Hilfe verteilter Algorithmen zu koordinieren. In dieser Arbeit werden Verfahren vorgestellt, die in großen Kameranetzen zu einem selbst-organisierenden Systemverhalten führen. Insbesondere werden Algorithmen zur räumlichen Aufteilung einer zu überwachenden Fläche vorgestellt. Somit wird ein dezentrales, verteiltes Verfolgen von Objekten mit mehreren Kameras möglich. Die zentrale Problemstellung ist hierbei, wie eine Vielzahl durch Aktuatoren drehbarer Kameras so ausgerichtet werden kann, sodass das kooperative Aufnahmeverhalten die Anforderungen des Nutzers bestmöglich erfüllt. Eine mathematische Betrachtung ergibt, dass dieses Problem zur Klasse der NP-vollständigen Probleme gehört. Daher kann, bei den zur Verfügung stehenden Rechenressourcen und den Anforderungen an eine geringe Laufzeit der Algorithmen, lediglich eine Annäherung von Lösungen mit Hilfe von Heuristiken erreicht werden. Der Entwurf und die Evaluierung solcher Algorithmen werden in dieser Arbeit vorgestellt.

Um eine Kontrolle der Kameras durch Nutzer vornehmen zu können, wurde ein Verfahren zur Interaktion zwischen Kameras und mobilen Steuergeräten untersucht. Dieses Verfahren stellt sicher, dass trotz der Autonomie der Kameras letztlich die Nutzer das System ihren Wünschen und Anforderungen entsprechend kontrollieren können.

Abstract

Keywords: Smart Camera networks, distributed algorithms for camera alignment, interaction between users and Smart Cameras

This thesis presents system management algorithms for Distributed Vision Networks. Future video surveillance systems are expected to consist of Smart Cameras. These cameras contain a computing unit that is used for analysing image data acquired from the built-in CCD sensor. Recent advances in the research areas of computer vision make way for scene interpretation and automated generation of alarms in case serious incidents are detected. Security staff can be informed by cameras by using mobile devices that are connected to the Smart Camera network.

The contribution to knowledge presented in this thesis is a class of algorithms that coordinates Smart Cameras in such a way, so that they act self-organising and with the least necessary amount of control by humans. The focus is on the alignment of multiple cameras' PTZ heads in order to observe the area the cameras are positioned on in the most efficient manner. Thereby, a decentralised, distributed system for a seamless tracking of objects with multiple cameras becomes feasible. The main problem is to coordinate numerous cameras in order to reach a system behaviour that suits the needs of its users. A theoretical analysis of the problem reveals that the problem which is related to the NP-complete *set-packing problem* can hardly be solved with the computing capacities of today's computing systems. Therefore, distributed heuristics are described which approximate close to optimal solutions to this problem.

In order to enable users to interact with a camera system, appropriate methods are introduced that make way for a bi-directional communication between cameras and mobile devices. Thereby we assure that the cameras behaviour can be supervised by humans and the system can be adapted to specific needs.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation: Intelligent Distributed Surveillance Systems

"The work on intelligent distributed surveillance systems has been led by computer vision laboratories perhaps at the expense of system engineering issues"

- Dr Sergio Velastin, Kings College London [1]

Due to the tense international security situation video surveillance systems have become part of our everyday life. Surveillance systems are used at airports and public transport facilities to detect and prevent acts of terrorism. In many public places cameras are installed for the prevention of vandalism. Benefits and threats arising from an increasing deployment of surveillance systems are discussed extensively. Apart from technical issues that are discussed throughout this dissertation, social sciences investigate the impact of surveillance systems on our live. In many inner cities, video based surveillance systems are used to make citizens feel save and secure - although privacy issues can not be neglected and are subject of continuous public discussion and legal investigation [2].

A new generation of surveillance systems relies on Smart Cameras and overcomes drawbacks of today's systems in terms of privacy protection, cost efficiency and robustness towards security threats. Smart Cameras consist of a CCD sensor acquiring images and a computing unit that allows to analyse the collected images automatically. No image data leaves the Smart Camera as long as no predefined incidents are detected. Advances in the research area of computer vision allow for object detection, tracking and recognition and

thereby enable Smart Cameras to understand scenes autonomously. The research project PRISMATICA[1], which has been funded by the European Union, investigated intelligent surveillance systems. Research results show that in future computers are able to analyse images in an accurate and reliable way - similar as humans do [1].

In case predefined, serious incidents are recognised, security staff is informed. Video data that is recognised as irrelevant is deleted automatically by the Smart Cameras. Thus, privacy is guaranteed. Persons surveyed by Smart Cameras are no longer prone to negative side effects of today's CCTV systems such as voyeuristic security staff or inattentiveness caused by fatigue - which might lead to dangerous situations passing unnoticed. By overcoming current privacy problems, other fields of application apart from security and surveillance arise. For gaining a deeper understanding of shopping behaviour, Smart Cameras can be used to count people and measure waiting times - e.g. for optimising queues at cash desks [3].
Installations consisting of numerous networked Smart Cameras are called Distributed Vision Networks. An international conference has been established recently (International Conference on Distributed Smart Cameras, ICDSC) where advances in research concerning Distributed Vision Networks are discussed. Several fields of research drive the development of these networks, especially advances in embedded and distributed systems and computer vision are important factors for future developments. Large camera systems (as e.g. used at international airports) are comprised of thousands of cameras. This thesis shows, that image analysis and camera alignment benefit from a self-organising system architecture. In Distributed Vision Networks, Smart Cameras analyse scenes together and anticipate dangerous situations or aggregate useful statistics. Apart from collaboration in terms of computational image analysis, management tasks need to be carried out. For example, cameras need to cooperatively adjust their fields of view by panning and tilting their CCD sensors in order to observe areas or track objects efficiently. These management tasks should -as the analysis of video streams- be carried out in a distributed fashion and should not rely on central components that might be prone to errors and attacks and even lead to system failure.

Today, the main component of a video based surveillance system is a central control console, where the video data from all cameras is delivered to. In contrast to this, Distributed Vision Networks rely on a distributed system. Each Smart Camera is a computing node that is able to analyse data without a central entity. In order to allow for cooperation between (neighbouring) Smart Cameras and users, the networked system architecture has to be designed in such a way, that cooperation becomes possible. Security

[1] $http://cordis.europa.eu/data/PROJ\_FP5$

staff can stay in contact with the Smart Camera system by using mobile terminals that are connected to the Smart Cameras. These users of the Smart Camera system can send requests to the cameras and thereby control the system behaviour. In case serious incidents are detected by the cameras, security staff is informed. Therefore video data can be transferred from one or more cameras to a mobile node. This information can then be used to detect false alarms or investigate the incident further. The notification process requires a bi-directional communication scheme: Security staff defines incidents that the cameras are expected to watch out for and sets parameters that control the system's behaviour.

This thesis introduces a class of management algorithms for Distributed Vision Networks. Namely a spatial partitioning algorithm (ROCAS [4]) and a tracking algorithm (DMCtrac [5]) are introduced. For the alarm management, a system is presented that allows for a bi-directional communication between Smart Cameras and human staff equipped with mobile terminals (AMiDiViN [6]).

1.2 Problem Statement and Contribution

The increasing performance of computing systems led to a design gap: On the one hand, today's systems can carry out more complex tasks in less time. On the other hand, it is difficult for humans to design and maintain these systems. Common camera networks consist of thousands of cameras[2] that can hardly be managed manually in terms of camera alignment let alone image analysis. During the last years, several approaches to address the problem of rising complexity of technical systems have been proposed and evaluated. Some of these approaches can be transferred directly to the management issues arising in large Distributed Vision Networks, some need to be modified and for several applications, completely new approaches need to be investigated. A detailed comparison is given in Chapter 3.

This thesis is influenced by the research initiative Organic Computing (DFG SPP 1183). The Organic Computing initiative aims at overcoming drawbacks of current top-down engineering approaches. Instead of designing a system as a static and thoroughly planned automaton with predefined states and behaviour, more flexible approaches are investigated. An Organic Computing system is able to develop and adjust itself to changing environmental influences. It exposes life-like properties, as described by the Self-*

[2]For example, at Athens airport 4.500 cameras are in operation which deliver their video data to central recording servers. In case serious incidents are reported, video data is analysed in the aftermath. Real-time detection of incidents is hardly feasible by manual image analysis.

properties of the system. The following terms have been defined that describe Organic Computing systems. According to [7, 8], an Organic Computing system is

- Self-Organising
- Self-Optimising
- Self-Healing
- Self-Explaining
- Anticipative

For Distributed Vision Networks, these self-* properties can be translated into concrete design features that have been implemented as a part of this thesis:

Smart Cameras are able to cooperate and investigate scenes without human intervention. Therefore, PTZ[3] cameras can be used to reorganise their fields of view by turning their heading. These Smart Cameras *self-organise* their fields of view. The user can set up constraints (priority regions, blind spots) but does not need to take care of the process in detail.

Smart Cameras are further able to arrange their fields of view so that an optimal surveillance coverage is achieved. In Section 5.2.3 it is shown, that the process of optimising the arrangement of the cameras' fields of view is an NP-complete derivative of the art gallery problem [9]. Distributed heuristics that help to find close to optimal solutions to this and related problems are presented in Chapter 5. The heuristics ROCAS and DMCtrac are examples for a distributed heuristics that bring *Self-Optimisation* properties to a Distributed Vision Network.

Self-Healing in Distributed Vision Networks implies that Smart Cameras are not only able to detect system failure but also to react in order to compensate its effects. Failure detection mechanisms allow the detection of fail-stop errors of single nodes. An appropriate reaction to a fail-stop error of a single Smart Camera is to re-arrange the cameras' fields of view so that the area covered by the failing node is covered as good as possible by neighbouring nodes. Hence, the overall system performance decreases (a failing node leads to a decrease in area covered by the surveillance system) but the Smart Cameras are able to partly compensate this loss in surveillance coverage (graceful degradation).

The interaction of Smart Cameras and humans (e.g. security staff) becomes possible with Mobile Alarm Management Terminals. For example, a Smart Camera system can guide security personnel to the position where an incident happened. Instead of just

[3] Pan, Tilt, Zoom

raising an alarm, a *self-explaining* system illustrates its actual behaviour (e.g. 'alarm on a certain area') by sending background information gathered by several cameras positioned in the area the incident happened in. The users of such systems intuitively gain a deeper understanding of how the cameras came to the conclusion to raise an alarm.

Since Distributed Vision Networks are expected to carry out surveillance tasks without human intervention, they need to *anticipate* critical situations in order to prevent fatal incidents. An example for anticipation in Distributed Vision Networks is the following situation: At a train station an unattended suitcase might be detected by a Smart Camera. Before raising an alarm signal, the detecting Smart Camera will communicate with neighbouring Smart Cameras in order to investigate the scene. In case a person has run away from this suitcase and left the building, an alarm should be raised. In case the cameras come to the conclusion, the suitcase has been left since it owner turned a few steps away to a timetable, then no alarm needs to be raised. Thereby, the number of false alarms can be reduced.

This thesis introduces a class of distributed system management algorithms that enable Smart Cameras to take over cooperative tasks by relying on Organic Computing features as introduced above. The following section summarises the major contributions of this work and explains its scientific focus.

1.3 Classification and Scientific Focus

Today's CCTV surveillance systems rely on centralised structures [1]. Video data is transferred from the cameras to centralised control rooms where it is analysed by security staff. Current research focuses on intelligent surveillance networks. This thesis presents a novel network architecture for Distributed Vision Networks, that is tailored to suit the needs arising in large networks. Smart Cameras form mesh networks by self-organising their network infrastructure[4]. This novel approach to the architecture of distributed surveillance systems serves as a basis for cooperative tasks that Smart Cameras carry out to understand scenes cooperatively. The following three main aspects are focused within this thesis and contribute to the system management in large camera systems.

- System architecture: Distributed Vision Networks as introduced above rely on numerous Smart Cameras. Such systems need to support several basic functions. A

[4]A mesh network is a subclass of wireless ad-hoc networks [10]. In an ad-hoc network, nodes connect spontaneously in order to forward data among them. Mesh networks are characterised by a static position of the network nodes. In contrast to this, MANETs (mobile ad-hoc networks) are designed for moving nodes. This thesis focuses on non-moving Smart Cameras.

novel Smart Camera middleware has thus been designed. It is tailored to the demands arising in completely self-organising Distributed Vision Networks. A special focus has been set on the deployment of this middleware on devices with limited computing power. It has been implemented in C++ and is therefore both lightweight and portable as described in Chapter 4. The middleware offers several abilities reaching from the invocation of existing computer vision algorithms to the message exchange between cameras. A special focus is on the overall system architecture.

Local neighbourhoods resulting from spontaneously connected cameras in sending/receiving range do not posses any knowledge beyond their own communication range. Although evaluation results show that the decentralised system performs well in general, specific shortcomings need to be addressed by a central entity. This central instance is needed for example for the notification process in case of alarms (only one camera is expected to call the police, not all cameras that detected an incident). The decision which camera becomes this central entity must not be fixed at system startup but needs to be determined by election algorithms. Thereby, the system becomes robust towards node and communication failure that may occur in wireless networks.

Furthermore, prerequisites that are taken to make the algorithms cope with lossy communication channels are presented.

- Spatial partitioning algorithms: Since the Smart Cameras investigated for this thesis have PTZ abilities, they are able to react to sensory input and use actuators to adapt to changing situations. A management task arising in such sensor/actuator network is to connect the coordination of sensory input and actuator output so that e.g. a tracking of objects with cameras and PTZ abilities becomes feasible. Another problem is the initial adjustment of a camera's heading in order to guarantee high surveillance coverage. The problem of partitioning an area under surveillance is related to a problem that has first been discussed 30 years ago in computational geometry. The art gallery problem considers an n-walled room that has to be observed by museum guards in the most efficient manner [9]. The spatial partitioning and tracking algorithms that are investigated in the following, are distributed heuristics that approximate solutions to the problem of aligning cameras' fields of view according to special user goals, which can be formulated as derivative of the art gallery problem. After having shown in Chapter 6, that the partitioning problem is NP-complete, heuristics are presented to approximate solutions to this problem.

- Alarm management in Distributed Vision Networks: Since Distributed Vision Networks are expected to act without a central control console, other mechanisms that allow for interaction between users and the system need to be provided. For example, users can switch between various modes of operation to control the camera network (e.g. start searching a specific object). For surveillance applications, guards are in action that patrol the surveillance area and react after they have been informed by staff working at the central control console. For some applications it seems useful to connect patrolling guards directly to the Smart Cameras. This is achieved by mobile devices that connect to the cameras by a wireless communication channel. As long as the user of such terminal remains in communication range of the camera system, relevant data can be transferred from the cameras to the mobile device.

  With this thesis, an election algorithm is presented, that allows cameras to elect a leader among them. This leader is in charge of informing security staff about the cooperative decision a set of Smart Cameras has agreed upon. An adaption of this algorithms allows to notify mobile guards quickly by discovering a network route to the Smart Camera which interacts with the mobile device of security staff.

1.4 Outline

This thesis is structured as follows. Chapter 2 gives a definition of the system model. Apart from formal problem statements, the current state of the art in the research areas embedded systems and computer vision is reflected. Several basic assumptions are taken from this analysis and motivate the need for system management in Distributed Vision Networks. Chapter 3 gives an overview of related work. Current research on Smart Cameras focuses on basic algorithms like cooperative calibration of cameras or object tracking with multiple cameras. These and other works are presented and analysed and their respective shortcomings are discussed. Chapter 4 describes the Smart Camera middleware that has been developed in order to provide an interface between computer vision algorithms and the system management algorithms that are investigated in the remainder of this document. Chapter 5 introduces distributed algorithms for system management. The evaluation of the algorithms' performance in both a simulated and real environment are given in Chapter 6. Chapter 7 contains the conclusion of this work and gives an overview of resulting future research opportunities.

Chapter 2

Distributed Vision Networks

This chapter contains a general overview of the anticipated system model. It introduces methods and paradigms from the research fields computer vision, embedded systems and ad-hoc networking which all have impact on the architecture and algorithms for Distributed Vision Networks as presented in this thesis. This Chapter introduces the reader to the research questions that are answered in this thesis, provides general background information and defines a set of assumptions taken.

2.1 Definition: Smart Camera

Each Smart Camera is an autonomous node containing a CCD sensor, processing capabilities (CPU, memory, etc.) and a communication interface. Common surveillance systems usually rely on cameras, that submit image data to central control instances. According to Velastin et al., 'Intelligent vision systems' are based upon central servers carrying out image analysis and storage [1]. The system structure of Smart Camera systems is different from those systems that are in operation today and 'Intelligent vision systems'. By analysing video data within the camera, no image data leaves the camera as long as no predefined incidents occur. Smart Cameras need to contain a computing unit in order to carry out image analysis and handle the cooperation between multiple cameras. An overview of existing Smart Camera prototypes is given in Section 2.3. All of these cameras have in common, that they collect image data from a CCD sensor (often by using an FPGA[1]). The typical resolution of today's cameras is $720 * 576$ pixel (TV quality) but cameras with higher resolution (e.g. QXGA, $2.048 * 1.536$ pixel) are beginning to gain

[1]Field Programmable Grid Array

more acceptance for surveillance applications. The recorded image data is analysed on DSPs[2]. Complex computer vision algorithms and higher image resolution require higher computational capacities. The performance of the computing units also has a tremendous impact on the nodes' power requirements: simple Smart Cameras capturing low resolution data and carrying out only trivial analysis of images are known to work with battery power only [11]. More sophisticated image analysis requires more energy which can currently only be provided in mains operation. A communication interface is necessary to enable Smart Cameras to cooperate with each other. Apart from wired networks (e.g. IEEE 802.3 Ethernet) wireless network devices can be used (IEEE 802.11 WLAN). The amount of data exchanged between Smart Cameras is typically lower than in today's camera networks. Instead of transferring video data, only aggregated information needs to be transferred. A user of the system is enabled to acquire further information from certain cameras in case incidents have been detected. Thus cameras transfer video data only on rare occasions and these data transfers affect only those parts of the network where an incident happened in.

A Smart Camera needs to have information about its position and orientation in order to cooperate efficiently with neighbouring cameras. For non-mobile cameras, this can be obtained by manual configuration when the cameras are deployed or by camera calibration techniques using feature points as described in [12]. For mobile cameras the current position may be obtained by appropriate positioning technologies such as GPS in outdoor scenarios [13] or by IEEE 802.11 WLAN positioning [14] in indoor scenarios. The Euclidean distance between cameras and objects under observation can be derived from computer vision algorithms, like described in Section 2.2. These prerequisites need to be met in order to allow Smart Cameras to self-organise their behaviour and form Distributed Vision Networks as described in the following of this thesis.

In the following computer vision algorithms that meet these prerequisites are introduced shortly. Since a detailed description is not within the scope of this thesis, the reader is referred to publications of the respective authors as denoted below.

2.2 Computer Vision

As stated before, large security systems that make use of video cameras are mainly used to record data rather than detect incidents in real-time. Current research advances in the area of computer vision allow for automated surveillance applications and have tremen-

[2]Digital Signal Processors

dous impact on the development of Distributed Vision Networks. Since computer vision is essential for Smart Camera applications and the design of Distributed Vision Networks, this section contains a short overview of the state of the art in the field of computer vision. This enables the reader to gain a deeper understanding of the calculus behind design decisions taken in this thesis. Existing computer vision algorithms offer abilities that are taken for granted for the architecture and algorithms as investigated in the following chapters. The architecture presented in this thesis has mainly been evaluated by simulation experiments. Since no real-world computer vision is used in this simulation environment, several assumptions must be made. These abstractions are introduced in the following. For example, the detection and recognition of moving objects is fundamental for the tracking algorithm that is described later on.

The robust detection of objects, their size and position estimation is part of current research. The computer vision algorithms used in Distributed Vision Networks are assumed to carry out the following tasks:

- (1) Position estimation and camera calibration
- (2) Movement detection
- (3) Object detection
- (4) Object recognition
- (5) Behaviour analysis

Current computer vision algorithms support the functionalities as mentioned above. Different approaches to each of these points exist and offer different capabilities. Some algorithms are more accurate than others but may be more compute intensive. Each problem is addressed in the following and different existing solutions are described and compared to each other.

2.2.1 Camera Calibration

Camera calibration is needed for several computer vision algorithms that are mentioned in the following. The basic idea behind camera calibration is to calculate real world distances from an image that has been captured by a camera. The calibration process of a camera reveals two important types of parameters: intrinsic and extrinsic parameters. Intrinsic parameters (such as focal length) allow to transform from the image coordinate system into the camera coordinate system. The extrinsic parameters allow for the reconstruction

of the world coordinates from the camera coordinates (camera position, translation and rotation). A Smart Camera measures the size of an object in pixels, and the position is derived from the image coordinate system. Camera calibration allows to compute the real world position and size of an object. Figure 2.1 shows the process of camera calibration.

A robust and well performing method to calculate both intrinsic and extrinsic camera parameters has been proposed by Tsai [15]. This method is aimed at computation of the external position and orientation of the camera relative to the object reference coordinate system as well as the effective focal length, radial lens distortion, and image scanning parameters. Nowadays, camera calibration has matured into products and is used for various kinds of applications. Recent advances allow for high accuracy measurement as e.g. needed for minimally invasive surgery [16]. For the following, we assume all Smart Cameras to be calibrated. Thereby the position and size of objects can be derived from the images captured by the camera.

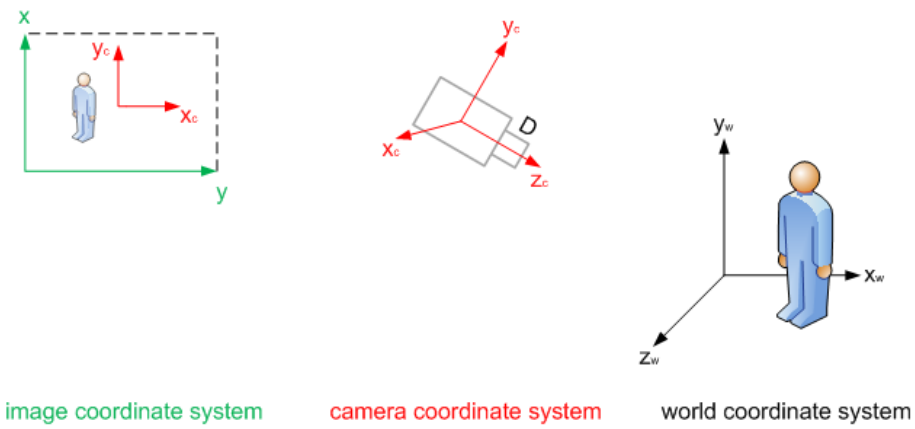

Figure 2.1: Camera calibration process: transform image coordinates to camera and world coordinates

Cheng et al. developed a method for obtaining vision graphs for Distributed Camera Networks with pre-installed, static cameras, see [17]. The vision graph allows to determine, which cameras share an overlapping field of view. The cameras observe an area from different viewpoints imaging large parts of the same environment. Each camera in the network encodes a set of distinctive and approximately viewpoint-invariant feature points and broadcasts them as a digest throughout the network. Each receiving camera decompresses this digest and constructs a vision graph of the camera network. Cameras being connected image parts of the same environment. The graph helps calibrating the network by passing messages along the graph's edges to recover a 3-dimensional structure and camera positions in a distributed manner. This work is an example for a distributed management algorithms that makes way for calibration in camera networks.

2.2.2 Movement Detection

A common method to analyse video data in order to detect objects and their movements is *background subtraction*. The foreground objects are detected by calculating the difference between the current frame and an image of the static background of the scene. Thereby a simple event detection becomes possible: if $(frame_i - background) > Threshold$, a change in the scene is detected. This simple algorithm allows to separate fore- from background and can be used to implement an activity monitoring. Acquiring the static background of a scene is difficult and the robustness heavily depends on the area of application. Several refinements have been developed over the last years in order to cope for example with illumination, motion changes and geometric changes of the background. While searching an appropriate algorithm for tracking piglets in cages, the running (or median) average has been introduced by McFarlane et al., see [18]. The formula that computes a running average over the whole scene to extract the background is

$$B_{i+1} = \alpha * F_i + (1 - \alpha) * B_i$$

The learning rate α determines how fast foreground objects are considered as background objects. A thumb-rule is to set $\alpha = 5\%$.
An improvement is to compute the background model as a chronological average from the history of each pixel. This leads to a rather high memory consumption of the algorithm, since the last n images of a picture need to be hold in a history. Keeping a history of each pixels values over the last n frames allows for more sophisticated foreground detection algorithms. By fitting a Gaussian distribution (μ, σ) over a histogram, a probability density function (PDF) is derived. By updating the PDF for each video frame the threshold adapts to changing backgrounds. By combining more than one PDF, the modelling of multimodal background becomes feasible, at least for a number of pre-defined modes of operation [19]. The usage of multiple Gaussian has been refined by Oliver et al. [20]: they defined eigenbackgrounds that base upon an eigenvector decomposition to reduce the input space and results show, that the calculation of a background model can thereby be sped up significantly.
To summarise this short introduction, one can state that moving object detection has been investigated thoroughly. Several robust and lightweight (in terms of memory and compute time consumption) exist. A good starting point for further information on this topic is given in [21]. Cucchiara et al. present a comparison between different approaches - some of which have been introduced above.

2.2.3 Object Detection and Recognition

The detection and recognition of objects is an important capability in Distributed Vision Networks. Detecting objects means to identify their positions in an image whereas the recognition of objects is concerned with identifying the object itself. E.g., a face detection algorithm returns the positions of faces in an image, including the position of noses and eyes. A face recognition algorithm matches the features of these faces to a database and returns the identity of the persons.

Object Detection

Object Detection has matured over the last years and is now also commercially available. Even standard digital cameras for the customer market offer a face detection mechanism that allows to focus faces automatically[3]. Viola and Jones [22] developed a powerful algorithm that allows to detect objects in images in real-time. Although it is often used for face detection, it can be trained to detect any kind of objects. Their approach bases upon so called Haar-like features. These features are sums of pixels in rectangular areas and can be described as basic Haar functions. A variant of the machine learning algorithm AdaBoost [23] is used to both select the best features and to train classifiers that use them. In order to enable real-time processing of images, cascades of classifiers are used. Strong classifiers that yield high detection rates are used prior to those that are weaker. A classifier cascade can be seen as a decision tree that categorises whether an image includes an object and determine its position in image coordinates.

Object Recognition

Recognising objects in images faces several challenges: partial occlusions, viewpoint changes, varying illumination and cluttered backgrounds are just a few of them. Depending on the field of application, different image features can be used for recognition. For tracking applications, the colour histograms of objects can be used [24]. In case an object needs to be handed over from one camera to another, the colour histogram of the object is transmitted and used for recognition on other cameras. The CAMshift algorithm proposed by Bradksi has first been used for tracking of faces [25]. More accurate results in terms of recognition rate can be achieved by the use of SIFT[4] features [26]. SIFT features are robust towards scale invariants and overcome drawbacks of histogram based recognition approaches. Since the complexity of the SIFT algorithm is rather high, current

[3] E.g. Casio EXILIM EX-Z1

[4] scale invariant feature transform

research results document mainly the use of histogram functions for tracking [27, 28]. For tracking in Smart Camera networks, the use of eigenfaces [29] for face recognition may be used. Similar to SIFT features, this complex algorithm is currently not applicable for real-time applications and serves as an example for algorithms that can be deployed on Smart Cameras in future as soon as their computing capacity allows for such complex computations in real-time. Shi and Tomasi [30] developed an algorithm, that detects *Good Features To Track* in an image. These features can be used to recognise objects that have been detected on a first camera and reappear on a second one. *Good Features To Track* are image areas that expose a high difference in contrast or brightness to surrounding areas or form edges. An application for the use of the Shi-Tomasi algorithm are e.g. mobile robot application where surrounding areas are scanned and investigated to allow for robust navigation. *Good Features To Track* can be used to detect the movement direction of an object. By calculating the movement vector between a pair of features in two subsequent frames, the direction of motion can be detected. This approach is based upon the work of Lucas and Kanade [31]. In this early work, the local calculation of motion vectors is described and evaluated for a stereo vision system.

2.2.4 Behaviour Analysis

Future Smart Cameras will be able to recognise human behaviour. In order to displace humans in front of monitor walls for manual video analysis, the detection and interpretation of human actions need to be designed in form of appropriate algorithms. The following section is way too short to give an extensive overview but gives two examples that describe how cameras can analyse human behaviour. The detection of human behaviour has been investigated by Cupillard et al. in [32] for the purpose of securing underground stations. By extracting foreground objects and measuring their speed, humans are detected and their behaviour is classified. A person moving fast and another person falling nearby indicates a fight between these two persons. The definition of regions of interest allows the detection of persons blocking entrances or jumping barriers. This system has been evaluated in a metro station in Paris and results show, that the system helps to unburden security staff from trivial monitoring tasks. The combination of rather low-level computer vision algorithms (foreground detection, definition of regions of interest) allows for the robust detection of serious incidents.

Recent advances in the field of human action representation have been published by Yilmaz et al. [33]. They present a system, that combines 2-dimensional shapes of moving persons from subsequent frames and form 3-dimensional volumes. Predefined volumes

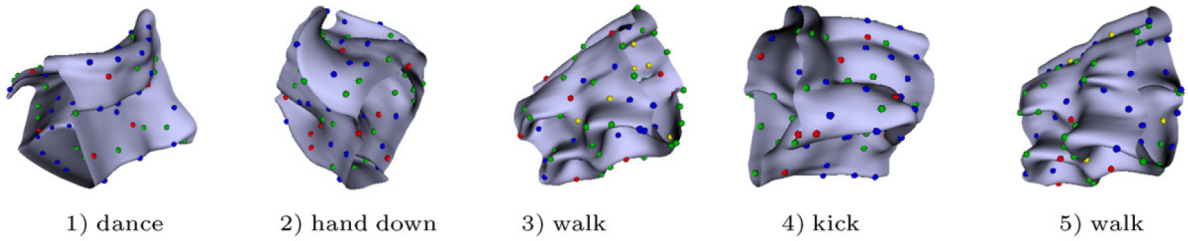

Figure 2.2: Examples for 3-dimensional movement shapes and corresponding human behaviour as shown in [33]

can then be matched to human actions. Figure 2.2 shows detected human behaviour.

The aforementioned works still lack the overall system aspect. Although computer vision seems to have matured over the years, it remains unclear, how thousands of cameras that are placed e.g. in underground stations can cooperate to analyse events and which demands arise in terms of network infrastructure. A centralised approach does not scale very well and might lead to a low performance and high cost for system enhancement. A Distributed Vision Network relying on Smart Cameras is able to solve these problems as presented in the following.

2.3 Embedded Systems: Smart Cameras

Smart Cameras are embedded systems, that consist at least of a CPU, memory, a communication interface and a CCD sensor. For different fields of application, various types of Smart Cameras have been built. Several research projects have been concerned with the problem of building high performance, low power Smart Cameras. These systems usually rely on FPGAs and DSPs. Other Smart Cameras base upon x86 processors that have recently become available in very small size, at low cost and with low power consumption. The following section describes some exemplary Smart Camera hardware platforms. Apart from Smart Camera prototypes built at research laboratories, early commercial products have lately become available. Their performance and field of applications is shortly described here, too.

Table 2.1 shows an overview of current Smart Cameras and highlights their special capabilities. These devices are introduced in more detail in the following.

At TU Graz, Rinner et al. developed the so called SmartCam, that consists of an Intel IXDP 425 development board with an Intel IXP 425 network processor (533 MHz). This system serves as the communication interface and connects the processing unit and the sensing unit via a PCI bus (133 Mhz) and provides basic networking functionalities (such

Vendor	Device	Processor	Performance
TU Graz	SmartCam	TMS 320 DSP	High
Stanford WSNL	MeshEye	Atmel AT91SAM7S	Low
University Aubière	SeeMOS	Altera Stratix	High
Matrixvision	MVblueCougar	Motorola MPC4825	Medium
SONY	XCI-V100	VIA Eden x86	Medium
Universität Hannover	SRA Smartcam	Intel Atom x86	Low

Table 2.1: Smart Camera Hardware

as USB, Ethernet and expansion slots for WLAN and GSM/GPRS). The sensing unit consists of a CMOS sensor and an FPGA. The image data has a resolution of $640 * 480$ pixels. The data acquired by the sensing unit is sent to the processing unit. The processing unit consists of up to four TMS320-C6415T DSPs (Texas Instruments). Each DSP offers up to 8.000 MIPS (when running at 1 Ghz). Again, an FPGA is needed for each DSP to provide programming interfaces such as I2C. These DSPs are equipped with 1 MB internal RAM, which allows to carry out computer vision algorithms efficiently. The peak power consumption of the system is estimated to be $30W$. The multi processor system offers a computing power of up to 32.000 MIPS, which indicates a rather high performance. Several applications have been implemented to evaluate this architecture. For example a traffic surveillance scenario for the detection of lost cargo in tunnels has been set up as well as a cooperative tracking of persons with two SmartCams, see Chapter 3 for more information on their work.

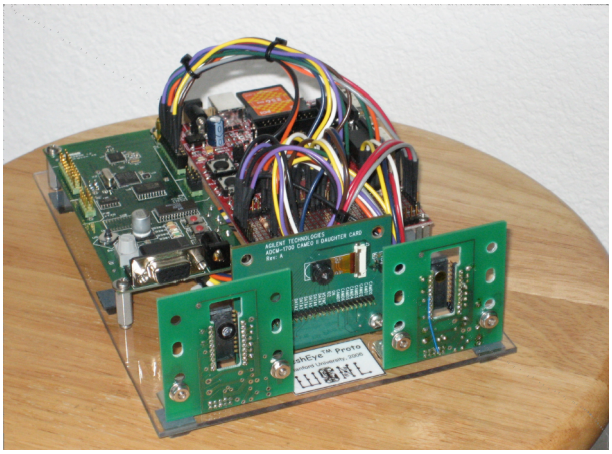

Figure 2.3: MeshEye mote, developed by Aghajan et al. [11]

At Stanford Wireless Sensor Networks Laboratory, Aghajan et. al developed the so called MeshEye Mote, see [11]. This device is based upon an Atmel AT91SAM7S controller. The 32 Bit RISC processor can be clocked at up to 55 MHz. It is connected to

three image sensors: two low quality sensors, that acquire images of $30 * 30$ pixels with a colour-depth of 6 Bit grayscale and one CMOS sensor that delivers $640 * 480$ pixels (VGA resolution). The mote can be extended to support eight low quality sensors. The rational behind using these low cost sensor is, that the node constantly observes the surrounding area for changes. In case a scene changes drastically, the VGA sensor is activated, so that a more detailed view of the scene can be analysed. The communication interface of the MeshEye mote is a IEEE 802.15 (ZigBee) device, that is connected to the main processors USB hub.
The MeshEye mote is equipped with rather weak computing resources. It is hence not suited for high performance computer vision tasks but in return offers very low power consumption. The device can be powered by 2 AA batteries for several days. The runtime of course depends on the number of incidents that need to be analysed with the high quality image sensor. Figure 2.3 shows the MeshEye mote.

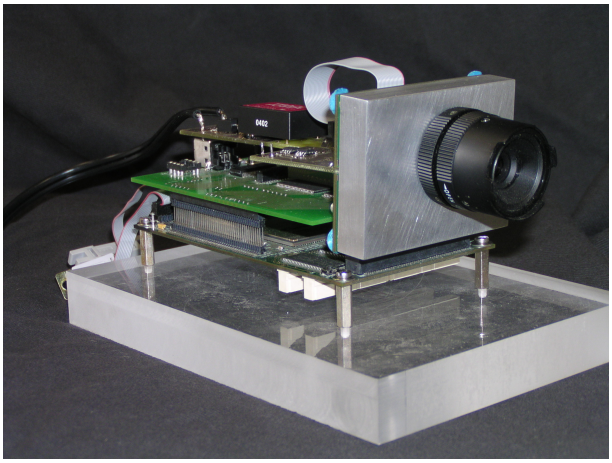

Figure 2.4: SeeMOS node developed by Berry et al.

François Berry et al. from LASMEA (Blaise Pascal University in Aubière) developed the SeeMOS node. This Smart Camera is split up into three layers, as can be seen from Figure 2.4. The CCD sensor (at the front) is connected to an FPGA board (ALTERA Stratix EP1S60F1020C7) that plays the central role in this system. The FPGA board is connected to a communication board and a DSP board. The communication board is in charge of connecting the SeeMOS node to other devices via USB and IEEE 1394. The DSP board is used to speed up basic functionalities such as FFT and filtering. In [34] an evaluation of this architecture is presented. A tracking algorithm that enables the camera to track an $32 * 32$ template over the whole visual field $(2.048 * 2.048)$ is described. The high performance of the architecture leads to a framerate of 55 frames per second.

Fleck et al. developed a Smart Camera software framework called SmartSurv [35]. By using SmartSurv, cameras are enabled to detect abnormal behaviour of persons. For evaluation purposes, in a residential home elderly people have been observed. In case the Smart Cameras detect a person falling, an alarm signal is sent to inform staff. The cameras are supplied by Matrixvision, a company specialised in computer vision for industrial applications. The Matrixvision cameras as used for SmartSurv consist of a CCD sensor, an FPGA (Xilinx) for low-level computation and a PowerPC processor for other tasks. The mvBlueCOUGAR uses a 400 MHz Motorola MPC 4825 CPU (including MMU and FPU) and an embedded Linux operating system. The camera is equipped with 64 MB SDRAM (64 Bit, 133 MHz FSB) and a 32 MB Flashdrive as background memory. A special 4 MB Flash memory is used to save the bootloader, kernel and system configuration. An IEEE 802.3 Ethernet interface is used for communication with other devices. The camera does not only send video data but also receives software updates through this interface.

Recently, x86 processor based cameras have become commercially available. SONY offer the XCI SX1, a CCD camera for industrial applications that incorporates a VIA Eden x86 processor. The system comes with Linux or Windows XP embedded and an image processing library that allows for edge detection. A network interface (IEEE 802.3) provides connection to other camera inside the system. The availability of commercial products shows, that Smart Cameras may soon be used for different application areas. The cooperative behaviour of large interconnected camera systems can only be ensured by the use of an appropriate system architecture as presented in this thesis.

Since commercial products are still only available at very high cost and scientific prototypes are not sold, a Smart Camera prototype has been set up with off-the shelf components for the evaluation of the architecture presented in this thesis. The basic hardware is a miniature sized ITX mainboard with an Intel Atom 330 CPU (Dual Core, 1.6 GHz). It is equipped with 1 GB RAM and 4 GB background memory (SSD). Interfaces to cameras are IEEE 1394 Firewire, USB and Ethernet. The communication interface can either be IEEE 802.11 WLAN or IEEE 802.3 infrastructure LAN. Although this prototype lacks the power of custom built FPGA/DSP based cameras, it represents a cross section of commercial and scientific Smart Cameras.

2.4 Wireless Sensor Nodes

A wireless sensor node is comprised by at least a battery, a processor, memory, a sensor and a communication interface. Although Smart Cameras are a special type of wireless sensor nodes, the term wireless sensor node is often associated with special devices and

software as introduced in the following section.

A typical sensor node is the Mica mote that has initially been developed at UC Berkeley. It makes use of an Atmega 128L processor and is available with various receivers. Powered by 2 AA batteries, Mica motes are able to operate for up to one year and deliver environmental data like temperature, humidity or air pressure. The memory of the Mica motes is up to 640 kByte, which suffices for a lightweight operating system and more than 100.000 measurements. Figure 2.5 shows an image of the Mica mote that is available commercially from Xbow Inc[5].

The work on wireless sensor networks is driven mainly by the need to acquire environmental information. For example, in order to investigate climate changes, sensor nodes have been spread over glaciers in Norway [36]. These sensors deliver temperature information, air pressure and tilt information to detect movements in the glacier. The nodes make use of a minimalistic computing unit that acquires sensor data periodically and emits this data on the network interface. All data gathered by the sensors is sent to a central server, where the data is stored and analysed. Another application of sensor networks is grape monitoring [37]. Sensors have been deployed on a vineyard and deliver exact temperature and humidity information. Thereby, precise plant care becomes possible. Again, the sensors use a low power computing unit to acquire and transfer data to a central server. Since the sensors are usually intended to be distributed over large areas, they are designed to work for a long time without their batteries being recharged. For outdoor applications, solar power can be used to recharge the batteries. The main challenge in wireless sensor networks is the power consumption of the sensor nodes. Especially, the communication between nodes is usually rather energy consuming. Hence, lightweight and well performing management algorithms have been developed that allow for efficient usage of battery power by adapting the network traffic to the current power state of the nodes. Some of these algorithms can be re-used in Distributed Vision Networks, that are a sub-class of wireless sensor networks as described in the following.

The sensors of Smart Cameras deliver much more complex data than common wireless sensors - the memory consumption of image analysis does by far exceed the capabilities of common sensor nodes like the Mica mote shown in Figure 2.5. Since the data transfer between cameras is bandwidth and energy consuming, video data needs to be analysed within the camera in order to derive abstract information that can be transmitted over the wireless channel. This bears the problem, that analysis of video data requires powerful computing resources that are not provided by classical wireless sensor as described e.g. in [39]. The research on cooperating Smart Cameras is therefore focused on wired networks

[5]http://www.xbow.com

Figure 2.5: Berkeley Mica mote [38]

with mains powered prototypes, as described in Section 2.3. In future, more powerful computing units will become available that allow to carry out image data analysis on small wireless nodes as described in [11]. Aghajan et al. present a first prototype of a wireless sensor node with CCD sensors for data acquisition.

In general, it can be stated that wireless sensor nodes have a software architecture, that is similar to the software needed for Distributed Vision Networks. Several examples are described in Chapter 3. The main differences result from the high computational requirements for image analysis (which currently prevents the use of battery powered camera systems) and the camera's ability to use actuators to change the alignment of their field of view. Furthermore, the context data acquired by cameras in a cooperative manner can be used for system management tasks (like key generation for encryption as filed for a patent [40]). Thus, applications beyond the scope of classical wireless sensor networks become feasible. The hereby arising demands with respect to distributed management algorithms are addressed throughout this thesis.

2.5 Summary

This chapter contains a description of the basic components of Distributed Vision Networks. Several computer vision algorithms are introduced that can be used in addition to the system management algorithms presented in this thesis. Furthermore, hardware platforms of Smart Cameras are introduced to give an insight where system management algorithms can be used and which prerequisites need to be met. Since research work on wireless sensor nodes is partly related to the work presented in this thesis, a short

introduction to the hardware and software used for common wireless sensor networks is given.

Chapter 3

Related Work

In future, Smart Cameras will act autonomously and cooperate to conclude online whether an event they detected is critical and an alarm has to be raised or not [41]. The burden of analysing large amounts of data will be shifted from human operators in control rooms to the computing units of networked cameras. As introduced in the previous chapter, the work on Distributed Vision Networks is approached by several research areas. This chapter focuses on system management algorithms for such networks and presents related work in this area.

Distributed Vision Networks are a sub-class of common wireless sensor networks. Thus, the impact of recent developments in this research area on Distributed Vision Networks is discussed in this chapter, too. Wireless sensors networks and problems arising in context of their deployment and management have been discussed thoroughly. A short overview is given in the following. Apart from networking issues and basic communication algorithms, a short introduction on middleware concepts that can be applied to Smart Cameras is given. Furthermore, in Section 3.3, a detailed overview of work in the area camera alignment is presented. The adjustment of PTZ heads to varying user demands and environmental changes is a fundamental system management task. Several existing approaches are introduced and their shortcomings are discussed. Finally, an overview of related work concerning user interaction and alarm management is given.

3.1 Basic Algorithms in Wireless Sensor Networks

As mentioned before, Distributed Vision Networks are a sub-class of wireless sensor networks. The most important constraints in sensor networks, caused by limited energy

resources, are the limited communication distance and bandwidth. Ad-hoc networks are deployed where no infrastructure exists and nodes with limited communication range form networks on-the-fly. A common communication scheme is broadcasting. Instead of sending messages to nodes by addressing each node separately, messages are sent to all nodes in communication range (one-to-all cardinality). In order to reach those nodes that are not in direct communication range, flooding can be used. By forwarding each incoming message to all neighbouring nodes, the message eventually arrives at all nodes inside the network. A problem arises, in case too many messages are being forwarded simultaneously. These so called broadcast-storms [42] may lead to a situation where the messages are extinguished due to collisions on the physical medium. In [43], the impact of this effect is analysed in detail.

Another problem is partitioning: in case nodes are out of each others communication range (e.g. due to movement of nodes or moving communication obstacles), the network may split up in different sub-networks. The partitioning of a network has tremendous impact on algorithms used for network management. A detailed specification of partitioning in ad-hoc networks can be found in [44].

Both of these problems, broadcast storms and partitioning can be addressed by repetition mechanisms. By adapting to the current situation of the network, nodes can avoid broadcast storms by forwarding only selected messages that have presumably not been forwarded yet, like introduced in [45] by Khelil. In order to cope with network partitioning, nodes can keep a history of formerly known nodes in sending range. In case a message can not be forwarded to these nodes any more, the network may have been split up. In this case, the message can be re-broadcasted. Thereby, a temporal partitioning can be overcome.

Figure 3.1 shows a typical Distributed Vision Network. Mobile terminals (short MAMT for Mobile Alarm Management Terminal as introduced in Chapter 4) are connected to Smart Cameras via unicast. The communication range of camera A is indicated by the dashed circle. The system is split up in two partitions.

All these communication algorithms depend heavily on the area of application. The node density, movement speed of nodes and timing constraints set by upper level applications constrain the usage of the algorithms. If not stated otherwise, the Smart Camera algorithms presented in this thesis rely on a broadcasting communication scheme. Some algorithms require the communication with direct neighbours only (single hop communication). Other algorithm require multihop communication which can be achieved by flooding based broadcast or more advanced alternatives, e.g. [45]. For alarm management in Distributed Vision Networks as presented in this thesis, reliable multi-hop communica-

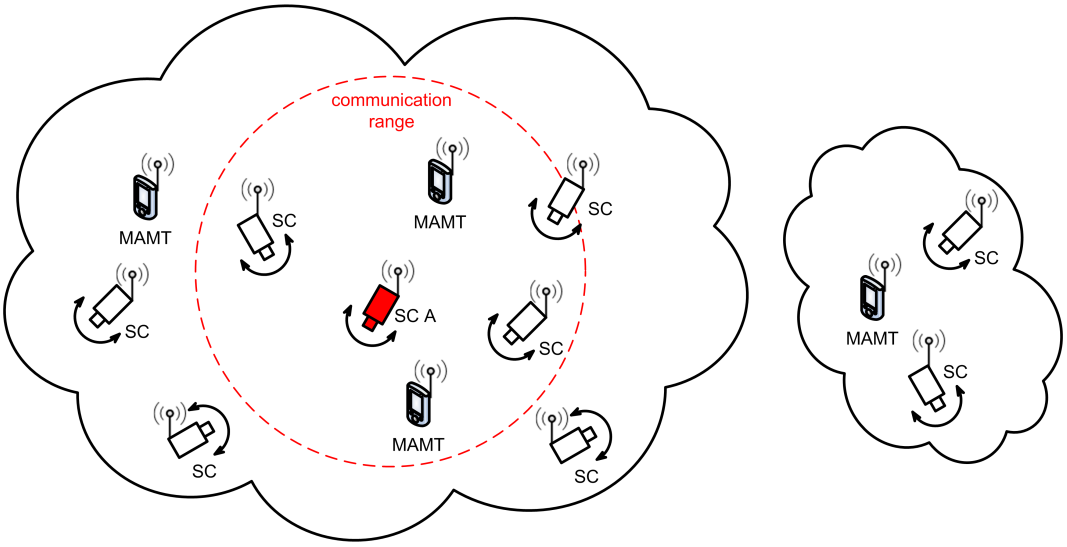

Figure 3.1: Partitioned network with limited communication range of nodes

tion is used. Therefore, existing on-demand-routing protocols (like AODV [46] and DSR [47]) have been adapted and enhanced where necessary in order to suit the special needs that arise in these special kind of sensor network.

3.2 Operating System and Middleware

Each Smart Camera node is running a software framework. The basic functionalities are provided by an operating system. For example, with TinyOS an operating system for wireless nodes is provided [48]. It is lightweight and designed for energy efficiency. Recently, Linux distributions have become available, that are adapted to resource constrained embedded devices, like Ubuntu MID [49]. Since Linux serves as a basis for many scientific and commercial Smart Camera devices, it is used as a basis for the middleware introduced in the following chapter, too.

Sensor nodes and Smart Cameras are usually equipped with a middleware providing functionalities that are not part of applications (that carry out high level algorithms) or the operating system (that carries out basic functionalities like task scheduling and memory management). Typical tasks for a middleware are e.g. event dispatching or the establishment of neighbourhood relationships, message generation and interoperability. Many middleware architectures have been presented until now and several of them might be suited for Smart Camera systems. To name a few, BASE [50] and AMUN (OCμ) [51] have been developed and provide component based flexibility and adaptivity in terms of changing environments. They offer many of those functionalities that are needed for Dis-

tributed Vision Networks. Still, they lack some important points. For example, AMUN comes with a vast variety of functions that allow for the design of organic computing systems. E.g., in [52] an artificial immune system for AMUN is presented. Thereby, malicious events can be detected and eliminated. Other middleware architectures, such as BASE, offer a component based approach. By adapting existing and adding new components, a BASE architecture that suits the needs of a Distributed Vision Network could be developed. In general, it remains unclear how well existing middleware approaches can cope with the demands arising in Smart Camera networks. Especially real-time capabilities are still subject of ongoing research. A further important aspect of a Smart Camera middleware is the close coupling of all algorithms to the image data that has been acquired by the CCD sensor. This data needs to be analysed in real-time and needs to be accessibly from different components (that might either be part of the middleware or high level applications). Apart from detecting events of interest, several system management algorithms can benefit from context information (e.g. functional monitoring and encryption). Therefore, an image data centric approach seems most appropriate and has been implemented in context with this thesis [53]. Instead of adapting an existing middleware to the needs arising in a Smart Camera system, a new middleware has been designed that connects special camera hardware (CCD sensor, PTZ actuator) and provides basic communication mechanisms. Thereby, a lightweight and highly specialised middleware becomes available, that exactly suits the needs arising in Distributed Vision Networks. This middleware serves as a basis for the system management algorithms that are discussed throughout this thesis. The detailed architecture of the customised middleware is presented in Section 4.2.1.

3.3 Cooperative Tasks in Distributed Vision Networks

Cooperative tasks, like camera calibration and scene analysis, are subject of research in the area of computer vision. In contrast to those tasks, the following sections present works, that are related to the system management tasks that are focused in this dissertation.

3.3.1 PTZ Camera Alignment and Spatial Partitioning

PTZ cameras suffer at first sight from several drawbacks in comparison to statically installed cameras: They are prone to mechanical failure since the drives used to pan/tilt and zoom are exposed to mechanical stress. Additionally, PTZ cameras are a lot more expensive than statically installed cameras. Despite of this, for several fields of application,

PTZ cameras offer benefits. A single PTZ camera is able to observe a larger area with a better image resolution than a statically installed camera. Furthermore, the cost arising for the installation of multiple static cameras usually outweigh the cost of a single PTZ camera. The use of numerous statically installed cameras may not be appreciated for aesthetic reasons or building constraints, too.

In large systems, cameras can use their PTZ abilities to achieve an efficient scene coverage. Efficient means, that the cameras adjust their fields of view in such way, so that the scene coverage is maximum. Cameras can switch (either automatically or due to users' requests) to other tasks like object tracking or stereo vision applications and change their heading and investigate scenes cooperatively. Usually, cameras will switch back to an efficient observation mode after they finished their task to save resources and avoid unnecessary wear-out of PTZ drives. This state of steady observation, with the goal to avoid cameras moving, is achieved by spatial partitioning algorithms.

Strategies to increase and measure surveillance coverage are proposed by Mundhenk et al. [54]. Their ideas have been simulated and tested in the iRoom of the University of Southern California. Several cameras that are connected to a central server track objects and periodically change their fields of view in order to achieve an even observation of an area. Results published consider movement strategies of a single pan/tilt camera. Cooperation between cameras is not discussed. The visualisation methods used by Mundhenk et al. have been proposed by Hew in [55]. Hew's approach of visualising surveillance coverage by fading colours on a 2-dimensional map is suited well for the visualisation of partitioning and coverage algorithms. It has therefore been implemented in the toolchain used for the evaluation of algorithms presented in this thesis.

Aligning cameras in order to cover a maximum area is related to the art gallery problem. In the 1970s, Victor Klee threw up the question how many guards are needed to completely observe an art gallery room. Vasek Chvátal showed, that $\lfloor \frac{n}{3} \rfloor$ guards are occasionally needed and always sufficient to cover a polygon with n vertices, i.e. an n-walled room [56]. This problem and derivatives have thoroughly been discussed, see [9]. A related problem is the adjustment of the cameras' viewshed in such way so that optimal surveillance coverage is reached.

Erdem et al. developed an application determining where to place cameras to satisfy task-specific and floor plan-specific coverage requirements, see [57]. This application allows for offline partitioning of a surveillance area under constraints and considers viewing obstacles, fields of view and regions of special interest that need to be observable in high resolution. This thesis focuses on the distributed online partitioning of a given surveillance area without any node having a global knowledge of the system, whereas Erdem et

al. present a planning tool for statically configured surveillance systems.

The aforementioned works base upon unrealistic guard capabilities, e.g. unlimited line of sight. A realistic and formally correct model of a cameras field of view is presented in [58]. A camera is installed at a position v with the coordinates (x, y, z). An area $\mathcal{D}$ is to be observed by the camera that has a limited line of sight r. The viewshed $V(v)$ is the set of points or cells on the surface $\mathcal{D}$ that are visible from v extending out to some maximum distance r from the viewpoint:

$V(v) = f(v, D, r) = \{\delta \in D \mid d(v, \delta) \ \leq r \ and \ \delta \ visible \ from \ v\}$

This definition also considers viewing obstacles: in case an object is placed between v and δ, the line of sight is blocked and δ is not visible from v. This camera model is used throughout this thesis (see Section 5.1 for details).

For mobile robot applications, Gonzáles-Baños and Latombe investigated how a 2-dimensional workspace can be explored in the most efficient manner. Hence, robots move through the workspace (modelled as a polygonal map) and stop at certain points where they start taking images of their surroundings. A solution to the art gallery problem is approximated by using random robot placement. The set of all positions a robot is placed on is the set $\mathcal{G}$. In order to acquire a complete view of the workspace quickly, the robots are expected to cover the total area with the smallest number of images. Out of $\mathcal{G}$, a set $\mathcal{S}$ is searched, that covers all observable points $\mathcal{X}$ on the workspace. This is the *set-coverage problem*, that has been shown to be NP-complete [59]. A common approach to approximate a solution to this problem is the greedy search. The algorithms starts to pick those positions, that contain the largest number of uncovered points. Thereby, a near-optimal solution can be found in linear time.

Murray et al. describe how an optimal positioning and alignment of cameras for security applications in 3-D urban environments can be achieved [60]. By relying on a realistic viewshed and a geographic information system[1], realistic scenarios can be investigated. Urban scenes are modelled with possible positions where cameras could potentially be installed. An offline optimisation is then carried out, to derive the minimum number of cameras and their positions and alignment in order to achieve an optimal surveillance coverage. This offline optimisation solves the Maximum Coverage Location Problem (MCLP). This problem can be formulated in such way so that it becomes solvable by a commercial optimisation tool. This mathematical description of the camera positioning problem and the optimisation approach discussed by Murray et al. offers the possibility to set up cost effective and efficient surveillance systems based upon a centralised offline

[1] A GIS is an information system comprised by hardware, software, data and applications. Spatial data can be acquired, modified, analysed and visualised.

optimisation

Both works mentioned above are closely related to the camera alignment problem that is addressed by the ROCAS algorithm presented in this thesis. Instead of calculating the camera alignment in advance, ROCAS is a distributed heuristic that approximates a solution to the coverage problem at runtime. No central planning instance is needed and the system becomes robust towards node failure and environmental changes.

Positioning cameras in order to achieve a complete scene coverage has been proven to be in NP [61]. Cole et al. use a 3-dimensional environment to place viewing stations in. The area is segmented since the height varies and viewing obstacles are present. Although the cameras have an unlimited viewshed, pits and walls limit the cameras' lines of sight. In case the walls and pits are arranged in a certain way, the general complexity can be described formally. By a reduction from *set-packing*, the NP-completeness of this task is shown. For an overview of complexity classes, cf. to Karp's work [59] or see Section 5.2.4 for a short introduction. In Chapter 5, it is proven that the problem of aligning cameras in order to maximise area coverage as investigated in this thesis is NP-complete, too.

Apart from a static camera alignment, the tracking of moving objects has been investigated. The following section describes approaches to this problem.

3.3.2 Tracking with Single PTZ Camera

In [62], Kang et al. present a system for tracking objects with a single PTZ camera. This system includes mechanisms for adaptive background generation, moving regions extraction and tracking. A mosaicing technique is proposed to project one view onto another view with different pan and tilt angles to allow for seamless tracking with a panning and tilting camera. The background generation relies on an adaptivity model as introduced in [18]. Since the computing complexity is rather low, real-time tracking becomes feasible. The transformation of the real-world to pixel coordinates is affected by the cameras pan/tilt angle and the zoom factor. All these camera parameters are taken into account for the tracking process. As a result, the input video can be presented to the user with the moving object indicated by a bounding box. The system was evaluated and results show, that it is possible to implement robust object tracking with a single PTZ camera. Especially, suitable computer vision algorithms have been refined and applied by the authors. For image analysis, a PC (Intel Pentium 4, 1.300 Mhz) is used. This PC has a computing power that is comparable to available Smart Camera systems.

3.3.3 Tracking with Multiple Cameras

Marking objects and tracking them with cameras that do not have overlapping fields of view requires the exchange of some condensed data describing those objects. In [24], an approach based on a fuzzy logic matching algorithm is proposed by Loke et al. to find the correspondence of multiple targets in a multi-camera network with non overlapping viewsheds. By using the CAMshift algorithm [25], a very high detection rate is achieved. In order to allow for more robust tracking, sophisticated features like SIFT-features [26] or *Good Features To Track* [30] can be used.

In [63], Everts presents a system that can be used to track objects with multiple calibrated PTZ cameras in a cooperative manner. Tracking and calibration results are combined with several image processing techniques in a statistical segmentation framework, through which the cameras can hand over targets to each other. A prototype system consisting of two cameras is presented that operates in real time. Evaluation focuses on computer vision techniques and shows, that realtime tracking of objects using multiple PTZ cameras is feasible. In contrast to the work presented in this thesis, there has been no evaluation how well the system performs for large numbers of cameras.

In [64] a biologically inspired approach to the coordination of two PTZ cameras is discussed. The behaviour of a chameleon's eyes has been studied and modelled in terms of control theory. The resulting PTZ camera control can be used for mobile robots in order to efficiently scan an area under observation with both eyes independently. In case a prey appears (an object the cameras are supposed to focus), tracking of this object with both eyes for stereo vision is performed. Again, no more than two cooperating cameras have been used for evaluation.

In [27], Wolf et al. present a peer-to-peer multi-camera system for multi-object tracking, where different CPUs are used to process inputs from distinct cameras. Instead of transferring control of tracking jobs from one camera to another, each camera in the system performs its own tracking and keeps its own tracks for each target object, thus providing fault tolerance. Experimental results demonstrate the success of the proposed peer-to-peer multicamera tracking system. For the message exchange between cameras, a message passing interface (MPI) is used. Thereby, the correctness of the communication protocol can be proved. A realistic simulation of the system's behaviour in large scenarios is not discussed, especially the impact of the communication network is not investigated. In contrast to Wolf's approach, the focus of this work is on a completely self-organising ad-hoc network establishing local neighbourhoods only.

3.3.4 Master/Slave Tracking

A promising approach towards reliable multi object tracking in Distributed Vision Networks is the master/slave approach. DMCtrac, the tracking algorithm presented in Section 5.4.2 builds upon this approach, too. The master is in charge of tracking an object actively whereas a slave knows about the objects existence but is not responsible for the tracking of this object. The following section shortly describes related work in the field of master/slave tracking algorithms.

In [28], a Smart Camera consisting of a two-stage computation unit is described: An image processing unit consisting of DSPs carries out image analysis, whereas a network processor carries out all high-level applications and communication related tasks. Within the DSP framework, the actual tracking algorithm can be loaded as a DSP task. The tracking is then done with help of the CAMshift algorithm. The handover between different cameras is implemented as follows: First, the subsequent cameras the object possibly moves to are selected and the tracking agent is being migrated into these cameras. During this step, several tracker instances exist. Then, the tracking task is initialised on the slave cameras. If the object is discovered by a Smart Camera in slave mode, this camera will take over the tracking task as a master. The master informs the other slaves which one of them got the tracking task and which one is to be terminated. Due to the usage of motion vectors, the authors minimise the amount of created slaves when following a single object with multiple cameras. Therefore, this approach scales well even in large networks, although the authors do not dwell on this aspect. The network communication in large networks and the reliability with respect to limitations arising by the usage of a wireless communication channels is not reported in the work described above.

For the work presented in [65], Margi et al. decided to use a master/slave approach. In contrast to the approaches introduced so far, they distinguish between *end nodes* and *internal nodes*. The main task of the *end nodes* is to discover objects which enter the network's monitored space and then deliver a message of detection to the next nodes. The *internal nodes* are alerted by end nodes and then track the object further. The main advantage of the distinction between *end nodes* and *internal nodes* is energy saving: in contrast to *end nodes*, which always have to search for possible objects, the *internal nodes* can save energy. Unless they are tracking, they can be on standby and only have to wake up from time to time and listen to messages from *end nodes*.

Ukita has been working extensively on Autonomous Vision Agents (i.e. Smart Cameras), see [66, 67] for major contributions. Apart from computer vision problems, also the tracking of multiple objects with PTZ cameras has been investigated in detail. In analogy to the system model used for this work as presented in Chapter 4, Ukita defines

an AVA-layer (i.e. a Smart Camera), an AVA-Agency layer (Smart Camera group) and inter-AVA-layer (Smart Camera system). Ukita's theoretical abstraction helps to define the neighbourhood dependencies for the tracking task. In case an AVA is searching for an object (freelancer AVA), the camera's PTZ head is screening the area by panning and tilting. After an object that needs to be tracked is detected, neighbouring cameras are informed. In case this object has not been tracked yet, an AVA agency is formed. All member-AVAs of this agency track the object cooperatively. In other words, the *masters* and *slaves* form a group that is called an agency. The 3-D position of an object tracked by the AVAs is calculated precisely and by combining the position retrieved by multiple cameras, the system can cope very well with noisy input data. Although the results of Ukita's work can be regarded as highly sophisticated, they lack an evaluation of their applicability in a real world system that has to cope with faulty communication channels, network partitioning and bandwidth limitations. Similar to all other works presented so far, the main focus of Ukita's work is on computer vision problems rather than on system engineering and networking issues. In contrast to this, we focus on distributed algorithms that allow for a coordination of large camera systems. Distributed algorithms are investigated that make way for a self-organising behaviour of the camera system. A special focus is on networking issues, as e.g. the robustness of protocols towards lossy communication channels. Furthermore, the scalability of these algorithms is investigated in depth in order to evaluate their performance in very large networks with hundreds of interacting cameras.

3.4 User Interaction and Alarm Management

Today's surveillance systems are controlled manually by operators in a central control room. Apart from the arising drawbacks (as lack of fault tolerance and scalability), this central instance is necessary since it is the only interface to the user. Although Smart Cameras act autonomously, several parameters need to be adjusted by surveillance staff. For example, the system might need to be switched from a modest, resource saving alarm level (steady observation achieved by partitioning) to a higher alarm level (tracking) manually. Since the system architecture proposed in this thesis aims at rendering the central control instance unnecessary, other approaches for user interaction need to be considered.

Intuitively, a closer coupling between humans (surveillance staff) and the surveillance system seems to be a valid approach to overcome drawbacks of today's surveillance systems. Several publications evaluate the use of mobile devices to display video data and

alarms raised in surveillance systems as introduced in Chapter 2. Other works report the use of mobile device as receivers for video streams. An important aspect is the encoding and routing of video data with respect to the bandwidth limitations of wireless networks. Although several works consider generic systems for streaming multimedia content from a sender to mobile devices (as e.g. Steiger et al. [68]), this section focuses on surveillance applications for mobile devices that allow users to communicate with cameras and in return acquire information from the cameras.

The alarm management in today's surveillance systems relies on a central control console. A supervisor is in charge of coordinating staff on the precinct (e.g. an underground station) and to detect unusual events with the help of a monitor console where the view of cameras can be projected on. Thus, the supervisor needs to select a set of cameras that will likely help to observe the scene. In [1], an in-depth description of alarm management in Victoria Station, London, is presented. The authors describe in detail which tasks a supervisor carries out and what is done in case an incident is detected. Staff communicates via full-duplex radio devices and a telephone is available for communication with police and firefighters. Since a supervisor is not only in charge of monitoring videos but also handles complaints of staff and passengers, his attention is not always focused on the video images. The authors state, it might therefore be necessary to develop notification methods that do not interrupt the supervisor suddenly but rather discretely catch his attention.

Li et al. present an early work dealing with mobile devices for surveillance applications in [69]. They describe a system called *PDA watch* that enables users to acquire video streams from surveillance cameras. A JAVA based framework has been developed, that allows users to login and register at cameras which in return deliver video data to the mobile device. The data sources (surveillance cameras) transfer still image to a central server which distributes the video data to the mobile devices.

A more sophisticated approach that also delivers streaming video is introduced in [70]. Cucchiara et al. present a home surveillance system that can for example be used to monitor elderly or disabled people. In case serious incidents are detected by surveillance cameras, a notification is sent to a PDA that informs relatives or service personnel. A central server is used to both acquire and analyse image data delivered by the video cameras and encode the video stream to transfer it to connected PDAs. An extensive evaluation with respect to categorisation of the persons behaviour is presented. Results are promising since the detection rate is very high.

In [71], Imai et al. present a system to connect cameras and mobile devices via the Internet. Cameras deliver still images that are analysed by a computing unit. The

images can be retrieved by mobile phones. Therefore, cameras are connected to a special, JAVA-based webserver. The mobile phones are connected to the Internet via a UMTS connection. In case incidents are detected, the server emits an email to registered mobile phones. Users can then decide to retrieve images from the webserver. Evaluation considers the time needed for transmitting image data from the webserver to requesting mobile phones. Results yielded are good and promising, the data transmission takes between $2s$ and $7s$, which seems appropriate for many surveillance applications.

Although several works as mentioned in the previous section consider the use of handheld devices in combination with cameras, up to now no bi-directional interaction has been considered. Since this interaction between Smart Cameras and users is an essential aspect for both basic management algorithms and high level applications, this thesis introduces appropriate algorithms. Apart from notification of guards, the system presented here allows users to control the camera system with a mobile device.

3.5 Summary of Related Work

Figure 3.2 shows an overview of selected related works and the abilities they lack in comparison to the algorithms presented in this thesis.

The design of Smart Camera hardware has been led by laboratories like the Stanford Wireless Sensor Network's Laboratory (Aghajan) or the Institut of Technische Informatik TU Graz (Rinner). The evaluation of the camera's performance in terms of computer vision algorithms has been analysed thoroughly. The aspect of networking and cooperative behaviour has only been discussed shortly in this context.

The most significant shortcoming of existing approaches to camera alignment (Erdem, Cole, Murray) is the lack of online optimisation. The related work on spatial partitioning covers a vast range of visibility problems and offers solutions and heuristics to approximate solutions. This is suited well for planning camera positions and initial alignment in advance but does not consider Organic Computing paradigms, such as self-optimisation at system runtime. A large camera system is prone to node failure and communication disturbances. Instead of planning all possible alignment combinations in advance, ROCAS enables the cameras to adapt to new situations and user goals. Approaches with mobile cameras as addressed by Latombe et al. aim at finding optimal guard positioning. This is related to the *set-coverage problem* and does not solve the task of optimal camera alignment as investigated in this thesis.

The tracking of objects is closely related to the coverage problem addressed by ROCAS. Related work mainly considers computer vision aspects of tracking (Rinner, Wolf)

and only few approaches to the design of self-organising tracking systems (for example presented by Ukita) are known. With DMCtrac an algorithm is presented in this thesis that combines the Master/Slave approach and a distributed system structure. Thereby, an evaluation of tracking systems with respect to requirements on the underlying network become feasible.

A bi-directional interaction of users and Smart Cameras is not documented until now. Existing approaches that suggest a coupling of PDAs and Smart Cameras focus on transmission of video data only. Furthermore, no experiments considering large networks (e.g. with thousands of nodes) are documented.

The summary of related works shows, that computer vision is ready for cooperative object tracking with multiple PTZ cameras. This thesis contributes to the area of networking issues since aspects of wireless communication arising in the context of Distributed Vision Networks have not been covered completely yet. There is little work available that considers the self-organising aspect of Distributed Vision Networks under real world constraints. These constraints are e.g. fault tolerance with respect to a lossy communication channel and the impact of mechanical stress put on PTZ cameras that carry out cooperative surveillance tasks. These issues are addressed throughout the following chapters.

	Spatial Partitioning	Cooperative Tracking	Real world evaluation	Realtime capable	System size, no of nodes	Evaluation w.r.t. network	Mobile User Interaction
Aghajan et al.	no	no	yes	yes	1	no	no
Erdem et al.	yes	no	no	no	theoretical approach	no	no
Cole et al.	yes	no	no	no	theoretical approach	no	no
Latombe et al.	yes	no	no	yes	theoretical approach	no	no
Murray et al.	yes	no	no	no	unlimited	no	no
Wolf et al.	no	limited	yes	yes	2	limited	no
Rinner et al.	no	limited	yes	yes	2	yes	no
Ukita et al.	no	yes	yes	yes	4	no	no
Imai et al.	no	no	no	no	2	yes	yes
Cucchiara et al.	no	no	no	no	2	yes	yes
Li et al.	no	no	no	no	2	yes	yes
Hoffmann	yes	yes	yes	yes	unlimited, at least 1000	yes	yes

Figure 3.2: Comparison of related work

Chapter 4

System Architecture

The most significant design feature of a Distributed Vision Network as introduced here is the avoidance of a central control instance. The decentralised system architecture presented in the following relies on a network of Smart Cameras and mobile terminals. Smart Cameras need to self-organise their behaviour in order to analyse scenes cooperatively and consider the requests sent by mobile terminals. This chapter presents basic functionalities concerning message exchange and failure detection. This serves as a basis for more sophisticated tasks as introduced in Chapter 5.

At first, the networked system architecture is presented. Afterwards, software components and several algorithms are introduced that comprise a middleware for Smart Cameras.

4.1 Networked System Architecture

We assume the exchange of data with neighbouring cameras to take place via a wireless ad-hoc network or a network with static infrastructure. For local communication between neighbouring nodes, a wireless network seems most suitable in terms of easy installation and flexibility towards camera replacement and is therefore investigated in this thesis. Since operators of today's surveillance systems might be interested in reusing the already existing wired network infrastructure, prerequisites are taken that allow to re-use these components. For example, cameras that are out of communication range can be connected by wired networks by using an appropriate routing protocol [72].

Cameras exchange information about their current state (alignment, position of objects, etc.). The use of broadcast communication allows for an efficient usage of the

wireless communication channel and enables Smart Cameras to establish local neighbourhoods. These neighbourhoods carry out tasks cooperatively. A small sized system, that incorporates all elements of a Distributed Vision Network is shown in Figure 4.1. In the following sections the components of such a system are described in detail.

4.1.1 Role Assignment

In a Distributed Vision Network, a Smart Camera can take over one or more of the following roles. The basic role, that is carried out by each camera, is the role of aligning a camera's own viewshed according to the user goals. The camera alignment inside the network is carried out by an algorithm presented in Chapter 5. This algorithm might make use of a central computing instance, a role that could also be taken over by a Smart Camera.

Furthermore, the following roles have been defined:

- In order to interact with users, Smart Cameras can start a webserver application. Thereby, the camera takes over the role of a Smart Camera Webserver (SCW) and provides human-readable information to Mobile Alarm Management Terminals (MAMTs).

- By becoming a gateway, cameras can route traffic to other networks (SC-GW). This is important for large networks that cover wide areas and may fall into partitions as explained in Section 3.1.

- For the alarm management, an election algorithm enables cameras to cooperatively judge whether to raise an alarm or not. Given appropriate computer vision algorithms, a set of cameras can vote for whether an incident is critical and an alarm needs to be raised or not. The elected Smart Camera Leader (SCL) is in charge of collecting votes and informing security staff.

All these roles are assigned dynamically by an election mechanism presented in Section 4.3.4 that chooses a leader among all cameras inside the network. Thereby, a single point of failure is avoided. The concept of roles as introduced above is now explained in more detail.

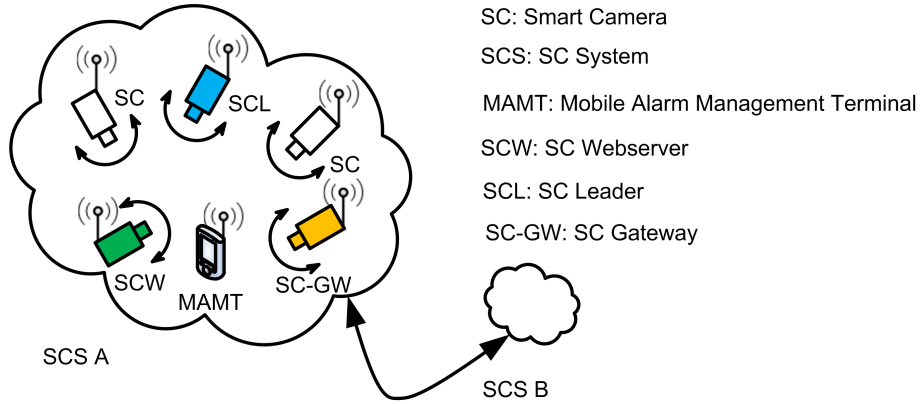

Figure 4.1: Networked system architecture

4.1.2 Smart Camera Webserver and Mobile Alarm Management Terminals

The interaction of mobile terminals and cameras relies solely on a wireless network since security staff is expected to be mobile instead of forced to stay inside a control room. A mobile terminal connects to any Smart Camera in transmission range. Thereby, the camera automatically becomes the routing end point for this mobile terminal. A webserver is started on a Smart Camera as soon as a mobile device connects to it. The Smart Camera that is connected to a Mobile Alarm Management Terminal (MAMT) becomes a Smart Camera Webserver (SCW). This webserver is in charge of routing all relevant traffic from the Smart Camera system (SCS) to the MAMT and vice versa. The communication between MAMT and SCW is a unicast communication, whereas for the communication between Smart Cameras a broadcast scheme is used. The impact of this design decision becomes obvious in the evaluation: frequent broadcasts demand a high bandwidth and are not suited for the transmission of video data. The unicast channel helps to reduce the bandwidth consumption. Without going into detail here, the Mobile Alarm Management Terminals are PDAs or mobile phones that have in common that they are at least able to establish a duplex connection to the camera network and come with a display and input device to allow for user interaction. More information about MAMTs is given in the evaluation in Section 6.7.3.

4.1.3 Establishing Hierarchies by Leader Election

For some applications, it is necessary to establish a hierarchical network structure: In case several cameras have spotted an event of interest, they are expected to analyse

cooperatively whether an alarm needs to be raised or not. This alarm should be sent by one camera only, thus the group elects a Smart Camera Leader (SCL) and lets the leader inform a MAMT.

Another application, a leader can be used for, is the partitioning of an area under surveillance as described in Section 5.3.2. In order to achieve a global maximum in surveillance coverage, local optimisation may not suffice. By electing a leader, global knowledge can be aggregated in the system and be used for global optimisation. It is important to notice that this globally collected data is only used in addition to the local optimisation. Therefore, the failure of a leader does not make the whole system stop working but causes only a decrease in quality (i.e., raising an alarm takes longer or the surveillance coverage reaches just a local maximum). The failure of a leader is detected by a heartbeat mechanism (Section 4.3.3) and automatically initiates an election of a new leader (Section 4.3.4). The evaluation of the system as presented here shows, that a leader is not able to carry out all management tasks arising in a network and that especially for the cameras' alignment for partitioning and object tracking distributed algorithms are suited better, see Section 6.4.10.

As mentioned before, the use of existing infrastructure networks might be necessary in order to bridge large distances between Smart Cameras that exceed their transmission range. A Smart Camera Gateway (SCG) can be elected in order to route traffic between Smart Camera systems. The SGW in Figure 4.1 connects SCS A and SCS B.

After the description of the networked system architecture, the following section deals with the software architecture of a single Smart Camera node.

4.2 Software Architecture

The system management algorithms presented in this thesis have been integrated in a Smart Camera middleware that is presented in the following. Figure 4.2 shows a block diagram of the software components.

Based upon an operating system, a three-layered architecture has been developed. Starting from the bottom, *Layer I* contains basic functionalities such as image acquisition and processing. Furthermore, the physical alignment of a camera's PTZ drives is handled in this layer. For the networking part, a message dispatcher has been incorporated. In order to secure the communication between Smart Cameras, a concept for key generation and dissemination in Smart Camera networks has been developed [40].

Layer II contains the Map Manager. Its main purpose is to keep neighbourhood information and aggregated sensor data (e.g. concerning viewing obstacles). Thereby,

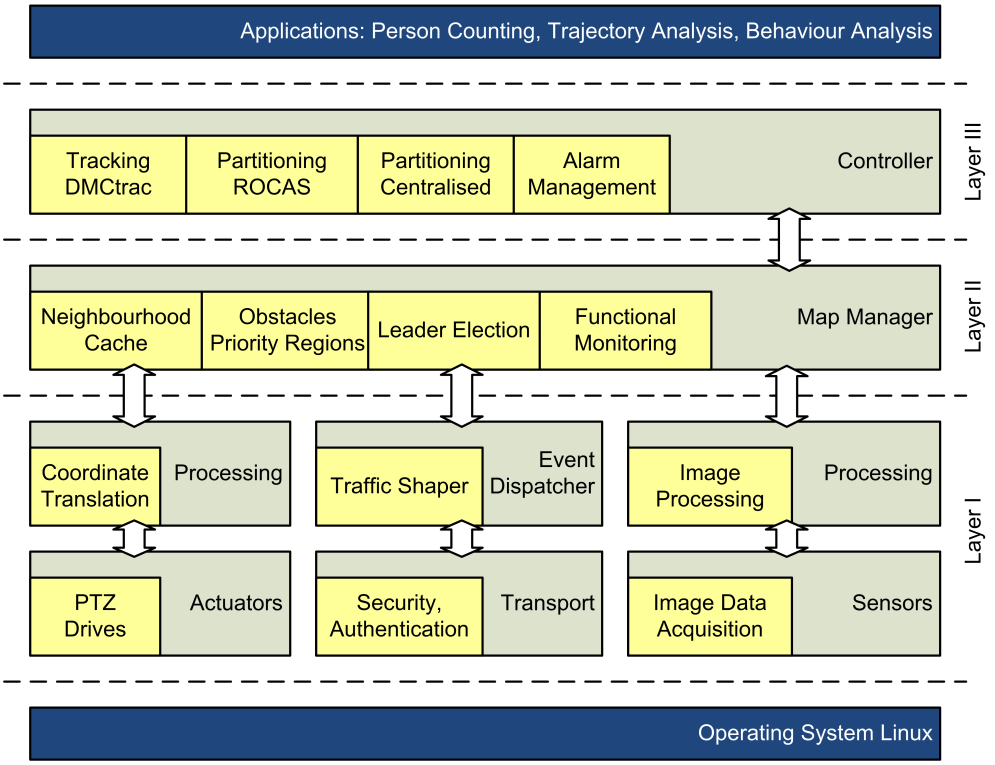

Figure 4.2: Software architecture for Smart Cameras

neighbourhoods are established and maintained as described in Section 4.3.1.

Distributed management algorithms are part of *Layer III*. The tracking algorithm DMCtrac [5] coordinates the tracking of multiple moving objects with multiple PTZ cameras. ROCAS [4] is a distributed heuristic approximating solutions to the maximum coverage problem in Distributed Vision Networks. The alarm management component [6] incorporates algorithms that allow for user interaction with Smart Cameras. All these upper-level algorithms are explained in Chapter 5.

The following section gives a short overview of the basic components in *Layer I* and *II*. Their extensive description starts with Section 4.3, where also more details about the implementation are presented.

4.2.1 Software Architecture in Detail

Since Linux serves as a basis for the software, no special prerequisites in terms of scheduling or memory management need to be considered. All hardware components comprising a Smart Camera are supported by appropriate device drivers. These low level functionalities are not discussed further.

4.2.2 Layer I: Basic Building Blocks

Starting from the bottom, three basic building blocks of the software framework can be identified: *Image Sensor*, *PTZ actuator* and *Network interface*.

The network functionalities are a critical part of the architecture and do not rely on the operating system. The Click modular router [73] is used to acquire messages directly from the network interface, process them (e.g. for routing) and emit messages on the interface. The Click router consists of numerous C++ classes, so called elements, that handle network messages. Click elements can be connected by a graph, with the messages floating along the edges. For performance reasons, message handover is done by passing a pointer from one element to another.

Click is a flexible and extensible open source program for IP based network programming. Click and the system management algorithms that have been implemented in context with this thesis can either be run inside a network simulator [74] or as a Linux kernel module on a real-world prototype. The simulation environment is described in Section 6.2. Apart from the *PTZ actuator* component, the *Image Processing* component and higher level applications, the node software is integrated in the Click Modular router.

The *PTZ actuator* component is loaded as a shared library. It is connected to the Click router by functional binding. This library offers a unified interface so that the computing unit can be attached to various PTZ cameras and control their pan and tilt angle, and the zoom setting. Apart from the camera's PTZ abilities, other settings like built-in autofocus and white balance can be controlled.

An extensive software library for computer vision (OpenCV [25]) is used for *Image Processing*. OpenCV contains a collection of computer vision algorithms that incorporates many of the features necessary for movement detection and object recognition as introduced in Section 2.2. OpenCV is further able to save still images, record video streams and encode them in different ways. These data can be accessed by Click elements, e.g. in order to detect feature points for *Functional Monitoring* or higher level applications as e.g. person counting. For method invocation and data transfer, a socket based communication scheme is used. The exchange of image data between the *Image Processing* element and other elements is done in shared memory. The shared memory block can be accessed from all elements that need to work on image data. The following components make use of this image data:

- Alarm Management
- Security, Authentication

- Functional Monitoring
- DMCtrac
- ROCAS (Obstacles and priority regions)

The access of DMCtrac and ROCAS on image data is a stub function on the real world prototype and has been evaluated in the simulator only. Future enhancement of the real-world prototype can incorporate appropriate computer vision algorithms in high level applications like person counting or behaviour analysis at this point.

For *Security and Authentication*, a Click element has been developed that contains an algorithm for generating a cryptographic key to encrypt the communication between cameras. Therefore, image data is analysed and feature points are extracted by using appropriate algorithms like [26] or [30] that have been introduced in Section 3.3.3. Two cameras with an overlapping field of view can generate a cryptographic key from what they have seen by deriving a fingerprint from the feature points they detected. The process of generating this key and an authentication process has been developed and filed as a patent [40]. The underlying algorithm can also be used for *Functional Monitoring* of the system. The networking part contained in the *Event Dispatcher* takes care of message repetition in order to cope with lossy communication channel and the aforementioned temporal partitioning of wireless networks.

4.2.3 Layer II: Map Manager

Information about neighbouring nodes is stored in the *Neighbourhood Cache* and is used by the algorithm presented in Section 4.3.1. It can be established in both WLAN networks (where neighbourhoods result from transmission range) but also in LAN networks, where geometric neighbourhood needs to be discovered first by appropriate routing protocols [72].

The *Map Manager* is in charge of holding all information provided by spatially adjacent cameras. Neighbouring nodes exchange information about their position and the geometry of their field of view. The *Map Manager* also maintains a cache including viewing *Obstacles* and *Prioritised Regions*. These values, defining which areas to observe with high priority and where no observation is necessary (e.g. viewing obstacles) need to be configured by the user. In future, sophisticated computer vision algorithms may be used to detect and store viewing obstacles automatically. ROCAS makes use of the *Map Manager* in order to approximate solutions to the alignment problem as introduced in Section 5.2.2.

For the *Functional Monitoring*, the same patented algorithm as for the *Security, Authentication* component is used. By deriving common feature points from image data, cryptographic keys can be generated. Since the details of the underlying algorithms are beyond the scope of this thesis, the reader is referred to [40, 75]. As implemented in all other components, the Click router is used for handling the message flow and the image data acquired from OpenCV (included in the *Processing* component).

The *Leader Election* component takes care of the election process that is necessary to find a suitable leader inside a network of numerous cameras, see Section 4.3.4. In case a camera has been elected as a leader, it uses global algorithms to carry out *Cooperative Tasks* as presented in the following.

4.2.4 Layer III: Controller

The *Controller* component contains the algorithm named ROCAS for spatial partitioning of Smart Camera networks. It is in charge of aligning the PTZ cameras' heads with respect to the positions of neighbours and local constraints (priority regions and obstacles). The tracking algorithm DMCtrac enables the cameras to cooperatively track multiple objects in a cooperative manner. In case an object leaves a camera's field of view, neighbouring cameras take over the object to achieve a seamless tracking. The *Alarm Management* component is used to detect events of interest and notify guards. All these algorithms are presented in detail in Chapter 5.

The software architecture presented above has been optimised to suit the special needs arising in Distributed Vision Networks. Image data can be accessed fast from different components by a shared memory concept. After introducing the anticipated system and node architecture, the following section provides a more detailed description of the basic algorithms in *Layer I* and *II* as well as the interaction of components.

4.3 Basic Algorithms for System Management

In the following, a short introduction is given on the communication between Smart Camera nodes and how the neighbourhood relationship in a completely decentralised system is established. The following sections present selected software components of *Layer I* and *Layer II* in detail.

Since a detailed description of several components mentioned below is beyond the scope of this thesis, these components are not discussed further. Instead, the reader is referred to works containing more background information:

- The adjustment of *PTZ Drives* and *Coordinate Translation* are introduced in [76, 77]
- *Image Processing* and *Image Data Acquisition* are described in [77, 78]
- For an introduction to *Security, Authentication* and *Functional Monitoring* cf. to [75, 40]

4.3.1 Event Dispatcher

Message exchange is coordinated by a thread running on each camera that manages a message bag for outgoing messages, see Algorithm 1. When the thread is started, a timer is initialised. In case the timer expires (line 4), all messages that have been collected in the message bag are sent. In case the message bag was empty, a heartbeat message is sent in order to inform neighbouring nodes that the camera is still alive. The heartbeat messages are called SPM (Smart Camera Position Message) since they include all information a camera has to share with neighbouring cameras or spatial alignment as introduced in the following chapter. This SPM (Figure 4.3) contains a message ID, the sender ID and the camera's position. An SPM further contains information about the camera's field of view described by the viewing range, the span angle and the viewing angle. The total length of an SPM is 28 Byte since all seven fields contain integer values.

Algorithm 1 Neighbourhood Management thread

```
1: init:
2: set timer t_i
3: init neighbourcache // start with empty NC
4: on timerexpire :
5:   send messages in sendbuffer
6:   send out heartbeat
7:   set timer t_i
8: end on
9: on incoming message :
10:  if message is a management message
11:  update neighbourcache()
12:  end if
13: end on
```

By varying the timing interval t_i, the timer can be used to shape the outgoing traffic: choosing a high value t_i causes the cameras to exchange messages seldomly. This saves bandwidth but causes the time to termination of the algorithm to become longer. Choosing a low value t_i intuitively results in a decreasing time to termination. It should be kept

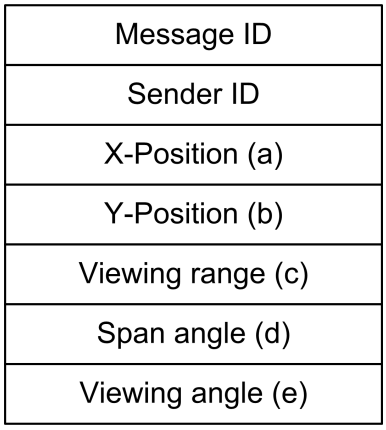

Figure 4.3: Smart Camera Position Message (SPM) as used for ROCAS

in mind, that the resulting higher bandwidth consumption provokes collisions and message loss. Thus, choosing t_i too small has a negative impact on the system's performance. The parameter t_i and its influence on the system's performance is extensively discussed in Chapter 6.

The timing interval can be varied by using randomisation techniques [79]. Thereby, the outgoing traffic can be shaped. A time period is chosen randomly in which a camera backs off from the shared media. In this time t_i, other cameras have access on the communication channel. This decreases the probability of simultaneous media access and helps to avoid collisions on the shared media. In Chapter 6 the influence of traffic shaping on the system's performance is investigated in detail.

The frequent exchange of messages allows cameras to keep a so called neighbourhood cache, where all known neighbours and their properties are stored. Incoming messages are processed immediately after they arrive. The neighbourhood management procedures (invoked from lines 3 and 11 of Algorithm 1) are part of the *Map Manager* component of the software architecture presented in Figure 4.2.

4.3.2 Failure Model

In order to decide, whether a component of a distributed system has failed or not, failure detection techniques can be applied [80]. Components are for example camera hardware, software processes or the communication system. Depending on the underlying system model, the failure of processes can be determined exactly or not at all. A common system model is the synchronous system: informally spoken, it is assumed that all timing constraints are known (i.e. message delay, processing times, clock drift). For such systems,

failure of components can be determined exactly [81]. In an asynchronous system, there are no timing constraints. Therefore, it is impossible to decide whether a component has failed or is just very slow [82]. Developing applications for synchronous systems is thus much easier than developing software for asynchronous systems. The advantage of using an asynchronous system model is, that applications can easily be ported to other platforms and communication systems since they are robust towards timing issues. Since failures can not be detected in asynchronous systems, some constraints need to be made in order to allow for practical failure detection (e.g. defining upper boundaries for message transmission). In the following, upper time boundaries are applied in order to allow for a practical failure detection.

4.3.3 Neighbourhood Cache

Apart from simple actions like adding and deleting neighbours from the neighbourhood cache, prerequisites have been taken to deal with failures as introduced above. A basic failure detection is used to maintain neighbourhood dependencies between Smart Cameras. By detecting fail-stop errors[1], neighbouring nodes can detect whether a formerly existing neighbour has failed or not. Hence, an upper timing boundary for message exchange and processing time needs to be defined. Processing and transmission time for SPMs are below $10ms$ on common Smart Cameras as described in Section 5.3. For security relevant applications of Smart Cameras we assume that detecting node failure within $1s$ is appropriate. Jitter and clock drift can therefore be neglected, since their impact on the upper boundary of message transfer is only marginal in comparison to the moderate timing requirements of upper level algorithms like ROCAS.

For the partitioning algorithms ROCAS and DMCtrac, a failure detector is implemented in the *Neighbourhood Cache* to detect devices that left the distributed system without properly notifying other nodes. A common method to detect such failures is the frequent exchange of heartbeat messages with a constant frequency $f_{heartbeat}$. In our case, we set the timer interval $t_i = \frac{1}{f_{heartbeat}}$ as introduced in the previous section. In case a heartbeat message is received by a device, the corresponding entry $t_{last\ heartbeat}$ is set to the receive-time of the message. In case a heartbeat has been missing for too long (longer than $t_{tolerated}$), the node is expected to have failed and is removed from the neighbourhood cache, line 3.

Newly joining nodes and updates of $t_{last\ heartbeat}$ are carried out by calling the *updateentry*

[1] A fail-stop error occurs when a component that has been running correctly, stops prematurely. Once a component crashes, it does not recover.

Algorithm 2 Update neighbourcache

```
1: for each neighbour
2:   if t_last heartbeat < ( get(time) − t_tolerated )
3:     removeentry(neighbourID)
4:   end if
5:   else if updateentry(neighbourID)
6: end for
```

function (line 5). Note, that formerly failing nodes that recovered from their failure are considered as newly joining nodes. In order to save bandwidth, heartbeat functionality is also embedded in SPMs. Only in case no management message has been sent for a certain amount of time, a camera decides to emit a heartbeat. This helps to reduce bandwidth usage.

A similar algorithm is used e.g. on the Internet by the border gateway protocol (BGP, defined in RFC 1657) to check the connectivity of the routing graph. More elaborate mechanisms rely on dynamic heartbeat intervals and an adaption process that helps to minimise the impact of temporal message loss on the communication channel, e.g. confer to Satzger et al. [83].

For the purpose of detecting node failure in Smart Camera networks, the approach as presented in Algorithm 2 suffices. ROCAS and DMCtrac can cope with inaccurate neighbourhood information, which has been shown by simulation experiments presented in Section 6.4.1. In an asynchronous IEEE 802.11 WLAN system, node failure can effectively be detected within at least one second (depending on t_i and $t_{tolerated}$).

4.3.4 Leader Election

A variant of the extrema finding algorithm proposed by Vasudevan [84] has been implemented as a part of the Smart Camera node architecture [85]. In addition to the basic functions described below, some modifications enable the cameras to elect a leader among them that carries out special roles as introduced in Section 4.1.1. The original variant of Vasudevan's algorithm has been extended in two points:

- It has been adapted to an IP based broadcast system
- Simultaneous elections are now feasible

In order to cope with lossy communication channels and network partitioning, Vasudevan's algorithm incorporates message repetition as used in the *Event Dispatcher*. Thereby, the algorithm is suited well for the usage in today's wireless communication systems. For

the usage in Distributed Vision Networks, the algorithm has been enhanced to support multiple simultaneous elections for different roles. A short introduction on the algorithm is given below, for an in-depth description cf. to [85, 6].

Vasudevan's algorithm relies on three phases. These three phases serve as a basis for several distributed algorithms. For example, Dijkstra and van Scholten [86] developed an algorithm for termination detection. Thereby, it is possible to detect whether a process in a distributed system has finished and terminated. Vasudevan's derivative of the algorithm allows to cope with disturbances of today's real world wireless systems like partitioning, node failure and message loss. For the usage in Smart Camera systems, it has been adapted to suit our needs:

1. Explode: any camera in a network can decide to start an election algorithm for a certain role, messages are broadcasted by all cameras (flooding)

2. Echo: in response to *explode* message, other cameras respond with their ID and a value a describing how well they are suited to take over the requested role

3. Information: after phase two is finished (i.e. each camera has emitted and forwarded *echo* messages), the camera starting to send *explode* messages informs all other cameras about the result of the election.

The algorithms terminates in case a leader has been elected. In case the election failed (e.g. due to message loss or node failure), the election is repeated. An evaluation of the leader election mechanism in a Distributed Vision Network is presented in Section 6.16. A special focus is on the timing behaviour since its impact on the system's overall performance is critical. On this basis, benefits and drawbacks of centralised components are discussed in Section 6.4.10.

4.4 Summary

This chapter presents a system architecture for Smart Camera networks. It introduced a networked system architecture and explained roles that Smart Cameras can take over dynamically, e.g. by using an election algorithm.

A middleware for Smart Cameras has been presented, that consists of several components tailored to suit the needs arising in Distributed Vision Networks. Lightweight algorithms have been explained, that allow for message exchange between Smart Cameras and make way for cooperative tasks as introduced in the following chapter.

Chapter 5

Distributed System Management Algorithms

This chapter presents algorithms that make way for self-organisation and self-optimisation in Distributed Vision Networks. Since the underlying problem of aligning cameras' heads in an optimal way can be shown to be NP-complete, heuristic approaches have been pursued. An algorithm that is approximating solutions to statically align cameras in the most efficient way is ROCAS (Robust Camera Alignment System [4]). For moving objects, a tracking algorithm is presented that makes use of the cameras' PTZ abilities (DMCtrac: Distributed Multi-Camera tracking [5]). Both alignment algorithms presented here are distributed heuristics addressing the problem of spatio-temporal camera alignment in large systems.

In the end of this chapter, another system management algorithm is presented. In order to let human staff interact with cameras, an appropriate algorithm for alarm management is introduced.

5.1 Pan and Tilt Alignment

The alignment of PTZ cameras in a cooperative manner is an important task arising in large Distributed Vision Networks. Cameras may be arranged in a way, so that their fields of view overlap. This overlap might be necessary for tasks like target tracking. In order to reach a maximum surveillance coverage of an area under observation, this overlap between cameras' fields of view needs to be minimal. Figure 5.1 shows a 2-dimensional view from bird's eye perspective on a Smart Camera system. On the left side, a misaligned

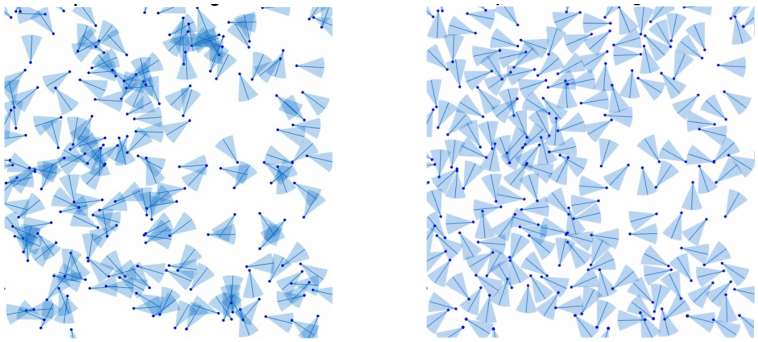

Figure 5.1: Example for misaligned cameras (left) and correctly aligned cameras (right)

configuration is shown. The right side displays the arrangement of cameras after a spatial partitioning has been carried out in order to maximise the area covered by the cameras' viewsheds.

In addition to this, cameras should focus on certain areas that are of special interest. The prerequisites concerning alignment might change over time. For example, at an entrance of a building a camera might be installed in order to capture images of frontal faces. A high level application may be in use to identify persons. Intuitively, a camera is expected to focus on the entrance door but in case a person needs to be tracked, the camera can decide to follow the person by using PTZ abilities. Thereby, the camera is misaligned, changes its heading and leaves the former region it observed unattended. Other cameras are informed, that they should take over the task of observing the entrance. In order to achieve an appropriate alignment for tracking and static surveillance, ROCAS and DMCtrac have been developed. These algorithms do not only consider the priority of regions, but also the alignment of neighbouring cameras in order to achieve a global optimum in surveillance and tracking coverage.

A surveillance system in operation adaptively switches between two modes of operation: While surveying a static scene, cameras aim at reaching the maximum surveillance coverage (in the simple example introduced above: focus the entrance door). As soon as the Smart Cameras need to investigate a scene more deeply and e.g. try to identify or track an object, they start to cooperatively reconfigure their fields of view in order to gather as much data as possible about this incident - therefore overlapping viewsheds are needed or at least tolerated.

In order to align cameras in a way that leads to a close to optimal surveillance coverage, distributed heuristics are introduced in this thesis. A formal analysis of this problem shows, that it is NP-complete. With ROCAS an algorithm is presented which is able to approximate a solution to the problem of static camera alignment.

Tracking objects with multiple cameras is another problem concerning camera alignment by panning, tilting and zooming. In order to allow for a seamless tracking of objects, cameras can follow objects by using their PTZ abilities. For object handover, cameras need to agree on a common migration region that needs to be focused simultaneously in order to take over objects. With DMCtrac, this thesis presents a heuristic that approximates a solution to the tracking problem.

5.2 Formal Problem Statement: Spatial Partitioning

Figure 5.2 shows the simplified viewshed of a camera and its geometry according to De Floriani's definition [58]. ROCAS relies on 2-dimensional geometries. The ground plane view is approximated by a triangular shape. Real world experiments show, that the ground plane view is a trapezoid but for the alignment process the impact of this simplification can be neglected [87]. By modelling the ground plane view as a triangle, the calculation of intersections between polygons becomes slightly faster and visualisation in form as triangles can be understood more intuitively. Triangular shapes modelling the cameras' fields of view have also been used by Erdem [57] for a positioning algorithm, as well as in [17] for the camera calibration. Extensions to a 3-dimensional model offers higher accuracy, especially in case cameras are positioned on differing heights. The calculation of volumetric intersections does in return require a much higher computational effort [60] and has not been investigated further in context with this thesis. The following theoretical investigations rely on a 2-dimensional model, but the assumptions and results might also hold for 3-dimensional models.

A camera SC is characterised by its position and field of view as described in Figure 5.2. Each camera is positioned on an area $\mathcal{A}$ at a position (x, y). Each camera's field of view FOV is defined by the focal length L, the viewing angle α and the camera's alignment δ. For now, we assume the cameras only to tilt, the pan and zoom abilities are neglected.

5.2.1 Formal Description

Optimal partitioning of an area $\mathcal{A}$ means to find an adjustment for all cameras' viewsheds on $\mathcal{A}$ (w.l.o.g. we assume that all cameras are on $\mathcal{A}$), so that the surveillance coverage becomes maximal. Intuitively, surveillance coverage is maximal in case of:

- the overlap of cameras being minimal

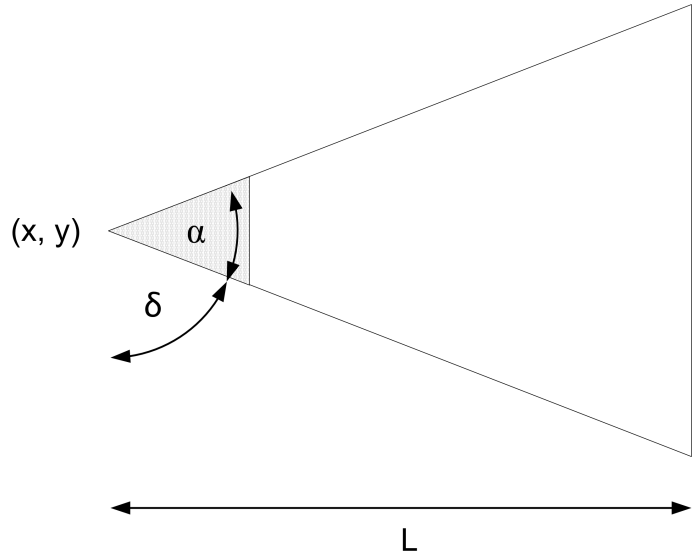

Figure 5.2: Geometry of a camera's viewshed

- the accumulated priority of the area focused by all cameras in the system being maximal

Depending on the degrees of freedom given, several parameters can be changed to achieve an optimal system configuration. For the classical art gallery problem the following assumptions are made: cameras have an unlimited viewing range, can be placed at any position on $\mathcal{A}$ and have a 360° field of view. Goal is to find a position for each camera, so that the observable area of a given polygon becomes maximal. For practical reasons (i.e. building constraints), a camera may not be placed at any position but only near those positions where several requirements are fulfilled. These requirements include for example power supplies, mounting possibilities or aesthetic guidelines to be met. In the system presented here, cameras are placed at fixed positions and have a limited viewshed.

Furthermore, we want those regions to be observed, that are especially prone to incidents. The area $\mathcal{A}$ the cameras are positioned on is divided into j subareas A_j. These subareas are prioritised by assigning values to them to weight the importance of the area. In the following we choose low values to indicate uninteresting areas, whereas large values indicate regions that should be observed in depth. Depending on the regions covered by a camera's FOV, the priority function $w(FOV)$ returns the priority of the area currently focused. The following section shows an example how regions can be prioritised.

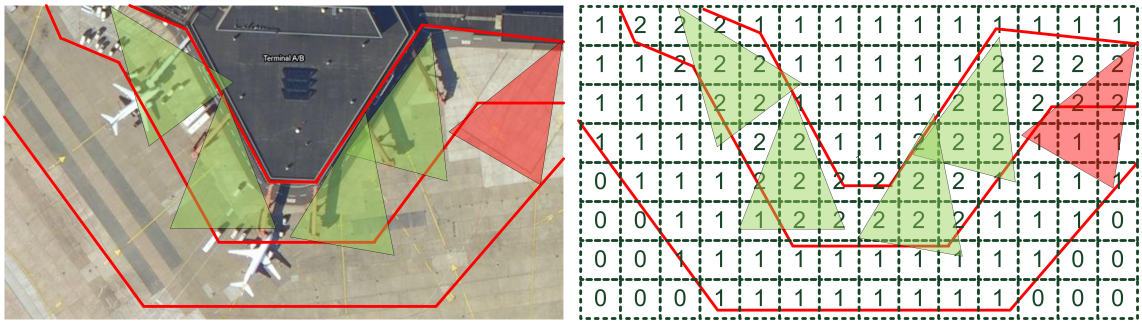

Figure 5.3: Scenario with at airport terminal: left side showing satellite view, right side showing camera's map with prioritised regions

5.2.2 Priority Regions

For real world applications, the area $\mathcal{A}$ is modelled as a polyhedral shape that contains viewing obstacles and regions of special interest in form of a 2-dimensional map. Trees, walls and large moving objects may block a camera's line of sight and might be of minor interest in comparison to other areas, that are expected to be observed with higher priority. In buildings, regions with a higher priority might be entrances or emergency exits. The priority of regions may be set in advance and kept statically inside the cameras at the time of their installation. With more sophisticated computer vision algorithms available, Smart Cameras may be able to interact in order to cooperatively rate regions in terms of priority. Regions that might frequently be subject of suspicious activities could be marked with a high observation priority. Smart Cameras need to align their fields of view to guarantee a complete coverage of $\mathcal{A}$. In contrast to viewing obstacles, priority regions need to be observed in detail.

The priority of regions is represented by assigning numerical values in order to weight the priority. Thus, the area $\mathcal{A}$ is divided into a set of subareas ($\mathcal{A} = \{A_1, .., A_n\}$). This discrete approach allows us to model the priority of regions in detail. In the following, we use a regular grid to represent polygonal regions. Figure 5.3 shows, how a map of precincts of Hannover airport is modelled in terms of prioritised regions. On the left, a satellite view of a terminal building is depicted. We assume areas close to the terminal building to be of major importance, since attackers might try to find a way out of the terminal and into aircrafts. Thus, the regions are prioritised accordingly: the map on the right side of Figure 5.3 shows the region values assigned. The cameras are expected to focus on those regions, that are close to the terminal building. The building itself does not need to be observed and for the taxiways and aprons further security mechanisms are existing, thus those regions are assigned a low priority (low values in the grid).

Variable	*Description*
SC	Configuration of a single Smart Camera
FOV	Mapping of camera configuration on an area
$\mathcal{SC}$	System configuration $\cup SC$
$\mathcal{FOV}$	Mapping of all cameras on an area $\cup FOV$
$\mathcal{A}$	Area under observation
A	Subareas, $\cup A = \mathcal{A}$
$w(FOV)$	Weight of subareas covered by $FOV(SC)$

Table 5.1: List of mathematical abbreviations

By summing up the values of all grid elements (subareas A) that are covered by a camera's field of view (FOV), a camera's alignment can be rated in terms of efficient coverage. FOV describes the mapping of a camera's field of view onto an area, $FOV : SC \rightarrow A$. By calculating $w(FOV) = \sum w(A_j), A_j \in FOV$, this is done for a single camera. In this example some areas A_j are only covered partly by a FOV. We count only those subareas A_j for the calculation of $w(FOV)$ that are covered by FOV to more than 50%. In the example depicted in Figure 5.3, $w(FOV)$ for the upper left camera results to $w(FOV) = 2 + 2 + 2 + 2 = 8$.

Aligning the camera to any other position would lead to a decrease of $w(FOV)$ and a decrease in coverage quality - assuming their positions are fixed. The camera indicated with a red viewshed on the upper right side of Figure 5.3 is misaligned, it is focusing an area of minor importance.

In addition to the intersection with priority regions, the intersection of cameras' viewsheds is considered. We are searching for an overlap-free camera configuration, i.e. $\cap FOV = \emptyset$.

Table 5.1 provides an overview of the mathematical symbols.

5.2.3 Proof of Problem Complexity

In this section, the complexity of the spatial alignment problem for Smart Camera networks is proved by reduction to the *set-packing problem*. The area $\mathcal{A}$ the cameras are expected to observe is therefore dissected into n sub-areas $\mathcal{A}= \{A_1, .., A_n\}$. The problem description given above allows for each element in $\mathcal{A}$ to be covered exactly by a single camera's field of view and each camera on $\mathcal{A}$ can select its alignment in such way, so that it has no overlap with neighbouring cameras. The task to achieve $\mathcal{FOV}(\mathcal{SC}) = \mathcal{A}$ with the highest possible coverage of priority regions $w(\mathcal{FOV})$ is equivalent to the *set-packing problem* which has been shown to be NP-complete by Karp [59].

5.2.4 Karp's Problem

A work by Stephen Cook laid the basis for an important step forward in computational complexity theory [88]. It has been shown that several problems, that have formerly been known to be difficult to solve efficiently, belong to a certain group of problems: the so-called NP-complete problems [59]. All these problems have in common, that once a solution to the problem has been found, this solution can be verified quickly - in polynomial time. The difficulty is to find such a solution: for NP-complete problems, no deterministic polynomial-time algorithms are known. Karp presented a list of 21 common problems and showed that they all belong to the complexity class NP. In case a deterministic polynomial algorithm is found, that solves one of these NP-complete problems, then all of the algorithms in the complexity class NP can be solved by that algorithm[1].

In order to solve the *set-packing problem*, a superpolynomial amount of time is needed in terms of the input size. Since we want cameras to be aligned in real-time, we need fast algorithms to find acceptable solutions to this problem. Hence, distributed heuristics are introduced. These heuristics approximate solutions to the alignment problem quickly and with sufficient accuracy, as shown in Section 5.3. The following sections formally describe, why the camera alignment problem is difficult to solve and belongs to the complexity class NP.

The Set-Packing Problem

The *set-packing problem* is defined as follows: Given a set of subsets $\mathcal{FOV}$ over a domain $\mathcal{A}$= $\{A_1, .., A_n\}$, the maximum number k of disjoint subsets is searched. After identifying all independent subsets, the optimal combination of subsets in terms of their cost is searched. In a camera network, we are searching for a system configuration where there is no camera overlap, i.e. disjoint fields of view are searched. Speaking in terms of cameras, no camera SC_A has an overlapping field of view with neighbouring cameras SC_B i.e. $FOV(SC_A) \cap FOV(SC_B) = \emptyset$.

The goal is to find an optimal configuration for all cameras $\mathcal{SC}$ with $w(\mathcal{FOV})$ being maximal. Note, that there may exist multiple possible solutions to the problem and we are searching for only one of them. Therefore, we define a set of optimum alignment candidates per camera D_{x_m}. The alignment of a camera SC_x is represented by the alignment angle δ_x. The resulting possible fields of view per camera are $FOV(SC(\delta))_x$

[1]The proof, that there is really no deterministic polynomial time algorithm to solve the NP problems does not exist yet. For such prove, a $1 million prize is awarded: http://www.claymath.org/millennium/P_vs_NP/.

with $\delta \in \{0°, .., 359°\}$. A camera can have m possible alignment candidates SC_x with $w(FOV(SC_x))$ being maximal.

The set D of m resulting alignment candidates for camera x results to

$$D_{x_m} = \{\delta_{x_m} \mid w(FOV(SC(\delta))_{x_m} \; is \; max, \; FOV_{x_m} \cap FOV_y = \emptyset \; \forall x_m \neq y\}$$

We are interested in disjoint fields of view, i.e. need the underlying *set-packing problem* to be solved or at least an approximated solution which is represented by selecting only those alignment candidates that have no overlap with any other cameras' possible FOV_y.

Finally, the resulting system configuration with optimal camera alignment is:

$$\mathcal{SC} = \{SC_1(D_{1_1}), .., SC_n(D_{n_1})\}$$

Without loss of generality, we chose the first configuration D_{x_1} fulfilling the conditions given above. This optimum configuration contains an overlap free configuration for each camera with the highest possible surveillance coverage with respect to priority of regions. Due to the enormous size of real-world systems, the problem can hardly be solved in real-time. The following example describes the problem in detail and offers a step-wise description of how the problem can be solved.

Example

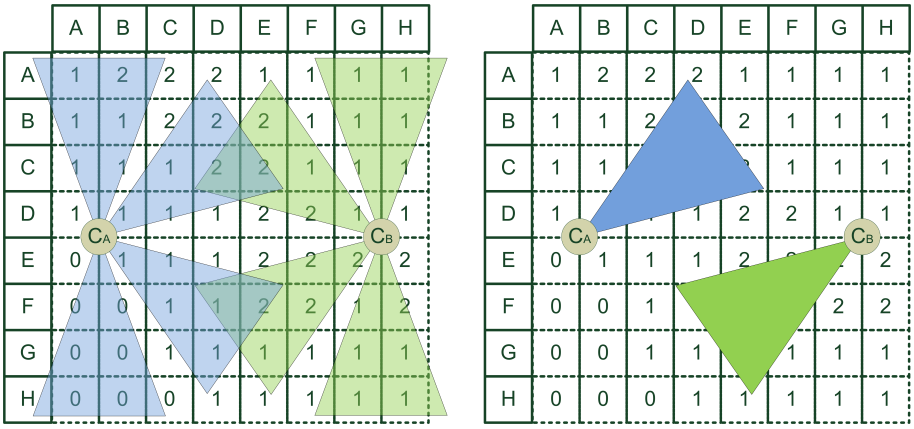

Figure 5.4: Example for camera alignment reduced to *set-packing problem*

The reduction of the camera alignment problem to the *set-packing problem* is in the following demonstrated by a modestly sized example. Two cameras are positioned on an area

$\mathcal{A}$ as depicted in Figure 5.4. This area is divided into subareas $\mathcal{A} = \{A_{AA}, A_{AB}, .., A_{HH}\}$. Each subarea carries a priority value $w(A_{ij})$.

In the following example, the cameras are expected to be able to change their alignment to four different positions each. This is a simplification, for ROCAS we assume cameras to switch between 360 different positions. Nevertheless, the example with 4 alignment candidates per camera suffices to explain the *set-packing problem* in Smart Camera networks. These positions are encoded by numerical values 1..4, as depicted in Figure 5.5. This encoding scheme replaces the alignment angle δ, i.e. camera A directing its heading to the north is encoded by $A1$, north-east by $A2$ and so forth.

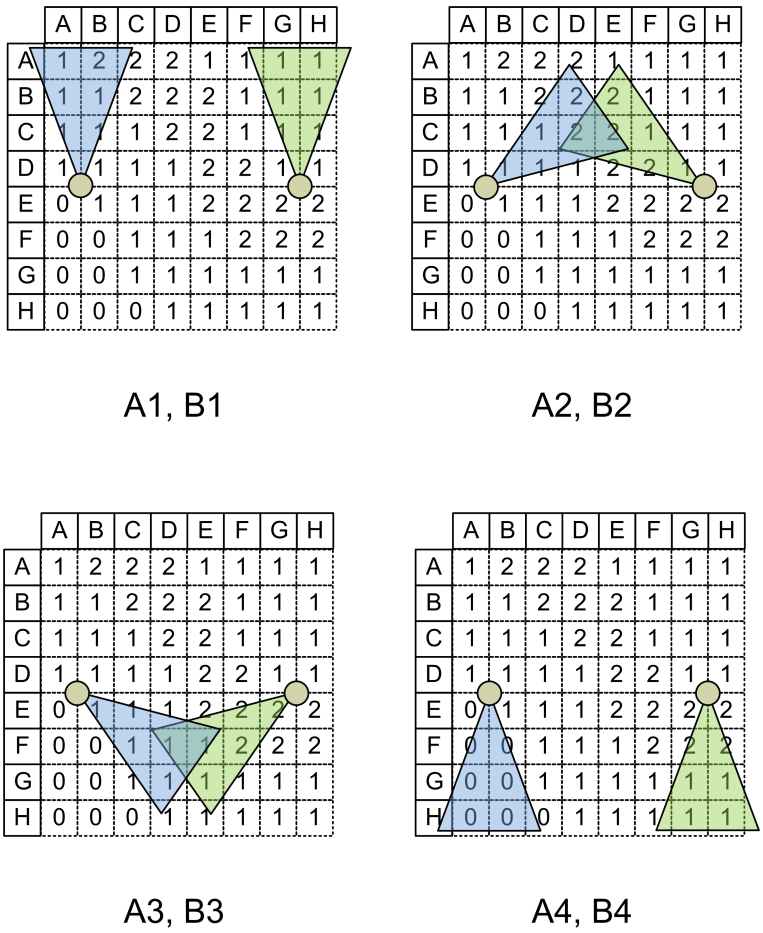

Figure 5.5: Camera positioning and encoding of alignment

Each camera alignment results in a different set of subareas A_{ij} to be covered by the cameras' fields of view. The complete list of subareas $S_{A1}..S_{B4}$ derived from the system configuration presented in Figure 5.4. For example, with camera A being aligned to position 1 of and their respective weight is shown in Table 5.2.

Sets of covered subareas A_{ij}	*priority* w_j
$S_{A1} = \{A_{AA}, A_{AB}, A_{BA}, A_{BB}, A_{CA}, A_{CB}\}$	$w_{A1} = 7$
$S_{A2} = \{A_{BD}, A_{CC}, A_{CD}, A_{CE}, A_{DB}, A_{DC}\}$	$w_{A2} = 8$
$S_{A3} = \{A_{EB}, A_{EC}, A_{FC}, A_{FD}, A_{FE}, A_{GD}\}$	$w_{A3} = 6$
$S_{A4} = \{A_{FA}, A_{FB}, A_{GA}, A_{GB}, A_{HA}, A_{HB}\}$	$w_{A4} = 0$
$S_{B1} = \{A_{AG}, A_{AH}, A_{BG}, A_{BH}, A_{CG}, A_{CH}\}$	$w_{B1} = 6$
$S_{B2} = \{A_{BE}, A_{CD}, A_{CE}, A_{CF}, A_{DF}, A_{DG}\}$	$w_{B2} = 9$
$S_{B3} = \{A_{EF}, A_{EG}, A_{FD}, A_{FE}, A_{FF}, A_{GE}\}$	$w_{B3} = 8$
$S_{B4} = \{A_{FG}, A_{FH}, A_{GG}, A_{GH}, A_{HG}, A_{HH}\}$	$w_{B4} = 7$

Table 5.2: Example for camera alignment problem

By comparing all sets S to each other, the disjoint subsets can be identified. Identifying these disjoint sets A_j has been proven to be NP-complete by Karp. After the disjoint subsets have been found, a selection process is started that selects those disjoint sets covering regions with a high priority w. Since $w(A2) + w(B3)$ is maximal, this system configuration is chosen. In Table 5.3, all possible combinations of the cameras' alignment are shown. Overlapping configurations are indicated with $w = -\infty$, i.e. they are practically occluded from the set of possible alignment candidates D_x.

	$w(B1)$	$w(B2)$	$w(B3)$	$w(B4)$
$w(A1)$	13	16	15	14
$w(A2)$	14	$-\infty$	16	15
$w(A3)$	12	15	$-\infty$	13
$w(A4)$	6	9	8	7

Table 5.3: List of disjoint subareas and their respective weights

This example shows, how an optimal camera alignment can be achieved by mathematical analysis. The following section presents ROCAS, a heuristic to approximate solutions fast and with high accuracy.

5.3 ROCAS

In the following, a lightweight distributed heuristic is presented that approximates a solution to the problem of aligning camera heads in Distributed Vision Networks. ROCAS calculates an overlap function w_a locally at each camera that determines, whether an overlap with other cameras occurs and returns the weight of the overlapping areas. A second function w_b is used, that describes the priority of the region the camera focuses and whether the camera alignment is on $\mathcal{A}$. The resulting overlap function is $w(FOV) =$

$w_b(FOV) - w_a(FOV)$. Viewing obstacles have a negative region value and cameras focusing obstacles calculate a lower $w_b(FOV)$.

The heuristic bases upon the assumption, that minimising overlap locally leads to a global maximum in coverage. That this assumption is true will be shown in Section 5.3.1 and has been evaluated quantitatively, see Chapter 6 for results. ROCAS is a distributed algorithm for dynamic reconfiguration of cooperating cameras and uses solely local (single-hop) communication and local knowledge. We assume, that the transmission range of a camera is larger than its viewing range.

Algorithm 3 Distributed Partitioning Algorithm

```
1:  init:
2:  set timer // randomisation used
3:  init neighbourcache // start with empty NC
4:  init mySPM // stores own position and geometry
5:  overlap ← 0
6:  on timerexpire :
7:    send SPM to all neighbours // send out heartbeat
8:    set timer and restart
9:  end on
10: on incoming SPM :
11:   oldoverlap ← overlap
12:   update neighbourcache // add/removeSCs
13:   overlap ← calculateoverlap // with all SCs in NC
14:   minimise overlap // by changing own position
15:   if ( oldoverlap − overlap ) > th_m
16:     send SPM to all neighbours
17:     change own position
18:   end if
19: end on
```

At startup, a camera generates an SC Position Message (SPM) corresponding to the message format depicted in Figure 4.3 (see Algorithm 3, line 4). A camera's networking unit is further in charge of parsing incoming messages from neighbouring cameras. If a message is received from a neighbouring node (see Algorithm 3, line 10), it is checked whether the neighbour is in viewing range or not. This is done by calculating the euclidean distance between the two cameras. If it is in viewing range its position and field of view is saved in a cache holding all information about neighbouring cameras, the so called *Neighbourhood Cache* (NC) that has been introduced in Section 4.2. Should the neighbouring camera be already known, a local timestamp (i.e. no clock synchronisation required) is updated. This timestamp is used for a heartbeat based failure detection. The NC is checked for changes every time a message is received, see Algorithm 3, line

12. In case of changes, the camera calculates its own overlap with all known neighbours and tries to minimise its overlap by tilting randomly either to the left or to the right, looking for the smallest angle to turn around. The polygon clipping algorithm used for the calculation of the overlapping area of several cameras' fields of view has originally been developed by Vatti, see [89]. In case the overlap has reached a local minimum, the camera switches back to listening for incoming messages and the viewshed remains unchanged. If the overlap can be lowered significantly, the camera generates a message and sends it out to all neighbours (see Algorithm 3, line 15-18). Significantly means, that slight optimisations (i.e. less than the movement threshold th_m) are ignored since sending a message for each of these slight changes causes unnecessary computational effort, network usage and mechanical problems due to dynamic fatigue of the camera mechanics. By varying th_m, the system's behaviour can be influenced. In case th_m is set to a small value, the cameras will achieve a better surveillance coverage, since they move more precisely. In return, the cameras will move more often and extend the time to termination of the algorithm as well as decrease the lifetime of the PTZ drives. In Section 6.4.6, the influence of th_m on the performance of ROCAS is investigated in detail. After sending a message, the camera begins to wait for incoming messages. If there are no incoming messages to be processed after a pre-set dead time, the camera sends out an SPM as heartbeat signal, line 6-9. In case failing nodes are detected, repartitioning takes place and old entries are deleted from the NC as described in Section 4.3.3.

5.3.1 Convergence and Termination

ROCAS adjusts the viewshed of a camera only in case an improvement of its local coverage is achieved. Each adjustment, which is not carried out concurrently with other adjustments of neighbours, leads inevitable to a global improvement as well. Concurrent adjustments are possible, but very unlikely as shown in the following.

In large camera systems oscillations are possible. Oscillation means that a scenario could arise, in which two cameras adjust their viewsheds almost at the same time. This leads to inconsistent NCs, which can result in two cameras choosing the same new field of view by coincidence. Inconsistent NCs can also arise from collisions on the communication channel, in which messages are lost.

The probability of an arising and continuous oscillation (i.e. lasting more than 4 time steps) in a system consisting of 100 cameras is very low ($< 1\%$). Due to the low probability of arising oscillations and the threshold of the algorithm (see line 15, Algorithm 3) the termination of ROCAS is practically given. The threshold secures that the algorithm

stops after finding a plateau of a beginning maximum by avoiding minor changes.

A scenario of two cameras SC_A and SC_B is considered. It is assumed that both cameras are situated on the same position and have the same orientation and configuration on start up. That is, the field of view of both cameras is identical, both cameras observe the same area. This is a worst case scenario for the probability of choosing the same new alignment, as explained later on. To start an oscillation, the two cameras SC_A and SC_B must be willing to change their alignment to improve the coverage of the area they observe. This is given by the assumption that SC_A and SC_B are positioned on the same place having a best match (that is, overlap completely) in their fields of view. Further, they calculate the same new best alignment of their viewsheds to improve the coverage ($event1$). Finally, they have to locally adjust their field of view and inform their neighbours about this. The communication times $t_0(SC_A)$ and $t_0(SC_B)$ are set to identical values to provoke neighbourhood cache inconsistency ($event2$). Both events are essential to start an oscillation. The probability of $event1$ is p_{evt1} and that of $event2$ is p_{evt2} respectively. Since both events have to appear, the probability p_{osc} for an oscillation is the product of the events p_{evt1} and p_{evt2}.

The probability p_{evt1} is calculated as follows. In the scenario of two cameras mentioned above, both cameras have 360 possibilities to change their viewshed assuming 1° steps for rotation. Since the implementation of ROCAS as introduced before makes the PTZ head turn randomly to the left or right for searching a new best field of view and looks for the smallest angel to turn around, the probability of choosing the same new field of view is 0.5, as both cameras are situated on the same position with the same orientation. Thus, the probability to chose the same new field of view is $p_{\text{evt1}} = 0.5 * 0.5 = 0.25$.

The probability for collisions in the communication channel follows a binomial distribution. This is equivalent to the so called birthday problem [90], in which the probability is calculated that k of n persons are born at the same date. The binomial distribution B can be calculated as follows:

$$B(\cdot|p,n) : \mathbb{Z} \to [0,1], k \mapsto B(k|p,n) = \binom{n}{k} p^k (1-p)^{n-k}$$

For the camera network we assume that k out of n cameras access the communication channel with a probability of p_c. In an IEEE 802.11 WLAN, p_c can be calculated as shown in the following paragraph.

For selecting the same new field of view one camera has to change its position and send this information to its neighbours at time t_0. The time step t_0 is chosen from an 100 ms interval randomly distributed by our messaging Algorithm 1. A collision arises

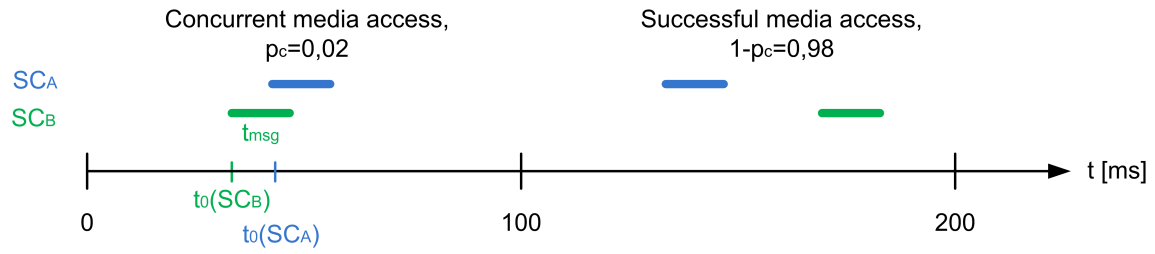

Figure 5.6: Example for inconsistency arising in neighbourhood cache

as soon as another camera adjusts its viewshed concurrently in the time of inconsistent NCs, in the interval $[t_0 - t_{\text{msg}}, t_0]$ or $[t_0 , t_0 + t_{\text{msg}}]$.

The time t_{msg} is approximated by the message transmission time and can be calculated to 2ms (assuming that a message has a size of 1.375 Byte and is send via an IEEE 802.11 network with 5.5 Mbit/s). That is, the probability of choosing any time t_0 from the interval of 100 ms is $p_c = 0.02$, assuming discrete time steps. In case less than two Smart Cameras decide to start sending, no collision will arise. In case two or more Smart Cameras start sending at the same time, collisions and message loss will occur. Intuitively, in a large network, the probability for collisions rises and for n cameras in our example results to $B(n) = 1-(B(0|0.2, n)+B(1|0.2, n))$. These collisions can only arise in case two or more cameras start sending simultaneously. The results for the cumulative probability (which is the sum of the single probabilities) are presented in Figure 5.7. The diagram also includes curves for the probability of collisions after five and eight sending attempts respectively. In case of 120 cameras sharing a collision domain, the probability for an oscillation lasting more than five timer cycles ($5 \times 100ms$) is below 1%.

An example for collisions arising on the shared media is shown in Figure 5.6. Two cameras SC_A and SC_B are positioned in such a way, so that their fields of view might possibly overlap. Camera SC_A decides to align its viewshed at time t_0 and immediately emits a message. Simultaneously, SC_B aligns its field of view and emits a message at time $t_0 + t_{d1}$ with $t_{d1} < t_{msg}$. Again, this situation might occur since the cameras choose a random backoff interval within a timespan of $100ms$. Since $t_{msg} = 2ms$, the probability for such situation is 2%. In this case, neighbourhood cache inconsistency occurs and the cameras might overlap. Such overlap is identified by cameras exchanging further messages frequently. In the example, camera SC_B resolves the situation by sending a message at time t_1. SC_A responds to that message at $t_1 + t_{d2}$ with $t_{d2}\, t_{msg}$. No collision occurs in this case.

As already mentioned above, the probability for starting an oscillation is the product of the probabilities for choosing the same viewshed and transmission collisions. To keep

an oscillation alive, this has to continue each following discrete time step t. As these events are independent from each other, the probability of an continuous oscillation is calculated as follows:

$$p_{continuous\ osc} = {p_{osc}}^{t}$$

Thus, the probability of a continuous oscillation decreases over time. After five time steps the probability for a continuous collision in a network consisting of 300 cameras is below 10% (see Figure 5.7). Since oscillations are very unlikely and our algorithm converges at the beginning of a maximum plateau, our algorithm terminates with high probability.

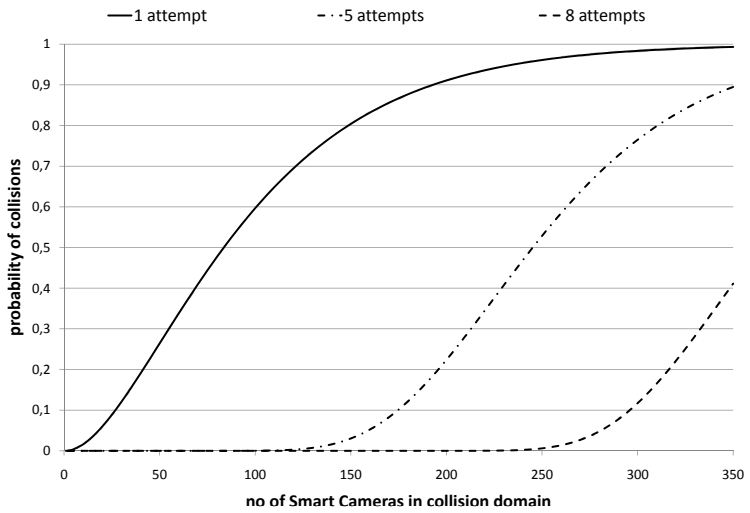

Figure 5.7: Probability of collisions by concurrent behaviour in a system with an increasing number of Smart Cameras

5.3.2 Centralised Variant of ROCAS

ROCAS, as introduced so far, uses local knowledge in order to find an optimal surveillance coverage. Intuitively, a central component, that has global knowledge about all cameras in the system, is able to find a solution that offers higher surveillance quality than the local optimisation. With global knowledge, local maxima can be overcome for the sake of a better overall system performance (Figure 5.10 shows an analogon derived from mathematical complex analysis). The evaluation of ROCAS has shown, that in some situations, local knowledge does not suffice to reach an optimal surveillance coverage. These situations may arise in case the cameras' physical neighbourhood is not represented

in their neighbourhood caches because of limited communication range. One of these special cases, taken from [91], is depicted in Figure 5.8.

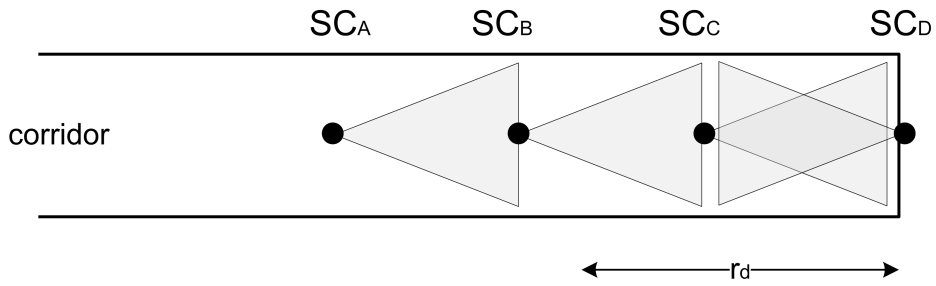

Figure 5.8: Example scenario for ROCAS finding local maximum

Four cameras $SC_A..SC_D$ cameras are expected to observe a maximum area of the corridor they are positioned in. Camera SC_D has a communication range r_d. It knows about camera SC_C but not about cameras SC_A and SC_B. A central server (e.g. camera SC_C) might be elected to gather global knowledge. Thereby, an appropriate solution can be found (i.e. cameras SC_A, SC_B, SC_C turn to the left as depicted in Figure 5.9).

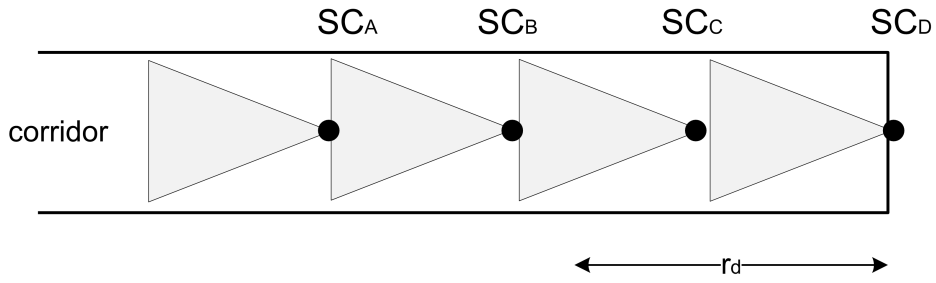

Figure 5.9: Optimal solution: global knowledge helps to overcome local maxima

The instance carrying out the centralised derivative of ROCAS is in charge of acquiring all position and alignment information from all cameras. This information is used to calculate an optimal alignment for all cameras and notify them about their new alignment. This instance, that carries out the overlap optimisation centrally, might either be a computation node or a Smart Cameras with special (larger) computing capacities like camera SC_C in the example introduced above.

In some special cases, a central instance is able to find better solutions to the partitioning problem than an algorithm relying on local knowledge does. In return, the centralised approach causes significant drawbacks in terms of scalability and runtime complexity as described in Section 6.4.10.

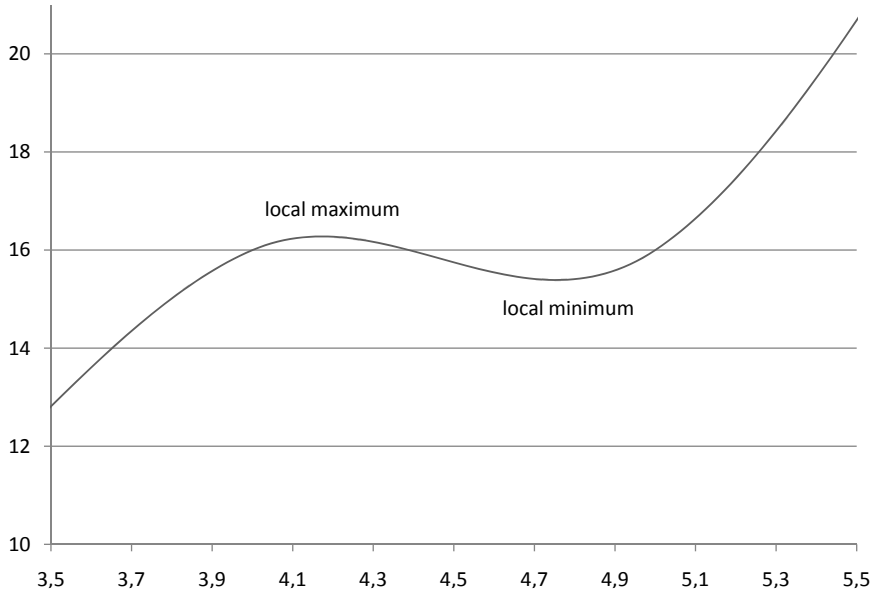

Figure 5.10: Optimisation problem - how to overcome local maxima

5.4 Cooperative Object Tracking

A common task in surveillance systems is to track objects of interest. Security staff switches from one camera to the next in order to follow an object on the monitor wall. In modern systems, PTZ cameras are used to follow objects. With DMCtrac, an algorithm is presented that allows cameras to take over the tracking of objects autonomically. Objects are searched for and followed by tilting the cameras PTZ heads. In case objects are assumed to move out of a Smart Camera's field of view, neighbouring cameras are informed to take over the object to track it further.

5.4.1 Formal Problem Statement

The following formal problem statement describes our theoretical approach to the tracking problem and explains the definition of the performance metrics that have been applied for the evaluation of DMCtrac.
$\mathcal{A}$ defines the spatial area of interest the cameras are observing at time and is derived from a floorplan of the building site or a map of the precinct that is observed. $\mathcal{A}$ may contain static and moving viewing obstacles covering or occluding a total area of $\overline{O}(t)$ that need to be considered by the cameras, i.e. $\overline{O}(t)$ is not observed by the cameras. Each camera has a 2-dimensional viewshed that covers an area $FOV(t) \subset \mathcal{A}$. We investigate a multi camera

and multi object system that uses n cameras for searching and tracking m objects on $\mathcal{A}$. In analogy with the spatial partitioning, the tracking problem can be formulated as one of Karp's NP-complete problems. A reduction to the set packing problem can be derived in analogy to the proof presented in Section 5.2.3. Moving objects replace the static subareas as investigated for ROCAS, i.e. $w(FOV)$ changes over time. The difficulty arising for the tracking task is that objects may move on unpredictable routes. The alignment of cameras' PTZ heads therefore needs to be adjusted continously by DMCtrac whereas ROCAS terminates in case a solution to the coverage problem has been found.

A camera with PTZ abilities can change its field of view (and thereby change $FOV(t)$) by panning, tilting and zooming. The time t is used as a discrete value. We further define the union of all k fields of views to be $\mathcal{FOV}(t)= \bigcup_{k=1..n} FOV_k(t)$. Objects that need to be tracked may either have been selected by human operators in a live video stream or may have been stored inside the cameras. Computer vision algorithms (e.g. based upon histograms) are used to let cameras decide which objects to track. The selection of objects for tracking is not discussed here, it is assumed that cameras know which objects to track. The focus is on tracking objects with multiple cameras, which includes object handover between multiple cameras.

The position of an object i to be tracked is given by its 2-dimensional position $\overrightarrow{p}_i(t)$. A trajectory $\overrightarrow{r}_i$ is comprised by a sequence of positions over time $\overrightarrow{r}_i = \{\overrightarrow{p}_0, ..., \overrightarrow{p}_n\}$ and the length of a trajectory $\overrightarrow{r}_i$ is given by l_{ri}. l_{ri} defines the temporal length of a trajectory in time intervals, not a distance.
We further define a function ω that indicates whether an object at position $\overrightarrow{p}_i(t)$ that is supposed to be tracked is currently seen by at least one camera at time t.

$$\omega(i,t) = \begin{cases} 1, & \text{if } \overrightarrow{p}_i(t) \in \mathcal{FOV}(t) \\ 0, & \text{else} \end{cases}$$

The aggregated number of successful tracking steps for an object i is $\Omega_i = \sum_t \omega(i,t)$. In order to reach a high tracking quality (i.e. tracked objects are under constant observation) we aim at maximising Ω_i. The ratio $\frac{\Omega_i}{l_{ri}}$ is used to define the tracking quality Q. Approaches to raise Ω are constrained by the fact that the system is supposed to maintain a high surveillance coverage of S. The surveillance coverage C is defined as $C = \frac{\mathcal{FOV}(t)}{\mathcal{A}-\overline{O}(t)}$, i.e. the union of all cameras' viewsheds over the maximal viewable area. Thereby, we avoid objects to be tracked by more than one camera at once. This might not be appreciated for all application scenarios, but for those where a large area is to be covered most efficiently and the tracking of an object with one single camera at a time suffices.

In other words, the goal is to find a system configuration $\mathcal{SC}(t)$ so that all object positions on $\mathcal{A}$ lie within the viewsheds of a minimum number of cameras. This is the

classical *set-packing problem* [59]. Although it is proven to be NP-complete, heuristics based upon greedy search yield close to optimal results. DMCtrac is described next and builds upon a greedy algorithm, too.

5.4.2 Tracking Algorithm: DMCtrac

Tracking objects is a dynamic task and needs to be addressed by another algorithm than the static scene coverage handled by ROCAS. The basic idea of DMCtrac is to let cameras switch between several modes of operation. In case no objects are focused and tracked, cameras search for objects by screening A with pan/tilt movements. In case a camera detects an object, it becomes responsible for further tracking. This underlines the greedy character of DMCtrac: objects are assigned to cameras on a *first come, first served* basis, i.e. the first camera detecting the object carries on tracking. Remaining cameras automatically take over other objects they find. DMCtrac is based upon a state-machine which will be described in detail in the following.

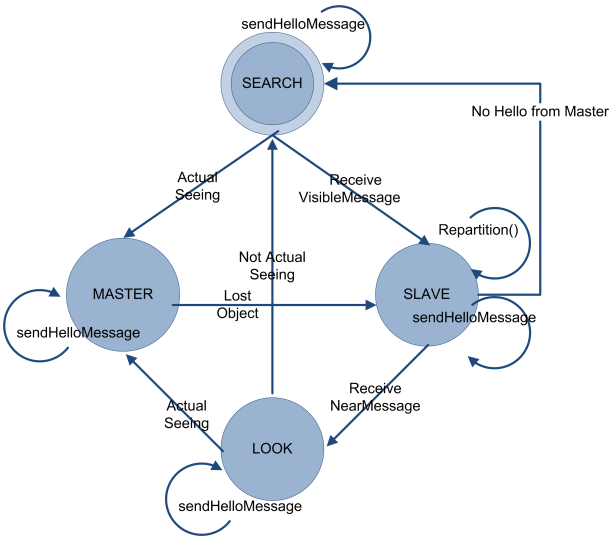

Figure 5.11: DMCtrac as a state-machine [5]

Figure 5.11 shows an overview of the different states of the state-machine. The *search* state is joined initially after the system has been switched from a static surveillance mode (managed by ROCAS) to tracking. A PTZ camera in *search* mode will pan and tilt saccadically and scan the surrounding area for objects that could possibly be tracked. In case an object that needs to be tracked has been detected, the camera becomes a *master*. In case objects got lost, neighbouring cameras are informed and switch to *look* mode to take over the object at an estimated migration point. Cameras in *slave* state move to a

position that leads to a maximum surveillance coverage of the area under observation. The *search mode* is started by the operator of a surveillance system in order to track specific, predefined objects (like persons or vehicles identified by histograms that are sent to the system by users as introduced in Section 5.5). The camera will then follow this object until it either loses this object or gets a message from a neighbouring camera stating that help is required for tracking another object. In case a camera decides to follow the neighbour's request, it will switch to the *look* state. The *look* state becomes activated in case a neighbouring node informed the camera that an object is going to get lost and might need to be taken over. The camera that lost an object will switch to *slave* state and search for a position to turn to that leads to a maximal coverage of the area of surveillance before starting to search for objects again. Each camera scans the surrounding area for objects that need to be tracked. As mentioned before, the detection and recognition is derived from an underlying computer vision algorithm. In case an object has been found, this object is analysed and marked so it has a unique ID that is used for the tracking of this object. All objects a camera has found in its field of view $FOV(t)$ are added to an object cache. Each camera chooses to track those objects that seem most promising to be tracked. A possible criterion for a promising object to be tracked is, that it is close to the camera and thus is expected to remain inside the cameras field of view longer than other objects. The distance between an object and the camera can be calculated in a calibrated system as introduced in Chapter 2. Since we assume the size of objects we track to be known, the pixel size of an object can easily be calculated to the real word size of the object. Thereby, the real-world distance between camera and object can be approximated. DMCtrac currently tracks those objects first, that are positioned closest to the camera. Other possible criteria for promising objects to track are face or body orientation that arise in different application scenarios.

The four different states DMCtrac builds upon are described in the following.

5.4.3 Search Mode

Algorithm 4 Search mode

```
1: init:
2:   set timer
3: on timerexpire :
4:   move randomly
5:   search for objects to be tracked
6: if object found
7:   switch to master mode
```

In *search* mode, a camera is not actively tracking an object but searching the surrounding area for objects to be tracked. The camera k changes its heading over time ($FOV_k(t)$) and thereby investigates the surrounding area randomly, lines 3 and 4 of Algorithm 4. More elaborate approaches than just turning randomly or performing an exhaustive search can be found in [55, 54]. The authors investigate how a 2-dimensional area can be observed most efficiently by PTZ cameras and introduce sophisticated algorithms that coordinate the cameras' movements to keep a constant surveillance coverage. In case an object is found, the camera becomes *master* for this object, line 7.

In case an object is detected by two cameras at the same time, both cameras become master of the object. This conflict is resolved by a simple election algorithm: both masters exchange messages and the camera positioned closest to the object remains *master* whereas the other camera retreats to its previous state. This temporarily causes a slight loss in surveillance coverage but does not affect DMCtrac any further as results show. I.e., usually only one camera is following an object at a time. This parameter may be changed, e.g. one might decide to track object from at least two different viewpoints. To achieve this, the *master* needs to send a *LOOK* command to neighbouring nodes which then would decide if to accept or deny this command considering their own tracking tasks. The movement of the PTZ head and time needed for image recognition is derived by measurements carried out on a prototype. The Axis PTZ214 camera used for the real-world evaluation pans/tilts with a speed of 45° per second. A camera with a span angle of 45° therefore needs 8 seconds to perform an exhaustive search for objects in its possible field of view when turning in 45° steps. The *search* mode can be left to *master* mode in case the camera has detected an object it needs to track or to *slave* mode in case a neighbouring camera became master of the object the camera searched for.

5.4.4 Master Mode

Algorithm 5 Master mode

```
1: on timerexpire :
2:   set timer
3: if object is inside FOV
4:   pushback position
5:   calculate motion vector for object
6:   move PTZ head in direction of motion vector
7: if no object is inside FOV
8:   send LOOK − message with motion vector
```

The *master* mode is reached from *search* mode and from *look* mode in case an object

has been found.
Being in *master* mode is the most computing intensive task for a camera since for each tracking step the object needs to be recognised and the motion vector is calculated, line 5 of Algorithm 5. After the object has been detected in *search* or *look* mode, the camera keeps a record of the trajectory, line 4. Elaborate mechanisms of how to retrieve position information of moving objects and calculate stable results (e.g. by use of Kalman filters) are described in [92]. Currently, our cameras follow object saccadically. I.e., they do not perform a continuous movement but rotates stepwise, line 6. The Axis PTZ214 camera we used is able to change its viewing angle in steps of 1°. Experiments considering the PTZ movement strategy are given in Chapter 6. The *master* mode is left towards *slave* mode in case an object got lost. The neighbouring cameras are informed by sending a *LOST* message.

5.4.5 Slave Mode

Algorithm 6 Slave mode

1: *wait and detect*
2: *repartition // runROCAS*

The *slave* mode is entered from *search* mode or from *master* mode. A camera turns to slave mode in case all objects are tracked by at least one camera. When the slave mode is entered, the camera waits for a predefined time in order to detect objects that may have changed their motion vector unexpectedly and returned instead of migrating to the predicted region. After this dead-time, the camera calculates its heading for an optimal position according to ROCAS, so that the maximum surveillance coverage is achieved or simply turns to a predefined home position, line 2 of Algorithm 6. The *slave* mode is left, in case the *Hello* messages from an objects master are missing for too long (camera switches to *search* mode) or a master sends a message that the object has been lost or is feared to get lost (camera switches to *lost* mode).

5.4.6 Look Mode

A camera enters the *look* mode, in case a corresponding request was emitted by a neighbouring camera in *slave* mode. *Slaves* may request other cameras to take over their tracking tasks in case an object they tracked has recently got lost. The *look* state is entered only in case the camera currently does not track an object as a master on its own.

Algorithm 7 Look mode

```
1: on timerexpire :
2:   set timer
3:   wait and detect
4: if object is inside FOV
5:   switch to master mode
6: if no object is inside FOV
7:   return to search mode
```

In case a camera detects an object within $FOV(t)$, it switches to master mode (line 5 of Algorithm 7. If no object is detected (within a predefined timespan), the camera returns to *search* mode (line 7).

By including weights of tracking tasks, object priorisation becomes feasible (comparable to the priorisation of areas for ROCAS). I.e., a *look* request is to be connected with an alarm level. Urgent incidents detected by cameras will lead to a scaling in the urgency of *look* requests. The *look* mode is left to *master* mode in case an object has been identified or to *search* mode in case the object is not found at the predicted position.

5.4.7 Summary of DMCtrac

Beginning with a formal problem statement, the previous sections introduced DMCtrac, a distributed algorithm for object tracking. The algorithm has been modelled in form of a state machine with four states. Each state has been explained in detail and examples have been given to subscribe the functionality of the algorithm.

The following section describes a further algorithm for system management in Distributed Vision Networks. By using appropriate notification algorithms, cameras and users can interact.

5.5 User Interaction and Notification

Until now, algorithms for the self-organisation of the alignment of cameras' PTZ heads have been discussed. In this section, another class of algorithms is presented and discussed. User interaction and alarm management in a decentralised system differs significantly from alarm management in centralised systems: In a centralised system, a compute server can be fed with user demands. For example, at train stations, the faces of criminals can be compared to faces in the video streams. In case faces match, an alarm is raised in the central control room. This approach lacks scalability. It is hardly possible to analyse video

streams of hundreds of cameras and perform a face recognition on these streams in real time. The task of analysing videos can be performed much better on Smart Cameras. A problem arising in this context is how to provide the input data to let cameras know what they are expected to search for. Furthermore, management algorithms like ROCAS and DMCtrac need to be adjusted to varying user demands. Since no central control instance is existing, another approach to feed this data into the network is needed. Furthermore, the alarms that have been detected by the cameras need to be transferred to surveillance staff to take appropriate action.

This section presents a lightweight system called AMiDiViN (alarm management in Distributed Vision Networks) to allow for user interaction with cameras. The focus is rather on underlying message dissemination than on usability and graphical user interfaces. We present an algorithm that enables human staff to send messages to the camera system. By using an appropriate forwarding scheme, cameras spread these messages inside the network. The requests are send to those cameras that they are addressed to. In return, cameras can notify staff, e.g. in case of alarms.

5.5.1 Partitioning: Modes of Operation

Depending on the user goals that are fed into the Distributed Vision Network, the Smart Cameras will adapt their behaviour. For the partitioning, this means the system can operate in two contradictory modes of operation the security staff can switch between:

- Exhaustive tracking, all cameras are turning their PTZ heads and search for objects
- Static alignment of all cameras with optimal surveillance coverage

Exhaustive tracking means, that all cameras in a subsystem try to track as many objects as possible. Such a scenario may arise in context of an alarm in highly sensitive areas, where a single person is searched and cameras try to focus all moving objects as long as possible in order to identify them. This goal can be achieved by using DMCtrac. A contradictory mode of operation is to keep the camera alignment fixed so that a constantly high surveillance coverage (as calculated by ROCAS) is achieved. In case an object that has previously been defined to be tracked enters a single Smart Cameras field of view, the system will switch over to track this single object.

Cameras that are constantly turning and searching for objects suffer from material wearout. On the other hand, special situations may arise that require intense search operations to prevent incidents. Cameras might not be able to switch between these

modes on their own because they lack background information. Therefore, human guards are in charge to identify such situations and adjust the system according to their goals.

5.5.2 Search and Detect Objects

We assume each guard to carry exactly one terminal. A notification sent to a mobile terminal is expected to be noticed by a guard that takes over the task of handling the alarm. All cameras $\mathcal{SC}$ in the system comprise the camera network $\mathcal{SC}= \{SC_0, ..., SC_n\}$. These cameras can be grouped by their capabilities, i.e. a camera placed in an elevator will not be able to detect cars, in contrast to a camera on a parking space. A group is a sub-set of N, and each camera is member of at least one group. For example, all cameras in elevators may be members of a group E. Each guard g can publish multiple search requests $R_g = \{r_{g0}, ..., r_{gn}\}$ and select which group of cameras the request is addressed to by defining a number of constraints c that need to be fulfilled by the searching cameras. A common constraint is to define an area where the object is potentially to be found, i.e. spatial constraints can be defined. Each request r contains information needed by the cameras to find an object (i.e. features). A camera that receives a search request R will check whether it fulfills c and carry out the requested tasks only in case c is fulfilled. Each camera holds a queue Q that keeps all requests. Figure 5.12 shows the flow of messages in case a guard publishes a search request. This request may be sent to a group of cameras $P = p_0, ..., p_n$ that are installed on a parking space. Each camera stores the image features contained in r and searches for these feature.

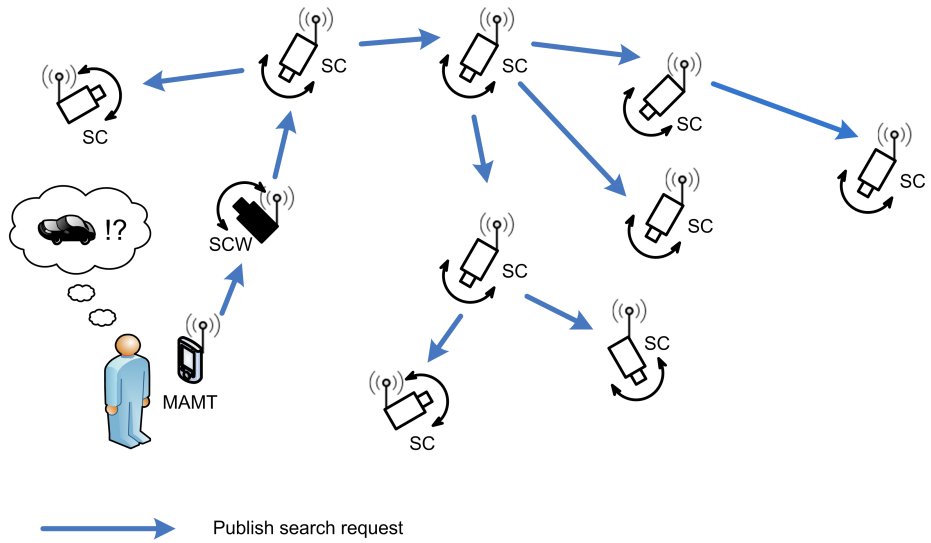

Figure 5.12: Guard sends search request to vision network

In response to a search request r, cameras can inform security staff about incidents

they analysed. In Figure 5.13, the notification process is shown. Since the position of guards changes over time, the cameras P need to search for the guard g_r in the system.

An appropriate algorithm that solves this problem is presented in the following section.

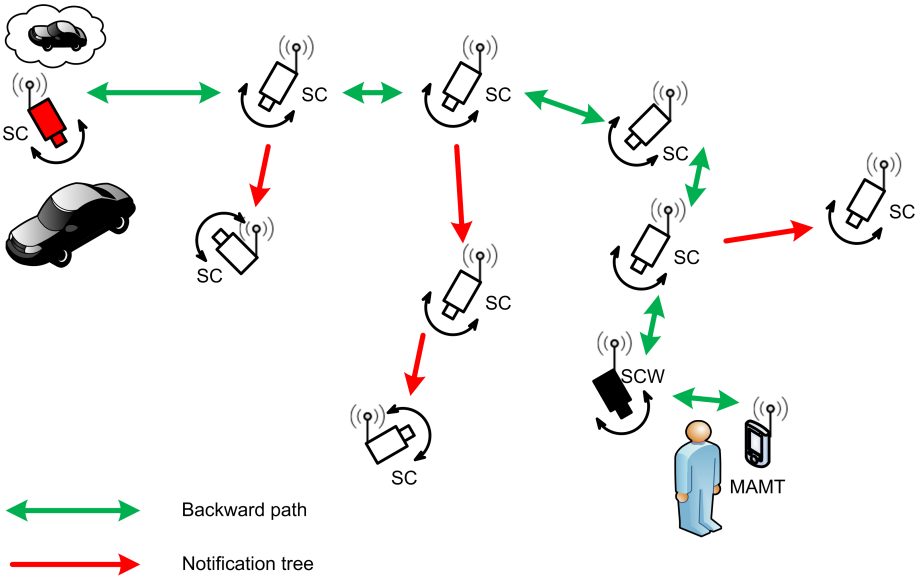

Figure 5.13: Notification after an object has been found

Data dissemination for alarm management

The dissemination of search requests and notifications for the publish/subscribe system is based upon a broadcasting scheme that first grows and afterwards shrinks a tree to allow for on demand route discovery. A simple search based upon broadcasts can be divided into the following three phases, which serve as a basis for several distributed algorithms as presented in Section 4.3.4:

- Explode: the camera that detects an event (*detecting camera*) sends a search request to find a guard
- Echo: each camera connected to a MAMT will return its position and inform the *detecting camera* that the search request was successful and a backward path is created that connects the MAMT and the *detecting camera*
- Inform: the *detecting camera* sends further information (video) over the backward path to the MAMT, the backward path is closed after some period of time

Each *Explode* and *Echo* message carries a payload of several hundred bytes containing position and a textual classification of the incident or the position of the guard. The

message exchange during Information phase relies on unicast communication via a backward path established during the Echo phase. The algorithm as described above needs to cope with multiple cameras detecting the same incident simultaneously. This situation arises in case the cameras are positioned close to each other and the failure of a node is detected by more than one camera. Therefore, the approach as described above has been extended and modified on the basis of an extrema finding algorithm that can cope with such situations. Vasudevan et al. describe a robust algorithm for leader election in ad-hoc networks in [84]. Their algorithm allows to find a maximum value (e.g. highest battery power) in a network and takes care of multiple nodes starting a vote (i.e. detecting an event) and can even cope with node failure and joining/leaving nodes during the election (search) process. Network partitioning is addressed by a re-broadcasting mechanism that helps to reconnect partitions. The extrema searched for is whether a guard is connected to any smart camera, which will then become leader (in our case: the routing end point) for a certain incident. For now, this value is binary (1=guard connected, 0=no guard connected). In future, this can be enhanced to a quality value (e.g. guard with car=100%, guard without car=50%).

The following enhancements have been carried out, to adapt Vasudevan's algorithm to suit the special needs of our system: Since no global routing tables are present, the neighbourhood cache as introduced in Section 4.3.1 is used. Each camera communicates with its direct neighbours and forwards all messages it receives, in case it did not deliver the same message earlier. Each search request is assigned a random ID and a sequence number to make it unique, see [93].

Since each node can start an *explosion* (election) at any time, Vasudevan uses IDs for each election process, too. In case a node receives multiple election messages, it will only participate in the election process with the highest ID. Since the cameras need to find guards simultaneously, the adaption presented here allows for multiple search instances. Algorithm 8 shows a simplified variant of the adapted notification algorithm. The basic idea is to create a spanning tree and find a guard. This is done by sending and forwarding *explosionmessages*, line 2 and 3. An *explosioncache* is kept locally by each camera in order to re-broadcast messages after timeouts, which is not shown here in detail (function call in line 12). In case a first guard is found (line 6), the tree is shrunken. This is done by broadcasting *ack* messages, line 7. In case a node receives an *ack* message for an explosion it already forwarded, this explosion phase is locally removed from its *explosioncache*. The *ack* message is re-broadcasted frequently until either a timeout occurs or data is send down the backward path to the mobile terminal. This data may contain images or videos and is not broadcasted but sent to unicast addresses only. The receiver of an *ack* message stores

Algorithm 8 Growing and shrinking the notification tree

```
1: on incoming explosion :
2:   explosion message → explosioncache
3:   if ( explosion message is new )
4:   forward explosion to neighbours
5:   end if
6:   if ( MAMT − connected )
7:     send ack to all neighbours
8:   end if
9: end on
10: on incoming ack :
11:   find corresponding entry in explosion − cache
12:   update explosioncache
13:   if ( ack message is new )
14:   set backward entry to first incoming ack
15:   broadcast ack to neighbours
16:   if ( ack for own request )
17:     send information over backward path
18:   end if
19: end on
```

the unicast address of the sender that thereby becomes part of the backward path. When the algorithm enters the information phase, this backward path is used to send information from the detecting cameras to the previously localised mobile terminal. The evaluation in Chapter 6 discusses the benefits of this approach. By using a unicast communication scheme, sending video and audio data requires less bandwidth than a broadcasting scheme would do. For better readability some points are not shown in Algorithm 8, e.g. explosion messages are re-broadcasted after a timeout in case no *ack* has been received for too long. The backward path is a branch of the tree that is established by all cameras on receiving an *ack*. By closing unused branches after some period of time in which no data has been received, the backward path is created that connects the *detecting camera* and a MAMT.

5.6 Algorithm Complexity

As mentioned before, Vasudevan intended only one instance of the election process to be active. The adaption to the notification problem explicitly needs to support multiple search requests to be active simultaneously as incidents might arise simultaneously, too. This has an impact on the algorithms message complexity. This section shows, that although a flooding based method is used, the algorithm is suited well for today's IEEE

802.11 WLAN ad-hoc networks. It is assumed, that a partition of Smart Cameras contains n nodes. In case a single alarm is raised, $2 * n$ messages (one for Explosion, one for Echo) are sent. The information phase is not investigated here, since it is unicast based and hence no flooding related problems can arise. The best case message complexity is $O(n)$. This number of messages is taken for an ideal system where no message loss and collisions appear and no partitioning arises. The impact of multiple events being detected simultaneously is high: in case all nodes detect an event at the same time, the message complexity is $2 * n * n$, i.e. $O(n^2)$. A timing delay is used in order to keep the message overhead low, see algorithm 8, line 4. Timing delays of $1s$ have been proven to be suited well and decrease the traffic on the wireless channel while still maintaining short response times. The timing constraint leads to a bandwidth usage of approx. $n * 1$ messages per second. Since the messages are comparably small (approx. 500 Bytes for transmitting an *explosion* or *ack*), no fragmentation takes place and the overall data rate that is needed is approx. $n * 8 * 500Bit/s$. The bandwidth needed results to $400kBit/s$ for a network of 100 cameras sharing a collision domain. This worst case scenario shows, that a common IEEE 802.11 WLAN with data rates above 5.5 $MBit/s$ is suited well for the notification algorithm presented here. Since the impact of collisions and timing delays is difficult to investigate theoretically, the reader is referred to simulation experiments presented in Section 6.6.

5.6.1 Conclusion

This chapter contains the description of two PTZ alignment problems arising in Distributed Vision Networks. After a formal description of these problems and a discussion of their complexity, heuristics have been presented to approximate solutions for these problems.

With ROCAS, an algorithm for static alignment of PTZ heads has been introduced. This algorithm enables Smart Cameras to align their viewsheds in such way so that the surveillance coverage becomes maximal. The algorithms takes viewing obstacles and the priority of regions into account and is suited for real-world applications.

DMCtrac allows cameras to align their PTZ heads for object tracking. By identifying objects and following them in a cooperative manner, Smart Camera networks can acquire trajectories of moving objects with multiple cameras. This is achieved by predicting the object's motion vector and a handover mechanism to pass the tracking task from one camera to another. DMCtrac can help to relieve security staff from the trivial task of tracking objects. Thereby, precincts observed by Smart Cameras may become safer and

incidents might be detected beforehand.

A notification algorithm is presented, that allows users and cameras to interact. Cameras can inform users in case previously defined incidents have been detected. The algorithms presented here allow for a closer coupling of humans and cameras and avoids a central control console.

The next chapter contains an in-depth evaluation of the algorithms that have been introduced so far.

Chapter 6

Experiments and Evaluation

This chapter contains the evaluation of the proposed architecture and algorithms. Relevant metrics that allow to quantise the system's performance are explained first. Afterwards, experiments are presented. The experiments that comprise the evaluation are derived from application scenarios for Smart Cameras, e.g. for security systems. Some experiments have been carried out in a simulation environment whereas others are carried out in a real-world testbed. Simulation experiments allow for large networks to be investigated (with up to hundreds of cooperating cameras). Real-world experiments have been conducted with up to ten cameras and confirm the assumptions taken for the simulation experiments (e.g. for image recognition complexity, movement speed of objects and camera heads).

6.1 Performance Metrics

Different metrics can be applied in order to measure the performance of the system presented in this thesis. The overall architecture can be evaluated in terms of how well it can cope with failure of Smart Cameras and how fast events of interest are transferred from the network to security staff. For the alignment of cameras, the surveillance coverage is of major importance, i.e. how well an area of interest is observed by the cameras. For the tracking of objects, the tracking quality in terms of detected movements is a valuable metric to gain deeper insight of the performance.

Each of the following subsections contains a short introduction to the management algorithm that is investigated and a number of evaluation aspects formulated as questions. These questions are answered in this chapter.

Performance Metrics for Basic Management Algorithms

The architecture of a Distributed Vision Network as presented in this thesis can be investigated with respect to its performance in terms of time and message complexity. Apart from costs for installation and maintenance (which are not discussed here), a user of such system may be interested in the following performance criteria:

- How much time is needed for system startup? Before system startup, we expect cameras to be turned off completely. When they are turned on at any time t_0, they need to learn to know each other and establish neighbourhoods. After the last camera is configured properly at t_1, the startup time $t_s = t_1 - t_0$ is calculated.

- What is the bandwidth consumption for the basic management algorithms? By counting the messages m with size s exchanged between cameras, we can calculate the bandwidth consumption to $b{=}s * m$. Thereby an insight is given on how well todays communication systems can cope with our algorithms.

- How long does it take to detect failing nodes? In case a node fails at time t_{f0}, we need to know how fast neighbouring cameras correspond to this failure and at what time t_{f1} the compensation process is finished. Our goal is to keep the time to compensation $t_c = t_{f1} - t_{f0}$ as short as possible.

- How long does it take to integrate new nodes in the system? After a new node (or formerly failing node) is switched on, it needs to be integrated into the camera network. The time between switching the camera on t_{in0} and its successful integration at time t_{in1} can be calculated.

- How long does it take to elect a leader? The leader election algorithm for role assignment can be started by any camera at time t_{e0} and finishes after at t_{e1} all cameras are informed about which node has been elected as leader. The time difference $t_e = t_{e1} - t_{e0}$ is used evaluate the performance of the leader election algorithm.

Performance Metrics for ROCAS

The spatial partitioning of an area under surveillance in a static environment (i.e. with no objects to be tracked) is addressed by ROCAS, a distributed heuristic introduced in Section 5.1. Apart from time and message complexity, the quality of the solution in terms of area coverage is important.

- Time complexity: How long does it take until all cameras have chosen their optimal headings (i.e. the algorithm terminates)?
- What is the computation complexity for the partitioning algorithm running on each camera?
- Message complexity: How many messages need to be sent, until the algorithm terminates?
- Robustness: How do disturbances in form of packet loss influence the system's performance?
- How accurate is the partitioning? How much does the surveillance quality increase?

Performance Metrics for DMCtrac

For the tracking of moving objects, DMCtrac has been developed. This network protocol has been designed for real-time tracking with multiple PTZ cameras. It is an extension of ROCAS and some evaluation aspects are already discussed within the evaluation section for ROCAS (Section 6.4). Apart from message complexity and robustness, the tracking quality as introduced in Section 5.4.2 is considered for the further evaluation.

- Quality: How well can objects be tracked? We therefore define the optimum tracking quality for an object Q_o, that could be reached in case the cameras had no further tasks they are expected to carry out. In a real world system Q_o can often not be achieved and an object's tracking quality results to a real tracking quality Q. The ratio $Q_r = \frac{Q}{Q_o}$ defines the resulting tracking quality.
- PTZ wearout: How does the movement strategy affect the tracking quality? In order to avoid cameras to fail because of excessive PTZ usage, we aim at keeping the number of camera movements n low. A reduction of n might lead to a decrease in tracking quality Q_r but prolong the lifetime of the cameras' PTZ drives.

Performance Metrics for User Interaction and Notification

The bi-directional nature of communication between humans and cameras requires the evaluation to be split up into two parts. On the one hand, guards emit search requests by using mobile terminals. On the other hand, cameras notify guards, in case objects of interest have been found or serious incident are anticipated. Timing issues are of major importance for both of these tasks.

In case a guard poses a search request, the time complexity is defined as the time needed for transferring the search request from a mobile terminal to all cameras in the network. For the notification task, the time complexity is defined as the timespan between the detection of an event by Smart Cameras and the delivery of an alarm message that instructs staff to investigate the scene further.

The following questions are answered in the evaluation section of the alarm management algorithm (Section 6.6)

- How long does it take to transfer a search request from a mobile terminal to all cameras? By calculating the timespan $t_s = t_{s1} - t_{s0}$ with t_{s0} being the time of the user emitting the search request and t_{s1} being the time at which the last camera receives the request.
- How long does it take to notify a guard in case an incident is detected? The timespan t_n between a camera detecting an event and the guard being notified is an important performance metric.

Several parameters are identified, that influence the systems' performance we have defined with the metrics presented above. All metrics introduced above are measured under the influence of at least one of the following parameters:

- System size in number of cameras
- Message loss on the communication channel
- Protocol timing (namely t_i as introduced in Section 4.3.1)
- Movement threshold to minimise number of PTZ turnings

6.2 Experimental Setup

The simulation environment that is used for the evaluation of the system architecture and algorithms is based upon the network simulator NS2 [94]. By using the Click Modular Router [73] and NSClick [74], the core algorithms can be run either inside the simulation framework or on real world Linux systems. An algorithm that has been implemented and tested inside the simulation framework can easily be ported to real world Smart Cameras.

For setting up simulation scenarios, a graphical user interface is used, that allows to place cameras on 2-dimensional maps. These maps can for example be floorplans of buildings or precincts and contain viewing obstacles (such as walls or trees) and prioritised

regions (like entrance doors or fire exits). Figure 6.1 shows a screenshot of the planning tool containing a map of Hanover airport. The field of view that cameras can possibly observe by rotating their PTZ head is indicated by a circle. The actual field of view of each camera is indicated by a triangle, see [95] for details. A visualisation frontend allows to display the alignment process of the cameras as initiated by ROCAS and DMCtrac and calculates the global results for observation quality. This is necessary, since the cameras act with local knowledge only and at no time a consistent snapshot of all cameras exists. After a simulation run has finished, the global history is created as described in [96] and [87]. Screenshots of these tools are shown in Figures 6.1 and 6.2. These tools were used for setting up and analysing all simulation experiments that are discussed in the following.

Figure 6.1: Planning tool

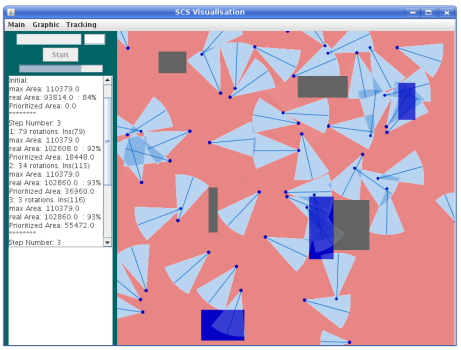

Figure 6.2: GUI for visualisation

6.3 Parameters of Simulation Experiments

This section shortly describes several parameters that have been used to model the environment for simulation experiments. The network simulator NS2 allows to model a realistic wave propagation model. For the IEEE 802.11 wireless LAN, the two-ray-ground model is used that models packet loss and reflections with high accuracy [97]. The communication range is set to $160m$, a valid assumption for IEEE 802.11 WLAN devices in an outdoor environment. The cameras are placed on a plain surface, $2m$ above the ground. The field of view of each camera is set to $50m$ viewing range and 45° viewing angle. The movement speed of the PTZ camera (Axis 214 PTZ) is 45° per s. The zoom and pan ability of the cameras is not used in the following.

Parameter	Value
Network size	10..1000 cameras
Area size	$100m * 100m..800m * 800m$
Transmission range IEEE 802.11 WLAN	$160m$
Viewing range	$50m$
Viewing angle	$45°$
Movement threshold	3%
Backoff interval	$350ms$
Startup interval	$1000ms$
PTZ movement speed	$45°/s$

Table 6.1: Parameters for simulation experiments

6.4 Startup and Failure Compensation

At system startup, cameras exchange state information and keep neighbourhood caches that store information gathered from surrounding nodes. This information includes position and alignment of the cameras' PTZ heads. In Section 4.3.1, an algorithm for establishing and maintaining the neighbourhood cache is described. As mentioned before, timing constraints are an important factor in camera networks. The time difference between cameras detecting events of interest and guards being informed has an impact on whether a critical incident can be prevented in time or not. System startup is expected to happen on rare occasions, e.g. after a power failure has occured or after an attack has been launched against the camera network. It is therefore crucial for the overall system performance to start up quickly. In the diagram displayed in Figure 6.3, the time-span needed to start the system is indicated by a blue bar. Since ROCAS is closely coupled to message exchange and neighbourhood cache, the performance evaluation of ROCAS gives an insight on how fast the neighbourhoods are formed (Figure 6.3 also includes experimental results for node failure, but for now the focus is on system startup). The diagram shows the area coverage in metres square over time. The ROCAS algorithm reaches a stable maximum in coverage after below $5s$. This indicates, that the neighbourhoods have been established in advance. This time span seems very reasonable for a camera system.

6.4.1 Detection of Joining and Failing Cameras

As proposed before, the algorithm presented in Section 4.3.3 allows for a reconfiguration of a system in case new cameras are installed. These new cameras start sending their SPMs and make themselves known to neighbouring cameras. This algorithm makes cameras hot-pluggable and renders manual reconfiguration unnecessary. Thereby downtime and

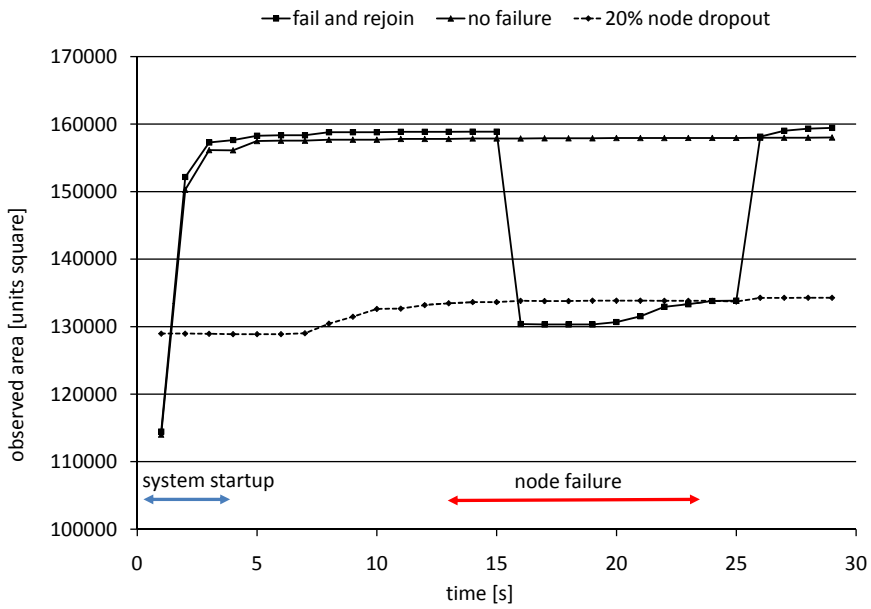

Figure 6.3: Overlap over time for a scenario with nodes failing and joining

costs for staff are avoided.

In large networks of cooperating cameras, failure of components becomes inevitable. ROCAS detects failing nodes by keeping timestamps in the neighbourhood cache. In case a neighbour did not send a message for a certain period of time, it is deleted from the list of possibly overlapping neighbours.

Figure 6.3 shows, how the failure of nodes is compensated. Since the failure of a large amount of cameras is more critical than failure of single nodes and results are more significant, a scenario with numerous failing/joining cameras is presented.

Three scenarios have been simulated in order to show how failure of nodes is compensated and how new joining nodes are integrated into the camera network. Two failure-free scenarios have been set up, showing the development of overlap for 80 and 100 SCs. For the third scenario 100 cameras are used at startup, but after $t_A = 13s$, 20 randomly chosen cameras fail. These cameras are those that are missing in the scenario containing 80 instead of 100 cameras. Thereby, the results of all three runs can be compared to each other. In reality, the sudden failure of 20 cameras may occur in the case of a failing power supply. For this experiment, after $t_B = 23s$, the formerly failing nodes are put back into operation. Figure 6.3 shows that the overlap caused by failing and joining nodes is nearly equivalent to the scenarios without failing/joining nodes. Detecting failing and joining nodes only takes a few seconds, depending on the interval of the heartbeat signals. I.e.,

given a heartbeat frequency of $f_{heartbeat} = \frac{1}{t_i} = 1Hz$, and a tolerated delay for messages of $t_{tolerated} = 2 * t_i$, failing nodes are detected in at least $2 * t_i = 2s$. This worst case detection time is $t_{tolerated}$ and results from a node failing immediately after sending a message. A receiving node will then wait for the full length of $t_{tolerated}$ before detecting the failure. In case a node fails directly before it was due to send a heartbeat message, the detection time results to $t_d = t_{tolerated} - t_1$.

For the example shown in Figure 6.3, node failure occurs directly after a message has been sent. Thus, the detection time is $t_d = 2s$ for a heartbeat interval of $t_i = 1s$ and $t_{tolerated} = 2s$

6.4.2 Scalability of ROCAS

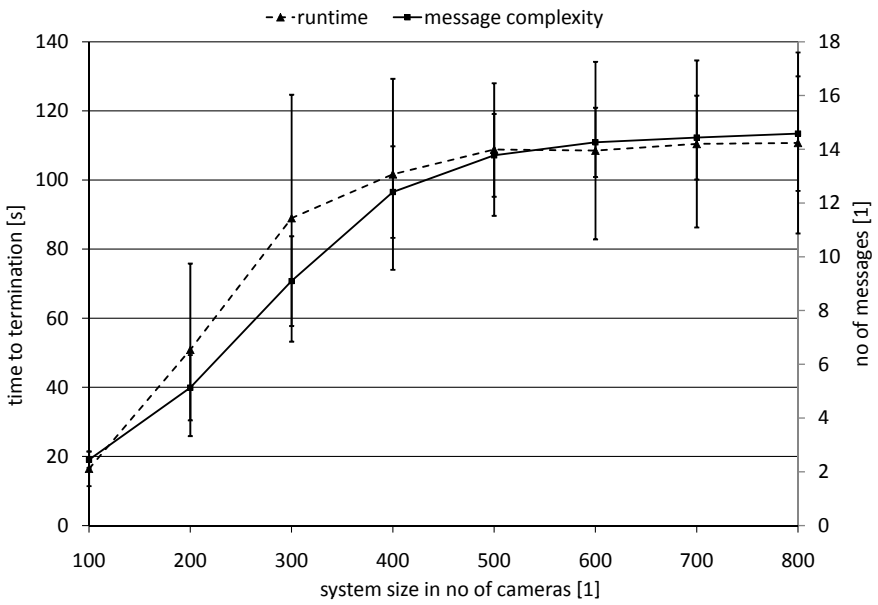

Figure 6.4: Scalability of ROCAS: Average time to termination and message complexity in networks of 100 to 800 cameras, error bars indicate standard deviation

The scalability of ROCAS is investigated with the following scenario. Smart Cameras are placed on an area $\mathcal{A}$ of $800x800$ metres square. $\mathcal{A}$ has been divided into $j = 640000$ subareas A_j. For this experiment, the priority $w(A_j)$ of the area under surveillance is kept constant. This is a worst case assumption, since the search-space for optimal camera alignment becomes larger due to the equivalent weights of all subareas. With obstacles and weighted regions present, the algorithm has less opportunities to choose the cameras' alignment and might terminate faster. The network size varies between 100 and 800 cam-

eras. For each scenario, 30 simulation runs have been carried out and for each simulation run, the initial alignment of each camera's PTZ head has been chosen randomly. In order to prove the robustness of the algorithm, the communication channel has been modelled with a packet loss of 80% as described in Section 6.4.4.

Figure 6.4 shows, how the time to termination increases with the number of cameras inside the network. Results show, that ROCAS can be deployed even in very large systems consisting of up to 800 cameras. The average time to termination does not exceed $120s$ even for a system with 1000 cameras, which is very promising for highly sensitive real-world applications. Smaller systems do need less time to find an optimal alignment than large systems. For systems with a size of 500 cameras and more, the time to termination is constantly high.

This is due to the fact, that at some point in time the cameras can not optimise their alignment further. This upper bound results from the network geometry and the impact of message collisions on the shared media as described in detail in Section 5.3.1. The impact of collisions and the limited horizon of each cameras' knowledge lead to chain effects in the optimisation process. Local optimisations have impact on far away cameras. Therefore the time to termination varies significantly, see error bars in Figure 6.4.

6.4.3 Bandwidth Consumption of ROCAS

The bandwidth needed by ROCAS can be calculated as follows: We assume an IEEE 802.11 WLAN in ad-hoc mode to be used. The frame size is low, since we send IP packets that contain just position and alignment information. The total length of a frame is 100 Bytes containing Ethernet header, IP information and the payload. The maximum message complexity can be derived from Figure 6.4. We assume a worst case scenario, in which 100 cameras share a collision domain and start sending messages simultaneously and 80% of the messages get lost on the communication channel. These 100 cameras sent $1s$ message per second, as introduced in Section 4.3.1. The bandwidth results to $B = 100 * 800 * 8Bit/s = 64kBit/s$. This bandwidth does not take retransmissions into account but in comparison to the minimum bandwidth of 5.5 MBit/s in IEEE 802.11 WLAN networks, the bandwidth amount needed by ROCAS is very reasonable.

6.4.4 ROCAS and Message Loss

As stated before, in a wireless environment the communication between Smart Cameras may be disturbed by message loss. Message loss occurs in case packets collide on the

shared media. As investigated in Section 5.3.1, the chance of message loss increases with the number of cameras sharing the same collision domain (i.e. the shared communication range). In order to evaluate, how a lossy communication channel influences the performance of ROCAS, the following scenario has been investigated: a number of 100 to 350 Smart Cameras is positioned randomly on a constant area. The loss on the communication channel is expected to be equally distributed. I.e., packet loss does not happen in bursts but single packets get lost randomly as caused by collisions on the shared medium or changes in the network topology due to node failure. The loss rate is set to 20% and compared to a perfect channel with 0% message loss. Figure 6.5 shows how the performance of ROCAS decreases with the raise in packet loss. In small systems of 100 cameras, the loss does not influence the quality significantly. For larger systems, the achieved surveillance quality is slightly lower. This is caused by alignment messages getting lost and neighbourhood inconsistency. For a network consisting of 350 cameras, the surveillance quality achieved with a loss free communication channel results to 92%. With 20% message loss, the quality results to 90%. The runtime of the algorithm has been kept constant for this experiment. Due to message repetition, the impact of message loss vanishes over time. This experiment shows, that ROCAS is robust towards losses on the communication channel and can be expected to work well in harsh real-world environments.

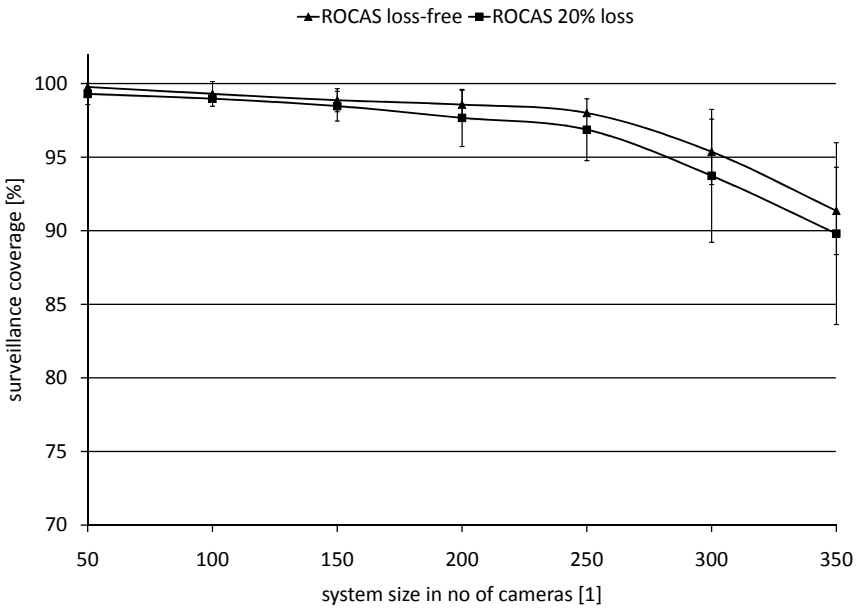

Figure 6.5: ROCAS and message loss: Surveillance quality decreases with message loss and system size.

6.4.5 Exploration of Parameter Space

As mentioned before, ROCAS can be adapted to various user demands by changing a set of parameters. Two main parameters are used to adapt ROCAS to changing requirements that might arise from varying user demands or changes in the Smart Cameras environment:

- Movement threshold as introduced in Section 5.3
- Backoff interval as introduced in Section 4.3.1
- Startup delay

The movement threshold th_m avoids that Smart Cameras carry out slight changes that might damage their PTZ drives. E.g., a movement threshold set to 3% avoids, that Smart Cameras carry out movements that lead to an increase of surveillance coverage below 3%. These slight changes might on the one hand improve the surveillance coverage but may on the other hand cause harm to the cameras' drives that tire from too many movements.

The backoff interval t_i specifies the timespan a camera backs off from the shared media after having sent a message. This parameter has been introduced in Section 4.3.1. A backoff interval that has been selected too short causes collisions on the shared media. Many messages might be sent in vain. A backoff interval that is chosen too long will cause the overall runtime of the distributed algorithm to be longer than necessary. The optimum parameter depends on the size of the collision domains in the network.

The startup delay is closely related to the backoff interval. At system startup, all cameras emit messages to establish their neighbourhood caches. Hence, the backoff interval is extended and defined as startup timer. The impact of the backoff interval is not investigated here but in the following Section 6.4.8 where the exploration of parameter space is discussed.

The following experiments have been carried out in networks comprised by 20 to 100 cameras. The movement threshold is varied between 1% and 10% and the backoff interval is changed between $100ms$ and $1000ms$. The cameras are positioned on a constant area $\mathcal{A}$, with an area of $285m * 285m$. Further simulation parameters are shown in Table 6.1. Parameter changes have impact on the systems performance in terms of time and message complexity, quality of surveillance coverage and the number of movements carried out by the cameras' PTZ drives.

6.4.6 Movement Threshold

The impact of the movement threshold on the time complexity can be seen from Figure 6.6. A decreasing movement threshold leads to an increasing time complexity. The accuracy of the solution found by the algorithm decreases with an increasing movement threshold. Intuitively, a precise alignment of cameras with 1% accuracy takes longer than a rather imprecise alignment where solutions are accepted only in case of a 10% accuracy. As expected, in large systems more time is needed to finish the partitioning than in small systems: in a system with 20 cameras, the algorithms needs about $1s$ to terminate, whereas in large systems with 100 cameras the time to termination is up to $4.5s$.

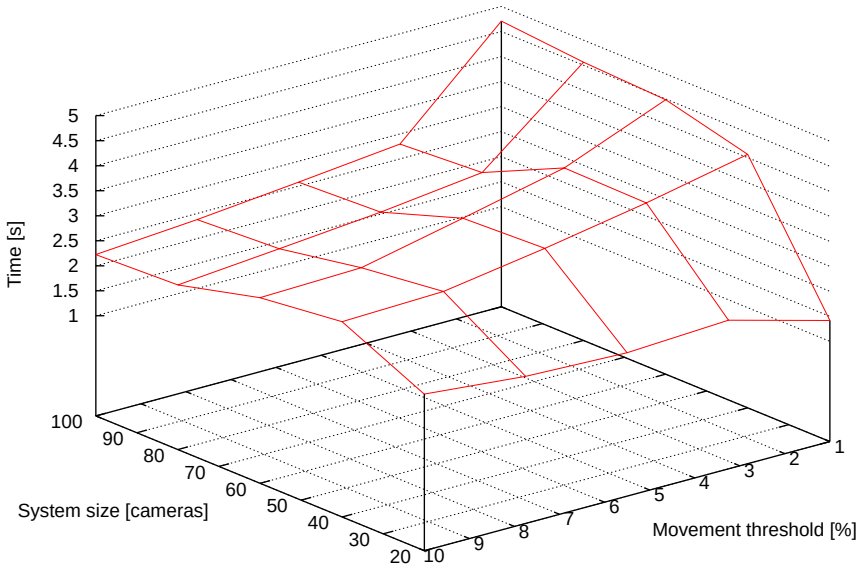

Figure 6.6: Time to termination depending on system size and movement threshold ($600ms$ backoff interval)

The increase in surveillance coverage achieved by ROCAS depends on the system size, but also on the movement threshold as can be seen from diagram 6.7. The larger the movement threshold is chosen, the lower is the increase in surveillance coverage. A movement threshold of 10% leads to an increase of 4%..12% for various system sizes. A 1% threshold allows for a quality increase of up to 16% for a scenario with 60 cameras.

Note, that the quality decreases for larger systems (movement threshold 1%, system size 80 and 100). This is due to overlap that can not be avoided by the cameras since they are already packed with high density on the area $\mathcal{A}$. The coverage can therefore not be increased any further.

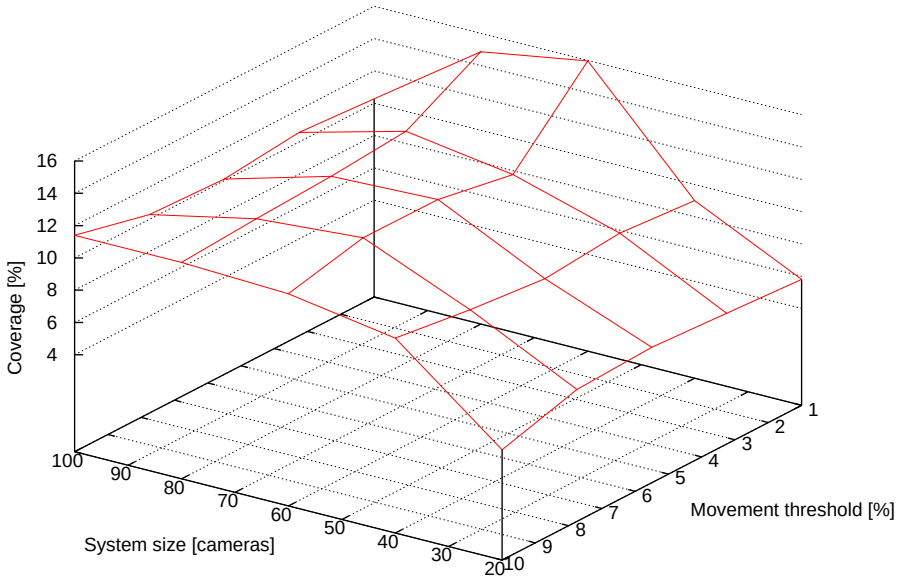

Figure 6.7: Increase in surveillance coverage depending on system size and movement threshold (600ms backoff interval)

The benefit of a large movement threshold becomes obvious in Figure 6.8: especially in large systems, the number of pan/tilt turnings can be minimised significantly by tolerating a 10% overlap in form of the movement threshold. In a scenario consisting of 100 cameras, the average number of turnings can be reduced from 8 for a movement threshold of 1% to 5 if a 10% threshold is chosen. The saving of nearly 40% in camera movements directly adds to the lifetime of the cameras. Operators of large networks might therefore switch between various modes of operation to find the most favourable working point.

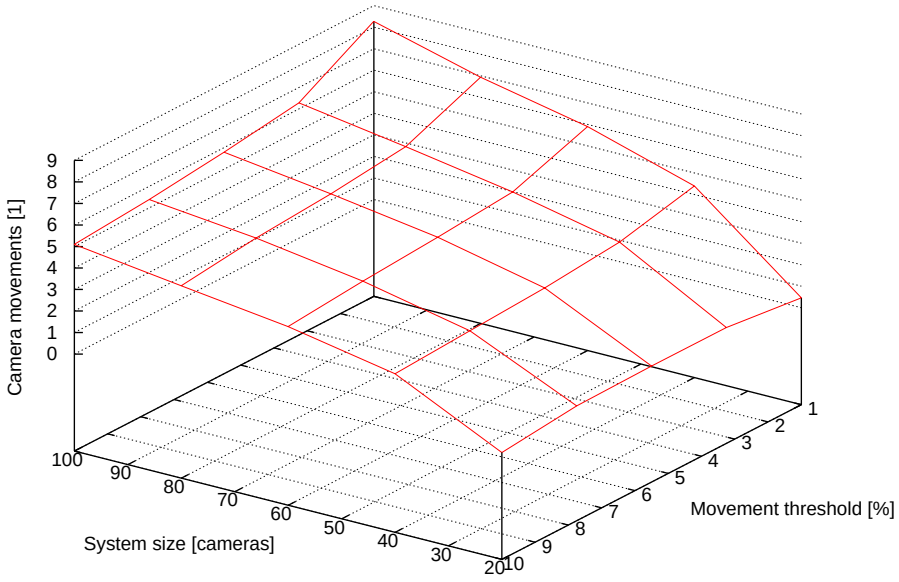

Figure 6.8: Average number of camera movements depending on system size and movement threshold ($600ms$ backoff interval)

6.4.7 Backoff Interval

The impact of the backoff interval t_i on the time complexity can be seen from Figure 6.9. As expected, an increasing backoff interval leads to an increasing time complexity. With $t_i = 100ms$, a system with 20 cameras is aligned in $300ms$. A larger system (with 100 cameras) needs $1s$ to finish the alignment process. If t_i is set to $1s$, the time to termination goes up to $5.8s$.

In return, the average number of messages that needs to be sent by the cameras increases significantly for small values of t_i. For $t_i = 100ms$, in a system consisting of 100 cameras each camera needs to send an average number of up to 2.6 messages until the alignment is completed. For larger values of t_i (e.g. $t_i = 0.5s$), only 2 messages per node need to be sent. Thereby, bandwidth can be saved and wireless nodes can save energy for data transmission (for example to prolong the lifetime of their batteries).

Figure 6.11 shows the impact of a rising backoff interval on the average number of movements carried out by the cameras' drives. Especially for a large system size, the number of movements decreases with an increasing backoff interval. This is due to the

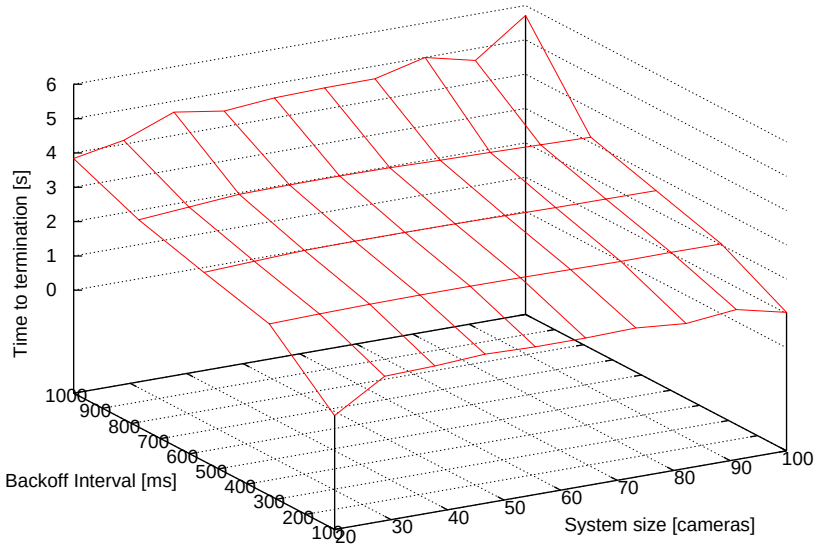

Figure 6.9: Time to termination depending on system size and backoff interval (4% movement threshold)

fact that the probability for collisions and simultaneous alignment as introduced in Section 5.3.1 is influenced positively by larger backup intervals. Thereby, unnecessary movements can be avoided by reducing conflicts arising from the backoff interval chosen too short.

6.4.8 Automatic Exploration of Parameter Space

The subsequent section presented experimental results considering different parameter settings. The complete exploration of the parameter space is hardly feasible. Thus, an evolutionary algorithm has been used to optimise the parameter setting for ROCAS. This flexible tool called POWEA (protocol optimisation with evolutionary algorithms [98]) can be used for various network protocols and has been used for the optimisation of ROCAS as described in the following.

At first, a simulation scenario with NS2 needs to be set up and the configuration space of the parameters needs to be provided. The process of network protocol parameter optimisation consists of three subsequent steps that are carried out iteratively: generate

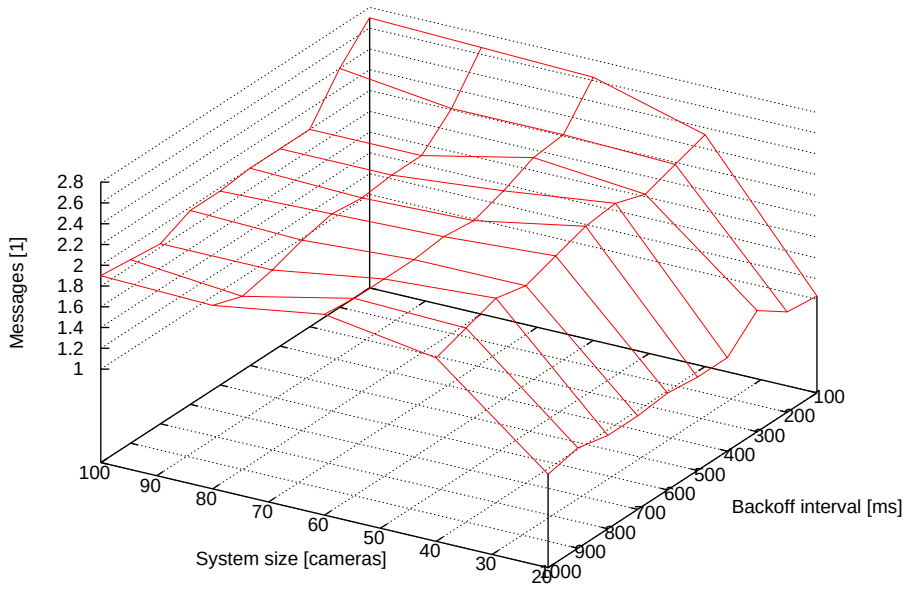

Figure 6.10: Message overhead depending on system size and backoff interval (4% movement threshold)

parameter sets with the EA (1), run simulation process (2), interpret results and calculate fitness (3). Therefore, the call procedure of the NS2 simulator has been modified with respect to providing parameters and reading simulation results. An interface between the protocol and POWEA bases on logfiles. The input parameters can thereby be changed without modifying the protocol logic of ROCAS. See Figure 6.12 for a graphical representation of the workflow.

The EA performs a predefined number of iterations (generations). The population-size as well as the number of children for the next generation have to be defined in advance. Another important factor is the mutation-rate, which describes the expectation value of the probability that one parameter (gene) is altered in that generation. For the basic configuration of the EA the values as described within Table 6.2 have been used. These values have been adapted from a configuration given in [99], where a traffic network is optimised by a similar EA.

The result of each simulation run (that evaluates one parameter set) is a logfile, which stores the sending and receiving events of the nodes within the supervised area. Thereby,

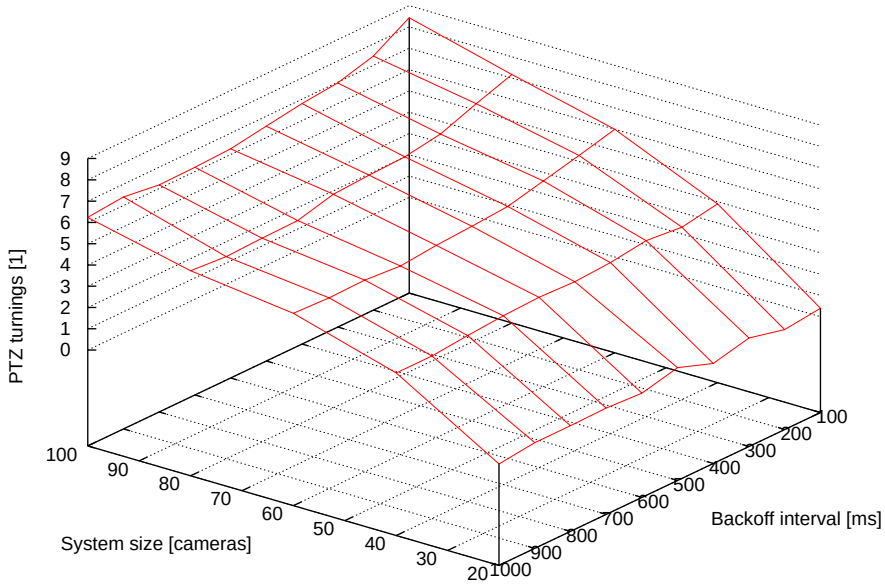

Figure 6.11: Average no of movements depending on system size and backoff interval (4% movement threshold)

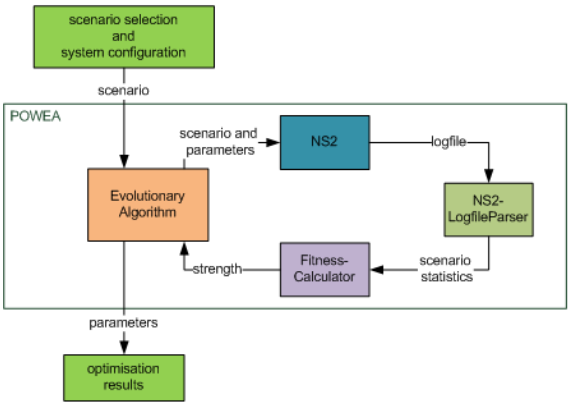

Figure 6.12: Workflow of the POWEA system

the overall performance of the parameter set can be measured. POWEA comes with a special parser to read and interpret these logfiles. The parser needs to be adapted by the user to new protocols in order to read the performance values that comprise the fitness function (e.g. coverage rate, message overhead, or combinations of these).

Variable	*Value*
Number of generations	18 cycles
Population size	10 individuals
Number of children per generation	6 individuals
Mutation rate	0.8 per child
Crossover rate	0.83 per child

Table 6.2: Configuration of the EA

The fitness function can be provided as a mathematical term in form of a JAVA class. E.g., for ROCAS the following weighted fitness function has been used: $fitness = messagecomplexity^{-1} * 40\% + timecomplexity^{-1} * 20\% + quality * 40\%$, where all metrics have been normalised (if the performance is better than the reference solution the evaluation-values are higher than 1.0 and vice-versa).

The optimisation process is performed until a stop criterion is fulfilled (e.g. a number of generations has been processed). If it continues, the EA generates a new generation by keeping the best individuals and substituting the bad (in terms of low fitness function values) individuals by generating new ones using genetic operations. By performing this process iteratively the performance of the scenario is increased over time.

Optimisation Results

The evaluation of the protocol is based upon scenarios with 8 to 80 Smart Cameras positioned on a grid. The parameters are varied between $100ms$ and $1000ms$ for startup delay and backoff time and 0..10% movement threshold.

Figure 6.13 shows the fitness values of generations the EA has delivered. Based upon a startup fitness of 1.0 for each scenario, POWEA always delivered better parameter sets than the reference solution. The best generations fitness is between 1.08 and 1.27. The best generation for each scenario has been chosen and 20 simulation runs have been carried out in order to further analyse the performance. Figure 6.14 shows an example for the optimisation concerning message complexity. The error bars indicate the standard deviation around the mean number of messages sent by each camera. About 10% of messages could be saved by using the best parameter sets discovered by POWEA. We expect this to pay back in terms of less power consumption which is important for battery powered nodes.

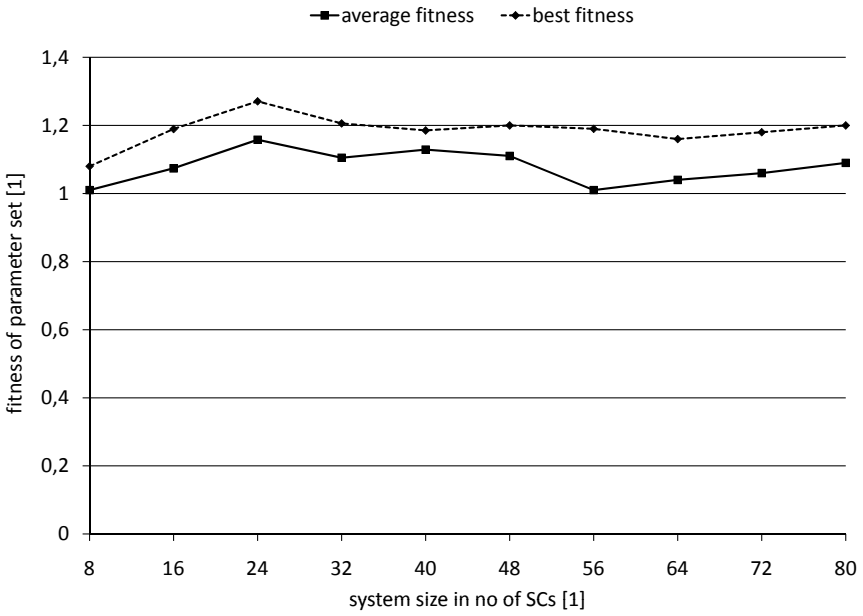

Figure 6.13: Average and best fitness values of parameter sets for ROCAS

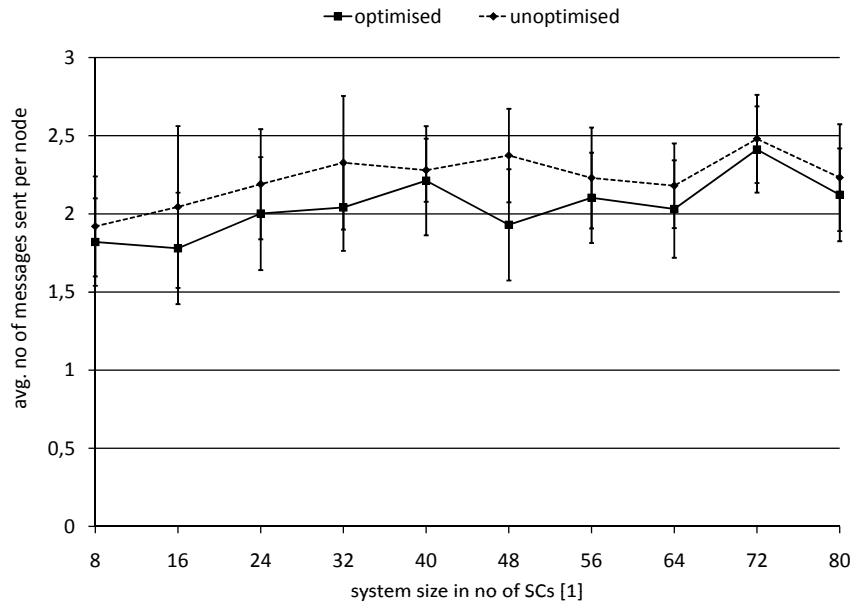

Figure 6.14: Optimisation results w.r.t. message complexity

6.4.9 Computation Complexity of Polygon Clipping

As mentioned before, ROCAS relies on a 2-dimensional approximation of the ground plane view, see Section 5.1. The overlap between cameras' fields of view, prioritised regions and obstacles is based upon a polygon clipping algorithm introduced by Vatti in [89]. The clipping algorithm has a time complexity $O(n \times m)$, with n and m being the number of edges of the two polygons. For ROCAS, only simple polygons (without holes and self-intersections) are used. Since a camera's field of view is modelled by a triangle and obstacles and priority regions are modelled as rectangles, ROCAS makes use of polygons with three to four vertices. Since each camera has to clip its own ground plane view with all neighbouring cameras and regions in order to find an optimal heading, the runtime complexity needs to be taken into account in order to evaluate the systems's performance. The clipping algorithm returns the resulting overlap polygon. The area of this polygon is calculated by the Gaussian trapeze formula $o_a = \frac{1}{2} \times \sum_{i=1}^{n}(y_i + y_{i+1})(x_i - x_{i-1})$. Figure 6.15 shows the time needed for polygon clipping on a Smart Camera: the camera calculates its overlap in field of view with 10 to 100 polygons. For each scenario 100 runs of the clipping algorithm have been carried out in order to derive average values and standard deviation. The time needed for the calculation of overlap until an optimal position is found takes below $150ms$ even in case of the 100 polygon scenario.

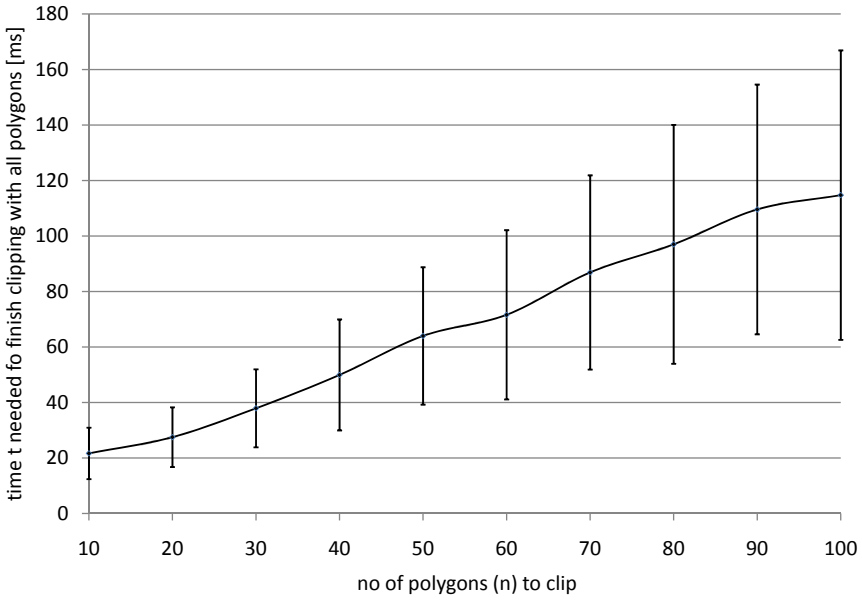

Figure 6.15: Runtime complexity of clipping algorithm for 10 to 100 polygons, error bars show standard deviation from mean

In comparison to the image recognition tasks carried out on the 1.6 GHz Atom based

prototypes, polygon clipping has only minor impact on the system's overall performance. For comparison: the Viola Jones algorithm [22] takes $200ms$ to detect faces in an image with 640×480 pixels.

An approach to increase the accuracy of ROCAS is to switch from a 2-dimensional ground plane view to a 3-dimensional model of the cameras' fields of view and obstacles. Since the 3-dimensional clipping of volumetric shapes is much more complex than just clipping polygons, the influences on the runtime of ROCAS need to be considered carefully.

6.4.10 Comparison of Distributed and Centralised Approach to Partitioning

As stated before, a centralised algorithm for camera alignment has been developed. By introducing an election scheme as presented in Section 4.3.4, the system elects a central entity that takes over certain tasks. An experiment has been conducted that uses a centralised algorithm for camera alignment in order to discuss benefits and drawbacks of a centralised approach. For the centralised variant of the ROCAS algorithm, a leader needs to be elected to carry out the global optimisation of overlap. With a simulation experiment, the time and message complexity of the leader election has been analysed as described in the following.

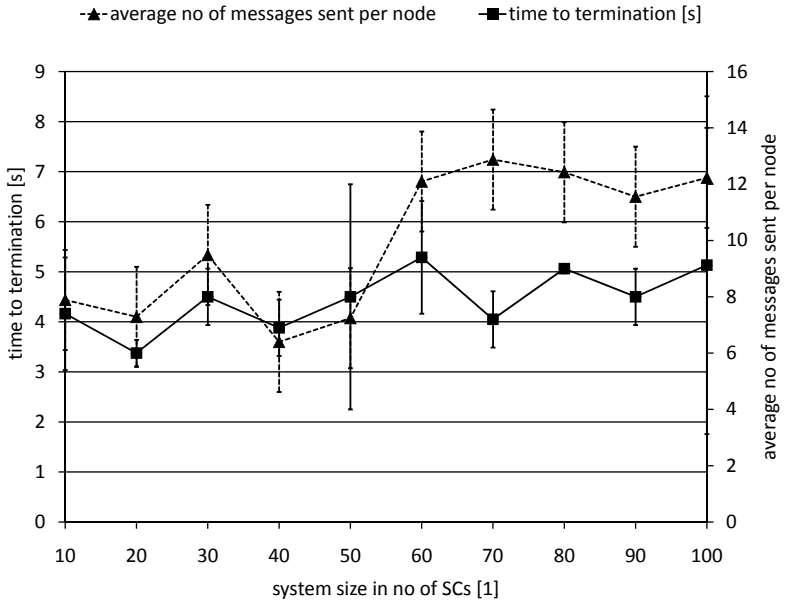

Figure 6.16: Evaluation of election algorithm

Experiment

For simplicity, this experiments does not establish a hierarchical but a completely centralised structure, i.e. only one leader is elected and all other cameras become slaves. The network size is varied between 10 and 100 cameras. Figure 6.16 shows experimental results concerning runtime and message complexity for an adapted version of Vasudevan's election algorithm [84, 85]. The average time needed for electing a leader is between $4s$ and $6s$. The mean number of messages is between 4 and 13 per node.

A complete evaluation of the election process would also focus on other aspects, e.g. how long a node is without a leader and investigate the robustness of the election process in case a leader fails. Since the time needed for leader election is rather high, and decentralised variants are much faster, the evaluation is kept short here. For an in-depth description of the election algorithm and its impact on the system's performance confer to our work describing the election algorithm for Smart Cameras in detail [6].

The experiment described in this section reveals a basic problem of the leader election mechanism. The flooding based broadcast requires all cameras to forward a large number of election messages. Since the election process is started on several nodes simultaneously, a large amount of these messages is sent in vain until the final leader succeeds and its ID is published throughout the network. Thereby, the algorithm becomes rather time and bandwidth consuming. In case a leader fails (as can be detected by the algorithm introduced in Section 4.3.3), the election might even be have to carries out multiple times, which causes a significant drop in the overall system performance. This drawback might need to be taken into account in order to achieve an increased surveillance coverage as described in Section 6.4.10.

As shown in Section 6.4.9, the time needed for calculating the overlap of hundreds of polygons is rather high. The attempt of transferring the ROCAS algorithm as described so far directly to a central server is doomed to fail in terms of scalability. Therefore, another approach to the partitioning problem has been developed with respect to the special circumstances arising when a central server is used. This algorithm is faster but less accurate than ROCAS, see Section 5.3.2.

At startup, the cameras elect a leader among themselves that serves as centralised master. The master collects all position and alignment data from its slaves and calculates a global approximation to the alignment problem. Afterwards, the cameras are informed about their new optimal alignment.

Figure 6.17 shows experimental results. The time needed to finish the partitioning process is taken as primary quality criterion. The pure partitioning and alignment of cameras on the basis of a central server is very high. It takes approx. $1s$ to determine

a camera alignment for networks of 10 to 100 cameras. ROCAS needs up to 2s for this task. The drawback of the centralised system becomes obvious when the leader election is taken into account. The simulation of the election algorithms has been carried out for 10 simulation runs in each camera scenario. The node that starts the election is chosen randomly and during the election process, the node with the highest computing capability (indicated by its quality value) is chosen. Since these nodes can be positioned very close or far away from each other, the time to termination varies significantly (see error bars in Figure 6.17 that indicate standard deviation from mean). The election algorithms takes between 4s and 14s to terminate.

These results underline, that the use of a centralised component is critical since the time needed to finish the election is much higher than the time needed for local coordination of camera nodes without a leader as provided by ROCAS.

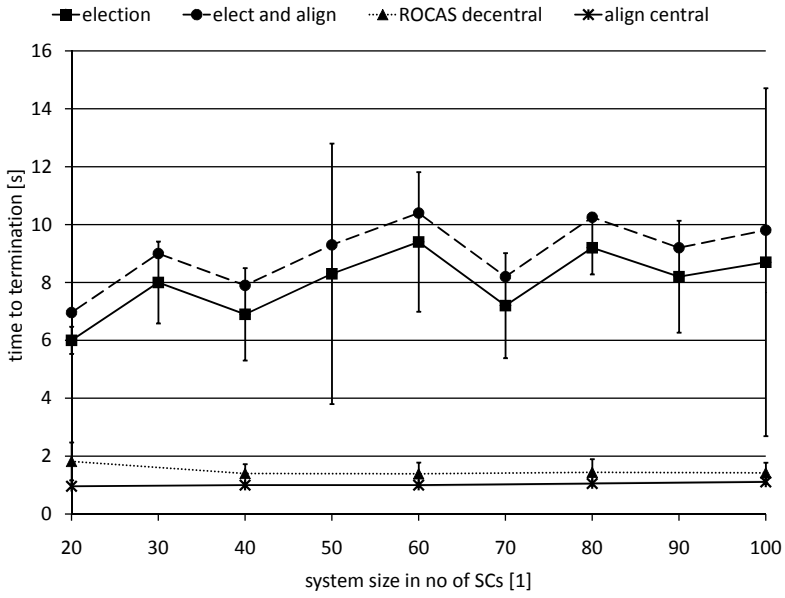

Figure 6.17: Centralised in comparison to decentralised partitioning

Maximum System Size

The decentralised system architecture allows to establish networks of potentially unlimited size. In case only local neighbourhoods are established, the system scales well.

With leader election and multi-hop routing, the system size is constraint. As presented in Figure 6.17, the central server approximates a solution to the coverage problem quickly (below 2s for a system of 100 cameras including data transfer).

subareas j	$Q_{central}$	t_{R_c} *in s*	*slowdown S*
16	92	4.7	2.35
25	97	4.7	2.35
400	99	5.98	2.99
1600	100	16.58	8.29

Table 6.3: Quality and time to termination of centralised ROCAS

For a small system of 18 cameras a comparison in terms of accuracy has been carried out. This system size suffices to show the limitations the centralised variant of ROCAS has to cope with. The cameras are positioned on a regular $3 * 3$ grid and observe an area $\mathcal{A} = 200 * 200m^2$.

The decentralised variant of ROCAS takes $t_{R_d} = 1s$ to finish and reaches a surveillance coverage of $Q_{decentral} = 100\%$. The area $\mathcal{A}$ is divided into $j = 200 * 200 = 40000$ subareas A_j with a size of $1m^2$ each. Depending on the number j of subareas A_j, the surveillance quality $Q_{central}$ and the time to termination t_t has been investigated. Again, t_t includes the time overhead t_e that is needed to elect a central entity and the time needed to calculate an optimal camera alignment t_a. For the simulation experiment carried out here, t_e has been measured to be $4s$. Table 6.3 shows experimental results. The left column contains the number j of subareas A_j. The grid is varied between $j = 25..1600$. The column entitled t_{R_c} displays the time to termination. In order to compare the results achieved by the centralised variant of ROCAS to the decentralised variant, the slowdown has been calculated to $S = \frac{t_{Rc}}{t_{R_d}}$.

This experiments shows, that the centralised approach to camera alignment suffers from performance drawbacks in comparison to the decentralised approach. Not only the time to termination is longer but also the quality decreases significantly. Only for special situations, where cameras are positioned in such way so that they block each others field of view, the centralised variant of ROCAS allows for an increasing surveillance coverage in comparison to ROCAS. See Section 5.3.2 for an example and a detailed description of such a scenario.

6.5 Tracking Algorithm DMCtrac

DMCtrac is a distributed algorithm for object handover in large networks of PTZ Smart Cameras. In the research field of computer vision, several algorithms are in use for object recognition and handover between cameras. DMCtrac builds upon these basic algorithm and allows for a self-organised tracking of multiple objects with multiple PTZ cameras.

In order to evaluate the performance of DMCtrac, a metric has been defined, that gives an insight how well the distributed algorithm can cope with varying numbers of cameras tracking different numbers of objects, see Section 5.4.

6.5.1 Simulation Environment for Tracking

With DMCtrac, Smart Cameras are enabled to track objects cooperatively. In order to generate realistic object movements, a pseudo-random trajectory generator is used. Objects tracked by the Smart Cameras in our application scenarios are either vehicles or humans. These objects usually do not follow random movement models but move on on predefined routes based upon a concrete intention. E.g., on an airport ground vehicles visit planes to refuel them or supply them with food. Tracking objects on their usual routes allows to detect potentially dangerous situations. These routes vary only slightly from each other and routes that differ significantly from common ones may be rated as suspicious. The trajectories that are fed into DMCtrac can be characterised by the frequency of their appearance. After having adjusted to a large number of unsuspicious tracks, the cameras may be required to detect a single suspicious movement. Thus, the simulation environment allows to let objects move on predefined trajectories. These trajectories are then replayed to the simulated Smart Cameras in order to evaluate their abilities to follow objects in the most efficient manner and detect anomalies as fast as possible.

6.5.2 DMCtrac: Evaluation

Figure 6.18 shows an overall evaluation of DMCtrac. The scenario is based upon a virtual testbed derived from a university campus. The typical routes taken by students and university staff are taken as input for the tracking scenario. In [5], the setting is described in detail. 12 spatially adjacent cameras are positioned in such a way, so that their FOVs potentially overlap slightly. The simulation has been carried out for a time-span of $720s$. The trajectory investigated has a length of $462m$, leading around the whole precinct. Therefore, each object stays inside the FOV of the SCs for $356s$ (persons move with a speed of $1.3m/s$). The first object starts at time $t = 0s$ and the last object at $t = 356s$. I.e., in case 10 objects are fed into the simulator, every $35.6s$ a new object enters the cameras' FOVs. A single object moving through the FOVs of the system could ideally be tracked all the time (i.e. $Q = 100\%$). Each camera tries to follow an object as long as it stays inside its FOV. For this experiment, the average length of a trajectory inside

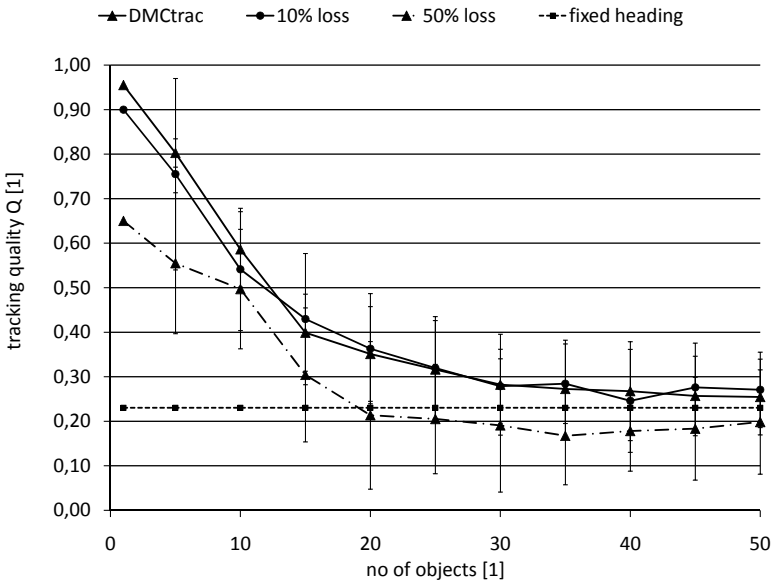

Figure 6.18: Evaluation of DMCtrac: tracking quality for fourteen SCs tracking multiple (1..50) objects

a single camera's field of view is $50m$ (calculated manually). This corresponds to a time of $38.5s$. Without searching for objects and no delays caused by SC movement, the ideal $Q = 100\%$ could theoretically be reached for up to $\frac{356s}{38.5s} = 9.2$ objects.

DMCtrac reaches $Q = 96\%$ for a single object. For a modest number of objects to be tracked, the usage of PTZ cameras pays back in full. The SCs are able to track 10 objects while maintaining an average $Q = 60\%$ per object. The standard deviation of Q remains low at about 8%. Q decreases the more objects are to be tracked. This is due to the fact that SCs are busy tracking objects and no capabilities are left to take over other objects. Cameras switch to *search* mode from time to time in order to detect objects that currently are not tracked. This takes some time since the rotation speed of SCs and the time needed for object detection need to be considered, too.

It has further been investigated, how message loss affects the performance of DMCtrac. When using a wireless communication channel, a certain amount of the messages that the SC send gets lost on the communication channel. It is assumed that this packet loss happens randomly and equally distributed (not in bursts). Figure 6.18 shows, that the performance decreases only slightly and DMCtrac can cope well with packet loss of 10%. In case a packet loss of 50% occurs Q decreases significantly. When tracking more than 20 objects, the performance even drops below the value achieved by statically installed SCs. This is due to the fact, that missing $LOST$ and $VISIBLE$ messages

make SCs search constantly for other objects to track. This seriously affects the system's performance. DMCtrac is a very lightweight algorithms: tracking two objects requires the SC to exchange an average of 0.16 messages per s. This value increases moderately for 24 objects under investigation where each SC sends an average number of 0.43 messages per s.

Movement Strategies

The movement strategy for the observation of static scenes has been investigated by Mundhenk et al. [100]. By continuously panning and tilting a PTZ head, a large area can be observed with few cameras only. Small linear drives are used to pan and tilt the cameras' heads. The lifetime of these drives is constraint and for cost reasons, the number of camera movements is kept low.

In order to evaluate how the mechanical stress the PTZ drives are exposed to due to DMCtrac, the following two experiments have been carried out. For the first experiment, we let cameras track objects seamlessly. I.e., a camera rotates in fine granular steps of 1° in order to follow objects. For the second experiment we set up a threshold of 45°. The span angle of a camera's viewshed is 45°, too. Cameras do not pan before the change they need to carry out is larger than this threshold. Thereby, the camera tilts to a new viewing angle with no overlap to its previous position. Figure 6.19 shows the tracking quality Q achieved by both configurations. DMCtrac without movement threshold achieves values for Q that vary between 96% for a single object and 40% for 24 objects. In comparison, DMCtrac using a movement threshold achieves about 20% lower values for Q. The impact of PTZ movement on computer vision algorithms is not considered here. Figure 6.18 shows, that by setting a threshold, the number of changes carried out for the PTZ heading can be decreased significantly. Without movement threshold, SCs carry out up to 1 change per s, whereas the movement threshold reduces the average number of movements to 0.7 per s.

This experiment shows, that the mechanical stress the cameras are exposed to is correlated to the accuracy of the tracking result. For a real-world application of DMCtrac, the use of different tracking modes appears to be of value. In alarm situations, cameras may pan and tilt constantly in order to acquire a deeper and more accurate understanding of the scene they investigate. In case the alarm level is low, the cameras can minimise the mechanical stress on their PTZ heads by using a turning threshold and thereby achieve a lower tracking quality Q. The movement strategy can be adjusted either by the cameras detecting events of interest or by security staff putting the system into an alarm mode.

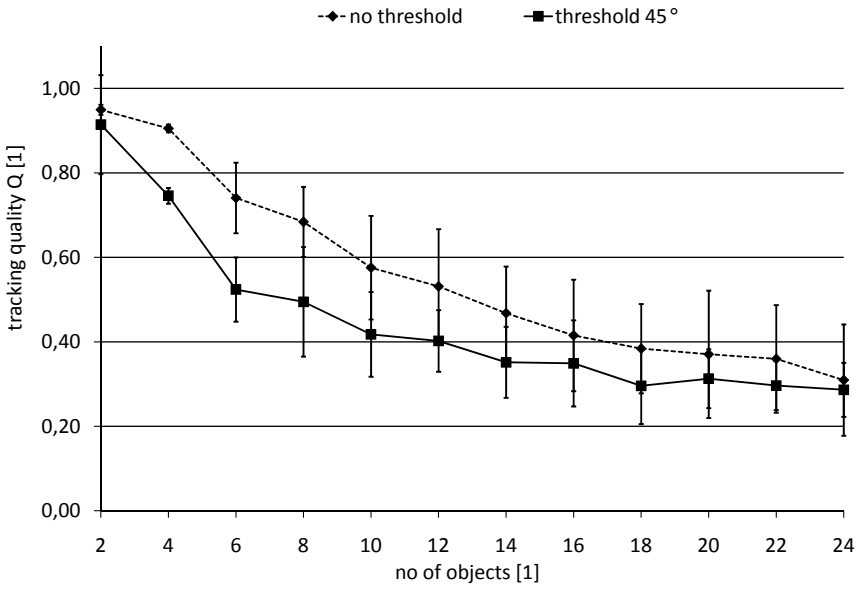

Figure 6.19: Tracking quality depending on movement strategy

6.6 Evaluation: Alarm Management

The evaluation of the alarm management algorithm has been carried out in a simulated environment as well as in a real-world testbed. Both experiments and results are described in the following sections.

6.6.1 Simulation Environment

For the simulation of the alarm management algorithms we assume the cameras and node to be positioned on an area with $800m * 800m$ square. The transmission range of a camera is set to $160m$ and the number of cameras simulated varies between 10 and 100. The performance criteria we applied for the evaluation of our algorithms are the following:

- Time needed to emit search request
- Time needed to notify guards
- Message complexity

The first two points consider the timing behaviour of the routing algorithm. For the simulation experiments, only network timing is considered. We do not model possibly unstable computer vision algorithms for event detection but use fixed values derived from

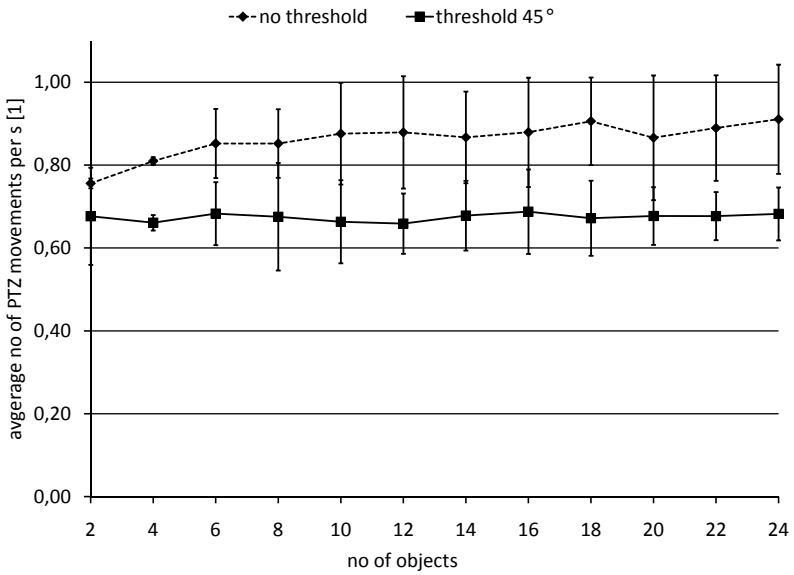

Figure 6.20: Impact of movement strategy on the number of PTZ turnings

our real-world testbed. We assumed a detection time of $2s$, which seems an appropriate value according to Figure 6.26.

In systems of varying size (x-coordinate of Figure 6.21), a node has been chosen randomly to emit SEARCH requests. A second node has been chosen to reply to the request with a NOTIFICATION message. The time between emission of a SEARCH request and its arrival on all cameras on the system has been measured. Average values have been measured between $2.5s$ for systems with only 10 cameras to $12s$ for larger systems consisting of 100 cameras (y-coordinate of Figure 6.21).

The notification of guards is much faster than the emission of SEARCH requests. This is due to the fact, that the notification is successful as soon as the first guard has been found whereas the emission of SEARCH requests finishes after all cameras have received the request. The notification algorithm terminates after all cameras have forwarded the corresponding NOTIFICATION, i.e. the overall time needed to notify all guards in the system is identical to the emission of a SEARCH request. Even in large system of 100 cameras, a guard is found in $4s$ and a backward channel is established to transfer video and image data from the cameras to the guard. In terms of timing behaviour, one can state that AMiDiViN is suited well for today's wireless networks.

A further metric that gives an insight on the performance of AMiDiViN is the average number of cameras that need to forward requests. Message forwarding is energy consuming and should therefore be kept at a minimum level - this is an important aspect

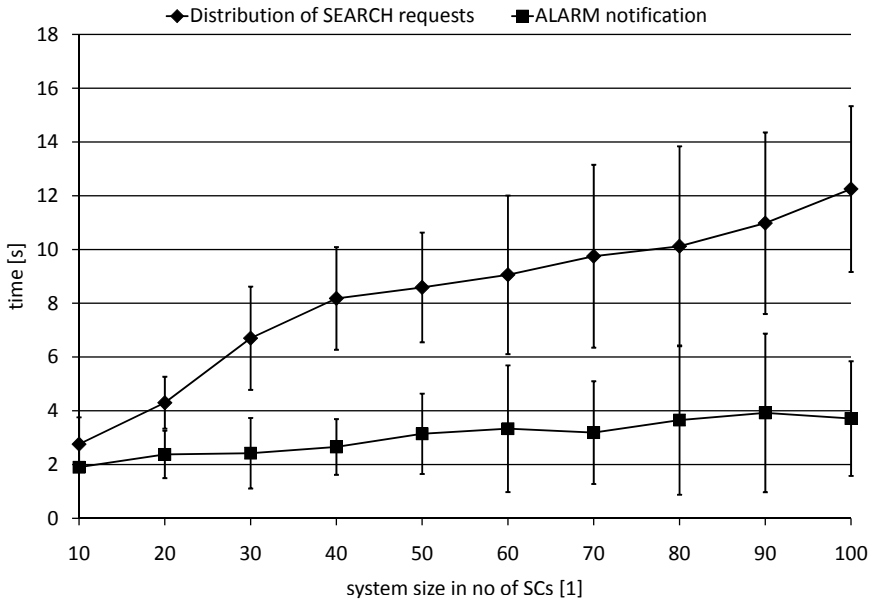

Figure 6.21: Time needed for disseminating a search requests to SC network of varying size

for battery powered systems. Furthermore, an ineffective broadcasting of messages may lead to a choking of network traffic and delays in transmission of data. A flooding based broadcast involves all cameras in a system, as depicted in Figure 6.22. In this figure, the number of cameras involved in the forwarding algorithm for both unicast and broadcast approach are shown. The set of cameras that forwards messages is a physical network segment where data packets can collide with one another for being sent on a shared medium. A system benefits from the small number of cameras involved in the routing process.

For lightweight notification tasks a broadcasting scheme is effective in terms of message and time complexity. The delivery of video data in large networks benefits from a low number of forwarding cameras achieved by a unicast communication scheme. AMiDiViN creates a backward path by establishing a routing table on each camera on the direct route between detecting cameras and guards. This routing system helps to minimise the number of packet collisions on the shared media and avoids broadcast storms. Figure 6.22 further shows the average number of routing hops on the backward path.

6.7 Evaluation in Real World Testbed

Large camera networks as installed at airports consist of thousands of cameras. Since such a large networks have not been available for evaluation purposes of the management

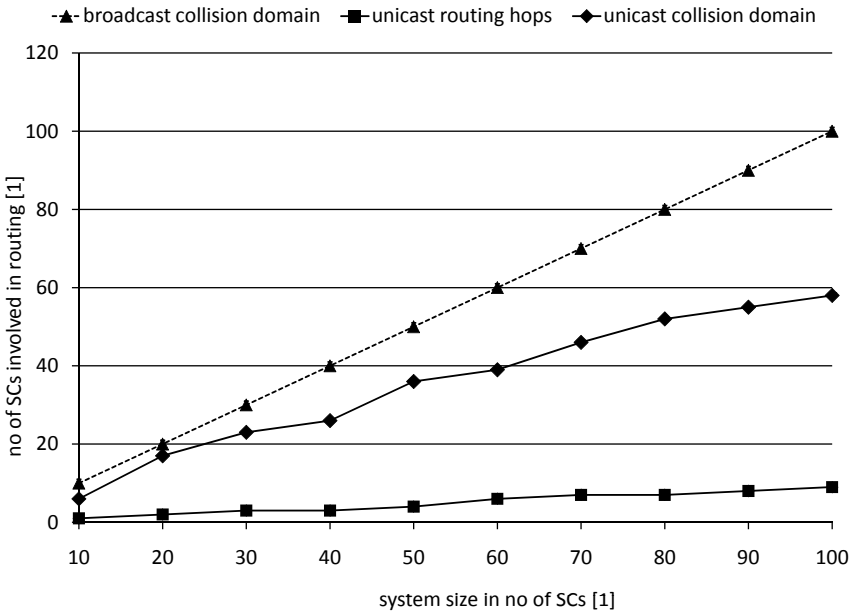

Figure 6.22: Number of cameras involved in routing: Unicast streaming on backward path in comparison to broadcasting

algorithms presented in this thesis, these large networks have been simulated. In order to provide a realistic simulation environment, several experiments have been conducted in a real world testbed. This testbed consists of several cameras, that are installed in a research lab and surrounding university corridors. As mentioned in Section 6.2, all system management algorithms can be transferred from a simulation environment to real world prototypes. The following section presents a real-world prototype as used for the evaluation.

6.7.1 Prototype

The software components introduced in Chapter 4 have been evaluated on a real world-prototype. A picture of the Smart Camera prototype is shown in Figure 6.23. The camera consists of an off-the-shelf PTZ camera (Axis PTZ214) and a computing unit. The computing unit is a miniature-sized, Linux-based embedded PC (Intel Atom 330 running at 1.6GHz with 1GB RAM). It can be connected to various kinds of cameras, reaching from USB and FireWire webcams to network cameras. This platform is used for the implementation the Smart Camera node software framework. For the communication with other nodes, an IEEE 802.11 WLAN or an IEEE 802.3 Ethernet device is used. Various interfaces are available to connect the computing unit to the camera. Since

Figure 6.23: Smart Camera prototype

their capabilities in terms of data rate and CPU usage heavily acts the overall system performance, the interface needs to be chosen carefully.

The framework uses Linux as operating system. This bears the advantage, that most hardware drivers are available in source code and can be adapted to the usage in a Smart Camera. A standard distribution (Ubuntu [49]) is used that comes with appropriate drivers to connect various kinds of camera to the system. The interface to the camera's video data is provided by V4L2 (video for Linux v2). The *Actuators* and *Sensors* are part of the PTZ camera depicted in Fig 6.23. The *Actuator* is the PTZ functionality of the camera. An extensible control library for *PTZ* camera movement has been implemented for this thesis [76]. After image data has been collected from the camera's CCD *Sensor* the computer vision framework OpenCV [25] is used for *Image Processing*.

6.7.2 Evaluation of Alarm Management

For the alarm management, a testbed with 5 cameras has been used. These cameras have been installed at the institute's corridor as depicted in Figure 6.24. The cameras use a Logitech Quickcam Pro 9000 as image capturing device. The aforementioned Atom based miniature PCs are used as computing units. The mobile terminal used for the connection between camera network and surveillance staff is an Apple iPhone with IEEE 802.11 WLAN interface in ad-hoc mode. The user of this terminal selected a predefined object (identified by its colour histogram) and publishes this search request. Each camera

capable of recognising this object subscribes to this request and starts searching for the object. In case the object has been found, the position of the detected object is sent back to the terminal. This experiment is in the following discussed in detail:

Five cameras are installed at the ceiling of the floor. The fields of view of the cameras overlap slightly so that a seamless trajectory estimation of moving persons becomes possible. This testbed has been used to evaluate the publish subscribe system for alarm management. A mobile terminal has been connected to one of the cameras.

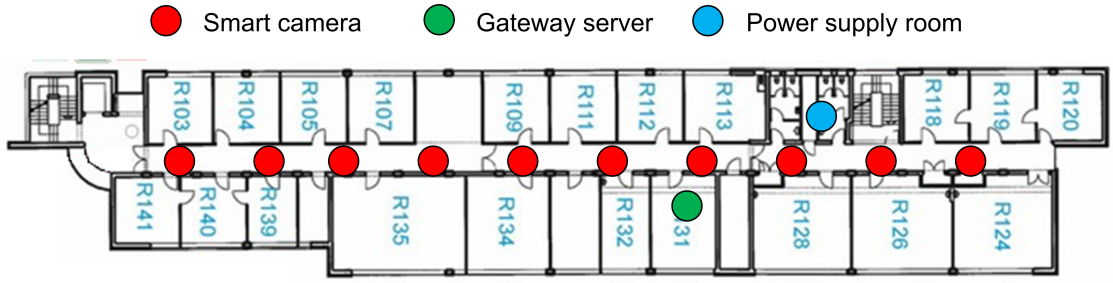

Figure 6.24: Real-world testbed

6.7.3 Object recognition

To give an insight not only on the networking aspects of the system presented in this thesis, we shortly present results of basic image recognition we used for the evaluation of the alarm management algorithm. The following algorithm is used for event detection.

Algorithm 9 Analyse captured frame

```
1: AnalyseCapturedFrame()
2: detect upper bodies U
3: for (each element u in U) do
4:    calculate colour histogram
5:    for (each element r in Q) do
6:       compare u to r
7:       if u is very similar to r then
8:          notifyguard
9:       end if
10:   end for
11: end for
```

The algorithm can be separated in three sections: detection of objects, tracking of these objects by features and location of objects of particular interest. The computer vision library OpenCV [101] is used for the basic computer vision algorithms. The object

detection module is an upper-body detection with Haar-classifiers for the detection of persons introduced by Kruppa et. al in [102]. Based on this detection, distinctive features can be calculated, for example colour-histograms [103] as used in our demonstrator. Since our research focus is on networking aspects, we do not dwell on more sophisticated algorithms that could be used instead. In consideration of their individual features, the persons can be distinguished and can be tracked. Each histogram is compared to histograms of persons which are already known by the Smart Camera. To locate persons of special interest, security staff marks a person to find in a still image which is depicted on the mobile terminal. Based on this selection, a colour histogram is calculated and transfused to the Smart Camera network. The guard will be notified, when the person is found.

The correlation coefficient d_{corr} between two histograms is calculated as

$$d_{corr}(H_1, H_2) = \frac{\sum_i H_1'(i) \cdot H_2'(i)}{\sqrt{\sum_i H_1'^2(i) \cdot H_2'^2(i)}}$$

with $H_k'(i) = H_k(i) - (1/N)(\sum_j H_k(j))$ and N equals the number of bins in the histogram. For AMiDiViN, 3-D colour histograms have been used with 125 bins. In case d_{corr} becomes larger than 0.8, the histograms are assumed to belong to identical objects (line 6). $d_{corr} = 0$ indicates total mismatch and $d_{corr} = 1$ indicates a perfect match. The size of a histogram results to 512 Byte which allows histograms to be transferred in the network without packet fragmentation. The impacts of this simplified detection algorithm becomes obvious in the following.

6.7.4 Object Detection

The colour histogram derived from moving persons is used for triggering alarms. By comparing the detected colour histogram with those histograms stored in the search cache, a camera decides whether to raise an alarm and notify guards or not. In order to evaluate the robustness of this approach, the following experiment has been conducted. The histogram of a test person's clothing is selected manually on a MAMT and distributed to the cameras in the testbed depicted in Figure 6.24. On the corridor, several persons pass by the cameras and each camera is expected to detect the person defined by the histogram stored in the search cache. Table 6.4 shows experimental results for a varying number of persons and appearances of the searched person in a single cameras field of view.

Results show, that the upper body detection works well. Only 1 out of 10 appearances of a person have not been detect in time (column: Alarms). In time means, we assume

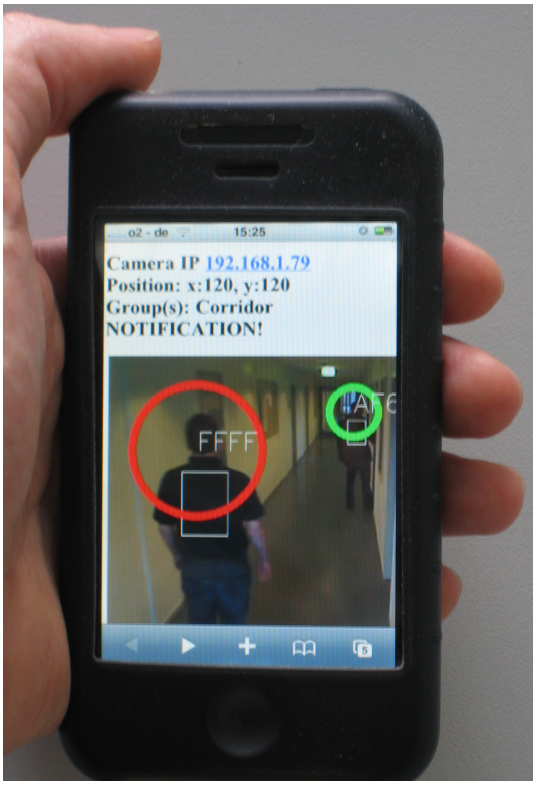

Figure 6.25: Webpage as displayed on MAMT (iPhone)

No of persons	Appearances	Alarms	False Alarms
1	10	10	0
2	10	10	0
3	10	9	2
4	10	9	4
5	10	9	4

Table 6.4: Detection results of upper body detection

that in case a person has been moving longer than $10s$ in the cameras field of view without being detected, the person is not detected at all. Distinguishing persons solely by the colour of their clothing introduced in Section 6.7.3 is not reliable. In case 3 or more persons are compared to a histogram, the false alarm rate is high (20% to 40%), see column: False Alarms.

In future, more sophisticated detection algorithms need to be deployed to make AMiDiViN able to cope with real-world settings where low false alarm rates are required. However, we assume that for the evaluation of AMiDiViN (with a focus on networking issues rather than on computer vision), realistic assumptions about the overall timing behaviour of the system can be made.

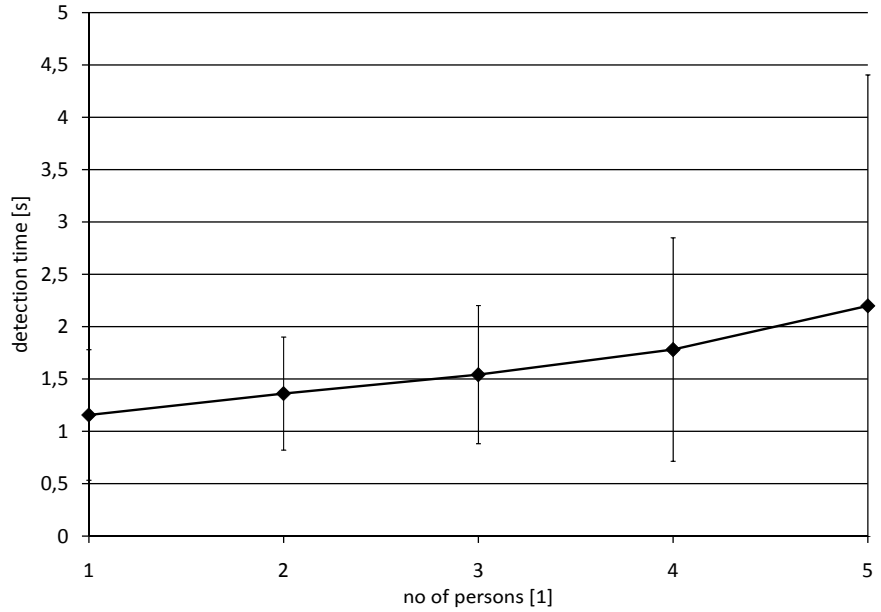

Figure 6.26: Detection time, errorbars show standard deviation

In order to measure the timing behaviour of our real-world testbed, test persons are walking randomly in the field of view of the cameras. At time $t_0 = 0s$ SEARCH request for one of these persons is fed into the camera system by a MAMT. Each camera receiving a SEARCH message starts the process of histogram comparison and in case the object is detected, an alarm is raised.

The time passing by between emitting a SEARCH request for a person (t_0) and the arrival of a corresponding NOTIFICATION t_1 has been measured for different scenarios as presented in Figure 6.26. In case a single person needs to be detected, the average time needed is $t_1 - t_0 = 1.3s$ (average values for 10 runs). In case 5 persons are inside the cameras field of view, the detection time is up to $8s$. The standard deviation is rather high ($0.6s$ to $2.2s$). This is due to the influence of the upper body detection. Persons are recognised only in case their pose allows for their detection which is the main impact on the detection time. The delay caused by the message transmission time is rather low: during the test runs, the average round trip time between the emitting camera and the detecting camera (1-Hop WLAN) was $152ms$ in average ($62ms$ standard deviation). These values can be neglected in comparison to the time needed for image analysis.

6.8 Overall Conclusion

The previous sections investigated system management algorithms for Distributed Vision Networks. The spatial partitioning of an area is achieved by ROCAS. Tracking of objects is handled by DMCtrac and the user interaction is taken over by distributed routing algorithms. Several metrics have been defined that allow to measure the algorithms performance. The main metrics are time and message complexity and accuracy. Several scenarios have been derived from real-world applications and the performance of the algorithms is evaluated in a network simulator. The size of the area the cameras are positioned on is up to $640000m^2$, i.e. rather large areas as they might appear in real-world.

In the first part of this chapter, ROCAS has been investigated. It has been determined, that ROCAS terminates in a modest amount of time: for large networks of 1000 cameras, no more than $120s$ are needed to finish the camera alignment, see Section 6.4.2. With this experiment it has further been proven, that ROCAS is suited well for networks consisting of at least 1000 nodes. With experiments presented in Sections 6.4.1 and 6.4.4, it has been shown that ROCAS is robust towards node failure and losses on the communication channel. The message complexity of ROCAS is low in comparison to the bandwidth available in todays communication systems, see Section 6.4.3. By adjusting several protocol parameters, the behaviour of ROCAS can be adjusted to specific needs arising in different situations (Section 6.4.8). A centralised variant of ROCAS has been presented, too. Results show, that for some situations a centralised algorithm achieves better quality results than the distributed algorithm, see Section 5.3.2. Usually, the distributed algorithm terminates faster and achieves comparable results as shown in Section 6.4.10.

For tracking objects with multiple cameras, this thesis introduced DMCtrac. The evaluation of DMCtrac showed, that depending on the underlying computer vision algorithm, a seamless tracking of objects is feasible in today's wireless networks. Section 6.5.2 shows, that single objects can be tracked without disturbances. A system with 12 cameras can track up to 50 objects simultaneously. A message loss rate of up to 10% can be compensated partly and does only slightly affect the overall performance. By applying different strategies for PTZ movement, the lifetime of cameras can be prolonged by accepting less accurate tracking results.

For the notification of guards a system for user interaction has been evaluated in Section 6.6. Even in large systems of up to 100 cameras, the dissemination of requests sent by a user to all cameras in a system does not take more than $14s$.

Image recognition and scene analysis are not investigated in detail here but their impact has been considered as defined in Section 6.7.3. Results show, that the management of Distributed Vision Networks benefits from the algorithms presented in this thesis. The time complexity of the algorithms can be considered as very low, aligning cameras and distributing notifications in the networks takes only few seconds – even in large networks consisting of hundreds of cameras. By a traffic shaping mechanism, the bandwidth consumption can be kept low. For example, ROCAS consumes below 8 kBits/s. This shows, that todays' wireless networks can cope with these algorithms. Furthermore, the heuristics presented are lightweight and have low computation complexity. The evaluation shows, that overall system presented in this thesis can be used on embedded camera devices that are available today. Operators of large surveillance systems can benefit from self-x properties and short reaction times.

Chapter 7

Conclusion

This thesis presents system management algorithms for Distributed Vision Networks. These algorithms enable Smart Cameras to cooperate in order to solve surveillance tasks. Based upon advances in the research areas of computer vision and embedded systems, this work introduces a system architecture that serves as a basis for high-level applications e.g. for securing public transport facilities. The analog camera systems that are used today in these areas are expected to be replaced by digital solutions in the future. Smart Cameras combining CCD sensor and computing unit are the basis for such systems. Appropriate software enables them to take over surveillance tasks in a cooperative way.

7.1 Summary of Contributions

Several distributed algorithms have been investigated and analysed thoroughly. They allow for self-organising camera alignment and target tracking with multiple cameras. Special prerequisites have been considered to make the algorithms cope with harsh environments causing communication failures like message loss. An evaluation with a realistic model of the underlying wireless network shows, that these algorithms are suitable for very large networks consisting of at least 1000 Smart Cameras. They terminate fast and by far exceed manually managed systems in terms of reliability and efficiency.

Several conclusions can be drawn in the end of this work. In order to build robust Distributed Vision Networks that act self-organising, self-optimising, self-healing and self-explaining, several issues have to be concerned. As stated in Chapter 1, three main problems have been addressed with this thesis:

- System architecture: A system architecture based upon wireless networked Smart

Camera nodes has been designed as described in Chapter 4. This architecture allows cameras to self-organise their behaviour and makes way for cooperative tasks to be carried out by the cameras. Different roles that cameras can take over in such system have been introduced and implemented to measure their performance under real-world constraints. Furthermore, a middleware for Smart Cameras has been developed that serves as a basis for high level management algorithms. This middleware is tailored to suit the needs arising in Distributed Vision Networks. Especially, a close coupling of image data and applications has been taken care for to allow for high speed processing on embedded systems with only modest amount of computing resources.

- Spatial partitioning algorithms: With a suitable system architecture as a basis, the problem of PTZ camera alignment has been discussed. Solutions to this mathematical problem, that has been shown to be NP-complete in Chapter 5, can be approximated by distributed heuristics. These algorithms enable cameras to self-optimise their alignment without using a central entity. For example, an algorithm has been investigated that enables cameras to track objects cooperatively by handing over objects from one camera to the next. By using a realistic network simulation, the algorithms have been shown to be well-suited for large networks in terms of functionality and performance.

- Alarm Management in Distributed Vision Networks: In order to let humans control Distributed Vision Networks, a system has been presented that manages the interaction of users and cameras. Requests can be sent by users e.g. from mobile terminals that are connected to the Distributed Vision Network. These requests may concern the system management in terms of alignment constraints but also application specific requests concerning computer vision. Moving objects can be selected by a user and cameras take over the task of searching for these objects throughout the camera network. In return, the cameras can alarm a guard as soon as they have detected such an object. An appropriate routing scheme for the dissemination of requests throughout the camera network has been designed and evaluated.

With the algorithms presented in this thesis, a class of management algorithms for Distributed Vision Networks has been introduced. All algorithms have been evaluated in scenarios derived from real-world applications. For the spatial partitioning algorithm, the precincts of an airport have been considered. These wide areas need to be protected against potential attackers. Simulation experiments show that cooperating Smart Cameras are suited well for this task. The area coverage becomes close to optimal so that

dangerous events can be detected beforehand. The camera alignment takes only short time, even in large systems of several hundred cameras less than one minute is needed until the cameras have chosen their final alignment.

The tracking of objects has been simulated with realistic trajectories derived from persons moving to and away from a university building. Multiple cameras can track multiple objects in the most efficient manner, e.g. so that frontal faces of persons (as needed for identification) are captured with high probability. The impact of message loss and drive wearout on the tracking quality have been investigated, too. With the algorithm presented here, operators of surveillance systems can increase security while at the same time avoiding a rise in cost for staff. Applications for these algorithms are, for example, security applications at airports and other public transport facilities. With advances in the areas embedded systems and computer vision, complex tasks can be carried out by cameras autonomously. Especially a real-time detection of potentially dangerous situations becomes now feasible. The privacy preserving character of Smart Camera systems may lead to an increasing security in areas, where today's camera systems can only be used to record data and analyse incidents in their aftermath. With this thesis, basic algorithms have been presented that allow for self-organisation and self-optimisation in Distributed Vision Networks even in large systems with thousands of cameras.

7.2 Future Research Opportunities

Based upon the management algorithms presented in the previous chapters, future research opportunities arise. For each of the three main contributions of this thesis, some possible links to future research topics are presented in the following.

Camera Alignment

The alignment of PTZ cameras has been investigated for a 2-dimensional model of the field of view only. This model offers sufficient accuracy for many applications, as described in Section 5.3. In case a 2-dimensional, triangular ground plane view does not suffice to reach an appropriate accuracy, e.g. for cameras being installed on different heights, a 3-dimensional model is needed. Currently, ROCAS makes use of a polygon clipping library in order to calculate the 2-dimensional overlap between cameras. In order to switch to a 3-dimensional model, this polygon clipper needs to be exchanged for a library allowing to calculate the intersections between volumetric shapes. With this enhancement, more sophisticated alignment tasks can be solved. In order to acquire stereo images of a scenery,

cameras could thereby calculate their pan/tilt alignment with higher precision.

Tracking

The tracking algorithm DMCtrac that has been introduced in Chapter 5.4.2 can be enhanced in order to achieve a better tracking quality. The success of handing over an object from one PTZ camera to another without overlapping fields of view depends heavily on the object's behaviour. An object might spontaneously decide to change its route while not being watched by any camera. Currently, the motion vector of an object is calculated and used to determine a possible migration region of the object. Possible candidate cameras for taking over the object will align their PTZ head in order to recognise the object. A first enhancement is to calculate the movement speed of an object and therefore allow for a more robust prediction of the objects arrival time at another camera. In order to cope with unexpected changes in the trajectories of objects, the use of machine learning techniques seems promising. Usually, objects move on common routes, i.e. planes move on a taxiway. Smart Cameras can learn these common routes and anticipate an object's behaviour in order to reach a more efficient tracking quality.

Alarm Management

The alarm management as presented in this thesis offers only basic notification mechanisms. The user interface could be enhanced in order to allow for a more comfortable interaction with the camera system. A future version of the interface might also include a graphical representation of the cameras' alignment and an online view of objects moving in sight of the cameras. In this context, the live streaming of video to the mobile device is of greatest interest, since it allows for a cooperation of human staff and Smart Cameras.

This list of research opportunities is way too short to cover all possible issues that need to be investigated in future. It contains some ideas that arose in context with this dissertation and might be considered in upcoming research projects. This work presents algorithm for network management. Especially in the fields of collaborative behaviour of cameras and high level algorithms for image recognition a lot of work still needs to be done.

Bibliography

[1] Sergio Velastin and Paolo Remagnino, *Intelligent Distributed Video Surveillance Systems (Professional Applications of Computing)*, Institution of Engineering and Technology, 2006.

[2] Marcel Winter, "Von der traditionellen Überwachung zur New Surveillance," *Universität Bielefeld, Fakultät für Soziologie, Bachelorarbeit*, September 2007.

[3] Martin Hoffmann, Michael Wittke, Jörg Hähner, and Christian Müller Schloer, "Spin-off: Autonomous Smart Camera Systems," in *Industrial Track of the 5th International Conference on Autonomic and Trusted Computing*, 2008.

[4] M. Hoffmann, M. Wittke, J. Hähner, and C. Müller-Schloer, "Spatial Partitioning in Self-Organizing Smart Camera Systems," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 2, no. 4, pp. 480–492, Aug. 2008.

[5] M. Hoffmann, J. Hähner, M. Wittke, Y. Bernard, and R. Soleymani, "DMCtrac: Distributed Multi Camera Tracking," *Distributed Smart Cameras, 2008. Second ACM/IEEE Int. Conference on*, Sept. 2008.

[6] Martin Hoffmann, Michael Wittke, and Jörg Hähner, "Design and evaluation of a notification system for alarm management in Distributed Vision Networks," in *29th International Conference on Distributed Computing Systems Workshops (ICDCS 2009 Workshops), 2009, June 22-26, Montreal, Quebec, Canada*, 2009.

[7] Jürgen Branke, Moez Mnif, Christian Müller-Schloer, Holger Prothmann, Urban Richter, Fabian Rochner, and Hartmut Schmeck, "Organic Computing – Addressing complexity by controlled self-organization," in *Proc. of the 2nd Intern. Symp. on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2006)*, 2006, pp. 185–191.

[8] Jeffrey O. Kephart and David M. Chess, "The Vision of Autonomic Computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[9] Joseph O'Rourke, *Art gallery theorems and algorithms*, Oxford Univ. Press, Inc., New York, NY, USA, 1987.

[10] C.K. K Toh, *Ad Hoc Wireless Networks: Protocols and Systems*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[11] S. Hengstler and H. Aghajan, "A Smart Camera Mote Architecture for Distributed Intelligent Surveillance," in *Working Notes of the International Workshop on Distributed Smart Cameras (DSC)*, 2006.

[12] Dhanya Devarajan and Richard J. Radke, "Distributed metric calibration of large camera networks," in *Broadband Advanced Sensor Networks (BASENETS)*.

[13] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Mobile Computing and Networking*, 2001, pp. 166–179.

[14] A. Kotanen, M. Hannikainen, H. Leppakoski, and T.D. Hamalainen, "Positioning with IEEE 802.11b wireless LAN," *Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003. 14th IEEE Proceedings on*, vol. 3, pp. 2218–2222, 2003.

[15] R. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 4, pp. 323–344, Aug 1987.

[16] D. A. Wang, F. Bello, and A. Darzi, "Augmented reality provision in robotically assisted minimally invasive surgery," *CARS 2004 - Computer Assisted Radiology and Surgery. Proceedings of the 18th International Congress and Exhibition*, pp. 527–532, Jun 2004.

[17] Zhaolin Cheng, Dhanya Devarajan, and Richard J. Radke, "Determining vision graphs for distributed camera networks using feature digests," *EURASIP J. Appl. Signal Process.*, vol. 2007, no. 1, pp. 220–220, 2007.

[18] N.J.B. McFarlane and C.P. Schofield, "Segmentation and tracking of piglets in images," *MVA*, vol. 8, no. 3, pp. 187–193, 1995.

[19] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2, 1999.

[20] Nuria M. Oliver, Barbara Rosario, and Alex P. Pentland, "A bayesian computer vision system for modeling human interactions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 831–843, 2000.

[21] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 10, pp. 1337–1342, Oct. 2003.

[22] Paul Viola and Michael Jones, "Robust real-time object detection," in *International Journal of Computer Vision*, 2001.

[23] Yoav Freund and Robert E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *EuroCOLT '95: Proceedings of the Second European Conference on Computational Learning Theory*, London, UK, 1995, pp. 23–37, Springer-Verlag.

[24] Y. R. Loke, P. Kumar, S. Ranganath, and W M. Huang, "Object matching across multiple non-overlapping fields of view using fuzzy logic," *The University of Adelaide Digital Library (Australia)*, 2006.

[25] Gary R. Bradski, "Real time face and object tracking as a component of a perceptual user interface," in *WACV '98: Proceedings of the 4th IEEE Workshop on Applications of Computer Vision (WACV'98)*, Washington, DC, USA, 1998, p. 214, IEEE Computer Society.

[26] David G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[27] Senem Velipasalar, Jason Schlessman, Cheng-Yao Chen, Wayne Wolf, and Jaswinder Singh, "SCCS: A scalable clustered camera system for multiple object tracking communicating via message passing interface," in *ICME*, 2006, pp. 277–280.

[28] Markus Quaritsch, Markus Kreuzthaler, Bernhard Rinner, Horst Bischof, and Bernhard Strobl, "Autonomous multicamera tracking on embedded Smart Cameras," *EURASIP J. Embedded Syst.*, vol. 2007, no. 1, pp. 35–35, 2007.

[29] Matthew Turk and Alex Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[30] J. Shi and C. Tomasi, "Good features to track," *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 593–600, June 1994.

[31] Bruce D. Lucas and Takeo Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, April 1981, pp. 674–679.

[32] F. Cupillard, A. Avanzi F. Brémond, and M. Thonnat, "Video understanding for metro surveillance," *The IEEE ICNSC 2004 in the special session on Intelligent Transportation Systems*, Mar. 2004.

[33] Alper Yilmaz and Mubarak Shah, "A differential geometric approach to representing the human actions," *Comput. Vis. Image Underst.*, vol. 109, no. 3, pp. 335–351, 2008.

[34] F. Dias, F. Berry, J. Serot, and F. Marmoiton, "Hardware, design and implementation issues on a FPGA-based Smart Camera," in *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, 2007, pp. 20–26.

[35] Sven Fleck and Wolfgang Straßer, "Smart Camera based monitoring system and its application to assisted living," *Proceedings of the IEEE, Special Issue on Distributed Smart Cameras*, vol. 10, 2008.

[36] K. Martinez, R. Ong, J. K. Hart, and J. Stefanov, "Glacsweb: a sensor web for glaciers," in *Proceedings of 1st European Workshop on Wireless Sensor Networks. EWSN '04*, 2004, pp. 56–62.

[37] R. Beckwith, D. Teibel, and P. Bowen, "Pervasive computing and proactive agriculture," in *Proceedings of PERVASIVE 2004, Vienna, Austria*, April 2004.

[38] XBow Inc., "Mica2 mote datasheet," http://www.xbow.com/.

[39] K. Römer and F. Mattern, "The design space of wireless sensor networks," *Wireless Communications, IEEE [see also IEEE Personal Communications]*, vol. 11, no. 6, pp. 54–61, 2004.

[40] Leibniz Universität Hannover, "Verfahren zur verteilten Erzeugung miteinander korrelierender Daten," *Anmeldung beim DPMA 1020090059784*, 2009.

[41] Ioannis Pavlidis and Vassilios Morellas, "Two examples of indoor and outdoor surveillance systems," *Vision based surveillance systems*, pp. 39–51, 2002.

[42] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu, "The broadcast storm problem in a mobile ad hoc network," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, New York, NY, USA, 1999, pp. 151–162, ACM.

[43] Christopher Ho, Katia Obraczka, Gene Tsudik, and Kumar Viswanath, "Flooding for reliable multicast in multi-hop ad hoc networks," in *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 1999, pp. 64–71.

[44] Jörg Hähner, Dominique Dudkowski, Pedro Jose Marron, and Kurt Rothermel, "Quantifying network partitioning in mobile ad hoc networks," in *MDM '07: Proceedings of the 2007 International Conference on Mobile Data Management*, Washington, DC, USA, 2007, pp. 174–181, IEEE Computer Society.

[45] Abdelmajid Khelil, *A Generalized Broadcasting Technique for Mobile Ad Hoc Networks*, Ph.D. thesis, University of Stuttgart, 2007.

[46] Charles E. Perkins and Elizabeth M. Royer, "Ad-hoc on-demand distance vector routing," in *In Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999, pp. 90–100.

[47] David B. Johnson, David A. Maltz, and Josh Broch, *DSR: the dynamic source routing protocol for multihop wireless ad hoc networks*, pp. 139–172, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, December 2000.

[48] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "Tinyos: An operating system for sensor networks," pp. 115–148. 2005.

[49] Kyle Rankin and Benjamin Mako Hill, *The Official Ubuntu Server Book*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2009.

[50] Christian Becker, Gregor Schiele, Holger Gubbels, and Kurt Rothermel, "Base - a micro-broker-based middleware for pervasive computing," in *PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, Washington, DC, USA, 2003, IEEE Computer Society.

[51] Wolfgang Trumler, Faruk Bagci, Jan Petzold, and Theo Ungerer, "Amun - autonomic middleware for ubiquitous environments applied to the smart doorplate project," in *ICAC '04: Proceedings of the First International Conference on Autonomic Computing*, Washington, DC, USA, 2004, pp. 274–275, IEEE Computer Society.

[52] Andreas Pietzowski, Wolfgang Trumler, and Theo Ungerer, "An artificial immune system and its integration into an organic middleware for self-protection," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, New York, NY, USA, 2006, pp. 129–130, ACM.

[53] Henning Perl, "Entwurf und Implementierung einer komponentenbasierten Middleware für Smart Cameras," *Leibniz Universität Hannover, Bachelor Thesis*, August 2009.

[54] J. Everist, T. N. Mundhenk, C. Landauer, and K. Bellman, "Visual surveillance coverage: strategies and metrics," in *Intelligent Robots and Computer Vision*, Oct. 2005, pp. 91–102.

[55] Patrick Chisan Hew, "Visualisation of Surveillance Coverage by Latency Mapping," in *Australian Symposium on Information Visualisation, (invis.au'03)*, Tim Pattison and Bruce Thomas, Eds., Adelaide, Australia, 2003, vol. 24 of *CRPIT*, pp. 11–16, ACS.

[56] V. Chvatal, "A combinatorial theorem in plane geometry," *Journal of Combinatorial Theory Series B*, vol. 18, pp. 39–41, 1975.

[57] Uğur Murat Erdem and Stan Sclaroff, "Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements," *Comput. Vis. Image Underst.*, vol. 103, no. 3, pp. 156–169, 2006.

[58] Leila De Floriani and Paola Magillo, "Algorithms for visibility computation on terrains: a survey," *Environment and Planning B: Planning and Design*, vol. 30, no. 5, pp. 709–728, September 2003.

[59] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds., pp. 85–103. Plenum Press, 1972.

[60] Alan T. Murray, Kamyoung Kim, James W. Davis, Raghu Machiraju, and Richard E. Parent, "Coverage optimization to support security monitoring," *Computers, Environment and Urban Systems*, vol. 31, no. 2, pp. 133–147, 2007.

[61] Richard Cole and Micha Sharir, "Visibility problems for polyhedral terrains," *J. Symb. Comput.*, vol. 7, no. 1, pp. 11–30, 1989.

[62] S. Kang, J.-K. Paik, A. Koschan, B. R. Abidi, and M. A. Abidi, "Real-time video tracking using PTZ cameras," in *Sixth International Conference on Quality Control by Artificial Vision*, K. W. Tobin, Jr. and F. Meriaudeau, Eds., Apr. 2003, vol. 5132 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pp. 103–111.

[63] I. Everts, N. Sebe, and G. A. Jones, "Cooperative object tracking with multiple ptz cameras," in *ICIAP '07: Proceedings of the 14th International Conference on Image Analysis and Processing*, Washington, DC, USA, 2007, pp. 323–330, IEEE Computer Society.

[64] Ofir Avni, Francesco Borrelli, Gadi Katzir, Ehud Rivlin, and Hector Rotstein, "Scanning and tracking with independent cameras–a biologically motivated approach based on model predictive control," *Auton. Robots*, vol. 24, no. 3, pp. 285–302, 2008.

[65] Cintia Borges Margi, X. Lu, G. Zhang, G. Stanek, R. Manduchi, and K. Obraczka, "Meerkats: A power-aware, self-managing wireless camera network for wide area monitoring," in *Working Notes of the International Workshop on Distributed Smart Cameras (DSC)*, 2006.

[66] Norimichi Ukita and Takashi Matsuyama, "Real-time cooperative multi-target tracking by communicating active vision agents," *Comput. Vis. Image Underst.*, vol. 97, no. 2, pp. 137–179, 2005.

[67] Norimichi Ukita, "Real-time cooperative multi-target tracking by dense communication among active vision agents," *Web Intelli. and Agent Sys.*, vol. 5, no. 1, pp. 15–29, 2007.

[68] O. Steiger, T. Ebrahimi, and A. Cavallaro, "Surveillance video for mobile devices," *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on*, pp. 620–625, Sept. 2005.

[69] Sheng-Tun Li, Huang-Chih Hsieh, Ly-Yen Shue, and Wen-Shen Chen, "Pda watch for mobile surveillance services," *Knowledge Media Networking, 2002. Proceedings. IEEE Workshop on*, pp. 49–54, 2002.

[70] Rita Cucchiara, Costantino Grana, Andrea Prati, and Roberto Vezzani, "Computer vision techniques for pda accessibility of in-house video surveillance," in *IWVS '03: First ACM SIGMM international workshop on Video surveillance*, New York, NY, USA, 2003, pp. 87–97, ACM.

[71] Yoshiro Imai, Yukio Hori, and Yuichi Suigiue, "A web-based surveillance system for mobile phones," in *MOBILWARE '08: Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, ICST, Brussels, Belgium, Belgium, 2007, pp. 1–6, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[72] Florian Blatt, "Entwurf und Implementierung eines Routingverfahrens für verteilte Smart Camera Netze," *Leibniz Universität Hannover, Bachelor Thesis*, February 2009.

[73] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, 2000.

[74] Michael Neufeld, Ashish Jain, and Dirk Grunwald, "Nsclick: bridging network simulation and deployment," in *MSWiM '02: Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, New York, NY, USA, 2002, pp. 74–81, ACM Press.

[75] Lars Friedrichs, "Analyse von Sicherheitsrisiken in Smart Camera Netzen," *Leibniz Universität Hannover, Bachelor Thesis*, May 2008.

[76] Anna Averbakh, "Entwurf und Implementierung eines Rahmenwerkes zur Steuerung von Kameras," *Leibniz Universität Hannover, Bachelor Thesis*, September 2007.

[77] Liang Han, "Entwurf und Evaluierung eines Bildverarbeitungsalgorithmus für den Einsatz in verteilten kameranetzen," *Leibniz Universität Hannover, Bachelor Thesis*, September 2007.

[78] Markus Thielecke, "Design und Evaluation eines Hardware Abstraction Layers für Smart Cameras," *Leibniz Universität Hannover, Master Thesis*, Oktober 2009.

[79] Alec Woo and David E. Culler, "A transmission control scheme for media access in sensor networks," in *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, New York, NY, USA, 2001, pp. 221–235, ACM.

[80] George F. Coulouris and Jean Dollimore, *Distributed systems: concepts and design*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988.

[81] Tushar Deepak Chandra and Sam Toueg, "Unreliable failure detectors for reliable distributed systems," *Journal of the ACM*, vol. 43, pp. 225–267, 1996.

[82] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, no. 2, pp. 374–382, 1985.

[83] Benjamin Satzger, Andreas Pietzowski, Wolfgang Trumler, and Theo Ungerer, "A new adaptive accrual failure detector for dependable distributed systems," in *ACM Symposium on Applied Computing*, 2007.

[84] S. Vasudevan, J. Kurose, and D. Towsley, "Design and analysis of a leader election algorithm for mobile ad hoc networks," *Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference on*, pp. 350–360, Oct. 2004.

[85] Ahmed Fares, "Untersuchung und Implementierung eines Wahlalgorithmus unter Voraussetzung eines unzuverlässigen Übertragungskanals," *Leibniz Universität Hannover, Diploma Thesis*, December 2008.

[86] Edsger W. Dijkstra and C.S.Scholten, "Termination detection for diffusing computations," *Inf. Proc. Letters*, vol. 11, no. 1, pp. 1–4, 1980.

[87] Philipp Kleybolte, "Entwurf und Implementierung eines Rahmenwerkes zur Steuerung von Kameras," *Leibniz Universität Hannover, Bachelor Thesis*, September 2008.

[88] Stephen A. Cook, "The complexity of theorem-proving procedures," in *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, New York, NY, USA, 1971, pp. 151–158, ACM.

[89] Bala R. Vatti, "A generic solution to polygon clipping," *Comm. of the ACM*, vol. 35, no. 7, pp. 56–63, 1992.

[90] D. M. Bloom and W. Knight, "A birthday problem," *Amer. Math. Monthly*, vol. 80, no. 10, pp. 1141–1142, 1973.

[91] Christian Schulz, "Entwurf und Implementierung einer adaptiven Systemarchitektur für Smart Camera Netze," *Leibniz Universität Hannover, Diploma Thesis*, Januar 2009.

[92] Shiuh-Ku Weng, Chung-Ming Kuo, and Shu-Kang Tu, "Video object tracking using adaptive kalman filter," *J. Vis. Comun. Image Represent.*, vol. 17, no. 6, pp. 1190–1208, 2006.

[93] M. Frans Kaashoek, Andrew S. Tanenbaum, Susan Flynn Hummel, and Henri E. Bal, "An efficient reliable broadcast protocol," *Operating Systems Review*, vol. 23, pp. 5–19, 1989.

[94] "The Network Simulator ns-2," http://www.isi.edu/nsnam/ns/, October 2009.

[95] Elena Mogilevska, "Entwurf und Implementierung eines Werkzeugs zur Planung verteilter Kamerasysteme," *Leibniz Universität Hannover, Bachelor Thesis*, September 2007.

[96] Ramin Soleymani-Fard, "Konzeptionierung eines Rahmenwerkes zur Visualisierung verteilter Kameranetze," *Leibniz Universität Hannover, Bachelor Thesis*, September 2007.

[97] Wu Xiuchao, "Simulate 802.11b Channel within NS2," Tech. Rep., School of Computing, National University of Singapore, 2005.

[98] Sven Tomforde, Martin Hoffmann, Yvonne Bernard, Lukas Klejnowski, and Jörg Hähner, "Powea: protocol optimisation with evolutionary algorithms," *GI-Tagung Informatik*, 2009.

[99] Holger Prothmann, Fabian Rochner, Sven Tomforde, Jürgen Branke, Christian Müller-Schloer, and Hartmut Schmeck, "Organic control of traffic lights," in *Proceedings of the 5th International Conference on Autonomic and Trusted Computing*, 2008.

[100] T. N. Mundhenk, N. Dhavale, S. Marmol, E. Calleja, V. Navalpakkam, K. Bellman, C. Landauer, M. A. Arbib, and L. Itti, "Utilization and viability of biologically-inspired algorithms in a dynamic multi-agent camera surveillance system," in *Proc.*

SPIE Conference on Intelligent Robots and Computer Vision XXI: Algorithms, Techniques, and Active Vision, Bellingham, WA, Oct 2003, pp. 281–292, SPIE Press.

[101] Gary Bradski and Adrian Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly, Cambridge, MA, 2008.

[102] Modesto Castrillon-Santana Hannes Kruppa and Bernt Schiele, "Fast and robust face finding via local context," in *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, October 2003.

[103] Michael J. Swain and Dana H. Ballard, "Color indexing," *Int. J. Comput. Vision*, vol. 7, no. 1, pp. 11–32, 1991.

Lebenslauf

Name:	Martin Hoffmann
Geburtsdatum:	01.01.1981
Geburtsort:	Bielefeld
Familienstand:	ledig
Schulausbildung:	1987-1991 Grundschule Nord in Leopoldshöhe
	1991-2000 Gymnasium im Schulzentrum Werl Aspe in Bad Salzuflen
	Abschluß: Allgemeine Hochschulreife
Studium:	2001-2006 Studium der Elektrotechnik Studienrichtung Technische Informatik an der Leibniz Universität Hannover
	2005-2006 Auslandsaufenthalt an der University of Bristol, UK
	Abschluß: Diplom-Ingenieur
Berufstätigkeit:	2000-2001 Zivildienst
	2006-2009 wissenschaftlicher Mitarbeiter am Institut für Systems Engineering, Fachgebiet System und Rechnerarchitektur an der Leibniz Universität Hannover
	seit 08/2009 Geschäftsführer der Volavis GmbH